Practicability of Blockchain Technology and Scalable Blockchain Network: Sharding

by

Abdoul-Nourou Yigo

A thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science

Computer Science

At

The University of Wisconsin-Whitewater

December, 2019

Graduate Studies

The members of the Committee approve the thesis of

Abdoul-Nourou Yigo presented on December 6, 2019

Dr. Athula Gunawardena, Chair

Dr. Jiazhen Zhou

Dr. Sungchul Lee

Practicability of Blockchain Technology and Scalable Blockchain Network: Sharding

By

Abdoul-Nourou Yigo

The University of Wisconsin-Whitewater, 2019 Under the Supervision of

Dr. Athula Gunawardena

ABSTRACT

Currently, some research has been done in the Blockchain domain on how to improve the efficiency of transmissions in Blockchain ecosystem. The effectiveness of the transmissions within a Blockchain network depends on the execution of appropriate consensus algorithms to ensure the security and fairness of the validation and transmission of transactions. The first consensus algorithm that has been used within a Blockchain ecosystem such as Bitcoin is Proof of Work (PoW). The second generation of consensus algorithm is Proof of Stake (PoS) that has been put into practice in the PeerCoin Blockchain ecosystem. One major issue is that those consensus algorithms are inefficient because they require a significant amount of energy consumption to enable the validation of transactions in a complete Blockchain platform. Therefore, the amount of transactions that can be processed and validated is limited and expensive in terms of time complexity. As a result, scalable approaches need to be designed to sharpen the efficiency (i.e., with low latency and high throughput) of the Blockchain system. The introduction of the sharding concept has given us the possibility to divide the network into small portions, and to enable the validation of transactions in parallel using the Byzantine-Fault Tolerant (BFT) consensus on those subset of nodes. Sharding the network can significantly improve the Communication Cost per Transaction $(CCPT)$ because each transaction is validated in its specific shard using validator nodes with appropriate proof size. In our implementation, the transaction would follow the shortest path, traversing nodes with small proof sizes minimizing the $CCPT$. Structuring our Blockchain network, proof sizes are randomly

assigned to the nodes to ensure unbiased network structure to stimulate and evaluate our approach to perform a global optimization within a sharded Blockchain network.

ACKNOWLEDGEMENTS

I still remember my first day in University of Wisconsin-Whitewater Computer Science Department during the Fall 2015. I needed to meet my academic advisor for the first time. Since then, I had the privilege to meet Dr. Athula Gunawardena. I greatly appreciate your advice and support for all these years. Most importantly, I appreciate the consideration of allowing me to work with you on various research projects. Your guidance on this thesis project is another privilege that I am grateful for.

Also, I would like to thank Dr. Sobitha Samaranayake to be one of my mentors during my graduate studies. I have learned valuable technical skills from your practical expertise.

I would like to thank Dr. Jiazhen Zhou for allowing me to be his undergraduate research assistant. This undergraduate research project had shaped my path to pursue my graduate studies in computer science. It has been a great pleasure to work under your guidance.

I would like to thank Dr. Sungchul Lee for accepting to be part of this research project. It is a great honor to get some guidance and advice from you in this research about this new technology.

Most importantly, I would like to thank all of the faculty members in the Computer Science Department for their great comprehension, and collaboration.

I dedicate this thesis to my family.

Thanks for giving me the opportunity to study in the U.S.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION AND BACKGROUND

The revolution of Blockchain technology is undeniable because its potentialities to improve human interactions in different aspects could open a new era of digital communication. For instance, the way organizations handle business transactions and privacy could take another lift with the use of the Blockchain platform. Therefore, it is crucial to make sure that this kind of system can effectively process information. As a result, the need of a Blockchain structure that could reduce the overhead of consensus agreement when it come to the validation of transactions could have a profound impact on the efficiency of the Blockchain system. This consideration of Blockchain system scalability would not be possible without its creation in the first instance. From its adoption, the Blockchain continuous evolution has taken different phases. In this thesis, we would explore these phases in detail.

1.1 Phases of Blockchain Ecosystem Evolution

1.1.1 Phase 1: The birth of Blockchain through Digital currency: Bitcoin Blockchain

The creation of a digital monetary system had been emphasis for years without sufficiently solving a major problem known as double spending problem. in 2008, Satoshi Nakamoto came out with a new cryptographic payment system to enable different entities to transact with each other without the need of a central authority [1]. In other words, the communication between the entities is cryptographically encrypted and does not require the entities to reveal their identities. For instance, such a system was able to eliminate the third party structure by structuring a peer-to-peer network structure where each entity within the network is represented as a node. When a specific node has executed some tasks, the information about these specific tasks would be broadcast across the network such that the other nodes would be tracking occurrences within the network. The system

also introduced a time stamping mechanism to ensure the chronology of the data processing within the network that ensures the unicity of the information.

The processing of this information is represented in a form of transaction. The transaction could be transferring digital coins from one node to another node. A digital coin is constructed with a set of digital signatures that would clarify the provenance of the coins. When a transaction is validated, it would be saved in a block. The blocks in the network would be structured in a specific order such that the successor block would point to the predecessor block creating a chain of blocks that would be piled on top of each to create a Blockchain as shown in **Figure 1.1**.

This type of structure has opened a new era on how people could use a distributed network to create a payment system that would not depend on a central authority. From that initiative, some innovation could be done to use this logic to the next level to establish smart contract [2]. This allows the advancement of the Bitcoin Blockchain to a state machine completeness [2].

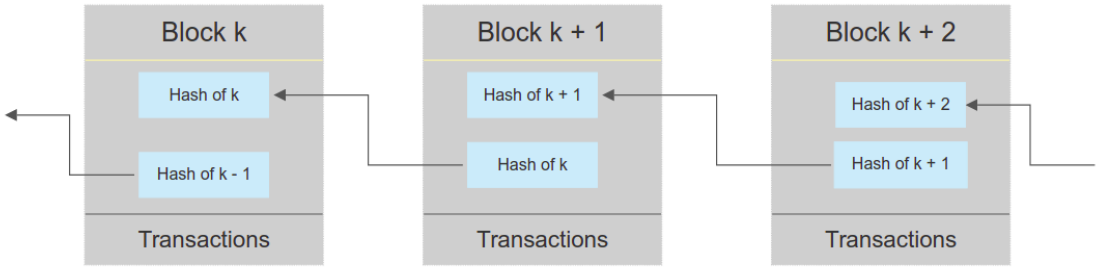

Figure 1.1: Chain of Blocks

1.1.2 Phase 2: The advancement of the Bitcoin Blockchain to a state machine completeness: Ethereum

As discuss above, the Bitcoin Blockchain network has presented some valuable features for a cryptocurrency system, but as a first version of a decentralized network structure, it has some limitations that could be exploited to create an innovative Blockchain system. The limitations could be cited as the following [2]:

- The lack of Turing-Completeness

- Value-Blindness
- Lock of state: the Unspent Transaction Output (UTXO) can either can be spent and Unspent
- Blockchain-blindness

These limitations had caused the birth of Ethereum, a programmable Blockchain platform for smart contracts. In Ethereum Blockchain, messages and transactions can have different interpretations. For instance, messages within the Ethereum Blockchain can be generated from a particular contract or an external resource whereas in the Bitcoin messages (transactions) could be created externally [2]. Most importantly transaction in Ethereum are represented in a form of signature that stores the message to be transmitted [2]. For instance, a transaction would contain the specific information about the sender and the receiver for its authentication. With these functionalities the Ethereum Blockchain network has pushed the Bitcoin Blockchain Network to another level outside of cryptocurrency platform. This advancement has shown the path to revolutionize Blockchain platform that could improve the use of cryptocurrency around the world.

1.1.3 Phase 3: The recognition of the Blockchain impact with the creation of Libra

The evolution of Blockchain platform has reached a sufficient stage in which cryptocurrency could be used at a global scale without the use a central authority. The creation of Libra [3] is going in this sense to give the possibility to million of people the ability to perform transactions without the need of banks. Libra has based its Blockchain network specification to aim for some specifics functionalities [3]:

- As the the previous Blockchain platforms, Libra is aiming security, reliability. Most importantly, Libra is projecting to ensure scalability within its Blockchain ecosystem.
- Libra introduces the notion of reserve within its Blockchain ecosystem that would ensure the maintainability of Libra's value.
- The main validators of the Libra Blockchain would be members of its association.

With these functionalities, Libra targets the implementation of Blockchain platform that could be scalable when it comes to the execution of millions of transactions at global scale. Therefore, the scalability issues within the Blockchain ecosystem should be taken with serious considerations. One of main consideration that could impact the Blockchain platform scalability is the overhead transmission within the network. More details would be given about the scalability issue in **Section 1.4**.

1.1.4 Futuristic Phase: The world monetary system would be digitalized to enable transactions to a virtual level between humans and machines to machines

The idea of using the Blockchain technology in different aspects of our society can be represented in many forms. For instance, this technology could have multiple use cases opening a new representations of existing societal structures. The advancement of technologies in different domain such as Cloud Computing, Internet of Things (IoT), Artificial intelligence (AI) is shaping the way we use technology nowadays. The integration of these technologies with the Blockchain technology could open a futuristic spectrum of new innovations.

Current Use Cases of the Blockchain Technology

We are going to enumerate some practical use cases that have been developed using the Blockchain technology.

1. **Adoption of the Blockchain Technology within the Supply Chain Ecosystem**

   The Blockchain technology could be used in the supply chain sector to enable more transparency by storing the information about the flow of products from suppliers to stores such that the provenance of the products could be traceable. In [24], they have mentioned that the Blockchain platform could be used to track the source of products information so that customers can make sure the products that they consume have been well conserved.

   With this functionality, some corporations have already combined Blockchain technology with their supply chain ecosystem to ensure more clarity. For instance, Walmart has used

Hyperledger Fabric to track the origin of their products [25]. For instance, Hyperledger Fabric blockchain-based for tracking food has reduced the tracking procedure from 7 days to 2.2 seconds [25]. With that result, Walmart is planning to track 25 more different products using Hyperledger Fabric [25].

