# A Study on Distributed Consensus Protocols and Algorithms: The Backbone of Blockchain Networks

Jayapriya Jayabalan
*School of Information Technology and Engineering*
*VIT*
Vellore, India
jayapriyabalan@gmail.com

Jeyanthi N
*School of Information Technology and Engineering*
*VIT*
Vellore, India
njeyanthi@vit.ac.in

2021 International Conference on Computer Communication and Informatics (ICCCI) | 978-1-7281-5875-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICCCI50826.2021.9402318

*Abstract*—**In a Blockchain network multiple nodes across the network verify each transaction and preserve them without having a centralized authority. Verification and insertion of transactions are achieved through distributed cryptographic mechanism called consensus protocol. Consensus protocol involves the application of reputable concepts like Distributed Computing (P2P Networks), Cryptography and consensus algorithms. Traditional motivation for application of Consensus Protocols is to warrant reliability in the distributed systems. The basic assumption behind this protocol is, a 'value' must have been proposed by some truthful node in the distributed system. Distributed consensus is attained once all the nodes present in the system agrees and approves the same value. At any given time, all nodes have a sequence of block of transactions, they have already reached consensus on. Also each node has a set of outstanding transactions it has heard about from other nodes. This consensus is achieved via various algorithms which serve as the backbone for Blockchain architecture. Once the transactions are verified, they will be included in the existing chain. The main advantage of such systems is the fact that they are immutable, transparent and distributed. Though stated simple, reaching consensus may be difficult due to various reasons such as crash of active node, presence of malicious nodes, faults in the network, latency (no global time) and not all pairs may be connected. A consensus algorithm ensures that the one and only version of truth gets added into the network, in addition to keeping malevolent people from tampering the network. It also makes sure that the system is fully decentralized. If everyone in the network is strictly honest, there is no need of consensus, however the probability of such existence is very less. Hence, we are undeniably in need of virtuous consensus algorithms for a strong, immortal Blockchain network. The process of selection and enactment of the suitable and right consensus protocol is the key for efficacious Blockchain network. This paper describes about distributed consensus and various algorithms which may be applied for implementation of a Blockchain network.**

*Index Terms*—**Consensus Protocol, Distributed Consensus, Consensus Algorithms, Byzantine Fault Tolerance, Distributed Computing, Peer-To-Peer Networking**

## I. Introduction

Consensus Protocols allow a decentralized network to arrive at an agreement about the state of the network. In a centralized system or federal organization all the decisions are taken by a single elected leader or a board of members. Whereas in a decentralized network, a leader will not be available to take decision independently, instead a group of system/nodes involve in decision making process. This process supports a decision, subject to the interest of all the people involved in the decision making process, called consensus [1]. The mechanism used to achieve consensus in a distributed or decentralized network is defined as consensus protocol. These consensus protocols are the backbones of the heavily disruptive technology 'Blockchain'. Blockchain is a decentralized peer-to-peer system with no centralized authority. By definition, this will be a system devoid of any type of corruption from a single point; but it results in some complex problems like reaching a collective decision and scheming mechanisms to get things done in such environment. Consensus Protocol solves the above problems in Blockchain by allowing the nodes to collectively reach an agreement on the transactions without relying on a third party. In voting systems, the solution with the majority of vote wins, there is no weightage for emotions or welfare of the minority. In consensus systems, decision is made for the benefit of the whole group scattered around the system, paving way to create a more democratic and nondiscriminatory society. The famous implementation of Blockchain is 'Bitcoin'; a cryptocurrency implemented using consensus protocol. Behind any cryptocurrency in today's world, there exists a consensus mechanism. Consensus protocols also ensure a mutual agreement is reached in the cryptocurrency and there is no double spending. These protocols are implemented in Blockchains using algorithms specifically designed for this purpose known as Consensus Algorithms. Due to their usage in distributed environments they are also known as Distributed Consensus Algorithms.

## II. Distributed Consensus Algorithms

Distributed consensus algorithms are the algorithms designed for reaching consensus in distributed environment. The reached consensus should be a satisfactory solution supported by every individual in the network, even though it is not their personal favorite solution. These algorithms make sure the next block/value added to the system is the one and only version of truth. This way we can define consensus as a vibrant way of reaching a general agreement and commonality of faith within the group. Fig. 1 shows illustration of consensus problem using three nodes, node 1, node 2 and node 3. Node 1 proposes 'value 2', Node 3 proposes 'value 1' to the other nodes, while Node 2 keeps quiet. Consensus algorithms should be designed in a way to address this problem model and reach on some value ultimately.

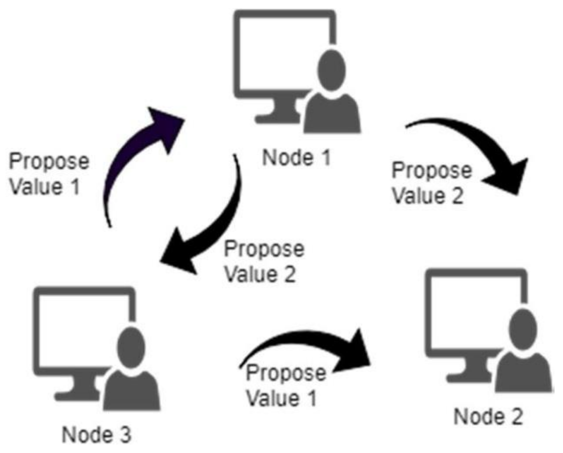

Fig. 1. Consensus Model

## A. Properties of Consensus Mechanisms

Distributed Consensus Algorithms have the properties shown in Fig. 2, to make the system work appropriately and prevent any kind of failure or glitch in the system.

