UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Tècnica Superior d'Enginyeria
de Telecomunicació de Barcelona

Study of consensus protocols and improvement of the Delegated Byzantine Fault Tolerance (DBFT) algorithm

A Master's Thesis Submitted to the Faculty of the Escola Tcnica d'Enginyeria de Telecomunicaci de Barcelona Universitat Politcnica de Catalunya by

GAVRIEL CHRISTOFI

In partial fulfilment of the requirements for the degree of MASTER IN TELECOMMUNICATIONS ENGINEERING

Advisor: FERNANDEZ MUNOZ, MARCEL and ESPARZA MARTIN, OSCAR

Barcelona, October 2019

Title of the thesis: Study of consensus protocols and improvement of the Delegated Byzantine Fault Tolerance (DBFT) algorithm.

Author: GAVRIEL CHRISTOFI

Advisor: FERNANDEZ MUNOZ, MARCEL and ESPARZA MARTIN, OSCAR

Abstract

Nowadays, blockchain is one of the most popular and innovate technologies over the world. Although this technology appears for first time in the Bitcoin cryptocurrency, in recent years, a lot of researchers and industries from different fields such as banking, financial, supply chain management, etc. have given more involved than ever before. The blockchain is implemented in decentralized and distributed ledgers in peer-to-peer (p2p) networks where non-trusting peers can implement digital asset transactions without the need of central authority. Then, other peers in the network, according specific rules determined by the network, validate these transactions, insert them in the block and append the block in the chain (ledger). The key contribution for the proper operation of the blockchain is the consensus protocols. Through these protocols, all the peers in the network or the majority of them, they have to reach an agreement for a specific block in order to insert it in the chain based on different blockchains rules.

In this master thesis, we will analyze in depth the general architecture of the blockchain and various consensus protocols that implemented in different blockchains in order to be able to improve the Delegated Byzantine Fault Tolerance (DBFT) consensus algorithm, which is used in the NEO blockchain technology. Finally, a development of a reputation mechanism is needed based on the improvement of the DBFT algorithm in order to measure the reputation of the peers for a specific day.

Revision history and approval record

Revision	Date	Purpose
0	15/3/2019	Creation
1	8/9/2019	Revision
2	11/10/2019	Correction
3	17/10/2019	Revision

Written by:	
Date	17/10/2019
Name	Gavriel Christofi
Position	Project Author

Reviewed and approved by:	
Date	
Name	
Position	Project Supervisor

Reviewed and approved by:	
Date	
Name	
Position	Project Supervisor

Contents

List of Figures

List of Tables

1 Introduction

We live in an era where the digital information plays a significant role in our life. Except for the internet, which appeared the last decades changing the life of people worldwide, another technology sparked the interest of many organizations, governments and academic to invest on this and is known as blockchain technology. Blockchain became known through the bitcoin and Satoshi Nakamoto which published on 2008 the first paper introducing the term chain of blocks describing an electronic cash system between peer to peer[14]. Furthermore, before the founder of bitcoin Satoshi Nakamoto, some other researchers referred to blockchain in the 90s but nowadays, the general structure and architecture of blockchains is mainly based on the bitcoin. Although the blockchain was used in the bitcoin cryptocurrency, at this moment it is used in different fields such as banking, digital identities, governments, in markets and so on.

Blockchain is referred to a chain of blocks where chain is the ledger and the block includes all the transactions that happen in the ledger. In addition, blockchain is a distributed or decentralized ledger based on peer-to-peer network where it can record all the digital asset transactions that happen to the network. Every peer in the network can implement different types of transactions directly with other peers without the need of third parties. This can be achieved through the hash function and the digital signatures. When a peer wants to implement a transaction, firstly creates the transaction, signs it with its own private key and then sends this transaction to the network. Then, other peers in the network according to the different rules verify the integrity of this transaction by checking the signature of the sender using its own public key and then insert this transaction in the block. The block expect from the multiple transactions that happened to the network, it consists also of the hash of the previous block that inserted in the chain. Through this technique, the transactions that are included in a specific block remain unharmed and the blocks immutable.

The security and the trust of the network can be achieved also through the distributed and decentralized ledger, the consensus protocols as the smart contract. In the decentralized and distributed networks, a number of peers records all the transactions that happen in the ledger and the rest of the peers in the network can get the copy of the ledger by requesting. As a result is impossible for an attacker to change the ledger on the grounds that it has to change it from all the peers that recorded it. In addition, the consensus protocols is a set of rules determined by the different kinds of blockchain and the purpose of them it is the agreement by all the peer or the majority of them for a specific block which includes all the transactions that happened in the network in order to append in the chain. The smart contracts is a set of computer programs that include codes and agreements in the form of business logic, between two or more parties without the need of third parties and it is executed in the top of the blockchain. When specific criteria are met, then the smart contracts are executed automatically in order to obtain the output and this result is stored and replicated in all participants in the network to provide immutability. A further explanation of the different kinds of blockchain and how these blockchains work will be explained in the next chapters of this thesis.

1.1 Purpose of thesis

This thesis has been implemented in cooperation with my colleague Aspasia Zoi. The aim of this thesis is the improvement of the Delegated Byzantine Fault Tolerance (DBFT) consensus algorithm, which is used in the NEO blockchain technology as the development of a reputation mechanism in order to measure the reputation of the peers for a specific day.

In order to improve this algorithm, a further and an extend analysis of different kinds of consensus protocols that are implemented in various blockchain technologies should be examined. Therefore, in cooperation with Aspasia Zoi, we examined these consensus protocols and after that I proceeded to the improvement of the DBFT algorithm.

1.2 Thesis structure

This thesis is divided in two parts. The first parts is consisted of chapter 2 and 3 which is the state of the art and it is implemented in cooperation with Aspasia Zoi and the second part is the chapter 3 and 4 where is the improvement of the DBFT algorithm and the conclusion. More specific:

- The first chapter is a brief introduction of what is the blockchain as the objective and the structure of this thesis.
- The second chapter is implemented with Aspasia Zoi and it includes an extensive explanation of the architecture of blockchain as the different types of networks and blockchains.
- The third chapter also is implemented with Aspasia Zoi and describes what is the consensus algorithm, the byzantine general problem and a deep explanation of the functions of different consensus algorithms where were implemented in various blockchain technologies.
- In the fourth chapter, the improvement of the DBFT algorithm is analyzed as the development of a reputation mechanism.
- The fifth chapter presents the conclusions as the future work.

2 Blockchain

2.1 Types of networks

When Satoshi Nakamoto [14] referred to the blockchain and the decentralised systems, a new era of different types of networks was born. Until 2008, the most typical type of network was the centralised network in which all the decisions were taken and the procedures in the system were controlled by a central authority. In order to implement a financial transaction between client A and B, a client A had to contact a bank to make the desired transaction by waiting some days for the transaction process paying some fees.

Once the blockchain was invented, users have more active role on the network as they can make transactions directly without the need of third parties or they can notice which actions are taking place in the network. This can be achieved through distributed and decentralized networks. hese three different types of network have many usages but in this thesis only the use of the blockchain is going to be analyzed. However, a trustful dynamic network requires some specific rules that users should follow. The transaction should be immutable. This can be achieved through consensus algorithms, which are not necessary in centralized networks.

2.1.1 Centralized

In this type of network, all the peers are connected to a central authority. For example, all the clients communicate through a single server. The central authority is responsible for determining its own rules that the rest of the network should conform to. Users have to send a request to the central authority in order to implement a transaction and after that they have to wait for the response to proceed.

The main advantages of the centralized network is that the decision making is taken in a fast manner as only the central authority is responsible for this action. In addition, the participation of users in the network does not require expensive and specialized equipment as the only duty of users is to execute transactions. Furthermore, in the centralized network, the information of the users such as identity and address is kept secure as any operation is implemented through the central server and not directly between the users.

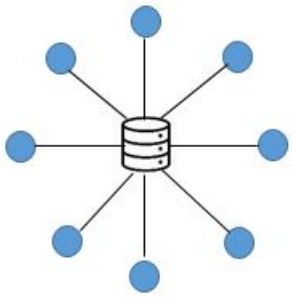

Figure 2.1: Centralized network

However, the centralized network suffers from a single point of failure. In case of any failure of the central server, the network goes down and then users cannot implement any request. In addition, another drawback of this type of network is lack of scalability: the case of high traffic could lead to the bottleneck where transactions would be implemented in a slow manner. Based on the high traffic issue, the centralized network is vulnerable to Denial of Service (DoS) and Distributed Denial of Service attack (DDoS) attacks where the attackers send a massive amount of data to the central authority in order to disrupt it to implement the proper procedures.

2.1.2 Decentralized

This network is the evolution of centralized network as eliminates some disadvantages, mainly the single point of failure. The network is controlled by multiple central nodes where users are connected to them. In addition, all the information is stored in all the central nodes unlike the centralized network where the data is stored only in a central authority. Furthermore, the peers can communicate directly each other without the need of third parties presence.

In case of failure of a central node, the network will keep operating as there are other central nodes to which users can connect in order to implement their transactions. In decentralized networks, the information is stored in multiple users and so it is difficult for any attacker to change this information in all the users. Furthermore the identities and the addresses of the users are difficult to be detected from a malicious node. It is impossible to track a specific user as the data is conveyed through different central nodes. Another advantage of this type of network is the absence of bottleneck as multiple nodes are responsible for different implementations.

On the other hand, users need to buy expensive equipment in order to act as central nodes for the validation of the transactions. Furthermore, central nodes can be selfish and can take decisions that act on their own interests and not on the whole network. Finally, the bigger the network, the more difficult could be to reach an agreement and, as a result, the speed for all the procedures in the network slow down.

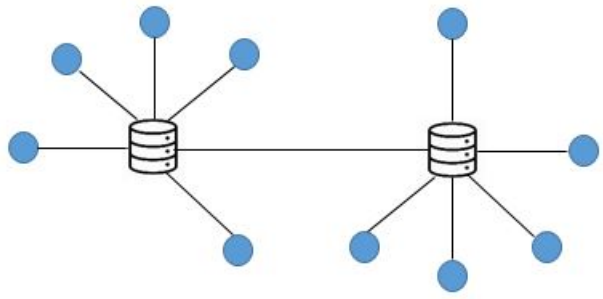

Figure 2.2: Decentralized network

2.1.3 Distributed

The distributed network consists of nodes that are all connected to each other and the communication is accomplished without the presence of a central node. All the participants

in the network can have an active role in the network. For example, one percentage of the nodes in the network are responsible to store all the data that happens in the network and the rest of the peers can get a copy of the recorded data by requesting. Another percentage is responsible for controlling if everyone conforms to the network rules. If a part of the network collapses, the network will continue to operate normally, since communication can be done through the rest of the nodes.

The main advantage of a distributed network is the low latency due to the absence of central authorities and the geographically spread of the network. On the other hand, the main disadvantage of this network is the fact that is difficult to reach consensus because it is highly dependent on the size of the network. More peers in network, more complicated is.

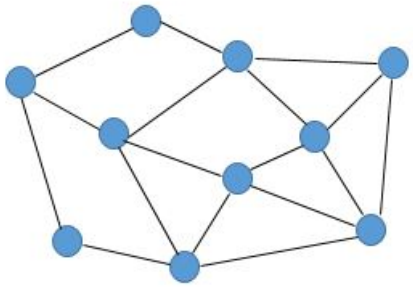

Figure 2.3: Distributed network

2.2 Blockchain

The term blockchain refers to a chain of blocks which was firstly described in 1991 by researchers, Stuart Haber and W. Scott Stornetta in order to timestamp a Digital document without being tampered [11]. In addition, in 1992, those researchers in collaboration with Dave Bayer improved the efficiency of this procedure by introducing the Merkle root, where the integrity of the data is checked in a efficient way [25]. In 2008, Satoshi Nakamoto published the first paper introducing the term chain of blocks describing an electronic cash system between peer to peer [14]. Through this paper, a new era of cryptocurrency was born and especially that of BITCOIN [14]. With the passage of the years, the term chain of blocks changed to what we call today blockchain.

Blockchain is a decentralized and distributed ledger across the network that can maintain and record transactions between peers unchangeably and securely. Each participant in the network can communicate directly each other without the need of the presence of the central authority. The P2P network determine some certain rules where all the peers who belong to this network should conform to. When the transactions are implemented between peers, every participant has to check the validity of these transactions according to the specific rules and then, these transactions are stored into a block. The blocks are linked to each other making a chain. Each block contains the hash of the previous block maintaining the transactions unharmed and making in this way the block immutable. So, the chain becomes more secure as the insertion of a new block verifies all the previous blocks. For example, if an attacker wishes to tamper the contents of the block, not only should she modify the data in that block but also the data of all the subsequent blocks [27]. The modification of the block is difficult to be implemented and it will be explained

in more detail in the next chapter.

According to the operation of the blockchain, this technology makes use of cryptography and hash functions in order to protect and keep the identity of each user unique in the network. Each peer in the network has a pair of keys, the public and the private key. When a peer wants to make a transaction should sign his own transaction with his private key. Then the rest of the participants have to verify this transaction by checking the signature of the sender using its public key [9]. This procedure eliminates the need of the central authority to control the whole network as peers can implement transactions through their keys. Furthermore, another reason of no need of centralized authority is that every participant in the network has a replica of the updated ledger which includes all the blocks of the chain with the corresponding transactions.

All the peers in the P2P network have the right to join or to leave the network at any time they wish. The issue of the public blockchain is that the participants do not know each other, and as a result peers cannot know whom they can trust in the network. This problem can be prevented with the consensus algorithms which require all the peers or the majority of them to agree to the network rules as to reach the finality of the blocks that are stored in the ledger.

Except for the distributed ledgers and the consensus, blockchain can assure, also, trust and security through the smart contracts. It is an application that was introduced in the second generation of blockchain and especially in Ethereum. Smart contract is executed on top of the blockchain, and it is a set of computer programs that include codes and agreements in the form of business logic, between two or more parties without the need of third parties. When specific criteria are met, then the smart contracts are executed automatically in order to obtain the output and this result is stored and replicated in all participants in the network to provide immutability.

Despite of all these advantages, there are some limitations in the blockchain, which does not make it as trustable as seems. Some of the most typical limitations of blockchain are:

- Scalability: due to the consensus mechanisms, each peer in the network has to validate the transaction as a result the bigger the network is, slower the performance of processing transaction will be.
- Anonymity and data privacy: although in the case of public ledgers, where the peers are anonymous, when the transaction is occurred it is recorded in the blocks where each participant of the network can know about this transaction. Through these records, anyone has the possibility to identify other users as to learn private information of these users.
- Security: there is a possibility of a 51% attack, where if more than a half number of the peers lies, these lies will become the truth and then can persuade the rest participants of the blockchain. As a result, it would be able to have the control of the blockchain in order to change or remove the transactions.

2.3 Types of Blockchain

The blockchain can be implemented in three different categories, the public or permissionless, the private or permissioned and the consortium or Federated blockchain. Each type of blockchain has different characteristics and when the producers or businesses want to create a new blockchain, they can choose the specific type according to their needs in attempts to implement their own services. Some of these characteristics are the number of users that can take part in the network as their rights of participation to the verification of the transactions or the insertion of the blocks in chain etc.

2.3.1 Public Blockchain

In the permissionless blockchain everyone who wishes to join the network can take easily part in that and has, also, the right to read and write to the ledger. When a user joins the network, he is updated by all the information existing in the network and he has the right to make and validate transactions as to insert the block in the ledger [6]. Furthermore, due to the anonymity of peers in the public network, the users do not trust each other and for this reason, the network uses an incentive mechanism making the users to participate in the network. When users verify and validate transactions and blocks, the network gives them rewards, in order to encourage more and more participants to join that.

The public networks usually are more secure than the other types of blockchain as there is no presence of central authority and peers can keep their anonymity. In addition, in case that some nodes fail, this does not affect the normal operation of the network as each node in the network is fully aware of the transaction in the ledger. Furthermore, the security can be achieved through the consensus algorithms such as proof of work (PoW) and proof of stake (PoS) [27]. Through these algorithms, users have to prove that they have spent an amount of energy (PoW) or hold an amount of money (PoS) in order to be active members in the consensus process. Bitcoin and Ethereum are examples of this type of the blockchain.

One of the main limitations of public blockchain is that it needs a huge amount of resources like computational power and an expensive equipment in case of proof of work in order to ensure consensus in the network updating all the nodes. According to the previous case usually the transactions fees in the permissionless network tend to be high. In addition, each node has to possess a huge space of storage in order to record the transactions. For example, in Bitcoin, the size of the blockchain file is at about 217.8 GB so far [3]. The verification of transactions as the validation of the blocks is too slow on the grounds that each one has to be validated by the whole network. Finally, public networks can suffer from what is known 51% attack where a group of miners manipulate the majority of computing power on the network and thus they are able to choose what transactions can be valid regarding to their own perspective.

2.3.2 Private Blockchain

Unlike the public blockchain where everyone can join the network, in private network, the network administrators give the permission to a specific number of users to participate in the network. This kind of network consists of a small group of participants who are known each other. The network usually does not require any consensus algorithm due to

the fact that all nodes are trusted. In case of malicious behavior, the peers become aware of this situation and they can solve the problem following the certain rules where network has established. In this permissioned network, every peer has different participation rights than the permissionless network where all the participants have the same rights. In this case, a small number of participants is responsible for the validation of the transactions and the insertion of the blocks into the chain. In addition, the ledger is not necessary to be replicated to the entire network. For example, a business in order to make an agreement, can communicate only with peers who are associated with this agreement and once the deal is carried out, then the whole network can be informed with this agreement if the business wishes.

Private networks present better performance than the public networks as the transactions are stored and validated only by a specific number of peers and not by the whole network. Furthermore, the cost of the transaction fees is cheaper as there is no need for huge computational power to reach finality. As described previously, when a peer behaves badly, the network can detect immediately this abnormal behavior, it fixes it and uses consensus protocols and it reaches finality much quicker unlike public network where achieving consensus could take time.

On the other hand permissioned blockchain have security issues. Due to the reason that the network consists of a limited number of peers, it gets easier for the attacker to take control of the entire network. In addition this type of network tends to be centralized as only some peers are responsible for the normal operation of the network. Finally, the participants of the network are controlled from the business or the organization which are the owners of the network. This means that as the information is not published in the whole network, they can control or modify the transactions in their own perspectives. Hyperledger Fabric and R3 Corda are some platforms that uses private networks.

2.3.3 Consortium Blockchain

The consortium blockchain is permissioned and semi decentralized due to the small number of the participants and the hierarchy of the peers. Instead of public network where everyone can have access and participation for the consensus to the network, this type of blockchain relies on the predetermination of selected peers for the validation of the transaction and the insertion of the block in the ledger. For instance, if a network consist of 30 members, only the 2/3 of the peers (20) have to sign in one block in order to be valid. Furthermore, in this blockchain each participant knows every peer in the network having different rights from each other like in permissioned network.

Consortium blockchains, more or less, present the same advantages and disadvantages as the permissioned blockchains. It is not based on PoW like in case of public, but it is adapted from Practical Byzantine Fault Tolerance (PBFT), Proof of Activity (PoA) and other consensus algorithms as the peers trust each other. In addition, MultiChain and Hyperledger platforms are highly based on this type of networks.

The table 2.1 summarizes the main characteristics of these three different types of blockchain.