2. **Decentralization of the Music Industry's with the Blockchain Technology**

   The variety of domain that could use the Blockchain technology could expand to the music industry. For instance, creating a "Music Blockchain" could give the possibility to artists to manage their own creations removing the aspects of central authorities [26]. Doing so, there is no need of intermediaries, but rather having smart contracts that would be used to manage the artistic creation usages.

There could be multiple use cases of the Blockchain technology because of its potentialities. These potentialities could lead to some futuristic projects that would enable the combination of the Blockchain technology with other improving technologies to satisfy the technological evolution trend.

Futuristic Usability of the Blockchain Technology

With the significant evolution of technology in different domain such as artificial intelligence that gives the possibility to have self-driving cars. The implementation and the configuration of smart devices that enable the executions of specific tasks. Furthermore, the tremendous evolution of Blockchain technology could enable the implementation of special technologies. For instance, the combination of those technological instances could enable a payment system in which smart devices would communicate and execute transactions with each other without the need for human interaction. Most importantly, when cryptocurrency systems would be used at large scale, smart devices could authenticate themselves as being bought as long as their payment contract is correct. For instance, a self-driving car could drive itself to the owner after its payment.

With this dramatic technological evolution, centralized big technology organizations gather sufficient information about their users to influence their thinking and decision patterns. The

rethinking of the technological ecosystem to avoid the monopoly of any sort of organizations could be crucial. Therefore, the need of a decentralized system that could eliminate the sense of monopoly could be essential in this era. As a result, the Blockchain has arrived to the right century to enable a more robust decentralization ecosystems. This type of ecosystem has been in need of efficient consensus algorithms to enable its efficiency, fairness, security, and reliability. On top of all that, the possibility to scale by maintaining those core properties.

1.2 Phases of Consensus Evolution

Since the Blockchain ecosystem is in a form of network structure that would have a set of nodes connected by a set of edges. Some consensus algorithm need to be put in place to maintain agreement within those nodes that would try to communicate using some particular edges. The major role of a consensus algorithm is the maintenance of fairness within the network. Therefore, consensus algorithms are the guardians of the Blockchain ecosystem; as a result, their executions can considerably impact the scalability of the Blockchain network.

1.2.1 Phase 1: Satoshi Nakamoto Consensus: Proof of Work (PoW)

The first algorithm that was adopted in the early version of the Blockchain platform was Proof of Work (PoW) [1]. This algorithm is structured such that the validator nodes ($miners$) would have to use their computation power to solve a cryptographic puzzle. For instance, the system would introduce a unique composition called ”nonce” within the transaction that needs to be validated by the validator nodes figuring out the sequence of the nonce with the transaction hash. When a specific miner is able to solve the puzzle, the miner has the possibility to create a new block. The problem with this type of validation is that it requires enormous use of computing power. For instance, the main issue with PoW is that it does not scale, this could be related to many reasons. The first reason could be related to the structural representation of transactions being saved within a block. The second reason is the constraints of the network structure. We are going to give some explanations about these concerns.

1. **Structural Representation of Transactions within a Block**

   One of the reasons that could negatively influence the scalability within the PoW based Blockchain network is the size of the information (transactions) that is being saved and the size of the block in which the transactions are gathered [27]. Most importantly, during the PoW execution, some executions need to be performed such that the consensus level would determine the winning validator node. For instance, in the Bitcoin Blockchain network, the size of transaction could be impacted by its inputs and outputs. When the transaction is validated, it would be added to a newly created block. Then, this block would be broadcast to the network so that every node in the network could keep track of what is happening within the network.

   With a logical analysis, we could notice that PoW creates a sense of competition, in which there is a winner node (miner). The issue with this logic is that the validator nodes that lost the PoW consensus effort would create unused partial blocks (stale blocks). For instance, in the Bitcoin Blockchain network, the stale blocks would not be added to the main chain, so they would be discarded. The computing power used to create these blocks is lost because rewards are only given to winning validator nodes [28]. Consequently, the uncle blocks could create an increase of bandwidth usages that could affect the performance of propagating the message within the network [28].

2. **Network constraint Within PoW Based Blockchain Network**

   One of the challenges that could arise when it comes the Blockchain network scalability could be related to the structure of the network itself. For instance, the PoW consensus needs to operate across all the nodes in the whole network such that the validator node that solved the puzzle could be found [27]. Therefore, the PoW is not designed to operate on subsets of nodes to enable a sufficient scalability.

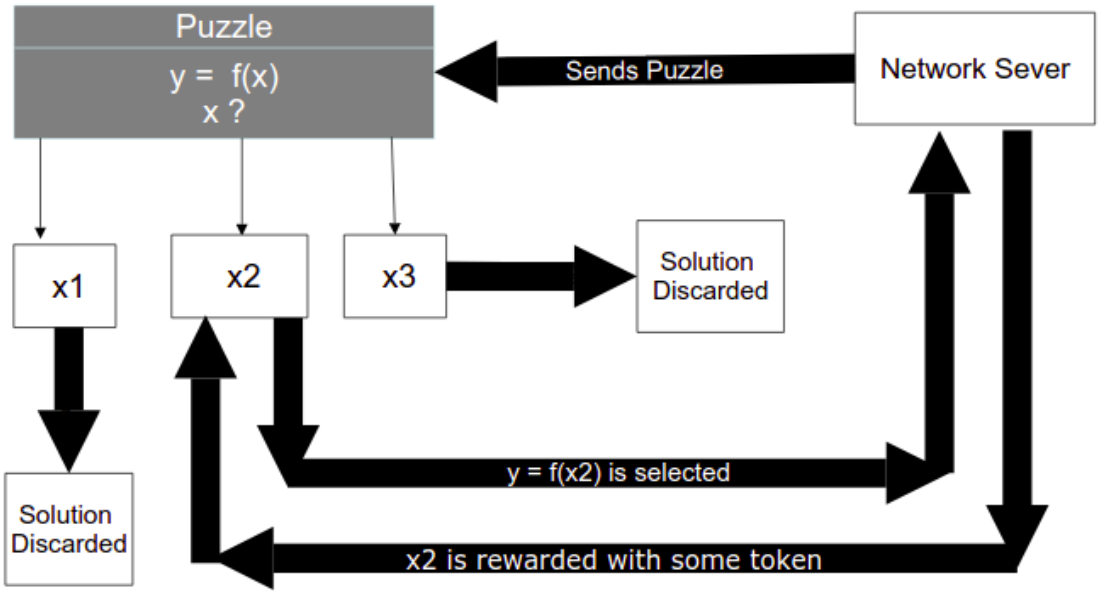

Figure 1.2: Proof of Work Concept

1.2.2 Phase 2: Proof of Stake (PoS) Algorithm

The Proof of Stake (PoS) algorithm is a modified version of PoW. it was designed to eliminate the burden of energy consumption from PoW [4]. The PoS algorithm was first used in Peercoin Blockchain platform [23]. It uses a different approach when it comes to processing the information. The PoS algorithm operates in the following way, the agreement on the creation of a new block is a competition in which participators hold a certain amount of coin (”stake”) [5]. Therefore, the participators that would have significant stakes could have an influence on the competition outcome.

1.2.3 Phase 3: Casper Protocol

The Casper protocol is a combination of two different protocols such as PoS and the Byzantine Fault Tolerant consensus theory [5]. The association of two different consensus algorithms is showing that some particular functionalities are being targeted. For instance, the PoS and BFT portions of the Casper Protocol play the following roles [5]:

- The PoS has the possibility to take a randomized approach to give a certain permission to stakeholders to generate new blocks.
- The BFT protocol has a property that as long as $> \frac{2}{3}$ of the nodes within the Blockchain network behave honestly, the algorithm won't be able to finalize the conflicting nodes within the network.

Those properties give the Casper protocol the ability to introduce new methods that could be put into practice in a Blockchain ecosystem. Those methods are the following [5]:

- **Accountability:** This method is to ensure that any validators would be responsible for their violation.
- **Dynamic Validators:** Since the protocol would be performing within a enormous network, the exchange of validator nodes at each consensus iteration could ensure the unbiasedness of the network.
- **Defense:** The protocol has incorporated a mechanism to be attack resistant to the "long range revision attack".
- **Modular overlay:** The protocol offers the possibility to build its functionalities on top of other consensus protocol such as PoW.

As we could observe, considerable advancements have being made for the consensus algorithms that are in use in the Blockchain ecosystem for its scalability. The next consensus algorithm we would observe is the BFT based protocols that are more considerable when it comes to Blockchain network scalabilty.

1.2.4 Phase 4: BFT based Consensus

The Byzantine Fault Tolerant concept has been researched for three decades. The principle of the Byzantine Fault Tolerant protocol is tight to "Byzantine Generals Problem" [6]. For instance, the abstract explanation of the problems can be expressed as the following: A Byzantine army forms

subsets of troops in different location around an enemy army. The troops need to communicate to come to consensus for a common strategy to attack the enemy army. They have to send messengers in their respective locations such that each party would be informed about the attack strategy that would be used. The major problem is that one or more messenger(s) sent could be compromised by a traitor. Most importantly, one the generals from the different locations could be a traitor by sending the messenger with falsify information to the other troops. Therefore, the problem is to define an algorithm that would guarantee that the ”loyal” generals would find a consensual common ground that would enable a successful attack. The message could be solved from two perspectives [6]:

- If the messages are orally sent, the problem could be solved if and only if $\frac{2}{3}$ of the generals are loyal
- If the message in a form of unhackable written message, the problem could be solved for any number of traitors.

These Byzantine generals problem had been translated to a computer science problem. For instance, its implication could be more effective within a distributed ecosystem in which each entity could be treated as a node in the network. Since in this type of system, some nodes in the network may have some unpredicted behaviors, the BFT based consensus permits to eliminate the assumption of adversarial behavior patterns within a distributed ecosystem. From these principles, different Blockchain protocols have been build to improve the scalability issues that a Blockchain network is facing. We are going to explain different varieties of the BFT based consensus that have been improving year by year.