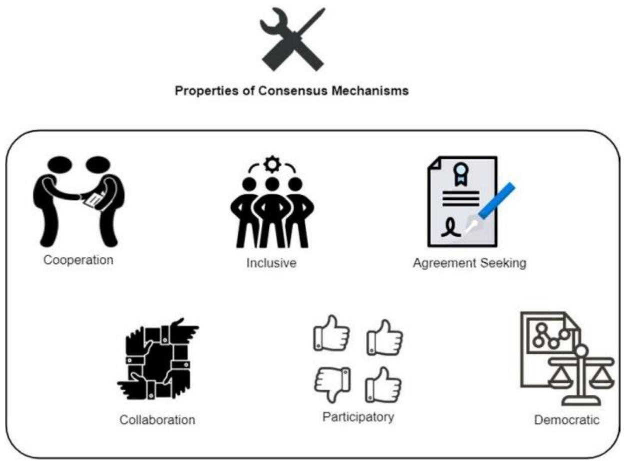

Fig. 2. Properties of Consensus Model

- Collaboration – All the nodes in the system should work together with the interest of the whole group as the objective
- Cooperation – System entities should work as a group rather than an individual with personal interests
- Inclusive – Make sure there is a maximum number of participation from the group
- Participatory – Active participation by the group is required for being successful
- Agreement Seeking – Bring as much as agreement from the individual nodes participating in the system
- Democratic – Each and every vote casted by the individuals should have an equal weightage

Consensus algorithm following the above mentioned properties forms the primary root structure of the revolutionary technology, Blockchain. Consensus algorithms deployed in various Blockchains makes them differ from each other. This functionality enables millions of nodes to exist in the same space; while they never exist mutually or hinder each other.

The processes highlighted in Fig. 3 shows the application of consensus algorithms in Blockchain. Such consensus algorithms reaches consensus if it satisfies the following conditions [2]:

- Agreement – Each one of the non-faulty nodes in the network should decide on the same output value
- Termination – Each and every non-faulty node should ultimately decide on some output value.

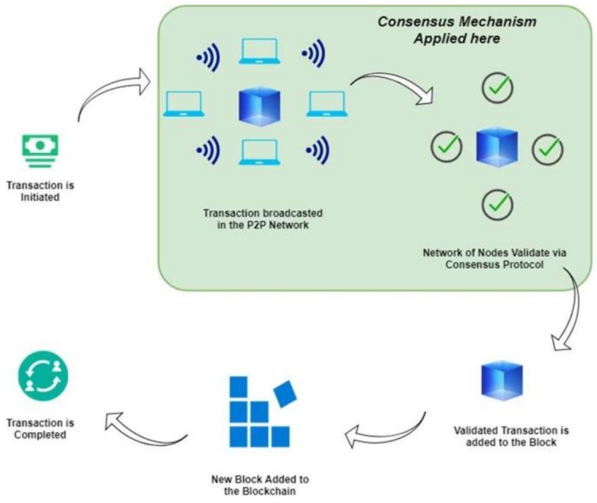
Consensus Mechanism Applied here
Transaction is Initiated
Transaction broadcasted in the P2P Network
Network of Nodes Validate via Consensus Protocol
Validated Transaction is added to the Block
New Block Added to the Blockchain
Transaction is Completed

Fig. 3. Application of Consensus in Blockchain

There are a lot of consensus algorithms designed and implemented in real world use cases, but each of those algorithms may vary in terminology, validity conditions (to achieve the final output value) and procedures on handling the decision making process.

## B. Network Synchrony

Network synchrony can be defined as the degree of synchronization between countless components in the system harmonizing with each other. Based on coordination, networks can be divided in to Synchronous and Asynchronous network.

*1) Synchronous Network:* In a synchronous network, a centralized clock synchronization service is used to coordinate the message communication in rounds. All the nodes in the system broadcasts the messages to each other in $i^{th}$ round and the nodes processes the received messages and broadcasts the output in $i+1^{th}$ round.

*2) Asynchronous Network:* There is no clock synchronization or coordination procedure in an asynchronous network. The network accomplishes all the tasks without any rules in an opportunistic manner. Since there is no rule or clock synchronization or upper bound on the message transmission delay, there is no guarantee the messages will be delivered properly.

### *C. Component Faults*

The network may have some faulty or dishonest nodes leading to failures in the systems. Depending on the type of fault created, the failure can be classified as crash failure or Byzantine failure.

*1) Crash Failure:* The node abruptly stops working and does not resume even after a good amount of time [3]. In these cases other components detects such a crash has happened and adjust all the decisions to be made locally.

*2) Byzantine Failure:* There is no absolute condition resulting in such failure; Byzantine failure occurs arbitrarily [4]. A node with such failure can either remain quiet or send conflicting messages to other nodes in the network. Such a node will not create any suspicion of the faulty process or malicious activity within the system.

### *D. Consensus Protocols*

Consensus protocols are the set of rules defining the consensus system's working, essentially implemented by three types of players.

- Proposers – Proposers are the leaders elected in the network
- Acceptors – These actors listen to value proposed by the proposers and provide response to the values
- Learners – The nodes which know of the final decided output value and updates the state are known as learners

Consensus model framework consists of Election, Vote and Decide steps involved in the implementation.

- Election – All the nodes combined selects the next leader who in turn will have the power to propose the next valid output value
- Vote – All the honest and non-faulty nodes in the network listens to the output value proposed by the leader, validate and process it as next valid value
- Decide – All the honest and non-faulty nodes must decide and come to a unanimity on the same truthful output value

Two rules define the consensus protocol's working: Message Passing Rule and Processing Rule.