	Public	Consortium	Private
Type	Permissionless	Permissioned	Permissioned
Read/write	Everyone	Specific number of peers	Only by administrator
Access	Open to everyone, Anonymity	Permissioned, No anonymity	Permissioned, No anonymity
Power consumption	High	Low	Low
Transaction speed	Low	High	High
Type of network	No centralized	Partial centralized	Tend to be centralized
Consensus algorithms	PoW, PoS	PBFT, PoA	Hyperledger Fabric, R3 Corda

Table 2.1: Summary of the three types of blockchain

2.4 Architecture of Blockchain

The main purpose of blockchain is to eliminate the central authority. In order to achieve this, blockchain had to invent new features to provide security in the network. In this section, the main features of the blockchain will be explained.

2.4.1 Hash

One of these features is the use of the Hash function which keep the identity of each user and the contents of each block in the chain unique. Specifically, the hash function is similar to identification of the person through its fingerprint. For example, not only does a transaction include the asset will be sent but also the address of the sender and the receiver. Through the hash function, the network can prevent the peers from imitating other participants in the network as it can confirm that a transaction originated from the real sender.

The hash function is a mathematical algorithm which converts an input of a variable size into an output of a fixed size. For example, Bitcoin uses the Secure Hash Algorithm SHA-256 where the input can be 2^{64} bits and the output of this algorithm is a fixed size of 32 Byte [27]. This function is characterized as not irreversible and the result of this algorithm represents a unique digital fingerprint which cannot be invertible. Furthermore, a slight difference in the input data leads to a new hash creation and as it is depicted in the figure 2.4, the deletion of only one exclamation mark changes completely the obtained output.

In blockchain, hash plays a significant role to keep the block secure and immutable in the chain as each blockheader of the block includes the hash of the previous block. As a result, its impossible for an attacker to modify the contents of a block because not only must he change the single block but also all the subsequent blocks in the chain.

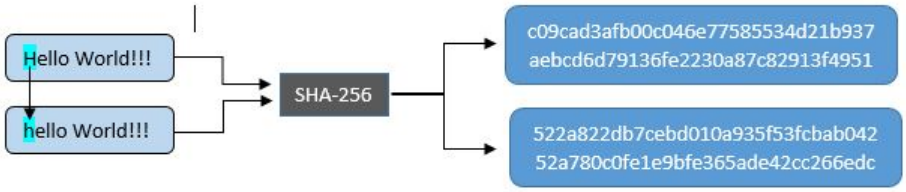

Figure 2.4: Example of hash function

2.4.2 Public and Private Keys

Every peer in the blockchain network owns a pair of asymmetric keys, the public and the private key. These keys play a significant role not only on the creation of the transactions but also the validity of these transactions. The private key is secret and is only known by each user. This key is used to digitally sign the transactions in order to prove the authenticity. The public key is visible by everyone in the network as it represents the address of each user by hashing this key. In addition, the public key is used to verify if the specific transaction was executed by the real sender. Furthermore, hashing the public key gives an extra privacy to the user.

The picture 2.5 depicts the whole procedure needed for the implementation of the transaction. The digital signature consists of two phases[1]: the signing phase where is generated by using the private key and the second one is the verification phase where anyone can verify the digital signature.

Alice wants to execute a transaction with Bob. In order to make this transaction, she has to follow some steps.

- Alice, firstly, has to create a hash of all the data that correspond to this transaction.
- In addition, she creates a digital signature by encrypting the hash which was obtained by the data of this transaction with her private key.
- Finally, she sends the digital signature as the data of the transaction to Bob. Then, Bob can verify if Alice is the real sender of this transaction.

On the other hand, once Bob has already received the data of the transaction and the digital signature of Alice, he has to implement two procedures to crosscheck the validity of Alices transaction.

- Bob has to decrypt Alices digital signature by using her public key in order to get the hash.
- Secondly, Bob has to use the same hash algorithm as Alice in order to calculate the hash of the data.

If the results of the hash coincide then it is clear that the verification was implemented successfully without tampering the data and Alice is the real sender of the transaction.

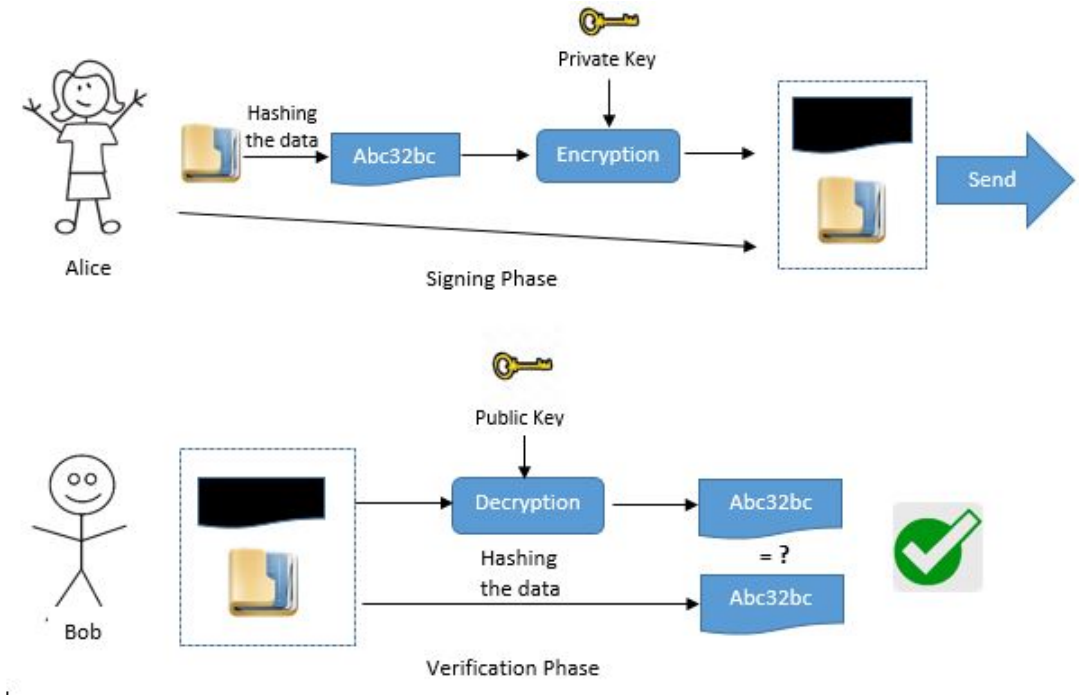

Figure 2.5: Implementation of transaction

2.4.3 Merkle root

In all types of blockchain and especially, in that of public network, the number of users that can take part in the network could be very high from thousands to millions of users as a result, the bigger the network is, more transactions are executed in the network. In order to be implemented, these transactions are stored firstly in the blocks before appending to the chain. Hundreds to thousands of transactions can be included in every block. Once a block has been inserted to the chain, then each user is informed about this situation and stores a full replica of the ledger. However, that operation cannot be implemented normally when the users have a limited capacity in their devices such as mobile phones and laptops, as there is no enough space to store the updated replica of the ledger. For this reason, the inventor of Bitcoin, Satoshi Nakamoto, was the first that made use of Merkle tree root in blockchain in attempts to resolve the issue of the storage space. This technique was invented by Ralph Merkle in 1979.

Merkle root is included in the blockheader of each block which belongs to the chain and represents a single hash value of the whole transactions corresponding to a specific block. Especially, Merkle root works as a binary tree which consists of hashes. An example of Merkle root is depicted in figure 2.6 supposing that a certain block consists of four transactions Tx1, Tx2, Tx3 and Tx4. These transactions are located at the bottom of the tree where a hash of each transaction is calculated separately hTx1, hTx2, hTx3 and hTx4 through a cryptographic hash function. Then, the hashes of these transactions known also as children are grouped in pair of two creating two new hashes hTx1Tx2 and hTx3Tx4 which are called parents. Especially, the concatenation of hTx1 and hTx2 is hashed and produces the hTx1Tx2. These hashes represent the parents and the leaves. This procedure is continued until reaching the final single hash known as Merkle root. As referred previously, Bitcoin makes use of SHA-256 cryptographic hash function and in the case of Merkle root, this function is used twice in order to obtain the hash result [1]. For instance, hTx1 = hash(hash(Tx1). Furthermore, the number of transactions included in

each block should be even as the following example with the four transactions. However, in case of existing an odd number of transactions in the block i.e. 5 transactions, then the fifth transaction has to be paired with itself to create an even number of transactions and subsequently this pair is hashed creating a new parent hTx5Tx5.

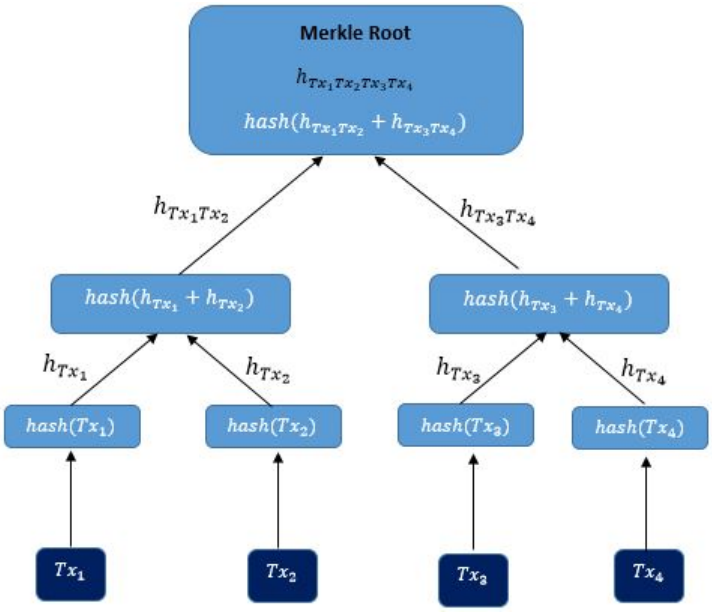

Figure 2.6: Structure of Merkle tree

On the other hand, not only was Merkle root used to overcome the storage issue, but also to crosscheck if a specific transaction or a number of transactions belongs to a certain block. This can be achieved by checking only some intermediate hashes of the Merkle tree. Given these hashes known as Merkle path, the verifier should calculate a single hash value and then he has to compare it with root hash. If both hashes are the same, then the transaction belongs to this Merkle root or it has not been tampered with. Lets assume that Bob receives the root hash hTx1Tx2Tx3Tx4 from a trusted source and Alice wants to prove to Bob that the Tx1 transaction exists in this specific root hash and the data is not tampered. So, Alice has to send the Tx1 transaction, the hTx2 and the hTx3Tx4 to Bob. He, in turn, should calculate the h'Tx1, the h'Tx1Tx2 and h'Tx1Tx2Tx3Tx4. Finally, he has to compare if both root hashes are the same hTx1Tx2Tx3Tx4= h'Tx1Tx2Tx3Tx4. In case of Bitcoin, where more than a thousand of transactions can be included into 1 Mb block, users have to produce only log2N hashes (from 10 to 12 hashes) to verify a single transaction. N is the number of the whole transactions that exist in the block. As a result, the computational effort can be reduced through the Merkle tree.

The Merkle root is a useful tool, especially for the users that they do not have available large space storage. In blockchain, the users can be divided in two categories: full nodes and light clients [26].

- Full nodes are responsible for downloading a replica of the ledger which consists of the genesis block until the last appended block in the chain as to verify the authenticity of the transactions which are recorded in the blockchain. Users are served as a full node need a huge amount of storage in order to store the updated replica. Nowadays, the size of the blockchain in Bitcoin has reached almost 250.6 GB [4] and this size is going to be increased as each block is generated every 10 minutes.
- Light clients, in contrast of full nodes, download only the blockheader of each block

which corresponds to 80 Bytes per block [14]. As a result, users with a limited capacity can have an active role in the network as light clients. They make use of the Simplified Payment Verification (SPV) [1][26]in order to check the authenticity of the transaction. When a user wants to verify a transaction, it makes use of Bloom filter (Bitcoin) in order to find only the associated transaction by asking for a full node. Once the full node has detected the desired transaction that corresponds to the Bloom filter, then the node informs the light client by sending to him the blockheader of the corresponding block as the Merkle path.

2.4.4 Structure of blocks

As mentioned previously, blockchain is a chain which consists of many blocks. Each block depends on the previous block as each one includes the hash of the previous block. Furthermore, such a feature makes a blockchain more secure because it is very difficult to modify the block without changing the subsequent blocks. As a result, the blocks inserted to the chain are immutable and neither can be deleted nor tampered with. The blocks differ from blockchain to blockchain and constitute the records of the transactions as the blockheader which contains some specific data.

When the transactions occur, every validator has a right to pick a number of transactions, verify them and insert them into a block. The number of transactions that can be stored in a block depends highly on the size of the block and the transaction. For example, the number of transaction in Bitcoin is limited as the size of the block is only 1 MB. In addition, validators can usually choose which transaction wishes based on the fees of each transaction. In Bitcoin, the transaction with the highest fees submitted by the producers of each transaction are selected quicker than those transactions with low fees.

Besides the transaction list which is included in the block, the blockheader is, also, part in that[1][28]. Blockheader consists of six fields:

- Version: it indicates the version number of upgrades and changes in the protocol used by a validator.
- Timestamp: it indicates the time where a block is generated.
- Hash of the previous block: this hash points to the previous block. Especially, all the data that are included in the block are inserted in a cryptographic hash function in order to obtain a single hash value. This hashed result is stored in the next block in the blockchain.
- Merkle root: as refereed to earlier, it is a single value of hash which represents the summary of all the transaction included in the block. Firstly, once the transactions have been hashed, they are grouped in a pair of two producing a new hash. This operation is continued until only one single hash is fount.
- Nonce: it is used in Proof of Work (PoW) algorithm by the miners and it is the only field in the block that can be changed. Modifying the nonce, miners have to solve the difficult mathematical puzzle in attempts to prove that the hash of the block header is lower or equal to the difficult target.
- Difficult target: it is a target threshold which is determined by the network every two weeks in order to follow the rules of Bitcoin where a new block is generated every

10 minutes. For instance, if the time that the blocks are inserted in the blockchain is less than 10 minutes per block, then the difficulty target should increase.

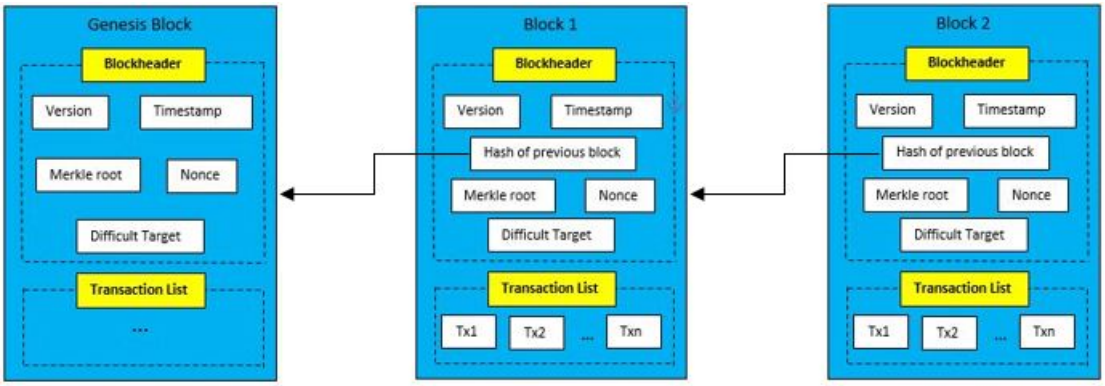

Figure 2.7: Structure of blocks

According to the sequence of the blocks, the block which is generated firstly in the chain is known as genesis block and is, also, referred to as block zero. Each block in the blockchain can be identified through a block height number which is always a positive integer number and increases by one every time that a block is inserted to the blockchain. For example, the second block which is inserted by the genesis block is referred to as block 1 and so on. A blockchain is depicted in the above figure 2.7 which consists of the genesis block and another two blocks. Furthermore, it illustrates how the block are linked to each other as the contents of them.

3 Consensus

3.1 What is Consensus?

In a centralized network, the central authority is responsible for the validation of the transactions providing security and trust to the entire network. However, in a distributed and a decentralized network, the peers interact only each other as there is not any central authority. In this case, for better maintenance of the network, all the peers have to follow a set of rules determined by the network and come to an agreement over a single block which will be inserted in the chain. This agreement in the blockchain is achieved through consensus algorithms. Nowadays, there are a lot of consensus algorithms where each one is used by a different type of blockchain. For example, in permissionless networks, the most common consensus protocols are Proof of Work (PoW) and Proof of Stake (PoS) while the permissioned networks make use of practical Byzantine Fault Tolerance (PBFT) etc.

The main purpose of consensus algorithms is to solve the problem that is created when any node in the network try to reach a consensus either in the presence of malicious behaviour of the peers or any faulty process exists in the network. These issues are known as forking and Byzantine General Problem. Forking can happen when two peers propose simultaneously a new block or when a peer wants to behave maliciously creating the same block twice. In both cases, consensus mechanism can solve this problem by choosing the longest chain of the blockchain. Furthermore, the byzantine general problem occurs when there is not unanimity in the network between the peers agreeing for a single piece of data on the grounds that some of them can misbehave sending erroneous messages or being faulty due to the crash of the system etc. As a result agreement can not be reached.

As referred previously, the distributed network suffers from some issues such as an abnormal behaviour of the peer or the network. However, two main categories of consensus mechanisms have been developed in order to overcome these problems [19]:

- Proof-Based consensus algorithm: it is used in permissionless networks where every peer in the network can be a validator to verify the block and insert it in the chain giving the sufficient proof to the network that a validator deserves to participate in. For instance, Bitcoin uses the PoW consensus algorithm and a validator (miner) has to consume a huge amount of energy in order to solve the mathematical puzzle and append the block to the chain.
- Vote-Based consensus algorithm: this type of algorithm is used in permissioned networks based on multiple rounds of votes in order to achieve consensus. For example, Hyperledger Sawtooth makes use of PBFT where a lot of messages are exchanged between the participants in the network in attempts to reach the consensus.

3.2 Byzantine General Problem

The "Problem of the Byzantine generals" was originally described by Marshall Pease, Robert Shostak and Leslie Lamport in 1982 [13] and was based on the scenario that parts

of the Byzantine army camped outside the walls of an enemy city are facing problems with the creation of a joint action plan due to the possibility of transmitting unreliable information. In detail, each part is administered by its own general and it has been decided that after a certain period of monitoring of the enemies, all the generals should agree on a joint action plan. So in order to achieve the necessary communication and the coordination of all generals, it is safeguards to transfer the necessary information and news from the one part of the army to another. At this point it should be noted that in the given historical period the transmission of information was made exclusively by messengers of each troop. However, this method of transmitting information poses risks of falsifying the information, because some informants who have betrayed the troop and allied with opponents, they have the ability to transmit false information in order to prevent (law-abiding) generals from coming to an agreement.

So, on the basis of the above, it is clear that the generals must use an algorithm which guarantees that:

- The action plan to be implemented is a unanimous decision by all the generals involved.
- The possible existence of a limited number of informants who have betrayed their troops will not lead to the manipulation of the data and therefore to the adoption of a harmful action plan.

Figure 3.1 illustrates two examples of Byzantine general problem in a presence of honest and dishonest soldiers. The example appeared in the left icon depicts the successful attack in the city as all the army following the general's command will attack at the same time. On the other hand, the second example represents the unsuccessful attack when there are some traitors trying to manipulate the order of the commander. In more detail, the general (Commander) gives an order of attack to the army but a part of the army are traitor and falsifies the order in retreat.