PBFT

The improved version of the BFT based protocol is the Practical Byzantine Fault Tolerant $PBFT$. The $PBFT$ was designed to work within an asynchronous network platform such as the Internet [7]. In such type of networks, malicious nodes or software errors could be a treat to the system

[7]. In other words, they could exhibit a Byzantine behavior. The "Practical" BFT algorithm for state machine replication [7] that could prevent the misbehavior of faulty nodes at most $\frac{n-1}{3}$ where n could be the number of faulty nodes in the network. In [7], they proved that if $2m + 1$ of nodes in the networks are correctly processing information within the network that would imply that the $2/3$ of the nodes in the network should be honest. $PBFT$ has been used in different Blockchain ecosystem. For instance, in [4], they mentioned when it comes to permissioned Blockchain ecosystem, the use of $PBFT$ is advisable because it enables consensus among subset of authenticated nodes (e.g, HyperLedger Fabric v0.5 [8]).

Tendermint

Tendermint is another BFT based protocol. Tendermint is an application based protocol that could be used to reinforce the security and stability of nodes within its distributed ecosystem [9]. It has two key technical components that consist of Blockchain consensus engine and generic application interface [9]. It could used as third party for organizations that do not wish to implement their own Blockchain protocol.

HotStuff

$HotSuff$ is another improved version of the BFT protocol. $HotStuff$ is leader-based Byzantine fault-tolerant replication within a synchronous network setting [10]. $HotStuff$ enables the possibility to elect a leader node that would drive the protocol to consensus[10]. In [10], they argue that $HotStuff$ is a more advanced protocol that combines BFT functionalities and Blockchain principles. The scaling approach of $HotStuff$ is based on two phases [10]:

- The first phase ensures that there is a unique proposal for the formation of quorum certificate (QC). A QC is a certificate proof that a minimum number of nodes has been correctly regrouped.
- The second step is to give the responsibility to the next leader to convince the nodes for secure proposal.

These specifications have pushed BFT principle to another level that could be in use in a more sophisticated Blockchain platform.

LibraBFT

The $LibraBFT$ has built its core foundation based on the $HotStuff$ protocol. The modifications that were done are to enable some particular characteristic within the $Libra$ Blockchain system. The major modification that were done are the following [11]:

- the $LibraBFT$ is more resistant to non-deterministic bugs because the validators directly sign the state of block instead of the transaction within them.
- They incorporate a "pacemaker" mechanism that would act as heartbit to enable the formation of a new quorum without relying on system clocks
- Most importantly, the leader election is based on the decision of the last quorum committee using randomization.

These modifications and advancements are aimed to ameliorate the concept of the BFT protocol that could operate on subset of nodes in a Blockchain ecosystem. When consensus algorithms are scalable, the Blockchain platform can also considerably scale. Within the Libra Blockchain they conjecture to execute 1000 transactions per second. This conjecture could be possible not only by strong and efficient consensus. It would require the network to be splitted accordingly due the notion of Sharding the network.

1.3 Graphical Representation of The Blockchain Scheme

Multiple Blockchain schemes have been used to enhance the efficiency of the Blockchain ecosystem. Those graphical representation have shown some advantages and some drawbacks.

1.3.1 Off-Chain

In this type of structure, the Blockchain system is formatted such that nodes in the system could have local representation of their transaction patterns such that the the global chain call the ”main chain” in the system would be solicited when new occurrences are encountered by sending the information about that specific task to update the global chain state [12]. The validity of the transactions in the isolated chains depends on the verification of the global chain. The issue is that this is creating a centralized Blockchain ecosystem.

1.3.2 DAG

Another approach that was taken to design and implement a Blockchain network was the use of a Directed Acyclic Graph. In this type of design the transaction is not structured in a form of chain but in a form DAG structure. The transactions would take paths according to the direction of the edges within the graph [13]. In other words, the validation of transactions from particular nodes would be influenced by their adjacent edges. In [12], they argue that the network could scale if all the nodes in the graph are not obligated to meet the complete graph property.

1.3.3 Sharded Network

The sharding approach is the possibility to artificially divide the network in appropriate portions such that transactions that land within those specifics shards could be validated using the specific shard validators. Different proposals have been evaluated to design a sharded Blockchain network [12][14][15][16]. This technique could be the most important approach when it comes to Blockchain scalability. The scheme it provides could be described in the following way: Since the nodes in the network are divided in small subsets they could be subject to some adversaries and vulnerabilities. Therefore, as we have previously mentioned robust consensus algorithm such as BFT should be run to eliminate those security constraint to ensure the shards fairness and correctness. Doing so would considerably reduce the Communication Cost Per Transaction ($CCPT$) enabling the global Blockchain network to considerably scale. in [13], they argue that running

the BFT consensus, the $CCPT$ in specific shard could be $O(g)$ where g is the size of the shard. They also observe that none of the sharding schemes proposed, were able to get $o(g)$ which they described as the condition to considerably scale out the sharded platform.

1.4 Consideration of Scalability within a Blockchain Ecosystem at different architectural Levels

The Blockchain ecosystem has multiple layers. These layers should be well structured to enable a correct functionality of the Blockchain platform.

1.4.1 Network Level

We could say that the network layer of the Blockchain could be the most important because it has the responsibility to connect components within the Blockchain platform by enabling communication among them. In the Bitcoin Blockchain, the network enables the propagation the transaction across the network [17]. In other words, nodes in the networks accept to broadcast transactions to other nodes if and only they were correctly validated. The information that is propagated across the network would be gathered at each transshipment nodes to create an immutable consistent record call ledger.

In Bitcoin Blockchain network, there two major inefficiencies that have been observed in [17]. For instance, to ensure the security properties of the system, a node that receives a transaction must check its validity before propagating it [17]. For instance, this validation procedure is necessary because it allows the validating nodes to prevent a denial of service attack by propagating incorrect transactions ingested by malicious nodes [29]. Most importantly, adopting this strategy is to ensure that a specific transaction meets the following validity requirements: the transaction must differ from previous transactions and its output must be correct. The second drawback is that when transactions are validated, the nodes that have validated the transactions are allowed to generate new blocks in which they need to save the transactions. The new block that was generated is also propagated across the network creating a double propagation. Therefore, this is negatively

impacting the efficiency of the network which should be solved.

Different network protocols are in use in the Blockchain network layer. For instance, we have the Overlay P2P Protocol such as Whisper [18], Telehash [19]. Another variant network protocol used the Blockchain ecosystem is Cryptographic Transport Protocols such us Ethereum Wire Protocol [20], and Kademia which is a peer discovery protocol [21]. All these protocols operate on top of the ISO protocol model.

1.4.2 Consensus Level

The Consensus level play the role of central authority to provide security and fairness across the Blockchain ecosystem. As we have explained there has been a tremendous evolution of consensus algorithms. The fundamental role of these algorithms is to maintain the consistency records across the Blockchain network [4]. Since the Blockchain network is a distributed network system, the nodes in the network need to be able to agree on a common state using the consensus parameters [4].

As we could observe, the effectiveness of the Blockchain network could be impacted by its consensus algorithm. For instance, this level could be described as an intermediary procedure that makes sure when an agreement is made between the nodes in the network such that the information they have agreed on would be recorded in the nodes' storage. The efficiency of this level could be significant because fast executions of the consensus could allow the network to efficiently scale. As we have described in **Section 1.2.1**, PoW could not scale because it requires validator nodes to use their computer power to participate to the consensus effort creating a burden on the network. Because of such issue, some of the Blockchain network such us Litecoin has sped up its PoW algorithm for more block generations than Bitcoin by sacrificing some security features [17][30].

Additionally, some amelioration of the PoW consensus algorithm have been proposed to improve its effectiveness. For instance, the GHOST protocol was proposed to reorganize the structural representation of nodes within the Bitcoin Blockchain ecosystem [31]. For instance, by doing this modification, GHOST allows the Blockchain network the improvement of its "mining power"

within a fair ecosystem [17].

With an analytical observation, the issue of scalability within the Blockchain ecosystem could be related to the consensus level because agreement between different entities within the network could be crucial. Without this consensual procedures, the Blockchain properties could be meaningless at some extent. We know that PoW needs to be executed within the whole network so that all the nodes in the network could participate on the consensus effort. Therefore, this is putting the burden on the network creating some scalability issues. Therefore, the use of consortium consensus such that the algorithm could operate on subsets of nodes in the network could significantly improve the performance of the network with high throughput reducing the latency [17]. For instance, most of consortium consensus algorithms are BFT based consensus which we have illustrated in **Section 1.2.4**.

1.4.3 Storage Level

As we can see, we are sequentially examining different parameters of the Blockchain network. The identification of nodes in the network is possible because the network system is able to use the right information at each node. Therefore, each node should have the storage capacity such that communication could be done in a sufficient manner.

Additionally, the Storage level could be define as the ”global memory” that ensure the availability of the information that was generated by the Network and Consensus Levels. In [17], they define the Storage level as an abstraction with two interfaces:

- The storage level ingest and process and memory-modification. That operation could be defined as a write operation. Most importantly, there could be a delete operation from the Consensus Level.

- The Storage level is structured such that any node in the network can make a read request.

Since the Blockchain network is a distributed system, the Storage Level also stores distributed records that are propagated across the network.

Problem statement:

We have realized that the scalability concern within the Blockchain network could be interrelated at multiple levels. The agreement about a specific transaction would require the consensus level to perform some validations and executions. The validation of the transaction would require some computation at the validator nodes. The validator nodes would need to perform some verification using the previous information they have acquired. When the verification is done, the transaction could follow its path to the destination. How can we measure the interrelation within a sharded network? We have exposed some questions in **Section 1.7.2**.

1.5 The age of Sharding arrives for the efficiency of the Blockchain Ecosystem

The concept of Sharding is widely used in distributed database platforms. For example, the manipulation of huge volume of data could be computationally expensive; therefore, distributed databases such as Dynamo, MongoDB, MySQL, and BigTable have incorporated the sharding technique within their ecosystems.

Since the conjecture of using Blockchain technology at global scale is coming to a reality, the amount of data that would be processed could be considerable. The use of sharding technique in the Blockchain platforms has caught the attention of researchers and industries.