*1) Message Passing Rule:* The rule followed by the components to broadcast or pass on the messages to other components within the network.

*2) Processing Rule:* Once a component receives the messages it should process the message and change the internal state of the system in response to the message.

### *E. Need of Consensus Algorithms*

All nodes present in the network may not be honest or non-faulty nodes. Hence consensus algorithms are needed for Blockchain or any other decentralized network to comply with the required set of functionalities to be implemented in the network mentioned below:

- All the components in the system decides on a commonly agreed value
- Every node in the network should behave in the same manner for a certain request
- There should not be any impact due to faulty or dishonest nodes and unreliable network communication

For the node to be non-faulty it should be devoid of all the possible failures like crash failure and Byzantine failure. The network is said to be Crash Fault Tolerant if it tolerates at least one crash failure and Byzantine Fault Tolerant if it tolerates at least one Byzantine failure. The main problem in Byzantine is to reach on a common value. If there is an event of error or failure in the system, the nodes cannot reach an agreement or the difficulty value might be higher. Consensus systems can work successfully only if all the actors in the network work in harmony. If there is a malfunction even in one system, the entire system may crash. Malfunctioning system can cause inconsistency and they are not ideal for decentralized network. Thus consensus algorithms are inevitable to wade of the failure or inconsistencies. Consensus algorithms do not face this type of problems and their primary goal is to reach a common agreement by some means making such systems more trustworthy and fault tolerant with better output.

In multi-agent systems or any other distributed systems the core obstacle is reaching consensus in the system. There is a lot of difficulty involved when a distributed network system is trying to come to an agreement or reach consensus on a value much needed for the decision to be taken during the computation process. Any component in such system is unpredictable, prone to failure making the network untrustworthy as a whole. The number of bad or dishonest nodes trying to tamper the network heavily influences the functioning of the network. If a system has only good actors with good intention and honest behavior, then consensus is not needed for them. Nevertheless distributed systems in real world scenario may have any number of bad actors making the consensus an essential and difficult to implement element.

### *F. Byzantine Generals Problem*

A reliable distributed computer system must be able to tolerate the failure of one or more components in the system. In case of failure the component may send conflicting or malicious information to other components in the system. To illustrate this scenario, the famous Byzantine Generals Problem has been considered. Several teams of lieutenants headed by generals are waiting outside a city planning for an attack. The generals will be able to communicate only through messengers and agree on a plan whether to attack the city or retreat. There might be traitors present in the teams who may prevent the generals to agree upon a single plan. This problem is famously known as the Byzantine Generals problem [5] [6]. We need an algorithm to guarantee a solution to the above problem guaranteeing:

- All the honest or loyal generals agree upon the same decision or plan for action (condition 1). In this case we assume loyal generals proceed as per the steps in the algorithm. Traitors may perform the next course of action as per their wish. The algorithm must ensure condition mentioned above is met irrespective of the presence of traitors. All the loyal people should agree upon a judicious action plan

- The presence of few numbers of dishonest people or traitors should not result in loyal people adopting a bad plan

### G. *FLP Impossibility*

The consensus process involves asynchronous systems which may be unreliable, such systems may not be able to come to an agreement or terminate even if there exists one faulty node. This nonterminating condition is known Fischer-Lynch-Paterson Impossibility (FLP Impossibility) result [7]. Consider a synchronous environment for example, in those systems messages will be delivered within fixed time duration. If the time frame is expired and the messages are still not delivered the system terminates, hence there is no possibility of non-termination, whereas in an asynchronous environment, there is no guarantee of the message delivery or no fixed time frame for the message delivery. We cannot make a decision on the maximum time limit for the message delivery; achieving consensus in such environment is much tougher than the synchronous systems. Even a single faulty node can make it impossible for the system to reach consensus in asynchronous environments, leading to impossibility result.
In a Byzantine Generals problem with one general and two lieutenants even one traitor (general or one of the lieutenants) can cause impossibility results demonstrated in Fig. 4 and Fig. 5. The FLP impossibility result in asynchronous system

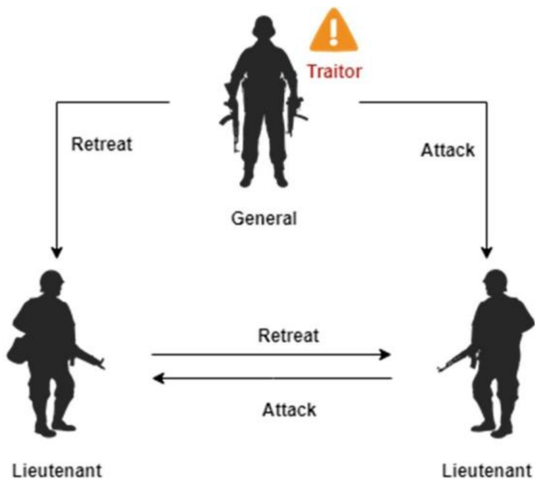

Fig. 4. General as Traitor

can be bypassed by 1. Synchrony assumption and 2. Non-Determinism. Various consensus algorithms have been designed using the synchrony and non-determinism assumptions which are explained in later sections.

## III. Types of Consensus Algorithms

With respect to the conventions existing to bypass the impossibility result, consensus algorithms can be broadly

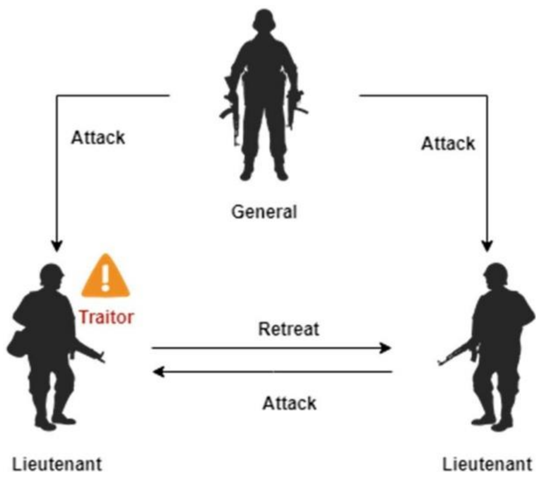

Fig. 5. Lieutenant as Traitor

classified into categories involving Synchrony assumption and Non-determinism.