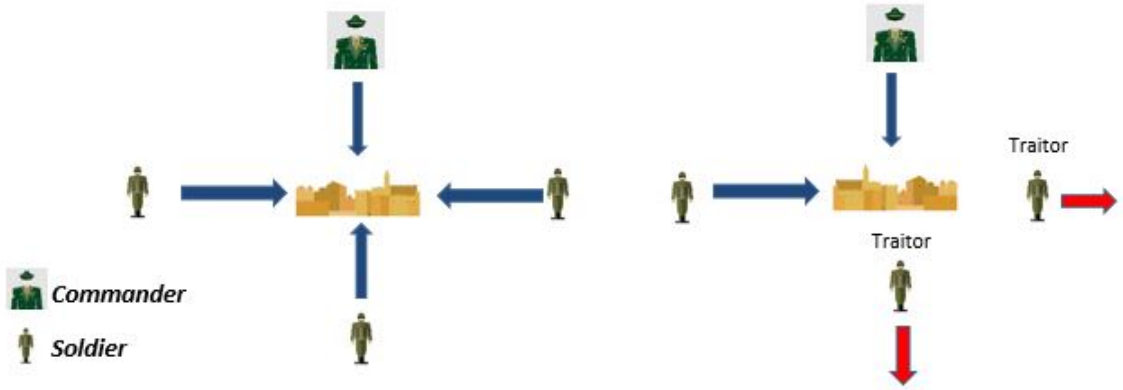

Figure 3.1: Coordinate and uncoordinated attack

It is therefore obvious that without the existence of synchronicity and integrity in the information exchanged, it is impossible to implement a unanimous and reliable action plan, because the risk of carrying forged orders from traitors informants always exists.

This method can correlate with the distributed system where generals represent the honest nodes and the generals who betray the troops represent the malicious nodes in the

network. Lamport, Shaostak and Pease proved that if at least the 2/3 of the network acts honestly, then there will be a unanimous decision in the entire network. Especially, they determined a condition which indicates that if the number of malicious nodes is equal or exceeds the 1/3 of all the nodes, then the system will not work properly.

$$n \geq 3f+1$$

Where n is the total number of nodes in the network and f the number of malicious nodes.

The above condition can work properly if the whole network consists of at least four nodes. In the case of three nodes where one of those nodes behaves maliciously, agreement cannot be reached between of them as each node receives two different messages. As depicted in Figure 3.2, there is one commander and two generals. Commander and general 2 act honestly but general 1 is a traitor and wants to change the decision of the commander. Therefore, general 2 cannot make a decision as received both attack from commander and retreat from general 1. In normal case, general 2 should trust the commander as he posses higher position in hierarchy than general. However, what happens if commander behaves maliciously? This leads to the same scenario as previously, since commander sends attack to general 1 and retreat to general 2. So, General 1 does not know what decision can take.

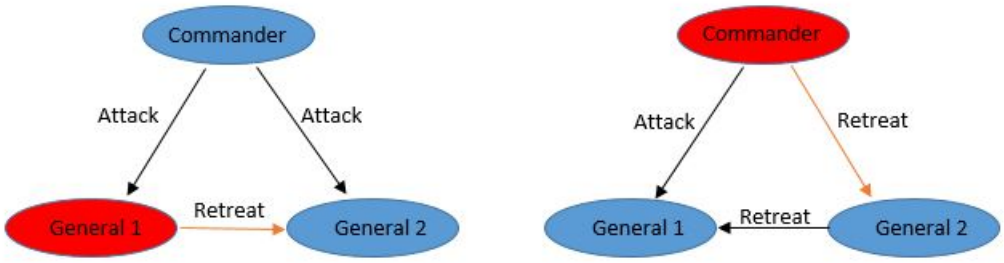

Figure 3.2: System with three nodes

In case of four nodes where one of them is a malicious node, generals can reach agreement as the 2/3 of the nodes in the system behaves honestly. Each general informs the other ones in the system about the command they have received. According to Figure 3.3, General 2 receives two attacks from the commander and the general 3 and one retreat from the general 1 acting as a traitor. As a result, general 2 takes into account the majority of the messages that he received and then he will attack. In addition, in case that commander sends erroneous messages, generals could be possible to reach in agreement, as two of the three generals will get the same messages. For example, all generals receive two attacks and one retreat.

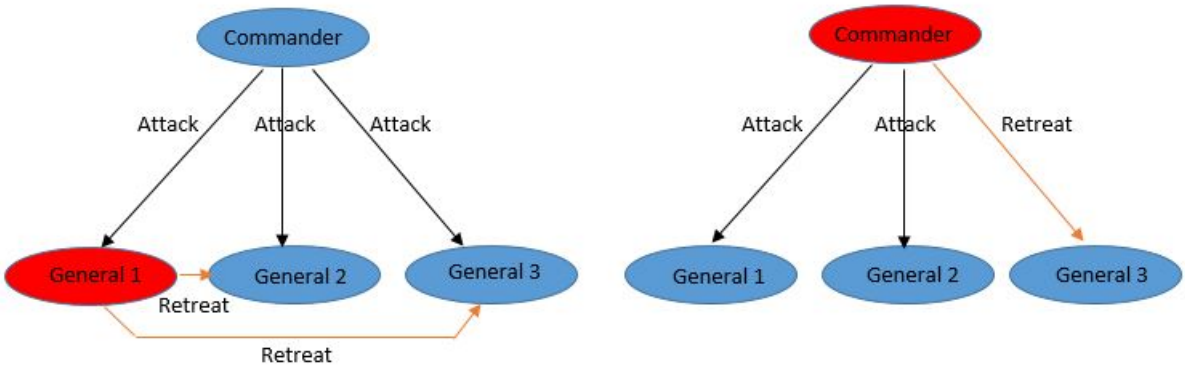

Figure 3.3: System with four nodes

3.3 Consensus Algorithms

3.3.1 Proof-Based consensus algorithm

3.3.1.1 Proof of Work (PoW)

The Proof-of-Work (PoW) is an algorithm in attempts to achieve consensus to ensure the validity of the transactions as the creation of blocks. Although, the concept of this algorithm was first developed in 1993 by Cynthia Dwork and Moni Noar trying to solve the problem of spam emails, the term "Proof of Work" was first referred and formalized in a 1999 paper by Markus Jakobsson and Ari Juels. Later, this algorithm became widely known through the founder of Bitcoin, Satoshi Nakamoto, who used this mechanism to reach consensus successfully between many nodes on the network and he used it as a way to secure the Bitcoin Blockchain.

To understand the operation of the proof-of-work algorithm should be taken account that each time a transaction is executed in the Bitcoin network, it is recorded and is stored in a temporary block. Thus, once the block has been confirmed with all the validated transactions included in that block, then this block is inserted to the chain sending a replica of this record to the rest of the participants in the network. The validity of those transactions as the insertion of the next block in the chain are implemented successfully through the use of Proof-of-Work consensus protocol. The above process is called mining and is done by the miners who are trying to solve a difficult mathematical cryptographic puzzle in attempts to obtain the appropriate result determined by the difficult target. Once miner obtains that, it informs all the participants in the network about the result and then the participants verify this validity in an easy way reaching the consensus.

Especially, every time that a new block is generated, all the verified participants should include all the valid transaction into the block as the HASH of the previous block. In addition, in the new block, the block header consists of some other fields like the Merkle root, difficult target and the Nonce. All the fields included in the blockheader as the procedure of the PoW are depicted in the figure 3.4.

In Bitcoin, using an encryption algorithm called as SHA-256 hash, miners have to solve the difficult mathematical puzzle by modifying the nonce in order to prove that the hash of the block header is lower or equal to the difficult target. Always, the output of a

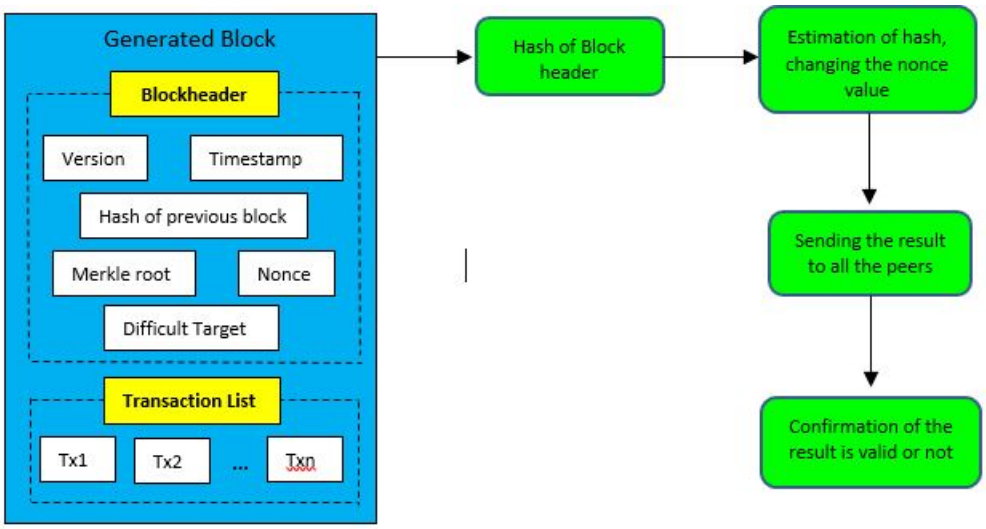

Figure 3.4: Procedure of PoW

SHA-256 hash will be 256 bits long regardless of the size of the input.

$$\text{hash}\ (\text{blockheader} + \text{nonce}) \leq \text{Difficult target}$$

For example, if the difficult target for the network is 0000 and the base string is "Hello, world!", then miners should find the desired result changing the value of nonce until the hash (Hello World! nonce) gives at least 0000. The rest bits of the hash does not affect the selection of the desired result.

Hash ("Hello, world!0") ⇒1312af178c253f84028d480a6adc1e25e81caa44c749ec81976192e2ec934c64

Hash ("Hello, world!1") ⇒e9afc424b79e4f6ab42d99c81156d3a17228d6e1eef4139be78e948a9332a7d8

Hash ("Hello, world!2") ⇒ae37343a357a8297591625e7134cbea22f5928be8ca2a32aa475cf05fd4266b7

...

Hash ("Hello, world!4249") ⇒c004190b822f1669cac8dc37e761cb73652e7832fb814565702245cf26ebb9e6

Hash ("Hello, world!4250") ⇒0000c3af42fc31103f1fdc0151fa747ff87349a4714df7cc52ea464e12dcd4e9

As shown in the example, the computer tried 4251 times to calculate the hash that gave the first 0000 bits. Furthermore, this obtained result is sent to all participants of the network and they, in turn, have to verify this result hashing the phrase with the corresponding nonce. If the validation is done successfully, then the block is verified and the miner can receive the rewards as a small amount of transaction fees. The miners, usually, choose the transactions with the highest fees in order to get included in the block. But the number of these transactions is limited because is fully depended on the block size which is 1 MB. The transactions fees are determined by the peers that execute the own transactions. More the fees that peers pay, faster the time to valid these transactions.

Especially, a reward should be given to the miner after the insertion of a given number of blocks. For example, in Bitcoin, a reward is given after at least 6 validated blocks appended to the chain making the block immutable. Nowadays, the rewards of miners are 12.5 bitcoins but during the 2016, the block reward was at 25 bitcoins. The amount

of reward is halved every 210000 blocks and this happens in interval of four years. This procedure of halving rewards will keep going until the total amount of bitcoins reach the 21 million in 2140. Then, the generation of bitcoins will be stopped as a result, miners will not get the rewards but they will keep receiving the transaction fees.

Due to the different CPUs that exist worldwide, some miners can solve the cryptographic puzzle in less than 10 minutes or they want more than the expected time to have a result. For this reason, the network determines a difficult target every two weeks in order to follow the rules of Bitcoin where a new block is generated every 10 minutes. If the time that blocks are inserted in the blockchain is less than 10 minutes per block then the difficulty target should increase and vice versa. The value of the difficult target can be obtained from the following equation:

$$New\_Difficulty = Old\_Difficulty * \frac{Actual\ time\ of\ last\ 2016\ Blocks}{20160\ minutes}$$

The period of two weeks consists of 20160 minutes. During these two weeks, if every new block is generated every 10 minutes, then it will consist of 2016 blocks. In case that the blocks are generated every 9 minutes, then 2240 blocks will be generated in two weeks, 224 blocks more than the normal situation. As a result, the network should calculate only the actual time of the first 2016 blocks needed to calculate the new difficult target. Therefore, the time that corresponds to 2016 blocks are 18144 minutes and having the old difficult target and the actual time of last 2016 blocks, the network can calculate the new difficult target.

There is a case that two or more miners can find the same or different result where the hash of the block header is lower or equal to the difficult target by solving the cryptographic mathematical puzzle simultaneously. Each miner tries to solve its own cryptographic puzzle as each block includes different transactions.

In that case, the miners should broadcast the result to the rest of the participants of the network, which leads to the creation of the forking. As depicted in the following figure 3.5, fork can consist of more than two chains. In a normal situation, miners will choose in which chain they are going to mine because every time that a miner tries to mine a block consumes a huge amount of energy. They have the right to mine in more than one chain but this is needless due to the reason that the miner's reward will be smaller than the cost of energy he is going to consume. Furthermore, the fork will be stopped when one chain overtakes the other chains. So, the longest chain is chosen giving the reward to the miners belonging to this chain and the other ones are discarded without miners taking any reward.

The main goal of PoW is to prevent cyber-attacks such as Sybil attack and Distributed Denial of Service attack (DDoS). Sybil attack occurs where a single entity creates multiple fake identities in order to control the majority of the network. If the attackers control more than the half of the peers in the network they can execute a double spend attack implementing which transaction they want and preventing transactions from other users to be validated. In addition, they can modify validated transactions that have already inserted in the block. In PoW, an effective attack is costly and time consuming on the grounds that every device wastes a huge amount of energy in attempts to solve the mathematical puzzle and that makes the PoW algorithm more secure. For this reason, Sybil attack can not be implemented. The purpose of the DDoS attack is to delay the transaction valida-

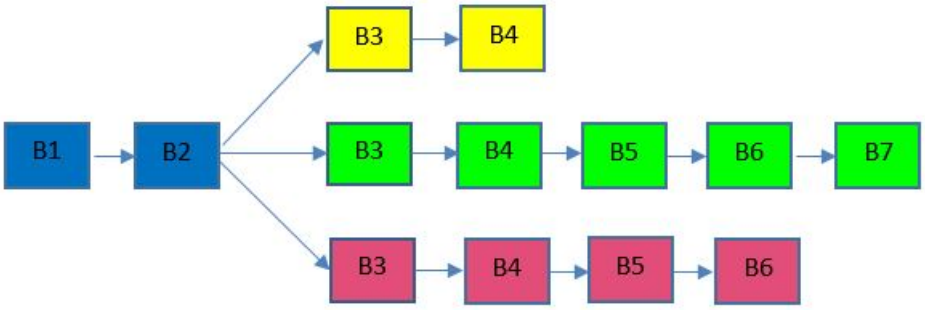

Figure 3.5: Case of fork

tion of the peers by sending a flooding of fake transactions in the network. The Bitcoin can validate up to 7 transactions per second, the size of each block is only 1MB and each block is inserted to the chain at about 10 minutes. As a result, only a specific number of transactions will be included in each block, some transactions will never be validated, and they will be discarded. Bitcoin, solved this problem by introducing transaction fees. Therefore, the miners include to their blocks the transaction with the highest fees.

The PoW suffers from what is known as 51% attack, where a group of miners manipulates the majority of computing power on the network and thus they are able to choose what transactions can be valid regarding to their own perspective. This can be easily achieved through the mining pools in which a lot of peers cooperate in order to find the solution of the cryptographic puzzle in a faster manner. They are given more chances to solve the cryptographic puzzle rather than other peers who are not included in the mining pool as a result they can control the network. In addition this cooperation leads to the centralization.

However, the presence of 51% attack leads usually to another issue that exists in the blockchain, called as double spent attack. Such an attack happens when a malicious peer wants to spend the same amount of coins on the blockchain twice. Every time that a transaction is executed, a vendor must wait n confirmations of blocks and then provide the product to the sender of the transaction. The n number of confirmations depends on the types of the payments, namely how fast or slow a transaction is implemented [10]. The slow payments are those that provide the most secure way of payments on the grounds that a vendor has to wait at least 6 confirmation in order to send the product and the time which is needed to the confirmation operation corresponds approximately to one hour.

On the other hand, the vendors do not need to wait 6 confirmations to make an exchange in fast payments. In this case, they usually do not wait for confirmations or just waiting for one or two confirmations. So, there is a possibility of a double spent attack where a malicious attacker creates a fork by putting its real transaction (a purchase of a product) in the block of the first branch and its fake transaction (the same amount of coins is sent to another account controlled by the attacker) in the block of the second one. The second branch, in essence, is produced by malicious peers and is not published to the entire network. This branch will be published when a malicious attacker receive the product and when this chain overtakes the real chain. As a result, all the transaction included in the real chain is discarded and the fake chain wins. According to the following figure 3.6, it illustrates the real chain in the light blue boxes where Alice wants to buy a t-shirt and the fake chain is represented by the yellow blocks in the second chain.

In addition, another drawback is that it has been proved costly and energy intensive

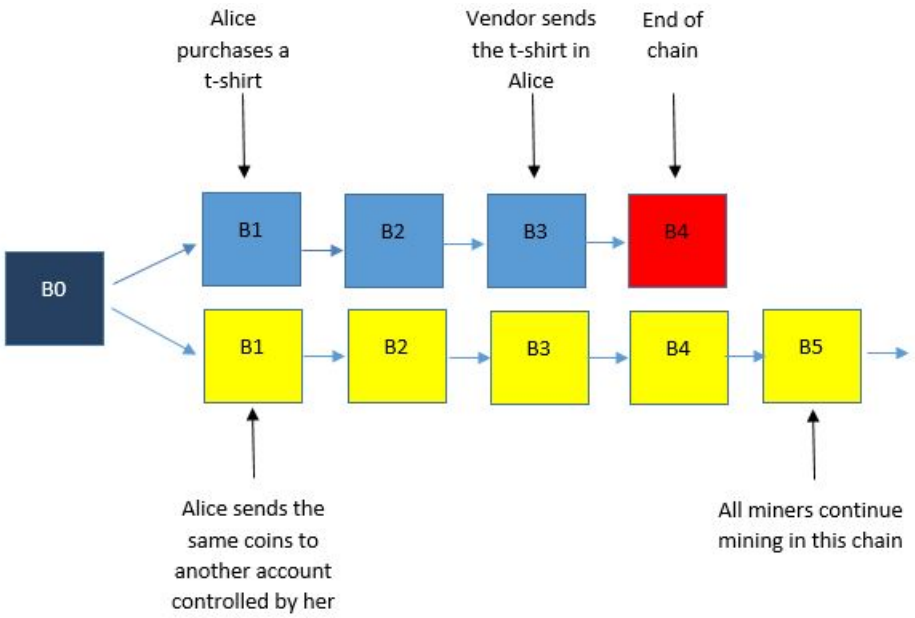

Figure 3.6: Double spend attack

as miners require a huge amount of computation power and expensive equipment in order to execute the mathematical algorithms. Furthermore, PoW has some performance limitations. The first one is the low throughput as the average number of transactions that can be committed are 7 transactions per second, unlike Mastercard that executes 10000 transactions per second [2]. Furthermore, PoW suffers from the high latency since a block can be considered as valid after at least 6 verified blocks (1 hour) appended to the chain. For this reason, cryptocurrencies like Nxt makes use of other consensus algorithms like a PoS in attempts to improve these limitations. There is also another cryptocurrency called Ethereum which makes use of PoW but it attempts to switch to PoS.