In [17], they introduce the sharding concept to address the Blockchain scalability issue. They argue that Sharding could be done at Consensus Level. They offer a scheme that consist of running consensus processing on subset of nodes to enhance the throughput at each node at the same time reducing each node processing storage. However, the scheme they have proposed could be limited. Why it does not have the possibility to shard the main components of the Blockchain system?

1.5.1 Different Structure of Sharded Blockchain Platform

Different Sharding concepts have been proposed to structure the network in certain ways that allow the network to efficiently scale out. The followings are different sort of schemes that have been

proposed:

- Sharding of Computation and Storage
- Sharding of Computation, Storage, and Communication

1.6 Related Work

In this section, we are going to explore the most relevant works that have been done in the context of sharding. Multiple Blockchain network structures have been proposed to try to improve this lack of efficiency and at the same time to ensure that the Blockchain network maintains its core properties such as security and decentralization. This profound need of scalable schemes is due to the fact that the consensus algorithms created by Satoshi Nakamoto [1] in the Bitcoin network ecosystem are inefficient and costly in terms of energy consumption. Researchers have tried to come up with different optimal schemes to enable the implementation of sharding. As we have explained, the concept of sharding is the ability to split the Blockchain network into small portions that would gather subset of nodes.

1.6.1 Elastico

In [14], they have designed a sharded Blockchain network concept called ”Elastico”. According to their analysis, Elastico could scale linearly on the computation power of the validator nodes in the Blockchain ecosystem. For instance, the execution of transactions is proportional to the computing power available in the network [14].

From a technical perspective, Elastico is using a rigorous sharing principle in which the Blockchain network is partitioned in shards making sure that subset of nodes are disjoint. This approach is to guarantee the sharded network would have disjoint ledgers. In order to achieve a great partitioning, the shards in the network should be in a appropriate shape such that variant of the BFT consensus protocol could efficiently operate on them assuming that the nodes in the network have proportionally the same computation power.

Since the network is disjointly sharded the processing on each shard will be done in parallel. This processing characteristics permit the network to locally minimize the cost of communication and execution in each sharded committee [14]. From the cryptographic perspectives, when the network has come to consensus on a set of transactions X, it will formulate a cryptographic digest of X to form a "hash-chain" with previous transactional agreement [14].

The network is structured in particular way such that the graphical connection between validator nodes is synchronous [14]. For instance, the transaction of the information between those nodes should be within a specific interval of time.

Even though, Elastico has presented some good features it faces some challenges that should be mentioned and examined appropriately. Most of the sharding properties are more beneficial within a permissioned Blockchain settings. Bringing those properties within a permissionless Blockchain setting could lead to the following challenges [14]:

- First, in a sharded network, honest nodes do not have the opportunity to use a trustful PKI. Therefore, malicious nodes can just create unreal processing to enable a large scale of sybil attack on the network. To limit this threat the consensus level could prescribe identities to "processors" at each consensus round. When the identities are establish, a variant of BFT algorithm could be run.

- The most challenging aspects is to randomly select "committee" knowing that randomization could be extremely challenging within a distributed network system.

- Another issue that should be put into consideration, is setting the right parameters that would limit the boundaries of the consensus protocol such that its operation would be restricted to any sort of corruptive activities. These limitations should be put in place because Byzantine failure and network failure could be unpredictable

These technical challenges could be solved using particular sharding design patterns. It is in this sense that Elastico was able to create some procedure to automatically split the available

computation power across multiple shards making sure that both states are growing proportionally. The running process of Elastico could take different epoch steps [14]:

- Identity Establishment and Committee Formation
- Overlay Setup for Committees
- Intra-Committee Consensus
- Final Consensus Broadcast
- Epoch Randomness Generation

Elastico has shown a model to implement sharding; however, sharding could be done in a more efficient way.

1.6.2 OmniLedger

In [15], they developed new sharding concept called "OmniLedger" build on top of Elastico. They argue that OmniLedger could operate within permissionless Blockchain such as Bitcoin [1]. OmniLedger could provide a secure sharded platform using a randomized protocol approach when it comes transaction validations [15]. The innovative feature that OmniLedger was able to create is the possibility to execute cross-shard transactions by ensuring the atomicity of the transactions across the sharded ecosystem [15].

Additionally, in [15], they argue that the throughput of OmniLedger could linearly scale the percentage of its validators to handle tremendous volume of transactions reaching the performance of centralized transaction system such as Visa.

Some clear procedures have been taken to structure its core functionalities [15]:

- Primarily, OmniLedger should have a strong security feature to group validators in each shard. The robustness of its security parameters a Sybil-attack, OmniLedger has partially incorporated PoW or PoS within its ecosystem.

- Secondly, OmniLedger must reduce the likelihood of corrupted shard formations. Most importantly ensure that they are bias-resistant.
- Thirdly, as being said, OmniLedger must ensure the atomicity of transactions within disjoint shards.

Since its core functionalities and boundaries have been established, OmniLedger introduces its foundational scalability and protocol route map for its transaction model [15]:

- OmniLedger introduces decentralized ledgers from sequence of ordered blocks within which there would be sequence of transactions
- OmniLedger has adopted the Unspent transaction output ($UTXO$) scheme to structure its ledger "state"
- Each node in the network is supposed to crawl the global ledger of approved $UTXO$ and their local storage level such that the evolving of the creation of a new block could be validated.

For its validators network model, OmniLedger has adopted the same principle as Elastico [14].

1.6.3 RapidChain

The previous sharding concepts we have described have only considered sharding at Computation, and Storage level. Therefore, these sharding concepts could be seen as limited. This limitation had giving birth to a new sharding concept "RapidChain" [16] that was able to elaborate a full sharding. This approach is very innovative because it has the ability to process intra-shard consensus agreement using the appropriate consensus algorithm. That allows the Blockchain network to considerably scale with high throughput via block "pipelining" [16]. Most importantly, they have incorporated a new broadcasting technique for considerably large blocks. One of their main achievement is a provable reconfiguration system to enable the Blockchain network to reset its functionalities accordingly. These innovative representations should be structured in particular manners that could outperform previous concepts. These evaluations are the following [16]:

- **Sublinear Communication:** RapidChain is proven to be the first sharding technique that requires sublinear exchange of bits during each transaction.

- **High Resiliency:** From a BFT based protocol perspective RapidChain is the first sharding techniques that pushes the boundaries to tolerate less $\frac{1}{3}$ corrupted nodes within its ecosystem.

- **Rapid Committee Consensus:** RapidChain transactions propagation mechanism over a $P2P$ network structure is faster in the interval of $3-10$ than the previous sharding techniques we have explained.

- **Secure Reconfiguration:** RapidChain has included the notion of reconfiguration in which the sharded network would reshuffle such that some nodes could be assigned to another shard during the reconfiguration. Reshuffling the sharded network could be very costly, so RapidChain used the Cuckoo rule [22] to protect its system against any $Byzantine$ adversaries. Also, the reconfiguration of the parameters could be done in such a way that nodes could join and leave without the interruption of the protocol execution within the network.

- **Fast Cross-Shard Verification:** RapidChain incorporated an innovative technique such that nodes in the sharded network would only acquire a small portion of the information within the global Blockchain for Cross-Shard validation.

- **Decentralized Boostrapping:** RapidChain allows open membership in permissionless Blockchain ecosystem without randomizing the membership feature.

RapidChain has defined its transactional procedures following the principle of Bitcoin Blockchain [1]. In other words, the transactions are sent to RapidChain protocol by an external user account to the protocol [16]. For instance, sets of transaction would be divided into disjoint blocks, let say k. Let assume that X_{ij} represent the $j-th$ transaction the $i-th$ block [16]. Let a protocol PI output a set X containing k disjoint subsets of nodes such that shards $X_i = x_{ij}$, for every $j \in \{1..|X_i|\}$ such that the following properties could stand [16]:

- $Agreement$: For every $i \in \{1..k\}$, $\Omega(log(n))$ validators nodes agree on X_i

- $Validity$: For every $i \in \{1..k\}$ and $j \in \{1..|X_i|\}$, $g(x_{i,j}) = 1$.
- $Scalability$: k grows linearly with n, the number of nodes.
- $Efficiency$: at each node the computation and communication complexity is $o(n)$. Also, the storage complexity at each node is $o(s)$, where s is the number of transactions.

With this conceptual structure RapidChain was able to improve in terms of performance. Their performances strategies is via pipelining as follows [16]:

- RapidChain creates a principle such that leader in each shard can create a new block at the time of re-proposing the headers of pending blocks. Using this strategy increases the network throughput.
- The protection of honest nodes is done through a strategy in which when a node accepts of a header it would not propagate it because there is a high probability that the header is correct.

With all these conceptual representations RapidChain has pushed the sharding concept to an appropriate level that could be exploited to create new sharding concept.

1.6.4 Spontaneous Sharding

As we could observe, there is a sequential evolution of the sharding concept within the Blockchain ecosystem. However, these concepts of sharding we have previously defined could have some limitation that could affect the efficiency of the Blockchain network. For instance, they could suffer from the deterioration of the decentralization and the effectiveness of the Byzantine fault tolerance [12]. As a result of that, they have come out with another structural representation of sharding call "spontaneous sharding".

Giving the thoughts of this structural realization, some fundamental concerns could arise because the goal here is to enable a sufficient "scale out" of the Blockchain network [12]. A scale-out could be permissible if $g = o(N)$ where g is the size of a specific shard and N the number of nodes in the shard [12]. The questions they have tried to answer are the following:

- The major aspect of transactions of values in Blockchain platform
- The possibility to enable the sharded Blockchain network to scale-out during transactional activities without deteriorating its reliability or its distributed representation.

Since answering these questions triggers their curiosity, they have pushed the boundaries to come out with a new design for their "Value-Transfer Ledger" model as follows [12]:

- Honest participants in the network would try to prove the authenticity of their activities within the Blockchain platform
- "Rational" recipients would only consider transactional movements that only impact their received transactions

Putting all the above constraints in one set, during a communication a transactions is passed from one node to another node. A proof is associated to that specific transaction such that the size of the proof increases with the number of node it traverses [12]. According this principle, the transaction would cycling in a small shard instead of the entire network to minimize the transmission cost. As a result of that, the network is naturally sharded by maintaining its distributed aspect and security.