### A. *Based on Network Synchrony - Traditional Consensus*

A number of algorithms exist to circumvent the FLP impossibility depending on synchrony assumption like Paxos, Raft, DLS algorithm and PBFT algorithm [8]. The older algorithms like Paxos and Raft are crash fault tolerant but not Byzantine Fault Tolerant making them an ineligible candidate for consensus on asynchronous networks like Blockchain. DLS algorithm cannot be implemented as well due to partially synchronous network requirement. Practical Byzantine Fault Tolerance (PBFT) algorithm is the one reliant on network synchrony assumption and works in asynchronous environment as well [3]. Hence let us discuss in detail about the above mentioned algorithms.

*1) Paxos:* The first real world implementation of practical and fault tolerant consensus algorithm was Paxos algorithm. It was one of the first and foremost of algorithms, widely adopted by high tech internet giants for implementing distributed systems. Let us consider a system with multiple processes proposing different values. The main objective of using consensus algorithm in such system is to choose a single value among the proposed value. If there are no values proposed by the process, then no value should be chosen. On the other hand if the system is able to choose a single value, then the processes should be able to learn the chosen value [8]. Multiple assumptions have been made for reaching consensus in those systems as follows:

- The chosen value must be selected from one of the proposed values
- Consensus algorithm must ensure only a single value is chosen from the set of proposed values
- Unless and until a value has been actually chosen the processes should not be able to learn the value

Ultimate goal is a single value must be chosen from a set of proposed values and the process should eventually be able to learn the chosen value.Three types of roles are played by actors present in this algorithm. More than one role might be played by a single process:

- Proposers: The processes who chooses new proposal numbers and sends requests to other processes within the network
- Acceptors: Acceptors are the processes who receives the proposed values from the proposers in the system
- Learners: Learners gains the knowledge about the chosen value after a majority of acceptors accepts the proposed value

Paxos algorithm has three phases implemented in achieving consensus listed below:

- Preparation phase
  - Proposer processes chooses a new proposal number ‘n’ and send it to the acceptors in the form of ‘Prepare request’
  - If the acceptors receives a prepare request with ‘n’ greater than the already accepted request, it responds with the acknowledgement stating it would not accept any proposal having number less than ‘n’ and replies with highest numbered proposal less than ‘n’, that has been already accepted
- Acceptance
  - Proposer receives the responses from multiple acceptors and aggregates the responses
  - It sends an accept request to all of them with number ‘n’ and value ‘v’, where ‘n’ is the number been sent in the prepare request and ‘v’ is the value from the responses having the highest number ‘n’
  - As soon as the acceptor receives a request it will accept it unless it has sent a response already with a number higher than ‘n’
- Learning Phase
  - If an acceptor is going to accept a request, then it will send the accepted value ‘v’ and number ‘n’ to all the learners
  - Learners after receiving the value ‘v’ from a majority of acceptors, will send the value ‘v’ to other learners
  - The learners decide on the value ‘v’ by means of the decide request

*Disadvantages*

Though Paxos was the first and foremost implementation of fault tolerant consensus, they had multiple drawbacks mentioned below:

- Important processes like election of the leader, failure detection are not defined properly
- Very difficult to understand and implement in real world

*2) Raft:* Raft is a distributed consensus algorithm as effective as the Paxos but simpler to understand. Multiple servers present in the network are involved in agreeing upon a shared state, even if there is presence of failures. The shared state is primarily a data structure sustained by replicated log. Majority of the system in the network should be up, to make sure the network is up and running. A leader is elected by the algorithm that will accept all the client requests and perform the task of replicating the log to other servers present within the network. The flow of data is always from the leader to other servers not in the opposite direction [8]. There are three important steps involved in reaching consensus through Raft algorithm:

- Leader Election – A new leader is selected once the process starts. The process repeats whenever there is a failure in the existing leader resulting in a new leader being elected
- Replication of Log – Leader should make sure the logs present in the other servers are consistent with its own copy. This is made possible by applying replication process for the replication of logs to the other servers
- Safety – Whenever a server commits an entry for particular index in the log, no other server should enter whatsoever in the same index of the log entry

Below properties of the algorithm should be true all times:

- Leader Election Safety – At any given term or duration of time only one leader should be selected
- Append-Only – A leader can make new entries or append entries on given index, though they will not be able to reverse, overwrite or delete an entry already present in the log
- Matching Logs – If two entries at a given index and term are identical between two logs, then all the entries in all the logs should be matching or identical up through the given index and the term
- Completeness – A log entry committed in the term of a given leader should be present in all the higher numbered terms following the current term
- State Machine Safety – Once a server present in the network inputs an entry at a given index of its state machine, then no other server should input a different log entry on the same index

Every server present in the network will be in one of the following states illustrated in Fig. 6:

- Follower – Followers are submissive servers; they will not send any new requests on their own, instead work on the requests from leader or candidates. If there is a scenario where a client contacts a follower, the follower will diligently redirect the request to the elected leader
- Leader – Leader is the one who is elected newly per each term and handles all the requests from clients
- Candidate – This state is used to elect a new leader; Each leader will be a candidate before being elected as a leader