3.3.1.2 Proof of Stake (PoS)

Even though the purpose of this Proof-of-Stake algorithm is the same as the Proof-of-Work algorithm, the process for validating the transactions is completely different. The main objective of this technique aims to avoiding long computationally expensive sequences of cryptographic operations leading to high energy consumption. The first idea of the PoS was proposed in the online forum bitcointalk.org in 2011, but the first cryptocurrency that made use of this method was the Peercoin in 2012 and Nxt followed in 2013.

Unlike the PoW algorithm where miners try to solve the mining operation, in the PoS, there are validators or forgers to proceed to that operation. The purpose of the forgers is to validate the transactions as to create new blocks. The selection of the forgers is done through the amount of coins (stake) that each peer holds in the network. The participant that owns a bigger amount of stake has a bigger probability to be a forger. For example, if Alice is holding 100 tokens, Bob 80 tokens, Daniel 60 tokens and Luis 20 tokens, then Alice has 5 times more chances to be the forger or validator than Luis. Despite PoW, where miners get the rewards when they solve the mathematical puzzle, in PoS, forgers receive only the transaction fees from the corresponding block.

But this situation leads to centralization, because the participant who owns the largest

amount of stake in the network, he will always be chosen as a forger for the creation of the block. As a result, he would be the only one who would earn the transaction fees controlling the whole network. In order to prevent the above problem different cryptocurrencies make use of different methods. For example, the selection of the forger in Peercoin is based not only on stakes but also the coin age where is the amount of time that the forgers hold this stake. In addition, Nxt cryptocurrency chooses randomly forgers based on their stake. Through the randomization and the coin age, PoS prevents the issue of centralization, something which happens in PoW through the mining pools.

The coin age that Peercoin [21] makes use, refers to a numeric value which indicates the number of coins that belongs to the user multiplied by the duration in days that a user hold the coin. For example, if Alice received 20 coins from Bob and held it for 30 days, then the coin age of Alice is 600 coins-days. In order to proceed to the verification of the transactions or to the creation of a new block, forgers should be waiting for a minimum of 30 days without spending any coin. Usually, the user which holds huger amount of coins and bigger duration than others, he will have the probability of being chosen to forge the next block. Every time that a forger validates a new bock, then a coin age of this forger is being zero and returns at its initial state waiting at least 30 days to be forged again. In addition, the maximum period where the participant can be forger is 90 days. Otherwise, the participant who held old and huge amount of stakes would be able to control the Blockchain.

In Nxt[20] cryptocurrency, the block is generated on average every 60 seconds. In order a peer to be selected as a forger, he should be waiting for a minimum of 1440 confirmed blocks without changing their amount of stake. Through this technique, Nxt prevents the users to using the same stakes in different accounts in order to valid the transactions and generate new blocks. After that, all users can be selected as a forger in a random way. When a validator is chosen then he can validate up to 255 transactions including them in the block. The block size in this cryptocurrency is 42 KB and a forger can choose which transactions wants, usually according to their fees. The Nxt can validate up to 100 transactions per second. Despite Bitcoin, in Nxt the transactions have deadline and this is 24 hours. This means that if a transaction remained in the transaction pool for more than 24 hours and one second then this transaction is rejected. In addition the block can be finalized after the insertions in chain of ten confirmed blocks at about 14 minutes.

One of the main advantages of PoS is that participants do not need to invest a huge amount of money in order to get the reward. There is no need for solving mathematical puzzles and this results less energy consumption as no expensive equipment. So, they can spend their money to buy more coins (stakes) in order to increase the possibility to be a forger and to receive the transactions fees. In addition, PoS provides faster processing of transactions compared to PoW. As in PoW, 51% attack is also possible in PoS but this is very difficult and expensive to implement due to the fact that a hacker should buy and control more than the half of the stake in the network where is a very huge amount of coins.

On the other hand, PoS suffers from what is known as nothing at stake. The issue can occur anytime the forks appear. Fork is happened when the chain of the blockchain is divided in two or more chains. This can be taken place for two reasons, the first one is due to malicious action and the second one is when two forgers propose a new block simultaneously. The last one is more difficult to happen in PoS as each forger is chosen by the system in a random way and it is responsible for inserting the block in a specific

timeslot. Furthermore, this issue appears more often in the PoW algorithm where all the validators try to solve the mathematical puzzle at the same time. The ideal scenario in case of forking is that the forgers should choose only one of the multiple chains, like in PoW. However, as referred above, it costs them nothing to take part in all the chains. So, the optimal strategy is that all the forgers can support multiple chains without the fear of having something to lose and always they can get the transaction fees from the longest chain of the wining fork.

Usually forgers use forks in order to implement double spent attack as they have nothing at stake to lose. An example of how forgers can proceed to double spend attack is illustrated in the figure 3.7.

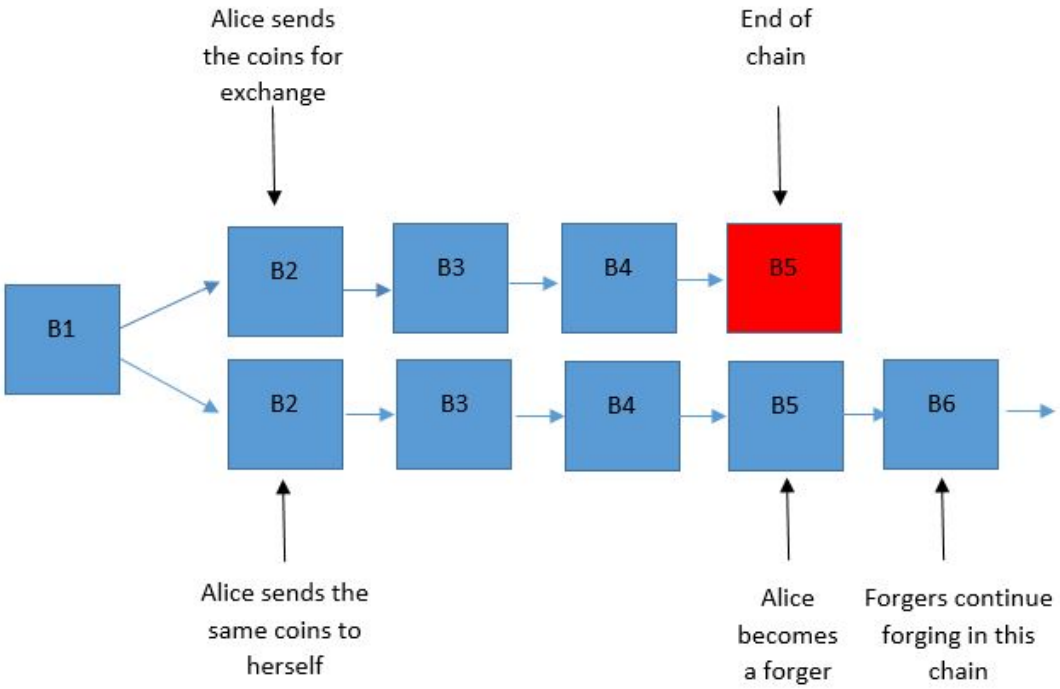

Figure 3.7: Example of double spending attack

Supposing that Alice is an attacker and she wants to execute double spent attack by changing her coins to bitcoins. Firstly, once fork has be created, Alice sends the coins to be exchanged in the above chain and the same coins are sent by Alice to herself in order to implement double spent attack. When the transaction becomes valid, then Alice can buy the bitcoins. In order to get the Bitcoin without paying any cost, when Alice is given the turn to validate the next block again, she will validate the block only in the second chain. While all forgers continue validating blocks in both chains, the second will be the longest chain as Alice stopped validating the block in the first chain. As a result, the first chain be dropped and Alice has achieved to add a new amount of coins at her stake by stealing the Bitcoin off an exchange as the transaction fees of the blocks that she insert in the chain. In addition, another cryptocurrency like the Ethereum tried to solve these problems by locking their stake as a deposit in order to be forgers. In this way, forgers are given the incentive to act always honestly, otherwise in case of fraudulent behaviour they are going to lose their deposit.

Another limitation of PoS is that the poorest participants in the network may never have the chance to be forgers in order to validate the transactions getting the corresponding transaction fees. This is unfair because the richer participants have always the greater

possibility to be forgers than the others and PoS makes stronger only the richer participants in the network. Leased Proof of Stake (LPoS) solves that domination problem as will be described in the next chapters.

3.3.1.3 Delegated Proof of Stake (DPoS)

Dan Larimer created DPoS in 2013 in order to overcome the limitations of PoW and PoS providing better scalability, flexibility and security in the network. Initially, this consensus algorithm was used in BitShares platform [5] and with the passing of time, DPoS was used from different projects like by Steem, EOS, Ark and Lisk etc. making some changes in the protocol. The main purpose of DPoS is to give the right to all participants that hold stakes (stakeholders) to be an active member through the voting procedure in order to solve the issues in the blockchain in a democratic and fair way.

Unlike the PoS where only the richest participants have the opportunity to take part in the validation of the transaction and the creation of the blocks in order to get the fees, in DPoS, all stakeholders have a significant participation role in the network. Stakeholders are responsible for electing the witnesses and delegates in order to execute the appropriate operations. Each participant has the right to vote in or out any number of witnesses or delegates he wishes in real time. The voting procedure can be valid only when more that 50% of stakeholders participate in election of the witnesses or delegates. In addition, each stakeholder cannot vote one specific participant more than once.

The figure 3.8 depicts the voting procedure by the stakeholders in order to elect the witnesses and the delegates. The yellow circles depict the peers that want to be voted from the other users in the network (witnesses) and the blue ones represent the stakeholders. The voting procedure happens in real time and the stakeholders can vote as many peers as they want to be a witness or delegate according to their stake they have. In BitShares, the list of the votes is updated every day and the number of witnesses amounts to 101. As a result, in the end of each day the top 101 candidates with the highest votes become the witnesses.

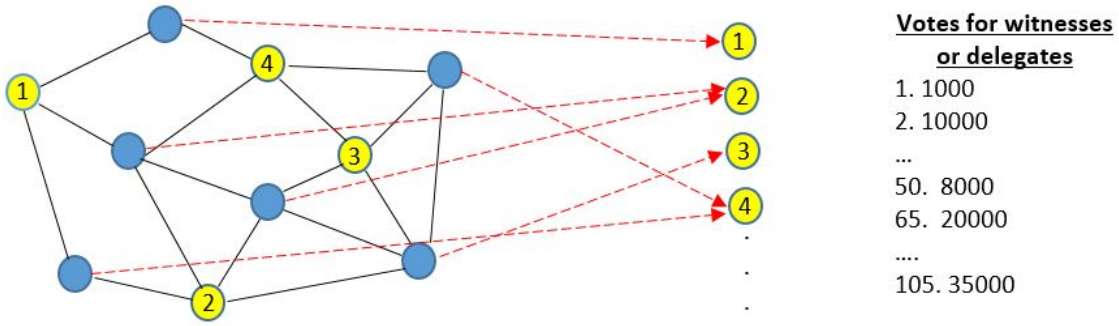

Figure 3.8: Example of voting procedure in DPoS

Furthermore, the votes of each stakeholder depends on the number of stakes that hold and this operation is known as Stake-weighted voting. For example, Alice has 100 coins and Bob 60 coins. If Alice wishes to vote two witnesses, Tom and Charlie, then both of them will receive the 50% of her voting weight. Bob, in turn, will vote only one witness, Bill, and so this witness will receive the 100% of his voting weight. As a result, Bill has a greater voting power than the others and then he will become the witness. Sometimes, this property benefits the weaker stakeholders as they can get the votes from

richer stakeholders and can be witnesses or delegates. In addition, another feature of DPoS is that users have the right to choose others trustful users in order to vote on their behalf.

Witnesses are responsible for validating the transactions and creating the blocks. As referred, they are elected by the Stakeholders of the network who have the right to vote out the witnesses at any time they want leading witnesses to behave honestly. In case of behaving badly, witnesses can lose the transaction fees of the block creation as their reputation. In addition, the number of witnesses is limited depending on the different projects that exist so far. For instance, EOS has a specific number of 21, BitShares with Lisk 101 and ARK 51 witnesses.

In BitShares, the list of active witnesses is updated every one day after the end of vote counting. The validation of the blocks in DPoS is done in rounds. Each block is produced by the witnesses every two seconds and the validation can be executed only once for each witness until the round is completed. The previous procedure is depicted in the figure 3.9. Assuming that the system has four witnesses the 1, 2, 3 and 4 which are responsible for inserting the block in the chain. In each round, each witness has a specific time slot that can validate the transaction and the insertion of the block in the chain. For example in the first round, witness 4 will be the first one which will insert the block, then follows the witness 1 and etc. In case that witness 3 misbehaves in its specific time slot, then the block is skipped and the next block is given the turn to be validated by the next witness which is 2. Due to that behaviour, witness 3 can lose its reputation and the transaction fees of this block. Once the round has finished, then witnesses are mixed randomly and the procedure starts again with witnesses validating new blocks as appears in the second round of the figure.

Figure 3.9: Example of validation procedure in DPoS

The majority of the projects that make use of the DPoS consensus protocol, except of the witnesses, presents, also, the delegates. Although delegates are elected in a similar manner to witnesses, their purpose is to propose changes for better maintenance of the network. Some of the proposed parameters that have to be changed are the block size, block interval and the amount of transaction fees that witnesses will receive. Once the changes have been implemented, the stakeholders, in turn, have a specific period of time (i.e BitShares two weeks) to decide if they proceed to the modification of parameters or not. Usually, changes in the network are not implemented easily because bigger the network is, more difficult for 51% of stakeholders to vote for changing the network. In few words, neither the witnesses nor the delegates are able to control the network, but the Stakeholders are they who have the greatest power in the network.

Compared to PoW and PoS, DPoS provides better performance in terms of confirmation of transactions. For example, PoW and PoS validate 7 transactions per second and 8 transactions per second respectively. However DPoS, theoretically, can achieve 100.000 transactions per second but in practice only 3300 transactions have been proven so far. In addition, all the participants have an active role on the network due to the real time voting, as a result, the network becomes more secure rather than the other consensus protocols.

Once the network has detected any malicious behaviour, is able to inform the stakeholders about this misbehaviour during 1 minute and then stakeholders, in turn, can vote out this specific witness. Another significant advantage is that even weaker stakeholders can be elected as witnesses getting the rewards.

Moreover, DPoS prevents the nothing at stake problem which is the main limitations of PoS. First of all, as it was referred previously, when a witness behaves in a malicious way, network realizes this situation fast giving the chance to the stakeholders voting out the malicious witness in order to replace him. As a result, witnesses are given the incentive to behave honestly to keep not only the future incomes but also their reputation. Furthermore, the order of the blocks in each round is known by all the participants in the network and in each round the witnesses are arranged in a mixed up order in order to validate the blocks. This means that in each round the witnesses validate the blocks in different timeslots. In addition, in case of forking, the DPoS algorithm chooses the highest chain in order to continue the insertion of the block in the chain. In that way, it would be very difficult for an attacker to create forking longer than the main chain. The figure 3.10 represents the canonical form of how the blocks are ordered in the chain with the elected witnesses. The outlined blocks indicate the start point of each round.

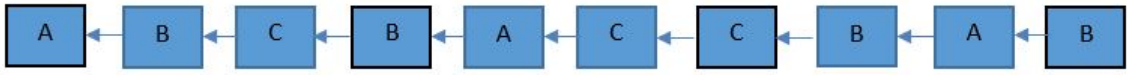

Figure 3.10: Form of canonical chain

In the figure 3.11, let us suppose that witness B wants to make a fork in order to behave maliciously. In that case, Bitshares always chooses the chain with the highest witness participation rate. This rate can be estimated by comparing the required number of produced blocks with the number of existed blocks. For example, if there are only 3 witnesses A, B and C, per round then, according to the figure 3.11 the participation rate of the top fork is 3 required number of produced blocks vs 2 existed blocks and on the bottom fork is 3 vs 1. So, the next witness C will continue validating the new block in the highest chain where A and B coexist, and not in the second chain where only B witness appears.

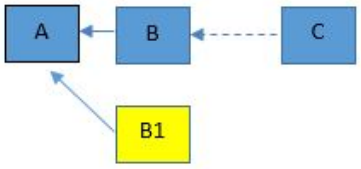

Figure 3.11: Fork with one peer behaving maliciously

Even in the worst case where a majority of the elected witnesses behaves maliciously creating multiple forks, the honest witness will define which chain will be the longest. Despite this situation, the DPoS algorithm has the privilege to identify the nodes which behave badly and to replace them with the honest ones. Especially, each witness is scheduled to valid one block for one time slot in each round and every time that witness produces forking creating more than one block, then there will be clear cryptographic evidence where the network can identify. For example, as depicted in the figure 3.12, the witness B created three blocks with the same timestamp and block height.

On the other hand, DPoS, also, presents some limitations. One of them is that a

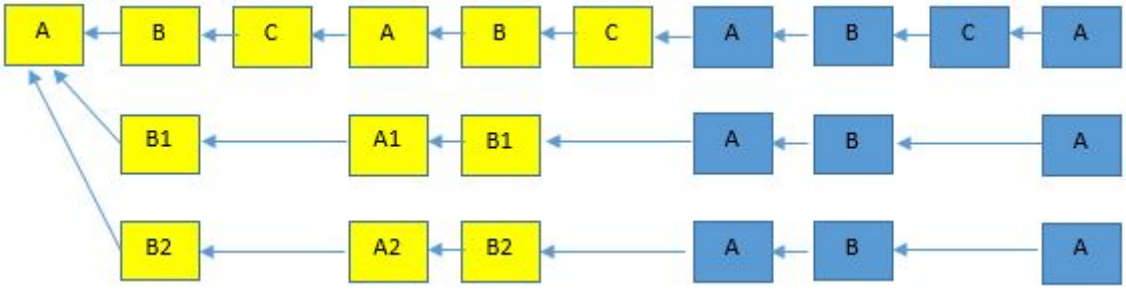

Figure 3.12: Fork with the majority of peers behave maliciously

stakeholder has the right not to vote in order to elect witnesses, as a result, the network becomes less secure and trustless. More peers vote, more secure and trustful the network is. Furthermore, the stakeholders who have the largest amount of stakes have the possibility to cooperate with other rich participants voting each other in order to control the network. As a result, this could lead to the centralization.

3.3.1.4 Leased Proof of Stake (LPoS)

The LPoS was invented to overcome some issues of the Proof of Stake (PoS) [22]. In PoS, the participants are chosen to be a forger based on their stake that hold. This means that, only the peers that hold a huge amount of stakes have the privilege to participate and not the weaker ones that have no opportunity to be chosen. The LPoS solves that limitation by giving the right to the weaker to be active in the network. In that case, the peers can lease their stakes to richer peers and when they validate the blocks, both of them take advantage of the reward.