1.7 Research Motivation

We realize that the most important factor that is impacting the scalability within a Blockchain network is the processing at each node. Enabling a fast transactional approach at each node could positively impact the performance of the network.

1.7.1 New Conceptional Representation of Sharding

We have decided to conceptually design a Blockchain in which we are going to combine the components of the Blockchain network. The structural components that we would take into account at each validator node are the following:

- Computation
- Storage
- Communication

Since the Blockchain network requires a node to gather information at each node creating a storage, those components are tightly related in many senses. For instance, when a transaction is performed from a node to another node within a sharded network, the transshipment's nodes would play the roles of transaction validators by doing some processing. The processing is the possibility to verify the validity and the provenance of the transaction. Therefore, the validator node would need to check its records by fetching the record from its storage. Assuming that the storage is a list of records that need to be scanned. We could see that the complexity of the computation would depend on the size of the records that need to be scanned. As a result, the amount of computation would depend on the size of the storage. We could observe that the time it takes to perform the computation at the validator nodes would negatively impact the latency in network. The size of the delay would be dependant of the computation and at each validator node. An increase on the delay in the network would be an increase on the time it takes to perform the communication within the network; therefore, it raises of the communication cost.

Speaking of sizes, we have decided to design an innovative way to design sharded Blockchain network, in which, we would conceptualize the the computation, the storage and the communication as proof size pz at each validator node.

1.7.2 Research Questions

In this research, we have been asking ourselves some fundamental questions on the structural representation of Blockchain network. Those questions could be related to the flow of information in the network:

- What is the main factor that could considerably impact the trajectory of the transaction within the sharded Blockchain network?

- Is it possible to design a system such that the transaction trajectory could avoid any network congestions at validator nodes? Such that the sharded network could significantly scale.

We are going to formulate these concerns into a mathematical model that would allow us to perform an implementation such that some manipulations could be done within the network helping us to give answers to those questions.

1.7.3 Our Contributions

The main contribution in this research project is the ability to come up with new conceptual representation of a sharded Blockchain network to perform some simulations to understand our assumptions. The main contributions would be:

- The creation of an optimization network model to analyse the flow of transactions within our network structure that is defined in **Chapter 4**.
- The assignment of random proof sizes to nodes in the network to evaluate their influence on the flow of the transactions within the network.
- Partitioning the network into shards to evaluate the effectiveness of the concept of sharding the network.

1.8 Definitions

Definition 1. Transaction from one node to another node.
The notion of transaction is the value that is sent from a sender node to a receiver node traversing a set of nodes to create a transaction path. In the network structure, the sender node would be represented as supply node and the receiving node would represented as demand node.

Definition 2. Proof size at each node
The proof size pz is the amount of computation that would perform at each validator node to validate a specific transaction. In [12], validator nodes in the network would try to minimize the

cost of the transmission for the transaction by trying to minimize the amount of the cost of proving the validity of the transaction. For instance, let $node_1$ and $node_2$ be two nodes in the network. $node_1$ would try to initiate a transaction with $node_2$. To minimize the communication cost, the transaction would use the validator nodes that already have proof about $node_1$ such that the cost of the validation could be reduced.

CHAPTER 2
PRACTICAL BLOCKCHAIN APPLICATIONS

In this chapter, we are going to explore some practical applications of the Blockchain ecosystems. These applications could be in various domain such us cryptocurrency, smart contracts, private Blockchain, distributed storage system. Those applications are showing the usability of the Blockchain system in the real world. These applications could have a tremendous impact in our societal environment.

2.1 Bitcoin

Bitcoin is the first Blockchain network that has initiated the concept of cryptocurrency. The realization of Bitcoin Blockchain network is a combination of cryptographic techniques within a distributed ecosystem [41]. For instance, the mixture of the following functionalities has enabled them the functioning of the Bitcoin ecosystem [41]:

- An overlay peer-peer network
- A public record ledger which refers to the Blockchain
- The establishment of some consensus rules for transaction validations
- Some algorithmic procedures that enable agreement within a decentralized platform (PoW algorithm)

2.1.1 Practical Functionality of the Bitcoin Network

Since the Bitcoin is a distributed ecosystem without the central authority property that enables the executions of transactions. The network needs to relay on some particular techniques. For example, as we have profoundly discussed in **Chapter1**, the execution of transactions is done by validator

nodes. In other words, the validators nodes in the network needs to guarantee the validity of transactions by discarding invalid and malformed transactions [41]. When the consensus iteration is performed, a newly block should be created such that transactions could be inserted in the new created block. Since the Bitcoin network is a permissionless Blockchain network, any nodes can join and leave the network at any moment. For instance, Bitcoin Blockchain validator nodes are spread out across the globe. In here, Bitnodes is a platform that is showing the operations of Bitcoin network validator nodes from different continents in real time. Those validators are the core foundation of the Bitcoin network. In the next steps, we are trying to illustrate some practical usability of the Bitcoin Blockchain platform.

2.1.2 Bitcoin Wallet

Figure 2.1 [45] would illustrate a structural representation of the Bitcoin Wallet. For sake of simplicity, we are giving an overview of the representation of the Bitcoin wallet.

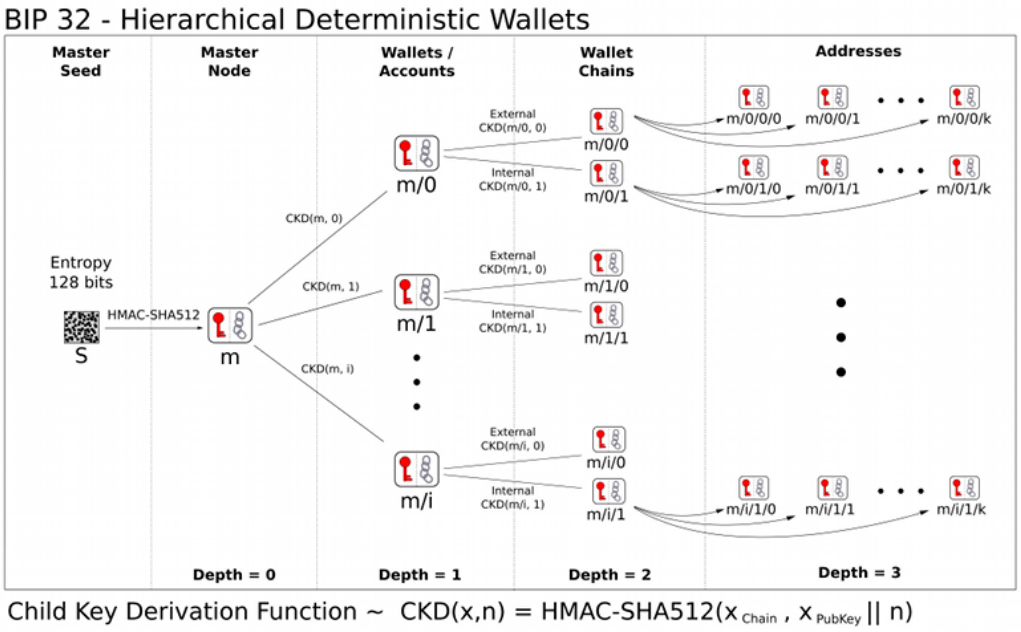

Figure 2.1: Hierarchical Deterministic Wallets [45][43]

In the physical world, one could use a wallet to gather their assets such us some Dollars, Euros, etc. The same logic could be also applied in the logical world, in which Bitcoin users would have

digital wallets to save their digital tokens. for instance, a Bitcoin Wallet is the association of the Bitcoin address with the private key associated to the wallet [42]. There are multiple methods of wallet generations in [42], they have given a client side application to generate a desired wallet. With the evolution of technological devices, different types of Bitcoin wallets have been created to adapt to the use of these devices [41]: Globally, a wallet is structured to store cryptographic private keys.

- **Desktop full client**

  This type of wallet was the first of type of wallet that was introduced for desktop operating system(OS) such as Windows, and Mac OS. For instance, Desktop wallets provide some autonomy in terms of usages. In [43], they had shown some advantages of using desktop wallets. Since the keys are stored within in local storage, the wallet desktop application can directly access the keys without require some additional authentication. Additionally, sufficient numbers of keys could be stored in the local hard disk because the size of the keys could be negligible at some extent. Lastly, the Bitcoin application can automate the key generation without the need of additional inputs from the user. However, Desktop wallet could present some security issues if they are not well configured [41]. For example, applications that would have access to the keys folder could be a threat if they could perform a read of the folder. Most importantly, a malware attack on the OS could be damaging for the keys [43]. In 2011, Symantec had reported a malware that could steal Bitcoin tokens [44].

- **Mobile wallet**

  With the tremendous evolution of smart phones, another mostly used wallet is a mobile wallet. For instance, this type of wallet could be found in different types of smart phones OS such as Android and Apple iOS. The security issues with the Desktop wallet could be also applied here.

- **Web third party wallets**

  With ubiquitousness of web applications has influenced the immersion of web wallets. Some

organizations have online web wallets such that users could use them as software service. The user could open an account in which the Bitcoin keys could stored. As we could observe, this type of structural representation of wallets could be assimilated to online banking [43]. The dangerous aspect of type of service is that it is creating a sens of centralized ecosystems in which when the hosting platforms are vulnerable from malicious attacks, their customers may lose their keys. Consequently, some thoughts should be made before using these types of applications.