All the terms in Raft algorithm are divided into arbitrary length, with each of the term starting with an election of the leader. Voting happens to elect a leader; if a candidate obtains majority then the candidate become the leader for the rest of the term. If there is no absolute majority then the term will end without a leader. The term number is incremental and each of the servers present in the network stores the term number and sends with each communication. Each server

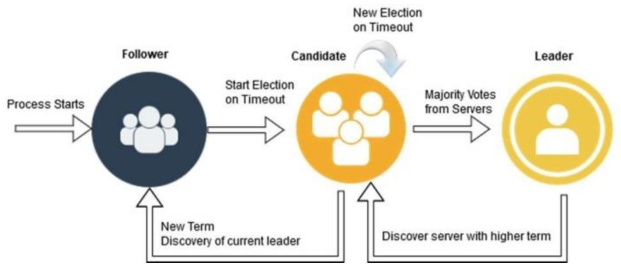
Process Starts
Follower
Start Election on Timeout
Candidate
New Election on Timeout
Majority Votes from Servers
Leader
New Term Discovery of current leader
Discover server with higher term

Fig. 6. Working of Raft Protocol

should ensure their current term is not lesser than the other servers. If there is any inconsistency noticed, it will update its current term to larger value from other server. Once the term of the leader ends, then it will revert to follower state immediately. If any of the servers receives a request with old term number in the communication, then the request is rejected.

*3) DLS Algorithm:* DLS Algorithm defined better mechanisms to achieve consensus in partially synchronous systems. This algorithm presented key developments and improvements in Byzantine Fault Tolerant Consensus. In asynchronous systems there is no definite upper bound for message delays; conversely in synchronous systems there exists an upper bound on the message delays. A partially synchronous system is an intermediate between these two conditions. There are two major assumptions in reaching consensus in partially synchronous systems, they are as follows:

- There exists a fixed bound on how long a message will take to be delivered; nonetheless they are not known earlier. Consensus should be reached notwithstanding the actual bounds
- There exists an upper bound for the message delivery, while they can be certainly beginning at some random unknown time. Regardless of when the time window starts, we should be able to reach consensus in the system

DLS Algorithms has two important properties acting as an assurance to achieve consensus in the system:

- Safety – All the correct or non-faulty nodes in the network should agree on the same value [3]. They must agree on the total order of the transaction logs even in case of failures and presence of dishonest nodes. The system will become inconsistent with two or more valid logs if the safety property is violated.
- Liveness – This property ensures every correct or non-faulty node in the network agrees on some output value eventually. It will make sure the Blockchain is continuously growing and not stalled.

DLS algorithm divides the working of consensus mechanism into rounds with two phases: ‘Trying’ and ‘lock-release’ phase. Let us assume there are ‘N’ nodes in the partially synchronous network [9]. Following are the steps involved in reaching consensus in the above mentioned system:

- In each round there exists a proposer who will accept the value communicated by each process; the processes communicate the value believed to be correct by them
- If at least N-x process has communicated the same value, then the proposer node proposes it
- Each of the nodes present in the network should lock on the proposed value and pass on to others in the network, whenever they receive the proposed value from the proposer
- If there are x+1 messages from the nodes to the proposer notifying they have locked on some value, the locked value is committed as the final value

*Disadvantages*

- We assume timeouts in this algorithm; In case of failure of the synchrony assumption this will lead to two valid transaction logs resulting in an inconsistent the system state. There is nothing useful in having an inconsistent and corrupt system
- If the nodes are not deciding on some output value, then there is no way consensus can be reached, the system just halts
- Even if we make synchrony assumptions for termination (timeouts), in case of failure of the assumption, the system will come to a halt

*4) Practical Byzantine Fault Tolerance Algorithm:* PBFT model is a fault tolerance algorithm primarily focusing on practical state machine replication overcoming Byzantine failures with independent node failures (dishonest nodes) propagating incorrect or faulty messages [10]. This algorithm will work on asynchronous systems with overall optimized performance and very little increase in latency. The system is modelled with one node assumed as a leader and all other nodes considered backups. All the nodes present in the system communicate with each other and the design is likely to bring all the honest nodes to an agreement through a majority [11]. There is a slight overhead in the communication of message since each message has to prove its origin and verify there is no alteration happened in the messages during transmission. PBFT works under the assumption, for a network size of ‘N’, the number of faulty nodes ‘f’ should not exceed (N-1)/3 for the network to be Byzantine Fault tolerant. It can be stated in other way $N \geqq 3f+1$ in a network guarantees to be a BFT system. PBFT is modelled based on three sub-protocols:

- Normal Operation Protocol - It is a session to session protocol with various phases like Client operation request, pre-prepare phase, prepare phase, commit phase and reply to client [12] [13]. A request from the client to primary is sent in the ‘client operation request’ phase. In ‘pre-prepare’ phase, the message is passed on from primary to all the backups present in the system. Each backup will forward the message to every replica in ‘prepare’ phase. ‘2f+1’ messages with same values is required for pre-prepare and prepare phase. A ‘commit’ phase occurs when each replica send the message to all the replicas. At last a reply is sent to the client if '2f+1' commit messages are available [3].

- Checkpoint Protocol - A stable checkpoint is maintained, which the system accepts and safely discarding the old transactions in the log. The most recently executed client request is recorded with the sequence number 'n'. Updated checkpoint is broadcasted as a 'checkpoint' message. All replicas updates the last checkpoint broadcasted if they receive '2f+1' checkpoint messages with same value [3]. All the nodes discard messages prior to the latest checkpoint.
- View-Change Protocol - A view change happens when there is a fault suspected in the primary node [3]. Whenever any of the backup is in pre-prepare phase of the normal operation protocol for a long time and stops receiving messages, the status is updated to 'view-change'. A view-change request is send to all the replicas to change the view state to 'v+1'. The primary node will broadcast a 'new-view' message if it receives 2f view change messages. The log will be updated and the primary node will proceed to normal operation. The replica nodes after receiving 'new-view' message, validate it, update its state and proceed to normal operation protocol.