The participants in LPoS can be divided in two categories: those who run a full node in order to be forger and that ones that leasing their stakes to the full node. The selection of the forger is executed like in the case of Proof of Stake. The higher the stake that a full node has, the more probability to be chosen in order to validate the transactions and create the next block. Once a full node has been chosen and the block has been inserted in the chain, then the reward of the transaction fee will be given to the full node. In addition, leaser will receive a percentage of the transaction fee. For instance, if a full node has 100000 coins and leaser leases 1000 coins to the full node, then the reward of leaser will be 1% of the transaction fees.

The main advantages of this algorithm is that all the participants in the network are active members making the network more secure. In addition, the leasing is safe as the stakes that are going to be leased are always locked in the owners wallet. Furthermore, each leaser has the right to cancel the leasing operation every time he wishes.

Despite these benefits, centralization is still concerned as a few members can control the whole network.

3.3.2 Vote-Based consensus algorithm

3.3.2.1 Practical Byzantine Fault Tolerance (PBFT)

In 1999, Miquel Castro and Barbara Liskov introduced the algorithm "Practical Byzantine Fault Tolerance" (PBFT)[7], which was the first practical solution against the problem of the Byzantine generals. It should be clarified at this point that the problem of the Byzantine generals is presented when different nodes in an unreliable network must reach a final decision, checking the data exchanged. Especially, the PBFT algorithm was regarded as the first practical high performance consensus algorithm that is suitable for use in asynchronous networks like the Internet.

PBFT consensus algorithm is used in private networks, where the number of peers is limited in comparison with public networks, which consist of millions of users. Furthermore, in permissioned network, the participants are known each other and they are given the permission by the administrator of the network to participate to that. This algorithm is based on state-machine replication and voting mechanism by exchanging a lot of messages among the peers in order to reach the consensus in a proper way. In addition, the algorithm can work properly if there is an equal or smaller number of (R-1)/3 faulty nodes in the network, where R is the total number of nodes. Especially, the network consisted of 7 nodes can tolerate up to two faulty nodes.

In a PBFT system, the participants are divided into two categories, the primary (or leader) node and the secondary (or backup) nodes. The primary is responsible for the validation and order of the transactions inside the block as the insertion of the block in the chain. Every node in the network can act as a primary node. After a specific period of time, the new primary replaces the old one through the operation of view change mode. In addition, the view change can happen when the primary node behaves maliciously or stop working. For instance, in Hyperledger Sawtooth [23], network switches to a new primary after the appendage of 100 blocks to the chain. Moreover, the backup nodes exchange messages each other to verify the block and to check if the primary node behaves honestly.

The procedure of committing the block in the chain in PBFT algorithm is divided in three phases:

- Pre-prepare
- Prepare
- Commit

Firstly, as described to the previous consensus algorithms, each node in a PBFT system has a pair of keys, the private (secret) and the public key. Not only does this algorithm make use of the digital signature, but also the Message Authentication Codes (MACs) [8] in attempts to authenticate all of the messages. Digital signature is used to authenticate only the messages that correspond to view-change and a new-view and the MAC is used to the rest of the messages making the system faster.

The MAC algorithm makes use of symmetric encryption where two peers A and B can verify the communication based on a share secret key in both directions. Specifically, the two peers make use of a couple of session keys k_{ab} and K_{ba} where the first one is used for the calculation of the MAC for a message that sends from peer A to B and the second key

for the reverse operation. The MAC of each message is computed by creating the MD5 of the concatenation of the share secret key with the corresponding message. Then the peer A sends both the MAC and the message to the peer B and peer B in turn, calculates the MAC of the message that received form peer A with the share secret key. In the figure 3.13 is depicted the previous procedure. If the two MACs are the same then the peer B authenticates the message and vice versa. In addition, in PBFT algorithm, only the last 10 out of 16 bytes of the MD5 are used in order to reduce the size of the MAC and make the procedure ever faster.

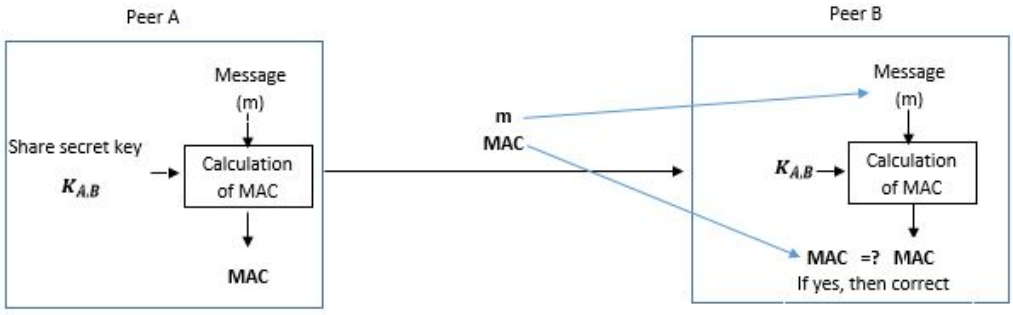

Figure 3.13: Procedure of message authentication code

However, through this technique a third peer cannot verify the authenticity of the message. This is one of the main reasons why digital signatures provide more security than the MAC. Despite this limitation, the developers of PBFT algorithm make use of the MAC by improving this algorithm. In case of multiple recipients, they use the authenticators instead of the digital signatures which is the vector of MAC and are symbolized as $\langle m \rangle \alpha_i$ where m is the message and a is the signature of the authenticator i. Messages like client request, pre-prepare, prepare commit etc. are signed by the authenticators but this doesnt happen in case of view change and reply messages. In the presence of one recipient, the digital signature is replaced by the MAC and it appears like $\langle m \rangle \mu_{AB}$ where peer A sends the message and the MAC to peer B. A reply message is an example where only one recipient appears.

According to the PBFT procedure, a node c sends a request to execute a transaction by sending a message m to the primary node. Every message m signed by the sender with its private key c includes the timestamp t which is the time of message creation, the c which is the name of the node and o which is the implementation of the state machine operation. This message has the form of $\langle REQUEST, o, t, c \rangle \alpha_c$.

Afterwards, the three phases of PBFT algorithm are executed. Once the primary node has received the request message m, creates a sequence number n of this message and then it sends a pre-prepare message accompanied by the request message m to all the backup nodes. The sequence number determines in which block the message belongs to. The pre-prepare message forms as $\langle \langle PRE-PREPARE, v, n, d \rangle \alpha_p, m \rangle$ where v is the current view of the message creation, n is the sequence number of the request, d is the hash of the request message m and α_p is the signature of the authenticator primary node. Notice that the request message m is not directly included within the pre-prepare message, but appended at the end. The objective is keeping the size of this message small. All the backup nodes store the message to their logs if:

- The id of the request and the pre-prepare messages are valid and if the hash of m is the same as the d.

- The view v of the pre-prepare message is the same as the current view of the network
- There is no other pre-prepare message in the logs of the nodes with different hash of the request message and the same sequence number and view.
- The sequence number of the message should be between the low water mark h and the high water mark H. Through this technique, malicious nodes are prevented from limiting the space of sequence number by choosing a number near to H. The h is equal to the sequence number n of the last stable checkpoint and H=h+k where k is a huge number in order to create a big space between h and H.

If the above criteria are fulfilled, then the backup nodes store the pre-prepare and request message on their logs and they move to the next phase where all backup nodes send a prepare message to all the nodes of the system. The Prepare messages have the form of $\langle PREPARE, v, n, d, i\rangle\alpha_i$, where i corresponds to each node that sends the message. Once the nodes have received the prepare messages, they check the correctness of signature of the prepare messages as their view is equal to nodes current view and the sequence number is between the h and H. If the previous conditions meet the requirements then, each node stores the message to its logs.

In order to proceed to the next phase, nodes have to ensure that they stored the request, the pre-prepare as the prepare messages in their logs. They have, also, to verify that the pre-prepare messages match to the prepare messages. If, specifically, the view, the sequence number and the digest of pre-prepare are equal to the contents corresponding to the prepare messages. Finally, if each node has, also, received 2f prepare messages from different backups, they move to the final phase.

In the commit phase, all the nodes, including the primary node send a $\langle COMMIT, v$, $n, D(m), i\rangle\alpha_i$, message to all the nodes in the network. After receiving the commit messages, they check the validity of these messages in order to store them in their logs like in the prepare phase. Afterwards, once nodes have received 2f+1 commit messages that match to the pre-prepare messages such as the view, the sequence number and the digest, then nodes send directly a reply message $\langle REPLY, v, t, c, i, r\rangle\mu_{i,c}$ to the node, where t is the timestamp of the corresponding request and r is the final result of that request. If node receives f+1 reply messages from different nodes with the same timestamp and result, then it accepts this result. Otherwise, it sends the request to all the nodes of the system and if the rest of the nodes has already executed the procedure of three phases then they send only the reply message. The figure 3.14 depicts the procedure of PBFT algorithm.

Besides the operation of the three phases, there is, also, the view change mode, which plays a significant role on PBFT algorithm. This mode occurs when there is a faulty behaviour of nodes or abnormal condition in the network and when the mandatory change of the primary happens after a specific interval of time (100 blocks in Hyperledger Sawtooth). When a primary node receives a request m from node c, then all the backup nodes start a timer to check that all the nodes will receive the corresponding message in a given time, which is determined by the system. On the other hand, unless they have received these messages before the timer expires, then they send a view-change message. Once the view-change procedure has started, all the nodes accept only the view-change, new-view and checkpoint message.

The view-change message has a form of $\langle VIEW - CHANGE, v + 1, n, C, P, i\rangle\sigma_i$, where v+1 indicates that the primary at view v is faulty and it has to be moved to the

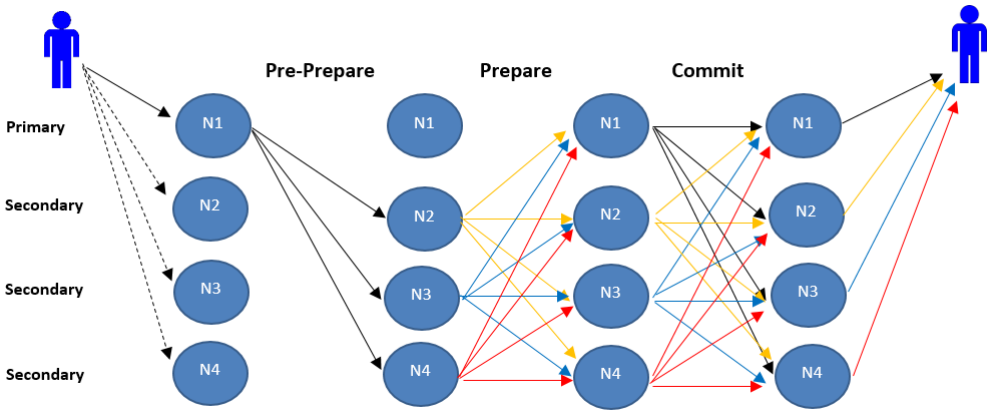

Figure 3.14: Procedure of PBFT algorithm

next view v+1 with the primary that accounts for that view. Furthermore, n is a sequence number of s, which is the latest stable checkpoint and C is a set of 2f +1 messages proving that the checkpoint s is stable. In addition, P is a set of the Pm messages where the sequence number of them is higher of the sequence number n of the stable checkpoint s. Pm includes all the valid pre-prepare messages and 2f prepare messages that match to the pre-prepare messages. When the new primary of v+1 view receives 2f valid view-change messages from different nodes send a new-view message to all nodes and starts the normal operation.

PBFT uses another technique to reduce a huge amount of messages that can be stored in the logs of each node. This method is called garbage collection. After an interval of time determined by the network, nodes create checkpoints by broadcasting a message $\langle CHECKPOINT, n, d, i \rangle \alpha_i$ to other nodes. The sequence number n indicates the latest message request m that executed in the system and d is the digest of the corresponding phase. When each node receives 2f +1 messages from different nodes with same both sequence number n and digest d, the checkpoint becomes stable. As a result, all the checkpoints and the messages that have lower or equal sequence number with n are deleted from the logs.

3.3.2.1.1 PBFT in Hyperleder Sawtooth

As mentioned previously, Hyperledger Sawtooth, except from the Proof of Elapsed Time (PoET) makes use of PBFT consensus algorithm [24]. In Sawtooth, a primary node is responsible for creating a block as to insert the block in the chain. each block is inserted to the chain approximately every 1000 msec. Firstly, the primary node creates the block and sends a signed request to its validator. Then, the validator sends this request to all other validators of the network to check if the signer of this block is the real one and then they store this request to their PBFT logs by informing all the nodes with a Block new update message. After that, a primary node assigns a sequence number to the request and sends pre-prepare messages without the request message m to all the backup nodes like in the classical case of PBFT.

Then, the procedure is the same as the classic PBFT until the end of the commit phase. In the final step of the insertion of the block, instead of all nodes have to send a reply message to the request node, in Sawtooth each node informs its validator to commit the block in the chain by sending a Commit Block message. This is done only if nodes have received 2f+1 commit and 2f+1 prepared messages that match to the pre-prepare

messages such as the view, the sequence number and the digest. Furthermore, validator sends to its node a Block Commit update message if the block is inserted to the chain successfully. Then the primary node starts the procedure for the creation of a new block.

In addition, when a primary node sends a pre-prepare message, all backup nodes start an idle timer where a primary has to send the pre-prepare message to all nodes in less than 30 seconds. If the primary exceeds this time, then backup nodes send a view-change message. Moreover, in the prepare phase, all nodes start a commit timer where each node has 10 seconds to continue the procedure until informing the validator to commit the block in the chain. If the time expires without any result then a view-change message will be sent to the new primary.

Unlike the classic PBFT where there is a garbage collection and the checkpoint to reduce the contents of the logs, Sawtooth has the Log Pruning procedure. Each node in the network implements this procedure when the log size of each node exceed the 10000 messages. They delete all the messages where the sequence number are smaller than the last block sequence number that committed in the chain.

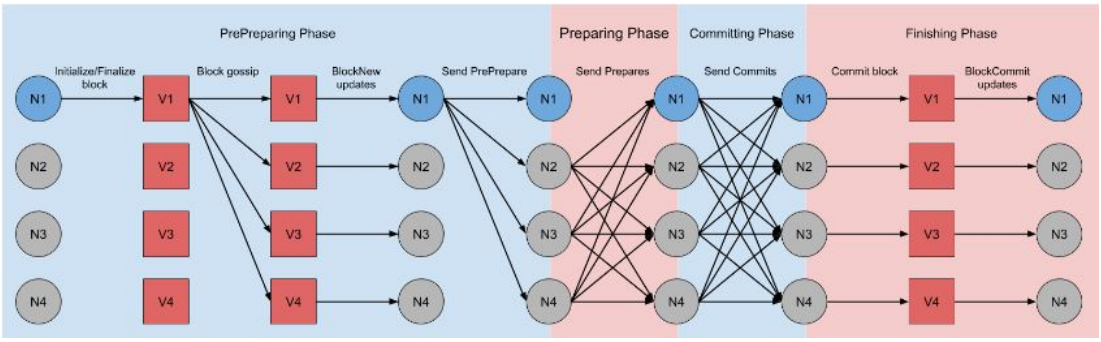

Figure 3.15: Procedure of PBFT algorithm in Hyperledger Sawtooth

The PBFT algorithm suffers from scalability when the number of participants in the network increases. All the participants in the network have an active role by exchanging a lot of messages each other in order to reach consensus. As a result, more peers in the network, more time is needed to get 2f +1 commit messages. Last but not least, the network suffers from the Sybil attack that occurs when a single entity can manipulate huge amount of nodes in order to control the majority of the network. This limitation can be avoided when the number of peers in the network increases.

3.3.2.2 Delegated Byzantine Fault Tolerance (DBFT)

Da HongFei and Erik Zhang introduced the DBFT algorithm in NEO blockchain. They, in essence, created the Antshares in 2014 and rebranded it as NEO in June of 2017[18]. The main purpose of DBFT algorithm is to overcome the limitation of the Practical Byzantine Fault Tolerance (PBFT) algorithm, which achieves consensus in networks with limited number of peers (private networks) in a faster manner. However, the DBFT was designed to reach consensus in public network just in few seconds.

This algorithm resolves the Byzantine Generals problem as it can work properly with some faulty nodes. The system can tolerate up to (n-1)/3 faulty nodes [18], where n represents a specific group of nodes (consensus nodes), unlike the PBFT where n corresponds to the all the participants of the network. In addition, NEO system creates each block

every approximately 15 seconds. Furthermore, the average number of transactions that can be committed is 1000 tps expecting to reach 10000 transactions per second which is excellent throughput in public networks.

The DBFT algorithm is implemented like the Delegated Proof Of stake (DPoS) algorithm based on voting procedure. All the participants in the network that hold NEO token are known as ordinary nodes. This kind of nodes is responsible for implementing transactions in the system and voting for consensus nodes in real time. The consensus nodes consist of the Speaker and the Delegates. The Speaker is responsible for picking the transaction from the memory pool, validating them and inserting them into the new block and the Delegates are in charge of doing the validation of the blocks through the voting process. Like in the PBFT, if the 2f +1 Delegates vote for a specific block, then this block gets validated and appended to the chain.

According to the operation of consensus nodes, the Speaker has a significant role, as it is responsible for creating the block as insert the validated transactions in the corresponding block. The speaker can validate up to 500 transactions with high priority and up to 20 transactions with low priority in each block. the priority is determined by the number of transaction fees that every peer pays. Transactions with fees equal or more than 0.001 GAS can be considered as high priority, otherwise transactions with lower GAS or zero fees can be considered as low priority. In addition, the size of each transaction should be up to 1024 Bytes. In the case that transaction is higher than 1024 Bytes, then the peers should pay 0.001 GAS obligatory plus 0.00001 GAS per extra Byte. For example, the fees for a transaction of 1034 can be calculated by TF = 0.001 + (10 * 0.00001) Bytes Every time that a block is generated, a new Speaker is chosen based on the formula p = (h-v) mod n where h is the height number of the block, v is the view number counting from zero and usually increases by one until the block appends to the chain. Finally, n is the number of consensus nodes. This number is determined and calculated by the network through the voting procedure and it depends on the number of peers in the network. The minimum number is 7 and the maximum will be 1024 consensus nodes. Nowadays, the number of consensus nodes fluctuates from 7 to 13 [17].

In the NEO network, there are two tokens, NEO and GAS token. NEO is the currency where all the users use it to make transactions. In addition, all the peers that hold NEO tokens can participate to the management of the network. Especially, each peer can vote the node that trusts in order to be the consensus node for the next block. The total number of NEO is 100 million which was created in the beginning of the network. The GAS token is produced by the generation of a new block. Every peer that holds NEO token will be able to receive an amount of GAS proportional to the NEO tokens that peers hold. For instance, a peer that holds only one NEO token will receive 0.0003 - 0.0004 GAS per day while other peers with more NEO tokens will receive bigger amount of GAS. It is estimated that the GAS is going to be stopped generating when it reaches the amount of 100 million around 2039. Upon creating a block, 8 GAS are generated every time. The founder of NEO introduced an algorithm where the GAS is decreased by one every two million blocks in order to achieve the desired amount in twenty years [16].

In the NEO network, every peer has the right to participate in the consensus procedure. The only requirement is the payment of 1000 GAS and the validated digital identity. All the peers that meet the requirements are located in a list and all the ordinary nodes can vote for the candidates through their public key. One NEO token corresponds to one vote. If a specific node votes for multiple candidates then each candidate will get the same number of votes as the amount of NEO tokens that the voter holds. For instance, if a

voter holds 50 NEO tokens and wants to vote for five candidates, then each candidate will receive 50 votes. Furthermore, if the amount of voters token is decreased after the voting, then the number of votes of the corresponding candidates will be decreased too.