2.1.3 Transaction Representation

In the Bitcoin Blockchain network, a transaction is a transfer of Bitcoin value that would be broadcasted across the $P2P$ overlay network. In other words, a specific transaction input could be the outputs of precedent transaction outputs [55]. As we could observe, a transaction is composed of inputs and outputs. **Figure 2.2** would illustrate the graphical representation of how transactions are structured in the Bitcoin Blockchain network [55]:

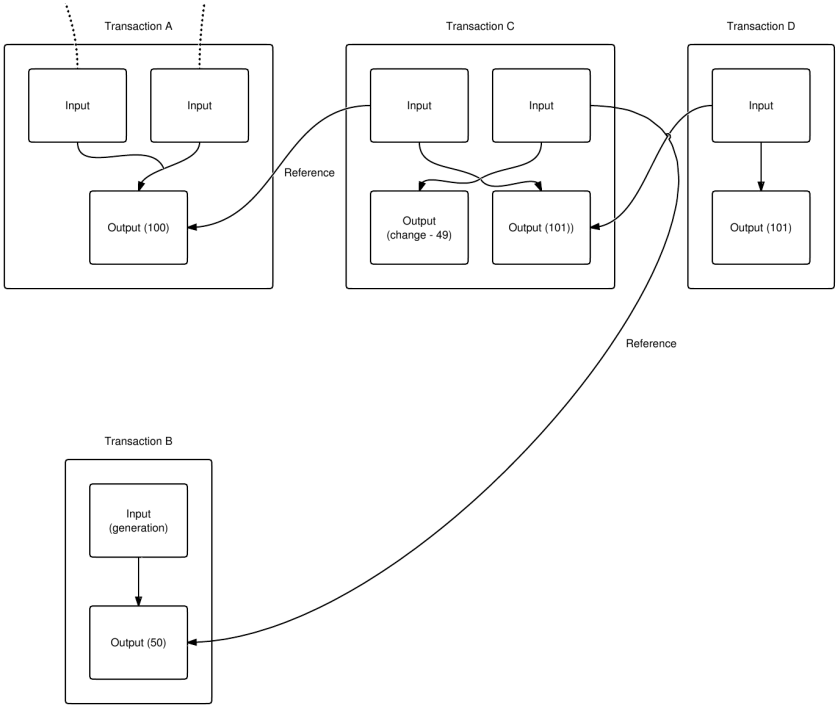

Figure 2.2: Transaction Outputs and Inputs

Transaction Input

As we could observe in **Figure 2.2**, a transaction could be represented by an input. The input of the transaction could be the output of previous transaction outputs. For instance, all transaction outputs that would reference a specific transaction input would sum up their transaction values. Most importantly, previous transactions would be referenced by their hash such that each transaction input would have an index [55]. Each transaction would be signed using some scripting procedures. The script of each transaction would have a signature and a hash of the public key of the interested output transactions.

Transaction Output

The output of a transaction is the second part of a transaction that is related to the input of the transaction. For instance, an individual transaction output would ensure the unicity of transactions inputs that would reference it. Transaction outputs and inputs are interrelated. On a practical settings, if the transaction input is worth 100 Bitcoins (BTC) and only 50 BTC could be spent, the system would create two outputs of 50 BTC such that one half would be spent and the other half would be in possession of the sender [55].

Transaction serialization

The main purpose of the Bitcoin Blockchain ecosystem is the ability to order transactional procedures such that they could be in the perfect order. The serialization of the transactions is a subset of this principle. For instance, when transactions are broadcasted across the network, they need to be structured using customized data structures such the transactional information could be transmitted one byte at the time [41]. In [46], they have given a regular transaction structure such that every transaction has a version ($nVersion$), an input vector (vin), and an output vector ($vout$), and a date that would indicates the occurrence of a specific transaction. **Figure 2.3** illustrates the data structure in which transactions are structured [46]:

Field name		Type (Size)	Description
nVersion		int (4 bytes)	Transaction format version (currently 1).
#vin		VarInt (1-9 bytes)	Number of transaction input entries in *vin*.
vin[]	hash	uint256 (32 bytes)	Double-SHA256 hash of a past transaction.
	n	uint (4 bytes)	Index of a transaction output within the transaction specified by *hash*.
	scriptSigLen	VarInt (1-9 bytes)	Length of *scriptSig* field in bytes.
	scriptSig	CScript (Variable)	Script to satisfy spending condition of the transaction output (*hash*,*n*).
	nSequence	uint (4 bytes)	Transaction input sequence number.
#vout		VarInt (1-9 bytes)	Number of transaction output entries in *vout*.
vout[]	nValue	int64_t (8 bytes)	Amount of 10^{-8} BTC.
	scriptPubkeyLen	VarInt (1-9 bytes)	Length of *scriptPubkey* field in bytes.
	scriptPubkey	CScript (Variable)	Script specifying conditions under which the transaction output can be claimed.
nLockTime		unsigned int (4 bytes)	Timestamp past which transactions can be replaced before inclusion in block.

Figure 2.3: Transaction Data Structures

Transaction Fees

In the Bitcoin Blockchain network, the validation of transactions is done by validator nodes called $miners$. Transaction fees is the process of incentivation of the mining processes to motivate miners to participate on the consensus effort (i.e, PoW: **Section 1.2.1**). For instance, a miner who mines a block would be rewarded with some coin ($Bitcoin$). In [41], they mentioned that transaction fees are calculated based on the size of the transaction in Kilobytes rather than the value of Bitcoin. As a result of that, transaction is based of market force of the network.

We have given a brief description of some Bitcoin Blockchain network components. The Bitcoin Blockchain network itself could be subject of an intensive research.

2.2 Ethereum

Ethereum is the second generation of the Blockchain ecosystem that has introduced the notion of smart contract. For instance, Ethereum is programmable Blockchain ecosystem that enables Blockchain developers to deploy various Blockchain applications.

2.2.1 Ethereum Wallets

The concept of wallets in the Ethereum Blockchain network could be assimilated to the Bitcoin Blockchain network wallets structure. For instance, wallets could be in multiple forms in different platforms. In the Ethereum Blockchain network, we could also have the following wallets [49]:

- **Paper Wallets**

  This type of wallet could identify as the safest type of wallet for Ether tokens. For instance, since this type of wallet is not stored within an electronic devices, it could be could safe from some malicious attacks. A paper wallet could be generated on MyEtherWallet. During the creation of the key, a private and public keys would be associated as $QR - codes$ to guarantee its uniqueness and secrecy. The problem with this type of wallet is that there could be some security issue because how the wallet could be downloaded without a computer and an Internet connection.

- **Mobile Wallets**

  This type of wallet could be assimilated to the Bitcoin mobile from the application perspective. More information about this type of wallet is given in **Section 2.1.2**.

- **Desktop Wallets** We have given more information about this type of wallet in **Section 2.1.2**

- **Hardware Wallets**

  These type of wallets could be defined as portable storage of cryptocurrency. For instance, they could be in form of specialized hard disks for cryptocurrencies that can be easily plugin to a computer to perform some transactional procedures. From that perspective, hardware wallets could be secured from some cyber-attacks.

2.2.2 Ethereum Transactions

As we have briefly given description in **Section 1.1**, the structure of transactions in the Ethereum Blockchain could have different representations. For instance, a transaction could be defined as

a set of cryptographic signatures owned by external account and structured in the correct data structure such that it could be submitted to the Ethereum Blockchain network [47].

There two types of transactions within the Ethereum Blockchain network: message calls, and contract creations. As the Bitcoin Blockchain network, a transaction in Ethereum Blockchain network has some specific specifications as the following [47]:

- **Nonce:** a unique set of numbers that identify a specific transaction.
- **gasPrice:** The amount of token that the sender of the transaction is willing to invest for the validation of the transaction.
- **gasLimit:** The maxmum amount of token that the sender of the transaction agrees to pay before the transaction is sent.
- **to:** This field would indicate the destination of the transaction.
- **value:** The value that is sent during the transactional procedure
- **v,r,s:** This is a special signature that identifies the sender of the transaction.
- **init:** It is a specific piece of code used to initialize the Ethereum Virtual Machine (EVM) to initiate a new contract.
- **data:** The data could reference the IP address and domain name of a specific account.

Figure 2.4 shows the representation of a transaction within the Ethereum Blockchain ecosystem [47]:

Ethereum Transaction Executions

Ethereum transaction execution is done differently from Bitcoin Blockchain perspective. The amount of computation a validator node could perform is determined by the number of Gas the validator is willing to use. For instance, a Gas could be defined as a fuel for to engage on some validation procedures. Additionally, like in the Bitcoin network the GAS could be defined as

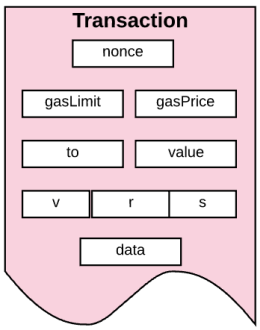

Figure 2.4: Ethereum Transaction Structure

method of incentivation in the Ethereum Blockchain network. The currency used in the Ethereum Blockchain network is called "ETH". As as a result, the Gas fee would be paid in ETH. **Figure 2.5** shows how the Gas is utilize for a specific transaction [47]:

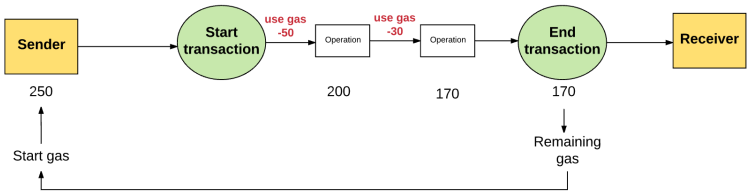

Figure 2.5: Gas Utilization for a Transaction

For instance, the miner who executed the transaction would be rewarded for the computation power used such that the transaction could be validated. Most importantly, Gas fees are not only limited to the execution of transactions. For instance, when transactions are executed, they need to be stored. As a result of that, Gas fees could be also used as storage fees [47].

We could observe that computation and storage at each validator nodes are the main method of incentivation of the Ethereum Blockchain network.

Since Ethereum is programmable Blockchain platform, different types of account could be plugin to its ecosystem as the following [47]:

Of course, there could be some internal mechanisms that would enable the exchange of information between account.