### B. Based on Non-Determinism

*1) Nakamoto Consensus (Proof of Work):* Nakamoto Consensus can be defined as a probabilistic approach non-deterministic in nature. In this model every node need not agree on the same value, instead agree on the probability of the value to be correct. This consensus mechanism is proved to be having the following characteristics through its famous implementation 'Bitcoin':

- Byzantine Fault Tolerance – There is no leader electionin this process, instead decision is dependent on the nodes solving the computational puzzle faster and getting a chance to propose its value
- Incentive mechanism – When a node solves the puzzle and proposes a value agreed by other nodes, there is an economic incentive rewarded to the winner node
- Sybil Resistance – Rather than using the conventional PKI authentication scheme or any other scheme, PoW (Proof-of-Work) mechanism is involved in the consensus process making sure the process is Sybil resistant
- Gossip Protocol – The peer to peer gossip protocol ensures messages are routed to all the nodes in the asynchronous network, assuming each node is connected to a subsection of nodes only

*Protocol Design* Nakamoto consensus is designed on the basis of the principle, all the nodes present in the Blockchain network runs an identical protocol and manages its own copy of the decentralized ledger by itself. This protocol depends on the fact, majority of the nodes present in the network are honest or non-faulty nodes. Important rules applied in the design of the protocol are mentioned below:

- Message Broadcasting Rule – All the transactions either they are generated locally or received from other peer nodes within the network should be broadcasted to the identified peers in a timely manner without causing any delay. The same conditions applies for broadcasting the blocks as well
- Validity – All the transactions and blocks received by each node should be validated before broadcasting. Once all the transactions in the blocks are validated along with the block as a whole, it should be appended to the Blockchain. Invalid transactions or blocks will be discarded in this process
- Longest chain Rule – In this consensus mechanism, any longest chain present in the network will be considered as the valid chain. Any honest or non-faulty node will build on top of the longest chain by appending the next valid block. Decision on where to append the next block will be reliant on the total amount of computation effort put on the longest chain
- Mining Process – Proof of Work is the backbone of Nakamoto consensus taking care of valid block generation process. Each node trying to propose a block should be finding a nonce (Number used once) to be inserted into the block header. The difficulty to find the nonce will be adjusted periodically to certify the average block generation speed remains constant within the network

Proof of Work consensus displayed in Fig. 7 has been successfully verified by the implementation of the protocol in the 'Bitcoin' Blockchain.

*Advantages*

- Nakamoto consensus is Byzantine Fault Tolerant though most of the traditional consensuses are not
- Any number of nodes can join or leave the network at any point of time assuring open participation
- The nodes in the network need not know all the participants in the network, still they will have a finite set of known peers
- There is no communication overhead involved unlike the traditional mechanisms
- Leader election process is not required in Nakamoto consensus

*Disadvantages*

- Nodes mining the next block try with brute computing force. Hence Higher energy and computational power consumption for calculating the hash
- Average size of block is limited to 1 MB and number of transactions processed per block is very less ( 7 transactions per second). Other established networks like VISA, PayPal process much higher transactions within the same duration ( 10000 transactions per second in VISA).

### C. Other Consensus Algorithms

*1) Proof of Stake:* A new consensus algorithm was designed to overcome higher use of energy in Proof of Work

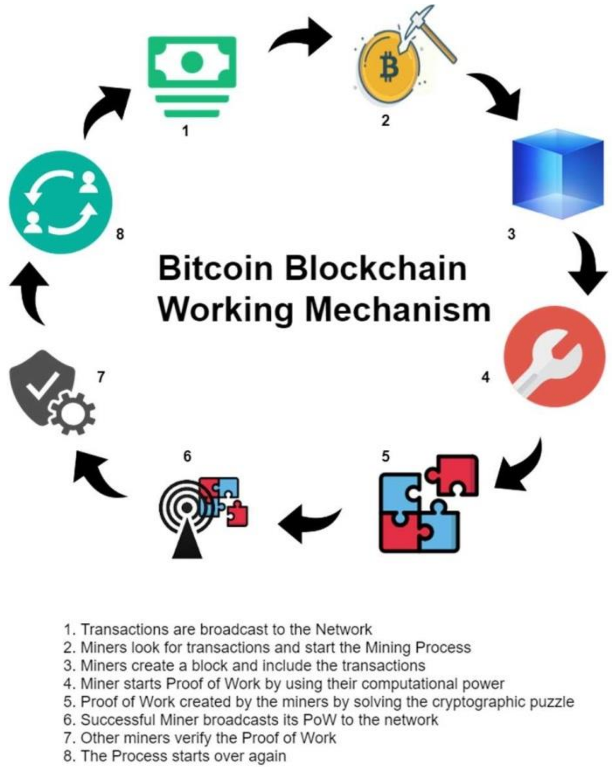
Bitcoin Blockchain Working Mechanism
1
2
3
4
5
6
7
8
1. Transactions are broadcast to the Network
2. Miners look for transactions and start the Mining Process
3. Miners create a block and include the transactions
4. Miner starts Proof of Work by using their computational power
5. Proof of Work created by the miners by solving the cryptographic puzzle
6. Successful Miner broadcasts its PoW to the network
7. Other miners verify the Proof of Work
8. The Process starts over again