In DBFT algorithm, the voting procedure happens in real time. Each time that a new block is created, the NEO network calculates the number of consensus nodes (n) that they will participate in the validation of the blocks and the speaker proposes which candidates are going to be consensus nodes for the next consensus round. The procedure of the candidates selection is the following:

- All the candidates are arranged in ascending order according to their votes.
- The network discards the first and last 25% of the entire list of candidates.
- The percentage of candidates that are left in a combination with the votes and the amount of NEO token that peers hold can be sorted.
- The top n candidates will constitute the consensus nodes.

Once a transaction is happened, all the consensus nodes after the voting procedure receive this transaction, store the data of this transaction in its memory and send this transaction to the whole network with the signature of the sender. When the consensus procedure starts, the view is set to zero and a new speaker is chosen through the formula p= (h-v) mod n. After that, speaker wait for a time t, which corresponds to 15 seconds. During this time, the speaker collects the transactions from its memory, validate them and insert them into the block. Once the time expires, speaker sends a Prepare message to all the consensus nodes. This message has a form of $\langle PrepareRequest, h, v, p, block, \langle block \rangle \sigma_p \rangle$ where h is the block height number, v is the view number starting always from zero, p is the name of the speaker, the current block and the block signed by the speaker.

Then, Delegates receive the Prepare message and check if the values are valid. For example, they confirm if the view and the height block number of the prepared message are equal to the current view and the height block number corresponding to that view. In addition, they check if the transaction has not already existed in the blockchain and if the transaction is not used twice. Afterwards, Delegates broadcast a Prepare Response message $\langle PrepareResponse, h, v, i, \langle block \rangle \sigma_i \rangle$ where i is the number of Delegate node. Otherwise, they broadcast a change view message in attempts to change the Speaker. Finally, if there are 2f +1 $\langle block \rangle \sigma_i$ messages with the same view and height block number, then the block is appended to the chain and published to the network. The CommitSent state is responsible for ensuring that each node agrees to the specific block and view change state doesnt exist. Then, Delegates nodes delete the transactions inserted in the block, reset the view and start the new round of consensus.

Except for the above conditions of view change, it also happens when a consensus node behaves maliciously by sending erroneous information or suddenly stop working. In addition, the view change can occur when the nodes do not reach consensus after a specific time which is defined as t*exp(v+1) where v is the current view and t is the time of 15 seconds. So, consensus nodes send $\langle ChangeView, h, v, i, vk \rangle$, where vk= v + k with k=1. If each consensus node receives 2f +1 view change messages with the same values in less than t*exp(v+1) time, then the consensus procedure begins again by selecting a new Speaker. Otherwise, the k will increase sending again change view messages until the operation of view change is completed[15].

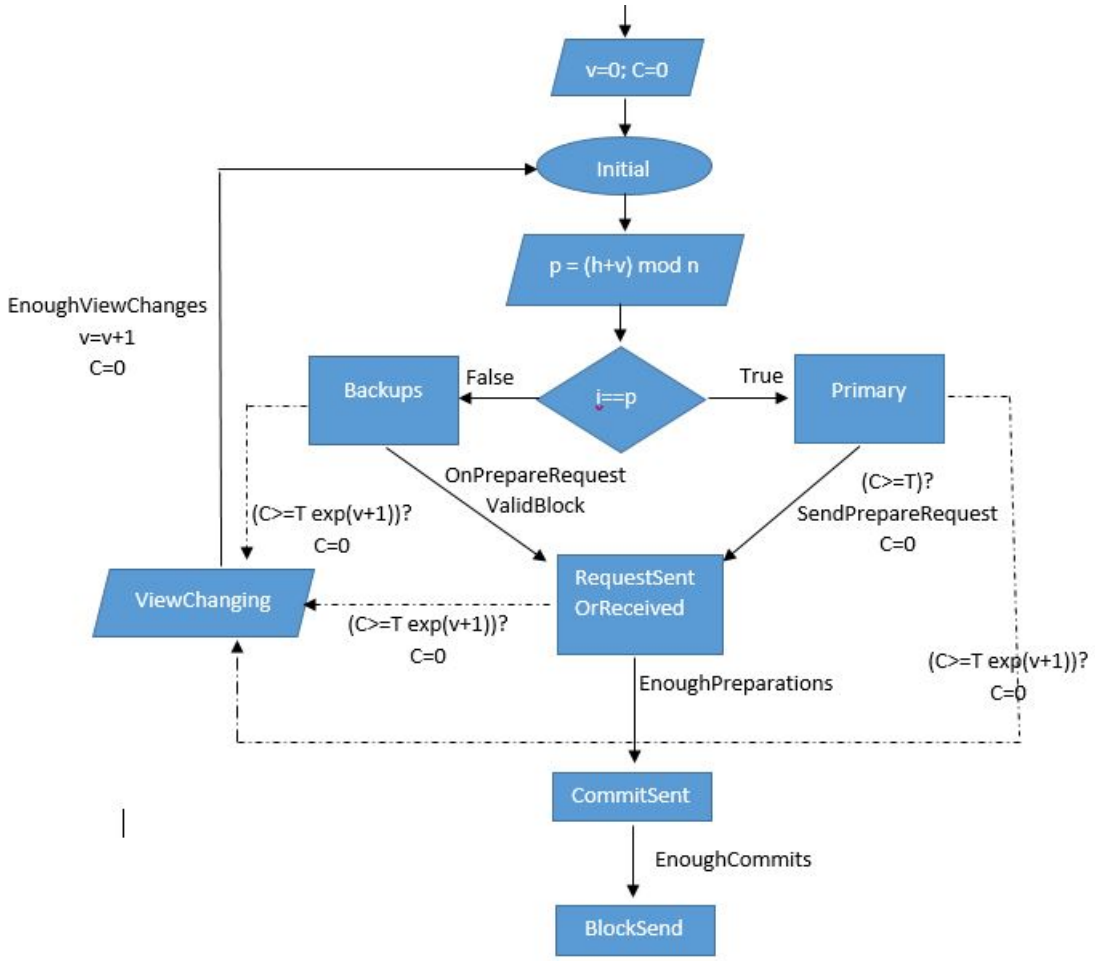

Figure 3.16: Procedure of DBFT algorithm

As depicted in the figure 3.16, if more than 1/3 of nodes fails or behaves maliciously then the system will never reach consensus. This problem will lead in an infinite loop changing only the view change state. In DBFT v2.0 developers solved this problem by introducing the Recover/Regeneration mode. Through this mode, the failed nodes that lost the communication due to some reasons can be updated by the last state of the consensus round in order to be able to continue to the participation in the consensus procedure. The figure 3.17 reprinted from [12] depicts the whole procedure of the DBFT consensus algorithm with the recovery mode.

All the events that take part in the consensus are stored in a local database. When a failed node re-establishes the communication, a change view message is sent to 0 [12]. Through this procedure, the failed node informs the rest of the consensus nodes which returns and it is able to participate in the consensus. Then the other consensus nodes inform the failed node by sending a payload and the failed node is updated and it can take part in the procedure by confirming the block as to commit the block in the chain. The failed node will receive payload only for equal or smaller number of f nodes where f is the byzantine failure nodes that the system can tolerate in order to work properly.

In the case of malicious nodes, the system automatically will remove this malicious node by voting it out or following another technique which could be explained in next chapter. Usually this kind of nodes is responsible for the delay of the messages and for sending messages where the payload of them are different from the payload of other

consensus nodes.

The DBFT algorithm presences some basic limitations compared to other consensus algorithms. Although the anonymity plays a significant role on public network, in DBFT anonymity cannot be achieved, as the candidates have to declare their real identities in order to be elected as consensus nodes. In addition, the fact that the whole network is controlled by a small portion of peers known as consensus nodes leads to the centralization. Nevertheless, the founders of NEO support that this is not a limitation for the system on the grounds that consensus nodes are going to exceed 1000 next years. Another drawback is the speed limit of the new block creation. As previously referred, 15 - 20 seconds are needed for each block creation, while BitShares needs only 2 seconds using the DPoS consensus algorithm.

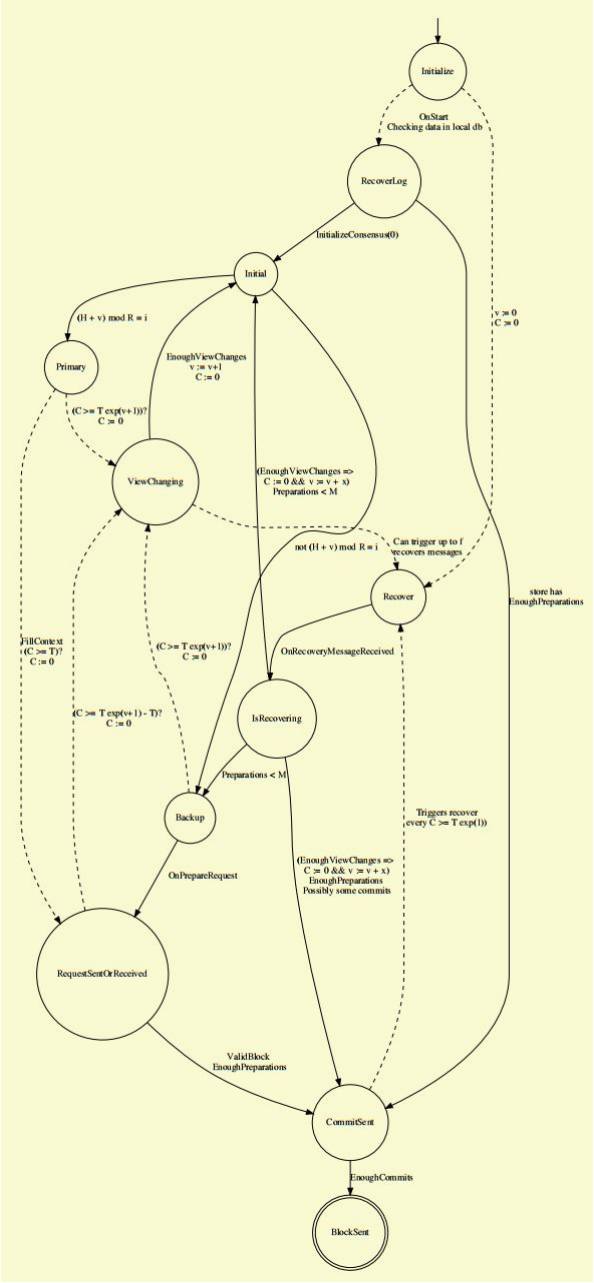

Figure 3.17: Procedure of recover mode in DBFT algorithm ([12])

3.3.3 Summary of consensus algorithms

The table 3.1 represents the main features of all the consensus algorithms that explained in this thesis.

	PoW Bitcoin	PoS NXT	DPoS BitShares	PBFT H. Sawtooth	DBFT NEO
Type of network	Public	Public	Public	Private	Public
Validating procedure	Mining, 10 min.	Forging, 60 seconds	Based on votes , 2seconds	Based on votes, 1 second	Based on votes, 15 seconds
Transactions per second	7	100	3300	Unknown	1000
Block size	1 MB	42 KB (up to 255 transactions)	Configurable by chain parameters	Unknown	Up to 520 transactions
Power consum.	High	Partial	Low	Low	Low
Finality	6 blocks 1 hour	10 blocks 14 minutes	1 block	1 block	1 block 15 seconds
Limitations	51% attack, High cost	Nothing at stake, rich become richer	Centralized,	Low scalability in high number of peers	Anonymity, Centralized

Table 3.1: Summary of the Consensus protocols

The PoW consensus algorithm is used in public blockchains and it has been popular though the founder of Bitcoin Satoshi Nakamoto which made use of this algorithm in order to assure the trust and the security into the whole network. This procedure is known as mining where each peer (miner) validates the transactions that happen in the network, inserts the transactions and the hash of the previous block in a new block and tries to solve a difficult mathematical cryptographic puzzle in order to be able to insert this block in the chain. This can be achieved by modifying the nonce, which is appeared in the block header by calculating the hash of the whole block in order to be smaller or equal to the difficult target. The first miner, which solved the puzzle, publishes the result in the whole network and then appends the block in the chain. A new block is generated every 10 minutes, this block is finalized after 6 blocks (1 hour) confirmations and the miner will get the reward in bitcoins as the transaction fees of the corresponding block. At this moment, this reward corresponds to 12.5 bitcoins but it will be decreased every 210000 blocks and the miners will stop to get this reward when the total amount of Bitcoins reaches the 21 million. On the grounds that the size of the block in Bitcoin is only 1MB, the miners usually choose the transactions with the highest fees in order to validate and insert them in the block. Through this technique, the PoW eliminates the presence of DDoS attack where the attacker can flood a multiple fake transactions in the network by delaying the validation of the real transaction.

One of the main limitations of PoW is the 51% attack, where more than the half peers can have the control of the whole network. This can be achieved through the mining pools where the peers cooperate in order to find the result of the puzzle. As a result, through this attack the attackers can control the validation of the transactions, implement double spend attack as to modify the contents of the blocks. Furthermore, in the PoW algorithm, the peers consume huge amount of power and need expensive equipment in order to solve the mathematical puzzle. In addition, PoW can only validate 7 transaction per second which is too low throughput compared to other consensus algorithms.

The PoS algorithm was invented to overcome some of the main limitations of PoW such as the huge power consumption, the low throughput and the time of the generation of new block. In this algorithm, the peers (forgers) do not have to implement the mining

operation in order to validate the transactions and insert the block in the chain. In PoS the forging procedure is operated in a random way, based on the amount of coins that each peer holds. The peers with the bigger amount of stake has higher probability to be selected as a forger. In Nxt, every block is generated at about 60 seconds. The size of the block is 42 KB and each forger can insert up to 255 transactions in the block. The reward of the forger is the fees of the transactions which inserted in the block. Furthermore, the Nxt can validate up to 100 transactions per second and can reach finality after ten confirmed blocks.

The main drawbacks of the PoS is the nothing at stake attack and the unfair selection of the forgers. In the nothing at stake attack when a fork is created usually due to malicious behavior, the forgers continue to forge in both chains, as they do not have nothing to lose. Through this attack, an attacker can implement the double spend attack where he can spend the same amount of money twice, one in each chain. In the first chain the attacker can implement a transaction in order to buy a product and in the second one he can send the same amount of money in a different account which is under his control. When the attacker receives the product, he will stop to forge in both chains. He will continue to forge only in the chain in which he sent the same amount of money in a different account. As a result, the longest chain will win and the transactions of the first chain will be discarded as a result the attacker will have both products and money in his position. The second drawback of PoS is that only the richest peers in the network have the chance to be forger since the selection of the forger is based on the stake that they have.

Dan Larimer tried to overcome the limitations of PoW and PoS by developing a new consensus algorithm known as DPoS. In this algorithm, all the peers in the network that hold tokens (stakeholders) can have an active role through the voting procedure. The stakeholders are responsible to vote for witnesses and delegates. Witnesses validate the transactions and insert the block in the chain. The insertion of the block is done in rounds, every 2 seconds and each witness can validate only one block in each round in a specific timeslot. Delegates are responsible to propose changes for better maintenance of the network such as block size, block interval etc. The DPoS has high throughput compared to PoW and PoS and this is 3300 transactions per second. Furthermore, due to the specific timeslot of validation and the shuffle of witnesses in each round, the nothing at stake attack is impossible in DPoS.

Despite of all the advantages, the DPoS is vulnerable from their stakeholders. First of all, in case of stakeholders inactivity by not participating in the voting procedure, the system becomes less secure and trustless. Furthermore, as the votes depend on stakeholders stakes, the rich stakeholders can cooperate in order to vote each other having the control of the network. Finally, both cases lead the network to be centralized.

The PBFT algorithm is the first algorithm which solves the byzantine general problem. It is used in private network where the number of peers is limited and the peers are known each other. In this algorithm, the peers have different rights and the system administrator defines these rights for each peer. The system can work properly having f faulty nodes. f=(R-1)/3 where R is the number of peers in the network. The PBFT protocol can reach consensus by exchanging messages between the peers. In every consensus round, the primary node validates the transactions, inserts them in the block and sends a pre-prepare message to the rest of the nodes known as backup nodes. Then, backup nodes check the validity of the transactions and the block and then they send a prepare message to all the nodes including theirselves and the primary node. Each node in the network checks that it received 2f correct prepare messages from different backups and then it proceeds

in the commit state. In that phase, all the nodes including the primary node sends a commit message to theirselves and to the rest peers of the network. If each node picks equal or more than 2f+1 correct commit messages, then the block is finalized and inserted in the chain. Due to the small number of peer's participation, the PBFT algorithm in Hyperledger Sawtooth has high performance and can generate block every one second. Although this is very important feature compared to other consensus algorithms, it could be also catastrophic as the attacker can implement a Sybil attack. In addition, due to the exchanges of the messages by increasing the peers in the network decreases the performance of the PBFT algorithm.

The DBFT algorithm in NEO blockchain was developed to overcome the limitations of PBFT protocol. This algorithm is used in public networks and is implemented like the DPoS algorithm based on voting procedure in real time. All the peers that hold tokens in the network known as ordinary nodes are responsible to make transactions as to vote in or out for the consensus nodes. The consensus nodes are divided in two categories, the speaker and the delegates. The first one is responsible to validate up to 520 transaction as to insert them in the block and the second ones are responsible to check the correctness of the block. If there are at least 2f+1 correct messages from the speaker and the delegates then the block is finalized and inserted to the chain. The DBFT algorithm can have from 7 up to 1024 consensus nodes. The system can work properly with up to f=(n-1)/3 faulty nodes. The n is the number of consensus nodes and not the number of all the peers like in the PBFT. In the NEO blockchain, the block is generated almost every 15 seconds and it can validate up to 1000 transaction per second. Every node in the system can be a consensus node only if it pays 1000 GAS and it declares its digital identity.

The main drawbacks of this algorithm are the speed limit, the anonymity and the centralization. Compared to other consensus algorithms that are used in public networks, the DBFT is too slow in terms of block generation. For example, the DPoS and the FBA algorithm can produce new blocks in 2 and 5 seconds respectively. In addition, the anonymity plays a significant role in the public network but in DBFT this cannot be achieved as the consensus nodes should declare their identity in order to be voted by the other nodes. Last but not least, the number of consensus nodes which take part in the consensus procedure is fluctuated from 7 to 13 consensus nodes at this moment. This small number of consensus nodes which control the whole network leads them to centralization.

4 Improvement of DBFT algorithm

4.1 Introduction

In the NEO DBFT algorithm, the voting procedure can happen in real time. Therefore, when a node suspects that any consensus node fails or behaves maliciously can vote out for this specific node by avoiding it to participate in the consensus procedure. However, this does not mean that any node will vote out these consensus nodes whenever they misbehave. As a result, the system needs a mechanism to delete or remove these misbehaved nodes without the need of the voting procedure. The purpose of this chapter is the development of this mechanism as a reputation mechanism.