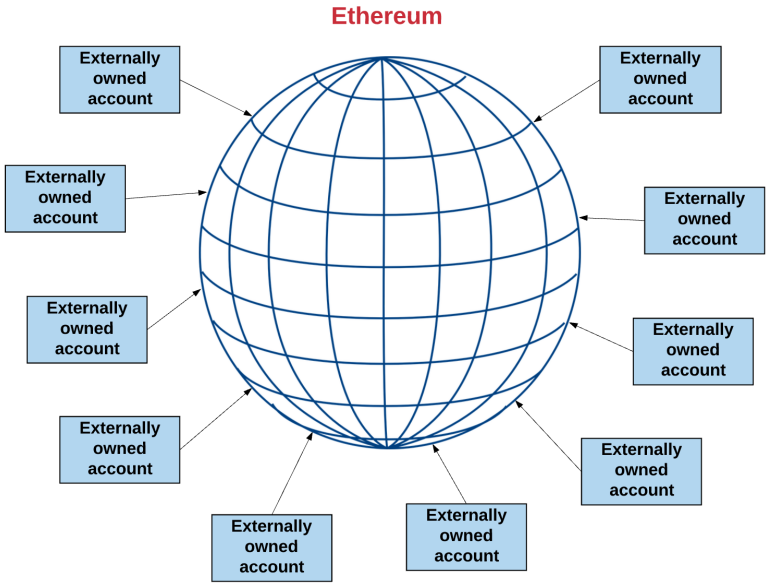

Figure 2.6: External Accounts Plugin to the Ethereum Ecosystem

2.2.3 Different Between **BTC** and **ETH**

The major differences between both cryptocurrencies is more conceptual. For instance, the generation of BTC is approximately halves each four years; however, there is not a variation on the generation of ETH. In other words, the generation of ETH follows a constant evolution [48]. **Figure 2.7** is showing the generations of both currencies.

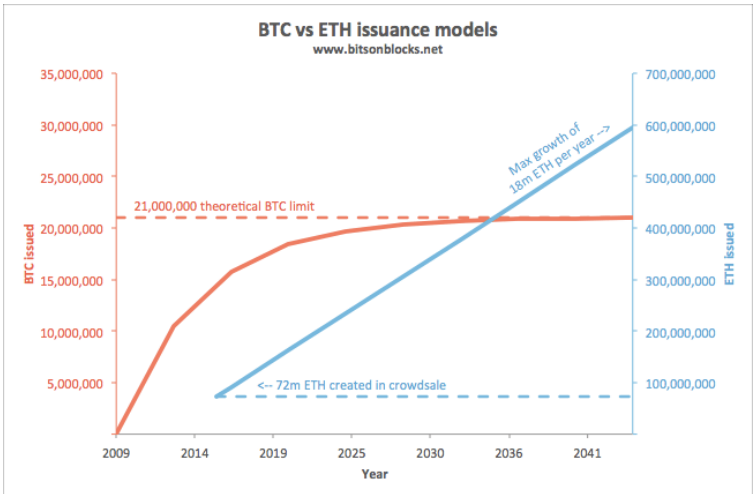

Figure 2.7: BTC vs ETH issuance models [48]

2.3 Hyperledger

The creation of Ethereum has opened new spectrum of Blockchain applications because it has pushed the Blockchain technology to another level as we have introduced in **Section 1.1.2**. From that perspective, an organization like IBM has seen a huge opportunity to exploit this design initiative to create a private Blockchain network in which businesses and industries could cooperate using smart contracts. The core functionalities of Hyperledger are the following [50]:

- Smart Contracts
- Digital Assets
- Record repositories
- A decentralized consensus-based network
- Cryptographic security

Most Hyperledger applications are based on smart contracts. For instance, a smart contract could be defined as a set of rules incorporated into a Blockchain network such that businesses can perform some business processes following those rules. In other words, a smart contract gives the ability to automate transactional processes between different entities removing any kind of intermediaries [50]. Most importantly, Hyperledger products could operate within permissioned ecosystem, they have customized their consensus procedures. For instance, they have integrated modular consensus platform such that their customers could decide the consensus algorithm they want to run to perform their transactional activities [50].

IBM has built multiple Hyperledger products to adapt to any aspect of the private Blockchain market.

2.3.1 Hyperledger Fabric

Hyperledger Fabric is the commonest Hyperledger product in use in most industries and businesses. This is due to the fact that Hyperledger Fabric incorporates some essential properties such

as confidentiality, flexibility, resiliency, and scalability [51]. For instance, Hyperledger Fabric permits the hosting of special smart contracts called ”chaincode” that businesses use to elaborate smart contract rules using customized consensus algorithms.

Additionally, Hyperledger Fabric could enable parallel executions of various applications built using different technology stacks. For instance, those applications could use different cryptocurrencies. This variousness flexibility is due to the fact that Hyperledger Fabric operates on non-deterministic smart contracts. From architectural perspective, Hyperledger Fabric would have two fundamental portions [52]:

- **A smart contract:** the chaincode is core part of the Hyperledger Fabric ecosystem because it enables the execution of distributed application.
- **An endorsement policy:** An endorsement policy acts could be defined as library for transactions' validation within the Fabric ecosystem. For instance, the endorsement policy could give some instructions to the chaincode to perform some transactional procedures on the endorsers of a specific transaction.

We could observe that, Hyperledger Fabric incorporates broad functionalities that could trigger some interest of various business applications.

2.3.2 Hyperledger Burrow

As we have previously explained Hyperledger products offer a modular consensus framework such that the consensus process could be customized. For instance, Hyperledger Burrow was designed to enable some particular consensus processes. Hyperledger Burrow provides deterministic smart contract based the Ethereum Virtual Machine (EVM). As a result of that, Hyperledger Burrow could perform the following functionalities [51]:

- **Consensus Engine:** This consensus engine permits the ordering of transaction across multiples nodes in the Blockchain network.

- **Application Blockchain Interface (ABCI):** This platform enables applications connected Hyperledger Burrow to trigger the right consensus.
- **Smart Contract Application Engine:** With the deterministic smart contract functionality of Hyperledger Burrow, developers could integrate challenging industrial applications to its ecosystem.
- **Gateway:** The gateway is an interface that enables the integration of numerous type of systems.

In global, Hyperledger Burrow offers deterministic smart contracts.

2.3.3 Hyperledger Indy

Hyperledger Indy is another product of Hyperledger that aims for some particular functionalities. For instance, Hyperledger Indy is distributed ledger that enables the creation of digital identities within a Blockchain network ecosystem. Therefore, Hyperledger Indy enables a clear identification of interacting peers across the network by using their digital identities [51]. As a result, we could enumerate the following functionalities of Hyperledger Indy: Self-sovereignty, privacy, verifiable claims [51].

2.3.4 Hyperledger Iroha

Hyperledger Iroha is also distributed ledger that enables the incorporation of various projects. For instance, Hyperledger Iroha main feature is the creations of portable applications for end users. Its main features are the following [51]: A simple structure, domain-driven C++ design, a chain-based BFT consensus algorithm named "Sumeragi".

2.3.5 Hyperledger Sawtooth

Hyperledger Sawtooth is also one of the commonly used Hyperledger product. One of its essential features is that it enables organizations regrouped in consortiums to make selfishness decision

about the type of Blockchain applications they would want to run within the Hyperledger Sawtooth ecosystem [51]. Therefore, Sawtooth has incorporated the following properties in its ecosystem [51]:

- **Dynamic Consensus:** This gives the possibility to exchange consensus algorithm at any given moment.
- **Proof of elapsed time (PoET):** This a modified version of PoW that requires less computation power.
- **Transactions Families:** This property gives the possibility to programmers to write smart contracts using their prefer programming languages.
- **Compatibility with Ethereum Contracts:** The combination of Ethereum application with Hyperledger Sawtooth could be possible.
- **Parallel Transaction execution:** Hyperledger Sawtooth enables parallel execution of transaction for better performance.
- **Private transactions:** Sawtooth could be configured to permit a high privacy level of transactional procedures.

The rise of Hyperledger has opened the applicability of the Blockchain Technology to a broad range of applications to facilitate the elaboration of trust between businesses and industries.

2.4 Filecoin

Filecoin is a distributed storage system using the Blockchain ecosystem principles.

2.4.1 The Interplanetary File System: IPFS

IPFS is distributed file system that operates within peer-to-peer network setup [53]. IPFS has built its ecosystem functionalities around previous peer-to-peer file system such as Distributed Hash

Table (DHTs), BitTorrent, Git, and Smart File System (SFS) [53]. IPFS is designed to take an innovative approach such that decentralized ecosystem could handle a large volume of data. Most importantly, IPFS was designed to give a decentralized approach to the web. Nodes within the IPFS ecosystem would be setup in a way that each node in the network would have a local storage for internal processing. Nodes are supposed to have the same privileges to avoid unbiased distributed network structure. These kind of design principles are technically similar to how the Blockchain ecosystem is setup.

Most importantly, $IPFS$ uses some fundamental properties to enable its functionalities. Since it is a peer to peer ecosystem, each node in the network should be identified by identity. $IPFS$ has introduced the notion of identities, in which nodes in the network would have a "Nodeid"[53]. That identification is the cryptographic hash of the node public key. Each node is responsible for storing their private and public key [53]. To maintain their integrity the $IPFS$ system has put an incentivation mechanism to reward nodes within the system.

The network structure of $IPFS$ uses particular protocol to permit the communication between nodes within the system. For instance, most of the communication within the $IPFS$ ecosystem is done over the internet. Therefore, nodes in the network could communicate with million of other nodes by maintaining integrity using the ICE NAT traversal technique. Moreover, message authentication is dong using $HMAC$ with the sender's public keys [53]. To find other peers within the network, $IPFS$ uses a routing method that would map other peers within the network from DHT references.

Putting all these together, they enhance $IPFS$ to a distributed file storage creating a virtual currency at the same time.

2.4.2 Enhancement of IPFS to Filecoin

$Filecoin$ is a distributed file system built on top of $IPFS$. In other words, $Filecoin$ is decentralized storage ecosystem using some algorithmic procedures to monetize cloud storage [54]. From this perspective, the $Filecoin$ ecosystem could be defined as a Blockchain platform such that

some consensus algorithm could be deployed to manage the network functionalities. The $Filecoin$ Blockchain has used a modified version of the mining principle of the Bitcoin Blockchain network. For instance, $Filecoin$ validator nodes (miners) would participate in competition to mine storage blocks that are computationally proportional to the capacity of the storage [54]. This operation could be lucrative to the miners because they could gather enough storage capacity that was rented to possible clients.