Fig. 7. Working of Bitcoin PoW

protocol to ensure mining activity is conducted in a more cultured manner without wasting energy or computational power. A group of validators will bet a specific amount as a security deposit for participate in the block generation process. They will buy cryptocurrencies instead of investing in buying equipment for rigorous computational activity. Each one who wants to participate in the block mining race will deposit some cryptocurrencies as a stake in the network [14]. The higher the stake is, higher is the possibility of the node to become a validator. Based on a random process the validator is selected to generate a block. The validator who generates a valid block will get incentives for their work. If the block generated is not included in the chain then the validator will be penalized and lose her/his stake. In Proof of Work mechanism, penalty for nodes generating invalid block is only in terms of wasted computational power and resource, whereas in PoS, nodes lose their stakes if invalid blocks generated or fraudulent behavior is exhibited. If all the correct or honest nodes in the network follows the protocol and own greater than 50% stake in the network, then the possibility of an already generated block to be revoked from the network drops exponentially [15].

*Disadvantages*

- A investor who is wealthy enough can invest some amount in the cryptocurrency, bet them as stake, get their rewards and reinvest making them richer
- Nothing-at-Stake Problem – A validator node can bet some value on top all the branches existing in the network without any additional cost. These kinds of bets are called 'double bets' where there is no penalty implications
- Even if a penalty scheme is incorporated in the algorithm, it would not be effective in case if more than 50% of the network stake is owned by a single validator

2) *Proof of Elapsed Time:* Proof of Elapsed Time (PoET) is consensus scheme working centered on a back-off mechanism that is purely random. Each validator node in the network waits for a time, whose length is random and then backs off after the time is elapsed. Once a node has finished its back-off, it will become the validator. It is a trusted back-off method for the nodes where each and every validator should be verified and trusted completely. The random back-off is achieved by the use of microprocessors precisely designed for this purpose. A new validator node while joining the network will get the back-off program from the peer nodes within the same network. The newly joined validator will send an attestation report to the host network, to ensure the required authentic program for back-off has been loaded successfully. Once the validator fulfills the back-off requirement for a random amount of time, it will generate a new block with the set all unprocessed transactions it has heard of. A certificate is generated for completing the trusted back-off mechanism and sent along with the block. The back-off mechanism proposed in the Proof of Elapsed Time algorithm can tolerate any number of faulty validators present in the network. A validator who is rich with a lot of resource can invest in multiple instances to cut short the expected duration of back-off. *Disadvantages*

- The microprocessor used in this mechanism is manufactured by Intel, hence the security of the system is bounded by the security of the hardware and reliability of their servers
- Dependency on a Trusted Execution Environment(TEE), in this case Intel's microprocessor
- This model greatly contradict the idea of decentralization proposed in Blockchain by using a specific hardware making the system centralized and dependent

3) *Ripple Protocol:* Ripple consensus protocol is being operated by an organization Ripple.Inc, facilitating exchange of currencies and settlements. It is a Real-Time Gross Settlement network regulated by known set of validators. These set of validators generally includes companies and institutions running Ripple servers and programs on their systems. The Ripple programs running in their infrastructure enable the company to accept transaction processing request from different clients and process them. Each client participating in the network submits their transaction requests to nominated validators existing within the network. While processing each transaction, a decision is taken centered on the Ripple consensus running in the validators in a distributed manner. Ripple consensus lets each node 'n' in the network to maintain a unique list of nodes (Unique Node List - UNL). The node 'n' along with the UNL forms a sub network within the network which needs to be trusted by 'n' partially. The protocol is designed so as,

- A list or set of entities with valid transactions is prepared as an initial step. All the new transactions along with the old transactions from the previous cycles constitutes to the set of valid transactions

- The candidate set prepared by the node ‘n’ will be combined with the candidate set prepared by the peers in the UNL list
- All the nodes present in the sub network will vote a ‘Yes’ or ‘No’ liable to the validity of transactions available in the combined set
- Each transaction will receive a vote from the UNL nodes within the candidate set. The transactions receiving votes count less than minimum threshold value will be discarded. The discarded transactions will be considered for the next cycle of voting consensus
- The above steps will be repeated in upcoming cycles until the final round is reached where the threshold for minimum number of votes will be increased to 80%. After the final round all the nodes append the remaining valid transactions and end the cycles

The transactions are finalized and updated into the list only if 80% of the nodes from the UNL approves. This Ripple consensus protocol is Byzantine Fault Tolerant making it advantageous to be implemented in the network. There is a special scheme for the authentication of the validators to ensure the true identity of nodes in the UNL is known to others present within the same sub network.

*Disadvantages*

- Any node in the network will be communicating with the peer nodes within the sub network (UNL) only. Each individual node in the network may have disconnected or disjoint set of UNLs resulting in Network Partitioning
- Different set of nodes participating in two unique UNLs might agree on different set of transactions and update the ledger in parallel in a conflicting manner. To ensure the above scenario does not happen, Ripple consensus should make sure at least 25% of nodes must be shared between two UNL groups
- The protocol is designed and maintained by Ripple Inc., supposedly under their direct supervision. Any changes in the protocol can be done by the company directly, hence the power is in a single hands rather than being decentralized. The above reason makes it unrealistic to be used in Public Blockchains

## IV. Possible Attacks

### *A. 51% Attack*

When a miner node controls multiple nodes with more computational resources than the rest of the network, i.e. at least 51% or a mining pool has 51% or more of the total network computation resources, then there is feasibility for these miners or pools to undermine the network with a peculiar type of attack known as 51% attack [16]. The miners or group of miners with 51% resource or more will be having the advantage of performing successful mining and claiming reward. The Proof of Work protocol famous for its implementation in ‘Bitcoin’ Blockchain is vulnerable to this kind of attack.