The DBFT algorithm can work properly with up to f faulty nodes. There are the ordinary nodes and the consensus nodes. The first ones can implement transactions as they can vote in or out for consensus nodes in real time. The consensus nodes are divided in two categories the speaker and the backups nodes. All the consensus nodes take part in the consensus procedure in order to append the block in the ledger. In the beginning of each consensus round the view v is equal to zero v=0 and the timer c is set also to zero C=0 and correspond to the step 1 as depicted in the figure 4.1. Then the speaker is selected through the formula p = (h+v) mod n where the step 2 includes this operation. Furthermore, the speaker has 15 seconds to prepare a block by validating the transactions and inserting them in the block. When this timer expires, the speaker sends a prepare request message to all the backups nodes with the current block, the block signed by them, the view and etc (step 3). Then, all the backups nodes which received the prepare request message form the speaker check the validity of this message and if it is correct, they send a prepare response message (step 4). If there are at least 2f+1 prepare response and request messages in the appropriate time of 40 seconds (step 5), then the system moves to the next phase which is the commit state (step 6). This state is the last step before the insertion of the block in the chain (step 7). Through this state, the system ensures that a consensus node signed a block with a specific height block number only once by avoiding the forks.

In the case where 2f+1 correct messages are not enough or the timer expires after 40 seconds on the grounds that the speaker or the backups nodes did not send the appropriate message, the system moves to the view change phase (step 8 in figure 4.1). This state is depicted clearly in, the figure 4.2. The consensus nodes send $\langle ChangeView, h, v, i, vk \rangle$, where vk= v + k with k=1 and v=0 (step A). If each consensus node receives 2f+1 view change messages with the same values in less than t*exp(v+1) time, then the consensus procedure begins again by setting v=vk and selecting a new Speaker. Otherwise, the k increases by one and the consensus nodes send again change view messages until the operation of view change is completed. When the view change is completed the system begin a new consensus round with view equal to the last vk value. As the k increases, the overall time of the system timers increases too.

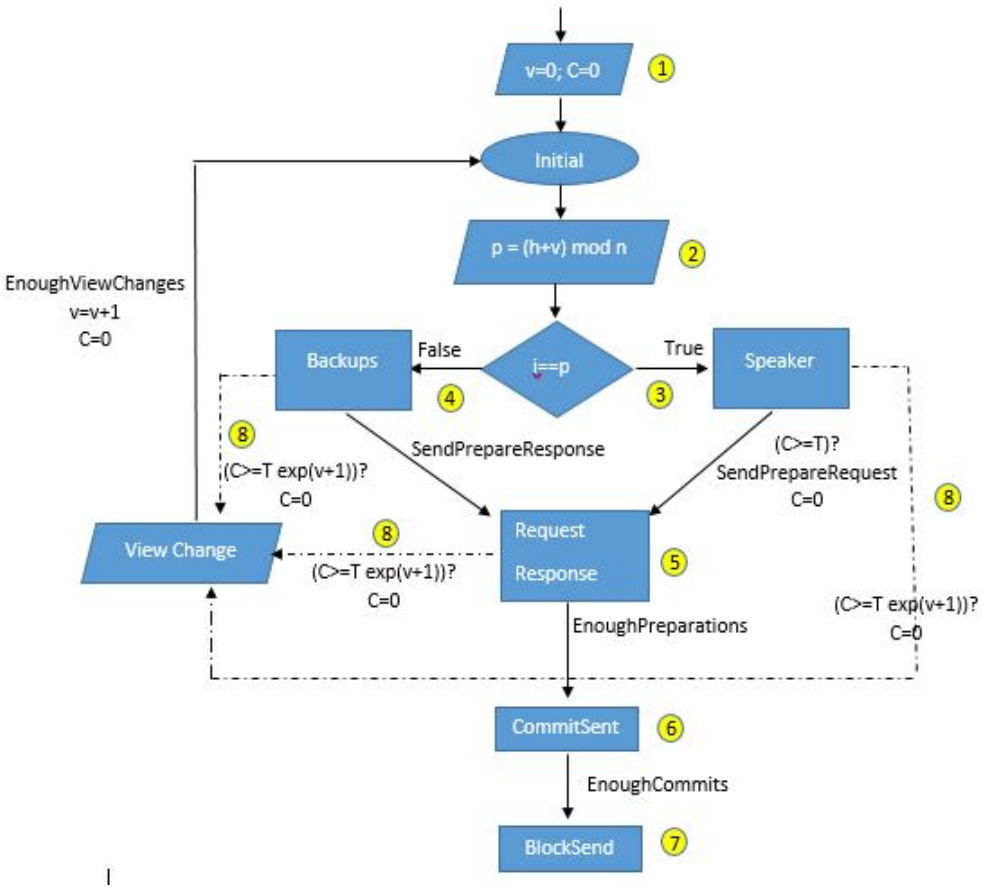

Figure 4.1: Procedure of DBFT algorithm

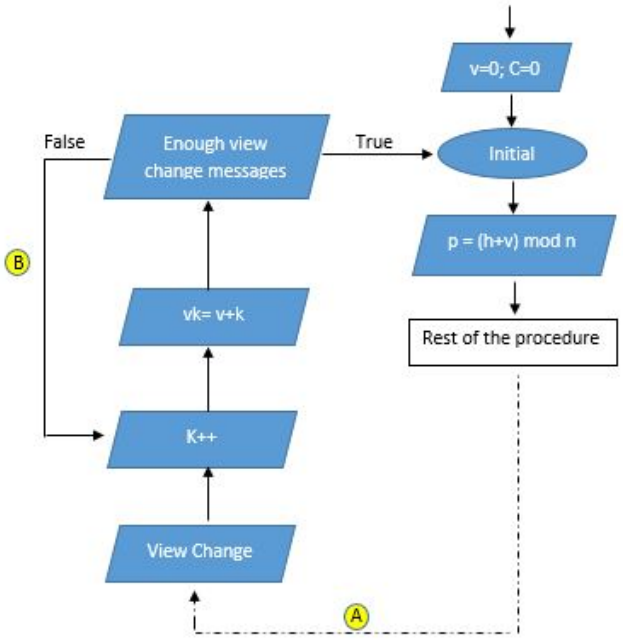

Figure 4.2: Procedure of DBFT algorithm in View change phase

Furthermore, The developers of the DBFT algorithm improved the safety of this algorithm by introducing the recovery mode. Through this mode, a failed consensus node due to hardware failure or loss of messages that takes part in the consensus rounds can be updated about all the information that needs when reestablishes the communication in order to be able to take part again in the consensus procedure. When a failed node reestablishes the communication sends a View change message to zero in order to inform the other consensus nodes that want to be updated. After that, f nodes where f is the number of Byzantine failure nodes can inform the failed nodes by sending to them all the necessary information in order to be able to take part in the consensus procedure or the view change phase.

For instance, the figure 4.3 depicts a consensus round which consists of 10 consensus nodes. As a result the system can tolerate up to f=3 faulty nodes. During the consensus procedure, the node 2 goes down and stops to take part in the procedure. After some time or some consensus rounds, the node 2 reestablishes the communication but it does not know the appropriate information in order to be able to take part again in the procedure. As a result, it sends a view change message to 0 in order the other consensus nodes to understand that node 2 is back and wants to be informed about all the necessary data in order to be able to take part in the consensus procedure. Then, node 2 can be informed by up to f nodes f=3 where f is the number of faulty nodes that the system can tolerate. For example, from figure 4.3, the honestly consensus nodes 3, 7 and 9 will inform the node 2 by sending the appropriate information to them i.e. the current view number, the height number of the block, the speaker on this consensus round and etc. In addition, in our proposal, the recovery procedure of the original DBFT algorithm will be used to inform all the new consensus nodes that have replaced the misbehaved nodes about the current state of the consensus round.

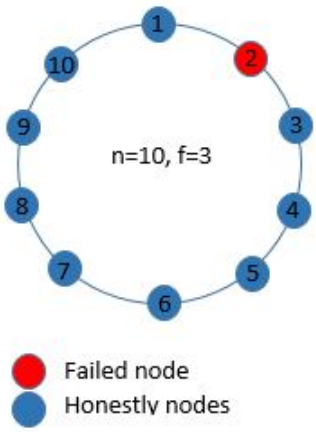

Figure 4.3: Example of recovering data

4.2 Categories of misbehaved nodes

In our proposal there will be two different categories of misbehaved nodes. According to the category, the system will act in a different way by removing or deleting these nodes. The first one category refers to the failed nodes and the second one to the malicious nodes. The failed nodes are the consensus nodes that do not send messages in the consensus rounds due to internet connection, crash of the system or any delay of the messages. Malicious nodes are those which send erroneous messages in attempts to make the consensus procedure

delay. In addition, the malicious nodes cannot create a fork for example to implement a double spend attack due to the reason of the commit state in the system.

The original DBFT algorithm has a timer in every state of the consensus procedure. If these timers expire without the system reaching agreement, then the system moves to the View change state. This usually happens when there are less than 2f+1 correct messages. First of all, the honestly consensus nodes should recognise if the nodes behave maliciously or failed as there is not central authority implementing that. This can be done through the exchange of the messages as all the messages traverse in the whole network making more difficult the possibility of the attackers to detect the location of the consensus nodes in the network. Ordinary nodes, also, can detect this behaviour through the of the messages and can vote out for these misbehave nodes. There should be different punishment in any case.

The punishment of the failed node will be more elastic than one of malicious nodes on the grounds that they do not account themselves for any failure. In this case, these nodes will be replaced by the next top consensus nodes that are located in the candidate list. This kind of nodes is going to lose only the votes that they had. In addition, they have the right to be a consensus node in the future, if they collect enough votes from other peers.

According to the malicious nodes, will be deleted and replaced by the next top available consensus nodes where are presented in the candidate list. The malicious nodes firstly will lose all the tokens that they have which are burnt by the system. In addition, they will be removed by the candidate list, lose the deposit that they paid in order to be candidate as losing the right to be a consensus node in the future. Furthermore, they will not never have the right to vote for any other consensus node. The only thing that they can do is to implement transactions in the network. In the end of the consensus round, the votes for all the candidates will be nullified because the votes of malicious nodes affect the overall voting procedure and a new voting procedure will start by informing all the peers that this specific node behaved maliciously and it loses all the rights.

4.3 Development of a mechanism to replace misbehaved nodes

As discussed in previous chapter, nowadays the number of consensus nodes fluctuates between 7 to 13 but the developers of DBFT support that the number of these nodes is going to be 1024 consensus nodes as the network is growing. The implementation of this mechanism will be based on n=10 consensus nodes. Therefore, the system can work properly with up to f=3 byzantine faulty nodes and this number is calculated through the following expression f = (n-1)/3.

The purpose of our algorithm is the automatically replacement of these misbehaved nodes after some unsuccessful consensus rounds in the case where the ordinary nodes did not vote out these nodes (normal procedure of DBFT). Furthermore, in the DBFT algorithm, when the system move to view change state, remains there, by implementing many view changes rounds until to get at least 2f+1 view change messages, as a result the delay of the system. So, our proposal is that after some unsuccessful view change rounds, the appropriate number of misbehaved nodes will be removed in order the system to work properly. Moreover, we will use the recovery mode which exists in the original

DBFT algorithm, to update the new consensus nodes that will replace the malicious or failed nodes in order to be able to get involved in the consensus rounds or in the view change states.

The design will be divided in three parts. The first part will consist of the failed nodes where it will analyze all the possible conditions in the presence of failed nodes while the second part will refer to how the system acts in case of malicious nodes. In the third part we will develop a reputation mechanism in order to measure the reputation of the consensus nodes.

4.3.1 Replacement of failed nodes

As discussed previously, a system with n=10 consensus nodes can tolerate up to f byzantine nodes which corresponds to three. In the case of rf=4 failed nodes where rf is the real number of misbehave nodes in the system, the system cannot reach agreement because there are less than 2f+1 correct messages with the same block, height of block and view. As a result, the system move to the view change state as depicted in the figure 4.4

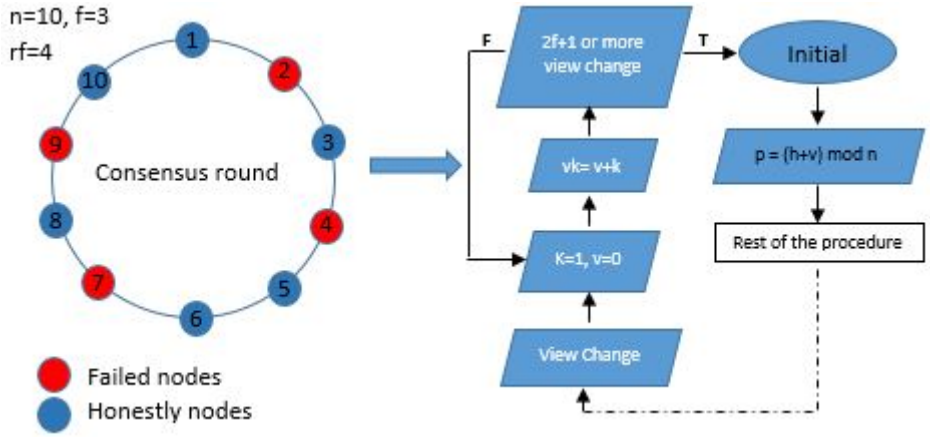

Figure 4.4: 1st view change round

In the first round of view change, the view v=0 since is the first time of the system that move in the view change procedure, the k=1 and then vk=1 as depicted in figure 4.4. If some of the failed nodes do not reestablish the communication, then the system will receive only 6 correct view change messages and when the timer expires after 40 seconds, the system will go to the second round of view change (See Figure 4.2). In this round the k will increase by 1 and will be equal to k=2, the view will be remained the same v=0 and the vk equal to vk=2. If there are not 2f+1 view change messages during the specific time of 40 seconds, the system will repeat the procedure for k=3 as depicted in figure 4.5

In the original DBFT algorithm this procedure would continued until at least 2f+1 correct view change messages by increasing, the overall time of the system. As a result in our proposal,in the third round of the view change procedure, the minimum number R of failed nodes with the lowest reputation will be replaced in order the system can work properly with the minimum cost. The number of replacement nodes R is calculated by the equation 4.1.

$$R = rf - f. \tag{4.1}$$

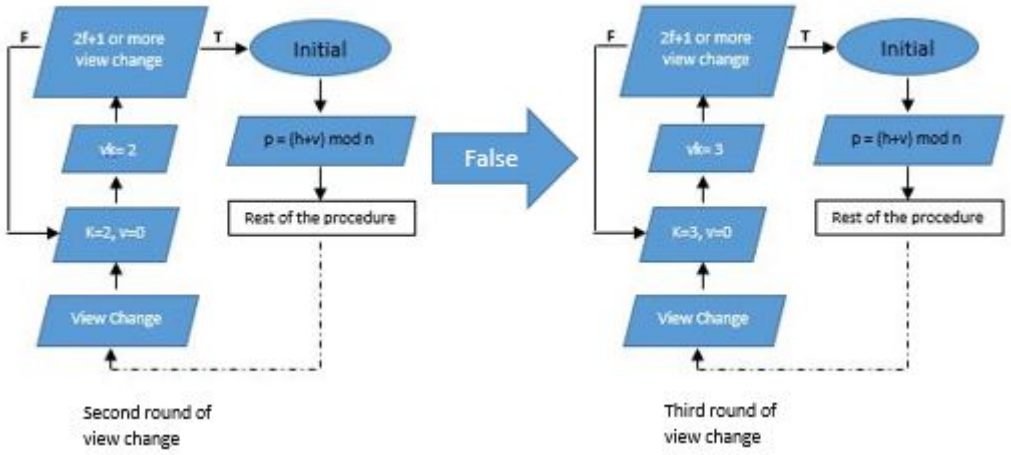

Figure 4.5: Summary of the rounds in the view change procedure

From the figure 4.4 the system has 4 real failed (rf) nodes and can tolerate only 3 faulty (f) nodes. So using the equation 4.1 should be removed the failed node with the lowest reputation. Assume that node 2 has the lowest reputation of the four failed nodes 2, 4, 7 and 9. This node will be replaced by the next top available consensus node where is presented in the candidate list. In addition, the new node will be updated about the state using the recovery mode of the original DBFT algorithm by sending a view change message to 0. Probably the view change state will complete with more than 2f+1 view change messages. Then the v will be equal with vk which is 3 and a new consensus round will start with view v=3.

Our proposal is the replacement of the failed nodes after two unsuccessful view change rounds which correspond to 1 min and 20 sec. This time was determined on the grounds that if the new speaker for view v=3 does not send a message or the system cannot reach consensus due to failed or malicious nodes (only 6 response/request messages) then the system will have to wait 14 minutes to insert in the new view change state where requires much time. The time of 14 minutes was calculated by the expression 4.2 where v=3, and t=15 seconds determined by the original DBFT algorithm.

$$t * exp^{(v+1)} = 15sec * exp^{(3+1)} = 819sec \approx 14min \tag{4.2}$$

As the consensus nodes inserts a new block to the chain after 15 or 20 seconds then the system will not insert more than 42 blocks in the duration of these 14 minutes. Therefore, all the transactions during this time will not be implemented as a result this cost to the system in terms of money and the reliability of the system. For example, if these failed nodes would have been changed in the fourth view change round, then the overall waiting time of the system would be 37 minutes.

Once the R failed nodes with the lowest reputation will be replaced, then the system will work properly but keeping some failed nodes. These nodes with high reputation will have the chance to reestablish the communication and take part again in the consensus procedure. In addition, as happens in the original DBFT algorithm, the ordinary nodes can vote out for the failed nodes which keep being consensus nodes without any activity in the rounds. If this does not happen, our proposal is that in the next time when the system will move to View Change state due to rf failed nodes i.e. the 4, 7, 9 from the figure 4.6 and a new one failed node which is the 1, it will replace automatically the R failed

nodes deactivated from the previous situation but now in the second round of view change instead of third. The deactivation time is based on the consecutive consensus rounds that take part without doing something in the network. If there are more than one failed nodes with the same deactivation time then the node with the lowest reputation from the nodes with the highest deactivation time will be removed.

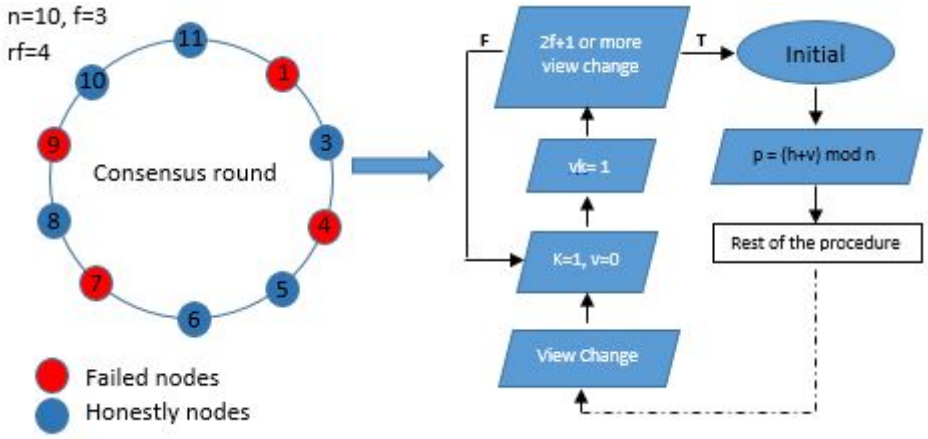

Figure 4.6: Moving in a view change state after some constantly failed nodes and the presence of new one failed node.

In the first round of view change state which is depicted in figure 4.6 the view v=0 and k=1, these failed nodes have the opportunity to reestablish the communication within 40 seconds. After that time, if there are not 2f+1 change view correct messages, then the k will increase by 1, as happened in the second round of view change in figure 4.4 and with k=2 the R failed nodes 4.1 with the longest deactivated duration will be replaced. Finally with view v=2 the system will begin a new consensus round.