### *B. Double Spending*

The ability of an individual to use the same coin or funds twice is known as double spending. The possibility of a 51% attack allows a miner to successfully double spend the amount by reversing the already committed transactions. We all know the definition, Blockchain are irreversible, immutable transactions, yet the ‘longest chain rule’, makes reversing and altering the transactions possible. A miner with 51% or more computational resource can create an alternate chain and add blocks on top of it. During a point of time, it will become the longest chain and the actual chain will be dropped making the attack successful. Say for example, User ‘u’ orders a high end laptop from a merchant by creating a transaction. The merchant waits for certain duration to confirm the transaction and ships the product. The user knows the product has been shipped and starts creating an alternate chain now. Once the alternate chain becomes the longest chain the actual chain containing the transaction to the merchant will be dropped. The transaction is reversed and the merchant loses the money. User ‘u’ can now happily use the money for some other transactions.

### *C. Sybil Attack*

When a single person in the network creates multiple identities, they can control multiple IDs in turn controlling multiple nodes. These IDs are used in an attempt to control the peer nodes in the network, by undermining the network using the multiple IDs at the same time. Also they can influence the decentralized network through additional voting power they acquire with multiple IDs. The main aim of this type of attack is to gain the majority of the influence in the network and carry out malicious activities disturbing the integrity of the network. Sybil attacks are difficult to detect and prevent as they involve false identities and hidden motives.

## V. Conclusion

Consensus algorithms are the soul of Blockchain Network. Consensus algorithms are for the systems in a distributed environment to reach a collective agreement. There cannot be any distributed or decentralized system without a consensus mechanism implemented to reach an agreement. The nodes within a network may not trust each other but they have to comply with the consensus algorithms to decide on a common goal. There may not be a single consensus algorithm common for all Blockchain implementations; however we need some consensus algorithm for the network to be successful and running. Blockchain may build blocks from innumerable transactions forming a database but they would not be decentralized in nature without the consensus algorithms. Blockchain is predominantly a framework working on decentralization concept, whereas consensus mechanism remains the soul of the network. There are numerous consensus algorithms described in this chapter based on various assumptions. Much more research is going on in this area and there are developments proposed every day.

## REFERENCES

[1]D. Puthal, N. Malik, S. P. Mohanty, E. Kougianos, and G. Das, "Everything you wanted to know about the blockchain: Its promise, components, processes, and problems," *IEEE Consumer Electronics Magazine*, vol. 7, no. 4, pp. 6–14, 2018.

[2]V. Gramoli, "From blockchain consensus back to byzantine consensus," *Future Generation Computer Systems*, vol. 107, pp. 760–769, 2020.

[3]S. S. Shetty, C. A. Kamhoua, and L. L. Njilla, *Blockchain for Distributed Systems Security*. John Wiley & Sons, 2019.

[4]Y. Zhao and F. B. Bastani, "A self-adjusting algorithm for byzantine agreement," *Distributed Computing*, vol. 5, no. 4, pp. 219–226, 1992. [5]L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," in *Concurrency: the Works of Leslie Lamport*, 2019, pp. 203–226.

[6]D. Dolev *et al.*, "The byzantine generals strike again," *J. Algorithms*, vol. 3, no. 1, pp. 14–30, 1982.

[7]M. J. Fischer, N. A. Lynch, and M. S. Paterson, "Impossibility of distributed consensus with one faulty process," *Journal of the ACM (JACM)*, vol. 32, no. 2, pp. 374–382, 1985.

[8]D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *2014 {USENIX} Annual Technical Conference ({USENIX}{ATC} 14)*, 2014, pp. 305–319.

[9]C. Dwork, N. Lynch, and L. Stockmeyer, "Consensus in the presence of partial synchrony," *Journal of the ACM (JACM)*, vol. 35, no. 2, pp. 288–323, 1988.

[10]K. Zheng, Y. Liu, C. Dai, Y. Duan, and X. Huang, "Model checking pbft consensus mechanism in healthcare blockchain network," in *2018 9th International Conference on Information Technology in Medicine and Education (ITME)*. IEEE, 2018, pp. 877–881.

[11]M. Castro, B. Liskov *et al.*, "Practical byzantine fault tolerance," in *OSDI*, vol. 99, no. 1999, 1999, pp. 173–186.

[12]Y. Amir, B. Coan, J. Kirsch, and J. Lane, "Prime: Byzantine replication under attack," *IEEE transactions on dependable and secure computing*, vol. 8, no. 4, pp. 564–577, 2010.

[13]J. Liu, W. Li, G. O. Karame, and N. Asokan, "Scalable byzantine consensus via hardware-assisted secret sharing," *IEEE Transactions on Computers*, vol. 68, no. 1, pp. 139–151, 2018.

[14]H. Guo, H. Zheng, K. Xu, X. Kong, J. Liu, F. Liu, and K. Gai, "An improved consensus mechanism for blockchain," in *International Conference on Smart Blockchain*. Springer, 2018, pp. 129–138.

[15]J. Kang, Z. Xiong, D. Niyato, P. Wang, D. Ye, and D. I. Kim, "Incentivizing consensus propagation in proof-of-stake based consortium blockchain networks," *IEEE Wireless Communications Letters*, vol. 8, no. 1, pp. 157–160, 2018.

[16] P. Chinnasamy, P. Deepalakshmi, V. Praveena, K. Rajakumari, andP. Hamsagayathri, "Blockchain technology: A step towards sustainabledevelopment." International Journal of Innovative Technology and Exploring Engineering (IJITEE), Volume-9 Issue-2S2.