4.3.2 Replacement of malicious nodes

On the other hand, in the presence of malicious nodes, although the procedure of replacement is going to be implemented in the same manner as the case of failed nodes, there is a different way of implementation according to the rounds in the view change state and the time. In the presence of rf=4 malicious nodes, i.e. 2, 4, 7 and 9 nodes from figure 4.7, the system will move to the View change state as happens in the normal operation of DBFT, on the grounds that there are less than 2f+1 messages with the same results.

Through our proposal, in the first round of view change, the malicious nodes have the chance to send correct message. A node can send an erroneous message in the system because it wants to delay the system. So, after 40 seconds if the same rf=4 malicious nodes continue to send erroneous messages, then in the second round of view change for k=2 the malicious nodes will be punished and replaced by the next top consensus nodes which have the turn in the list of candidates nodes. Then the consensus procedure will start with view v=2.

Furthermore, another case of having rf=4 malicious nodes should be considered. In this case when the system is in the view change state, it can be successfully completed in the first round if one or more malicious nodes revised their activity and decided to act honestly sending correct messages i.e. node 2 from figure 4.7. So, the system can work properly once there are 2f +1 correct messages and then the consensus round can begin

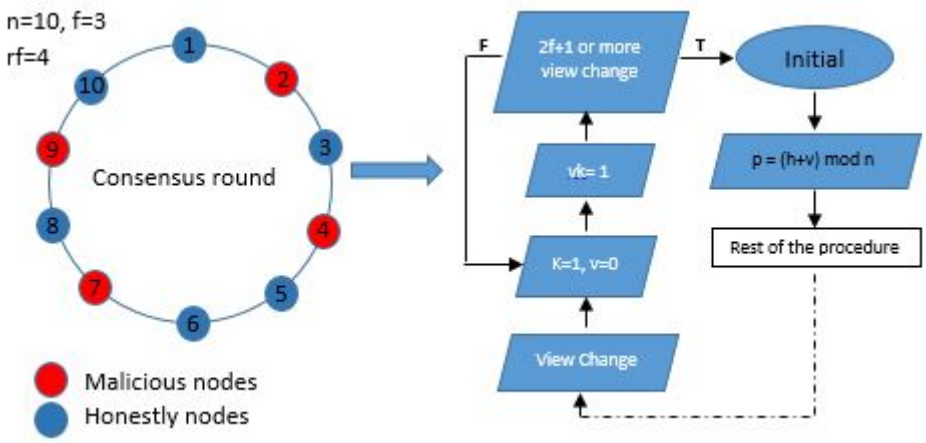

Figure 4.7: View change in case of malicious nodes

with view v=1. During the new consensus round with view v=1 if the same rf=4 i.e. 2, 4, 7 and 9 or less malicious nodes i.e. 4, 7 and 9 keep misbehaving by sending erroneous messages then will be punished and replaced automatically.

As it has been discussed so far, the system of n=10 consensus nodes can reach agreement having up to f=3 malicious nodes. The nodes in the whole network know which consensus nodes misbehave because all the messages traverse through the network. As a result, they can vote out for some misbehaved nodes. If there are up to rf=3 malicious nodes, which take part in the consensus procedure without entering in the view change state, then in our proposal these malicious node will be deleted after three successful consensus rounds.

In the case of view change state before completing the three consensus rounds where rf=4 due to three malicious nodes and one failed node or four malicious nodes, the system will give the opportunity to the malicious nodes to behave honestly in the first round of view change. Maybe if the failed node reestablish the communication during the first round of View Change, the agreement can be reached. Therefore, in the next consensus round for v=1, if the malicious nodes continue misbehaving, will be deleted. Otherwise, if there is not agreement for k=1 in the first view change round, then the malicious nodes will be replaced in the second round for k=2 by loosing all the rights and the system will execute new consensus round with view v=2.

4.3.3 What happens in the case of existing two same groups of consensus nodes that supporting different blocks

There is, also, a rare case of even number of consensus nodes i.e. n=10 where 5 consensus nodes supporting i.e. block a and the other 5 supporting block b. In this case, our proposal is to remove all the nodes automatically and replace them with the next top 10 nodes from the candidate list. These nodes will not lose their rights but only their votes. They have the right to be again a consensus node if the other peers vote for them. Furthermore, the candidates will record this behavior as suspicious one and if this happen again in the future with some of these nodes, then these nodes will be considered as malicious and they will lose all the rights.

4.4 Development of a reputation mechanism

4.4.1 Overview of reputation mechanism

Except of the development mechanism to delete or remove these misbehaved nodes without the need of the voting procedure, another mechanism should be developed in order to measure the reputation of the consensus nodes. The measurement should be done in a daily manner and the peers will measure only the nodes that take part in each consensus round. Three or more candidates according to the size of the network, in each consensus round will implement these measurements for a specific consensus node. Furthermore, these measurements will be implemented by candidates considering that they have enough storage to record all the measurements. In order to be a candidate except from the 1000 GAS that they have to pay, they also should have a minimum of 100 GB hard drive. In addition, the measurements will be stored also in the database where it is used by the original DBFT algorithm for the recovery mode.

The reputation mechanism will be based on three categories and each one will have different weights according to their importance. The sum of the weights shall be equal to 1. In the first category, the grades of the rate will be 0 or 5. In the other cases, the process rating will start from 1 to 5 with 5 the top grade. The top 20% of the consensus nodes will be rated with 5, the second 20% with 4 and this continues until the lowest 20% which will be rated with 1.

1. In the first category, the consensus nodes will be measured according to the messages that exchange during the consensus round. If the consensus nodes send correct message will be rated by 5, otherwise in the case of erroneous message or no message, they will be rated by 0. The weight value of this category is 0.5 as it is the most important field due to that the measurement is based on the behavior of each consensus node.

2. The second category is based on the tokens that each consensus node holds. The first 20% of consensus nodes with the higher amount of tokens will be rated with 5 and the rest of them will be rated with less grades depending on their positions. This category has the lowest weight of 0.15.

3. In the last field, the consensus nodes will be rated according to the votes that they have. As the previous cases, the top 20% of peers with the highest votes will get 5 and so on. The weight grade for this field is 0.35 and it is the second highest grade as the votes are a proof that the participants of the whole network trust this consensus node.

4.4.2 Procedure for grouping candidates

According to the original DBFT algorithm all the peers that want to take part in the consensus procedure they have to pay 1000 GAS, as to have available at least 100 GB hard drive and there are known as candidates. From all the available candidates only a specific number can take part in each consensus round and this is based on the votes and the tokens that each candidate has. As a result candidates will implement also the reputation measurements of all the consensus nodes.

In each consensus round there will be 3 or more candidates to measure the reputation for each consensus node in order to provide to the system more accurate results in the presence of misbehaved nodes. The number of possible different groups combinations of candidates will be calculated according to the formula 4.3.

$$c = \binom{nc}{g} = \binom{11}{3} = \frac{11!}{3!(11-3)!} = 165 \tag{4.3}$$

Where: c the number of available combinations, nc the number of the available candidates for a specific day (So far only 11) and g the number of candidates in groups (in group of 3).

Then, the candidates which will be responsible to measure the reputation for a specific consensus node with index number i will be calculated by the expression 4.4. The selection of the candidates according to the formula 4.4 for the measurement of a specific consensus node will happen every consensus round as the voting procedure occurs in real time and in every consensus round probably different candidates will take part.

$$ng_i \ = \ h + i \ mod \ c \tag{4.4}$$

Where: i is the consensus node index and the value range will be $1 \leq$ i $<$ n and h height block number at the corresponding consensus round.

All the candidates have to declare their digital identities in order to be elected as consensus nodes and the peers in the network can vote for them through their public keys. As a result, in each consensus round different candidates will take part and the identity of them will correspond to a number i index according to the number of consensus nodes in this specific round. For example, assume that at the moment there are 13 candidates and the system needs only n=10 consensus nodes in order to participate in the next consensus round as depicted in table 4.1. As a result, for this specific round, each candidate will have different number i index as it is fluctuated from $1 \leq$ i $<$ n. For instance, a candidate with digital identity A has index i=9 and candidate D has index i=1 as appeared in the table 4.1. This procedure is done in the normal procedure of the DBFT algorithm in order to simplify the selection of the new speaker in every consensus round. Instead of making use of public key of each consensus node they are given to them a different number index i for each consensus round.

The result ng_i in equation 4.4 will be associated with the number of the list with the possible combinations, which is calculated through the matlab. Each number in the matlab list indicates the three candidates which will be responsible to measure the reputation for a specific consensus node. Through the formula 4.4 there is a possibility where one of the three full validators will measure their-self but this is not unfair as there are another two or more full validators which will measure also this one by providing more accurate results in the network. The table with the desirable results can be obtained in the begin of each day according to the number of available candidates. Furthermore this table will be obtained by using the following commands in matlab.

i=11, which is the number of available candidates

v=1:1:i, through this line, the v increases by 1 until it reaches the number i.

Number of Candidates	Identity of Candidates	Corresponding index i for 10 consensus nodes
1.	A	i=9
2.	B	i=7
3.	C	i=3
4.	D	i=1
5.	E	i=4
6.	F	i=2
7.	G	i=8
8.	H	i=5
9.	I	i=10
10.	J	i=6
11.	K	-
12.	L	-
13.	M	-

Table 4.1: Table of candidates with different number index i

C=nchoosek(v,3). This formula is used to determine the possible combinations in a group of 3.

Then, the table C with all the combinations of the candidates is appeared. The figure 4.8 depicts an example of the previous procedure. For example, the possible candidates for the consensus node with index number 3 and block height number 4317382 will be 7, 9 and 11. These candidates are obtained by the ngi formula 4.4 where the result corresponds to the Matlab table.

$$ng_3 = (4317382 + 3) \bmod 165 = 160$$

Number of combinations	Candidates combinations
1	1, 2, 3
2	1, 2, 4
...	
50	2, 3, 8
51	2, 3, 9
...	
160	7, 9, 11
161	7, 10, 11
...	
165	9, 10, 11

Figure 4.8: Procedure of selecting candidates

In addition, if the same consensus node takes part in the next consensus round, then the possible candidates that they will calculate the reputation of the consensus node will be 7, 10 and 11. As a result, in every new consensus round, different groups of candidates will measure the reputation for a specific consensus node.

4.4.3 Calculation of reputation for a specific day

In this section the overall procedure of the calculation of reputation will be explained. In each consensus round three candidates will measure the participation of a specific node ni by calculating the average of the previous three categories multiplying by the corresponding weights in the end of the consensus round using the expression 4.5.

$$res_j(ni) = \frac{(1st\_grades) * 0.5 + (2nd\_grades) * 0.15 + (3rd\_grades) * 0.35}{3} \tag{4.5}$$

Where: j is the number of the corresponding candidate and ni the number of consensus node with index number i.

Moreover, the three candidates will calculate the average of these three results in order to get the total measurement for the consensus node ni in this specific consensus round using the formula 4.6 and then they will publish the total result:

$$res_j(ni\_conrounX) = \frac{res_j(ni) + resj(ni) + resj(ni)}{3} \tag{4.6}$$

Where: X is the number of the corresponding consensus round.

For instance, in the previous example the result of ng3 was 160 and the obtained candidates 7, 9 and 11 were responsible to measure the reputation of the consensus node ni=3. So, these three candidates will calculate the average of these three results for ni=3 according to the expression 4.7:

$$res_{7,9,10}(ni\_conrounX) = \frac{res_7(3) + res_9(3) + res_{10}(3)}{3} \tag{4.7}$$

After that,the three candidates will record the data for the node ni and they will send the results in the network. In addition, node ni, in turn, will also record the results from the three candidates and these results will also be stored in the local database.

The candidates will send the result and the hash of the result by signing them with their private keys. Through this technique, everyone in the network that wants to cross-check the results can do it by using the public key of the candidate.

A day consists of at about 4320 consensus rounds cr. Therefore, in the end of the day, 3 new candidates will calculate the total reputation of a candidate ca_j by calculating the average of all the $res_j(ni\_conrounX)$ as it is depicted in the formula 4.8. This result will be the reputation for candidate ca_j for the next day.

$$Tot\_rep_j(ca_j) = \frac{rep_j(ni\_conround1) + rep_j(ni\_conround2) + ... + rep_j(ni\_conroundcr)}{cr} \tag{4.8}$$

The three new candidates will be calculated through a new formula which is depicted in equation 4.9 and it will be based on the position in the candidate list for each candidate ca_j according to the voting procedure in the end of the day. Consequently, the result of equation 4.9 will correspond to the matlab list as in the example of the figure 4.8.

$$ca_j \ = \ c \ mod \ z \tag{4.9}$$

Where: ca_j is the digital identity of the candidate, c the number of available combinations through the equation 4.3 and z the number of candidate position in the candidate

list according to the votes. For example, if candidate ca_j=10 has the 100th position in the candidate list, then, through the equation 4.10, the three candidates which will calculate the total reputation for a candidate 10 for the next day should be the candidates from the matlab list with number equal to 65.

$$ca_{10} \; = \; 165 \; mod \; 100 = 65 \tag{4.10}$$

The three average results for the measurement of a consensus node ni for each consensus round should be the same. In addition, the final three results for the total reputation for a candidate ca_j of the whole day should be, also, the same. In case of having one or more different average results in each consensus round or different results of the total reputation for the whole day, then the system will examine the reason of these results. In the case of failed nodes where there may be only 2 honest results or in the rear case 1 honest result instead of three, the system will implement nothing. In the case of malicious node where there will be different results, the system will punish this candidate that sent error result by taking back their deposit which is paid by each node in order to be candidate.

5 Conclusion

The purpose of this final section is the conclusions of this master thesis after the analytical clarification that has been done for what is a blockchain, the general architecture of this and an extensive focus on the different consensus protocols. Furthermore, the main objective of this thesis was the improvement of a consensus algorithm, in our case the DBFT algorithm as the development of a reputation mechanism, which is necessary through the improvement that we have implemented.

The blockchain is a very promising technology that can be applied in multiple areas of human activity such as financial, banking, health fields and so on. In addition, it provides security and trust to the peers in the network through the decentralized and distributed ledger, the smart contracts and the consensus protocols. Some of the advantages of this technology are the direct asset transactions between peer-to-peer network without the need of third parties, the integrity and the reliability of the system as the immutability of the transactions in the network through the consensus protocols. Although the blockchain provides many advantages at this moment, it is still in immaturity level and it will take many years by researches and different development applications in order to become a necessary and daily tool in the humanity.

During the analyzing of different consensus algorithms, it seems that a DBFT algorithm is one of the most promising and fair protocol according to the needs of all the peers since is implemented in democratic way through the voting procedure, and all the peers have an active role in the network. In addition, it is difficult for an attacker to implement any kind of attack in this blockchain as the consensus nodes have to pay 1000 GAS for deposit, and in every consensus round, different speaker is selected based on the formula p = (h+v) mod n. Furthermore, all the exchange messages are traversed through the whole network and everyone in the network it is able to know about the behavior of the consensus nodes. Moreover, through our improvement the malicious nodes will loose all the rights that they have in the network as the deposit that they paid in order to be consensus nodes. In addition, based on our improvement, we solved another limitation of DBFT algorithm where is the lack of interest of ordinary nodes to vote out for misbehaved nodes. As a result, the system can work properly in case where the ordinary nodes dont care about the operation of the system.

As the improvement of the DBFT algorithm was based only in theoretical concept, the implementation of this algorithm in a practical way is possible for future work in order to see the behavior and the results of the system in real conditions according to our theoretical improvements. In addition, a practical development of the reputation mechanism can be done in order to examine the behavior of this based on our formulas and measurement categories as an improvement of all these parameters since the reputation mechanism can be implemented in many different ways.

Bibliography

[1] Andreas M. Antonopoulos. Mastering bitcoin. *Book*, June 2017.

[2] Dr. Arati Baliga. Understanding blockchain consensus models. *Book*, April 2017.

[3] Charts bitcoin. Blockchain size. `https://charts.bitcoin.com/btc/chart/blockchain-size#5ma4`, August 2019.

[4] Bitinfocharts. Cryptocurrency statistics. `https://bitinfocharts.com`.

[5] BitShares. White paper of bitshares. `https://bitshares.org/technology/delegated-proof-of-stake-consensus/`.

[6] Vitalik Buterin. On public and private blockchains. `https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains/`, Aug 2015. Accessed on 6 August 2015.

[7] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. *Article*, February 1999.

[8] Miguel Castro and Barbara Liskov. Authenticated byzantine fault tolerance without public-key cryptography. *Article*, June 1999.

[9] KONSTANTINOS CHRISTIDIS and MICHAEL DEVETSIKIOTIS. Blockchains and smart contracts for the internet of things. *Article*, 3 June 2016.

[10] Elli Androulaki Ghassan O. Karame and Srdjan Capkun. Two bitcoins at the price of one? double-spending attacks on fast payments in bitcoin. *Article*, 2012.

[11] Stuart Haber and W. Scott Stornetta. How to time-stamp a digital document. *Article*, 1991.

[12] Peter Lin Erik Zhang Igor M. Coelho, Vitor N. Coelho. Delegated byzantine fault tolerance: Technical details, challenges and perspectives. *WhitePaper*, March 14, 2019.

[13] ROBERT SHOSTAK LESLIE LAMPORT and MARSHALL PEASE. The byzantine generals problem. *Article*, 1982.

[14] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Article*, 2008.

[15] NEO. The dbft algorithm. `https://docs.neo.org/developerguide/en/articles/consensus/consensus_algorithm.html`, 2018.

[16] NEO. Neo design goals: Smart economy. `https://github.com/neo-project/docs/blob/master/en-us/whitepaper.md`, 2018.

[17] NEO. A statement from neo council. `https://neo.org/blog/details/3067`, 2019.

[18] NEO. White paper. https://docs.neo.org/en-us/basic/consensus/whitepaper.html, 2019.

[19] Giang-Truong Nguyen and Kyungbaek Kim. A survey about consensus algorithms used in blockchain. *Article*, February 2018.

[20] Nt. White paper of nxt. https://nxtwiki.org/wiki/Whitepaper:Nxt, Feb 2016. Accessed on 7 February 2016.

[21] Peercoin. White paper of peercoin. https://docs.peercoin.net/, Dec 1988. Accessed on 2012-11-11.

[22] Waves platform. Leased proof of stake (lpos). https://docs.wavesplatform.com/en/blockchain/waves-protocol/leased-proof-of-stake-lpos.html.

[23] Hyperledger Sawtooth. White paper. https://sawtooth.hyperledger.org/docs/pbft/nightly/master/architecture.html#message-types, 2018.

[24] Hyperledger Sawtooth. White paper. https://sawtooth.hyperledger.org/docs/pbft/nightly/master/configuring-pbft.html, 2018.

[25] W. Scott Stornetta Stuart Haber and Dave Bayer. Improving the efficiency and reliability of digital time-stamping. *Article*, March 1992.

[26] Florian Tschorsch and Bjorn Scheuermann. Bitcoin and beyond: A technical survey on decentralized digital currencies. *Article*, 2015.

[27] Wikipedia. Blockchain. https://en.wikipedia.org/wiki/Blockchain#Types_of_blockchains, April 2019.

[28] Hongning Dai Xiangping Chen Zibin Zheng, Shaoan Xie and Huaimin Wang. An overview of blockchain technology: Architecture, consensus, and future trends. *Article*, 201.