Therefore, the $Filecoin$ ecosystem elaborates interactions between different parties such as clients, storage miners, retrieval Miners. For instance, the clients pay to store and retrieve their data from the $Filecoin$ Blockchain network. The storage miners get some incentives for providing their storage. Finally, the retrieval miners get rewarded by making the data available to clients [54]. Those interactions are made possible by running the right consensus algorithms, and networking protocols.

Proof-of-Spacetime

The main protocol that operates within the $Filecoin$ Blockchain ecosystem is Proof-of-Spacetime ($PoSt$). For instance, $PoSt$ is used to check the proof of validity that of a specific data within a specific interval of time [54]. The principle behind the $PoSt$ is the following [54]: a prover P must convince a verifier V that P is storing data D within a time slot t. The following figure illustrates the execution of $PoSt$.

More specification about this protocol is given in [54]. This protocol enables the coordination of processes within the $Filecoin$ Blockchain network.

The Network

The network aspect of the $Filecoin$ Blockchain is structured in specific manner. For instance, the network is the skeleton of protocol executions, and management. Participants nodes within the $Filecoin$ Blockchain network would keep track of available storage, validate pledges, and audit the storage proofs. These processes would be saved within an immutable ledger [54]. This Blockchain

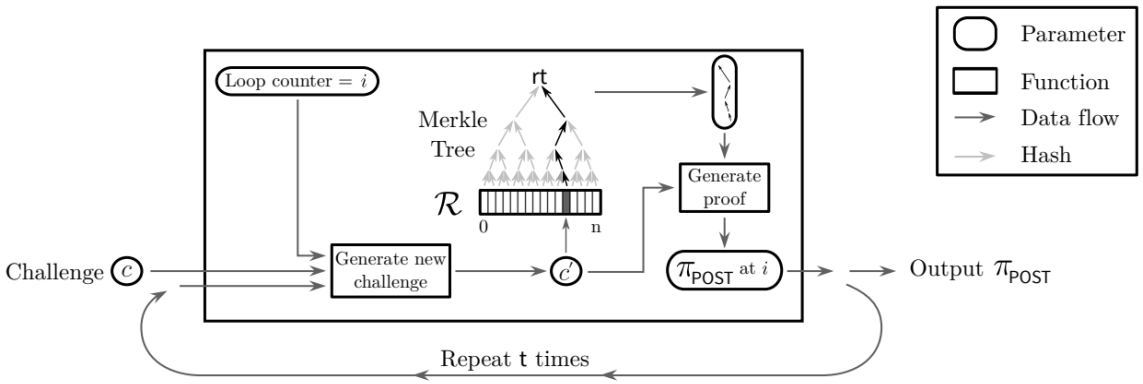

Figure 2.8: Illustration of Underlying mechanism of PoSt [54]

network structure has stimulated a lot interests creating an interesting Blockchain storage market.

Filecoin Market

The $Filecoin$ Blockchain platform has created two important markets:

1. **Storage Market**

   With the dramatic production of data created by the increasing trend of technological evolution. The storage market is in need of innovative approach such that the huge volume of data could be efficiently saved. As a result of that, the $Filecoin$ Blockchain platform offers the opportunity to potential clients, the management of their data in different aspects. The $Filecoin$ storage market has been designed by taking some particular approaches to meet certain requirements [54]:

   - **In-chain oderbook:** This requirement makes sure that available storage capacity and cost information is made public such that clients could make beneficial decisions for their own interests. Most importantly, occurrences in the network should be gathered in the oderbook such that accurate information could be presented to the network.
   - **Participants committing their resources:** This procedure is to make sure that storage miners and clients could prove the availability of their resources before proceeding to

any operations.

- **Self-organization to handle faults:** This procedure is to make sure that the network proof correctness for every action that happen within the ecosystem.

2. **Retrieval market**

   The $Filecoin$ Blockchain network enables the storage of information at the same time the retrieval of information is monetized. For instance, any nodes in the $Filecoin$ ecosystem could be a retrieval miners for some $Filecoin$ tokens. The retrieval within the $Filecoin$ Blockchain ecosystem do not have to store any data. Also, some requirements should be met to permit those functionalities:

   - **Off-chain oderbook:** Clients would request a retrieval process from retrieval miners without going through the $Blockchain$ network. This is done to boost the network efficiency.
   - **Retrieval without trusted parties:** This procedure ensures that, all the retrieval miners are honest. For instance, the processes between the clients and retrieval miners is done in a way that miners could get paid for the piece of information they able to retrieve. At the same time, the clients would receive the data according to the amount of tokens they have invested.
   - **Payments channels:** The payment channel is making sure that interactions clients and retrieval miners are preprocessed before any retrieval actions. The $Filecoin$ Blockchain network enables off-chain payment to enables efficient transactional processes. As such, the $Filecoin$ Blockchain ecosystem optimizes its efficiency for the satisfactory of its users.

The main goal of chapter is to illustrate the practicability of the Blockchain ecosystem. We have shown that the creation of a specific Blockchain platform could inspire the creation of other Blockchain platforms. For instance, from the creation of the Bitcoin Blockchain network has stimulated the creation the Ethereum Blockchain network. The creation of the Ethereum Blockchain

platform has stimulated the creation of private Blockchain networks such as various Hyperledger products. Most importantly, $Filecoin$ has taken another approach to use the Blockchain ecosystem in an innovative way.

CHAPTER 3
SCALABILITY WITHIN A BLOCKCHAIN ECOSYSTEM

The notion of scalability within the Blockchain ecosystem has attracted the interest of researchers. As we have explained, the use of the Blockchain platform at large scale requires scalable Blockchain platforms. We are going to examine various Blockchain network designs that aim for the improvement of scalability within the Blockchain ecosystem.

3.1 Previous Approach of Scalable Blockchain Platforms

Some structural Blockchain networks have been designed to enable scalable Blockchain platforms.

3.1.1 OffChain Technique

The off-chain technique was introduced to address the scalability issues with the Bitcoin Blockchain platform. For instance, in the Bitcoin Blockchain, transactions need to be broadcast to all the nodes in the network, so if the Bitcoin Blockchain network needs to be used to replace all the financial services such us VISA, and Master Card. This propagation of transactions across the network would create a huge volume of data that could not be managed [33]. In [33], They argue that to address the scalability issue within the Bitcoin Blockchain ecosystem, the system needs to limit some specific procedures. For instance, nodes in the network may only be interested on information that impacts their state such that the broadcasting of information could be reduced.

In addition, in response to the scalability issue in Bitcoin, another approach was taken to enable the creation micropayment channels such that a particular entity in the network could send a significant amount of information without affecting the network efficiency [33]. For instance, the micropayment channels could be structured in a way that put different parties to initiate transactional procedures such that only a single information about this transaction is recorded within the Blockchain network [33]. Using this off-chain technique could enable the network significantly

scale because nodes in a specific channel would only interact with nodes in its entity creating less overhead in the network. Although this technique could present some sufficient scalability features, it could introduce some weaknesses that could be exploited because of the isolation from the global network state.

3.1.2 Side-chain Technique

Due to some limitations of the off-chain technique, another technique was elaborated namely "side-chain". The side-chain technique performs in a particular way. For instance, a side-chain could work on a subset of transactions by only tracking values associated to those transactions [4]. By doing so, the network throughput could significantly increase because multiples of side-chains could be created through network. Since only portions of transactions are processed within a side-chain in the network, the nodes in the side-chain only track information about transaction they are interested in. Most importantly, the side-chain approach guarantees that the value that is carried by a particular transaction would not be double spent across the network. For example, special proofs are associated to each transaction during cross-chain transactions so that each side-chain could verify the provenance and validity of the transaction [4].

This type of representation enables the reconfiguration of the Blockchain ecosystem. This reorganization ensures that the independence of each side-chain within the Blockchain network [34]. In other words, side-chains in the network could be illustrated as a subset of blockchains within the Blockchain. From this understanding, when a value is transferred from one side-chain to another one, a transaction is created from the initial side-chain such that the value associated the transaction would be locked enabling the receiving side-chain to cryptographically prove that the cryptographic signature was correctly signed by the initial the side-chain [34]. This verification permits the side-chains to introduce a trusted communication channel without the altering the Blockchain efficiency.

The side-chain presents some interesting features that could improve the throughput within the Blockchain network. This technique could have some limitations that could damage some

properties of the Blockchain ecosystem.

3.2 Why those approaches are still not performing well

The Blockchain network models we have illustrated have some drawbacks that could significantly affect some fundamental parameters within the Blockchain network. For instance, the off-chain technique works by isolating the execution of a specific transaction across network to increase the network throughput. This kind of procedure could introduce some fundamental issues [4]. For instance, since a transaction is not broadcast across the network, there is a relaxation of the consensus level. By doing this relaxation, transaction values could be double spent [4]. Most importantly, the off-chain technique is creating a sense of centralized Blockchain network.

The side-chain technique could create some security issues. For instance, since each side-chain independently executes transaction without the need of agreement of the whole Blockchain network, an attack of a side-chain would not be detected by the other part of the network [35]. For instance, this security parameter could create a sense of isolation of the side-chain from the global network; therefore, the side-chain could be defined as a centralization of subset of chains in the Blockchain network. Moreover, the expectations of designing a side-chain Blockchain ecosystem could present some structural challenges. For instance, in the side-chain ecosystem, the consensus protocol in the receiving side-chain does not have the possibility to observe the ”state” changes in the sender side-chain. This situation creates a ”non-interactive” consensus proof of occurrences of the side-chains [4].

3.3 The age of Sharding

Since the previous scalability schemes have shown some limitations. Different approaches need to be taken to encounter these limitations enabling the network to effectively scale without compromising the core properties of the Blockchain ecosystem such as decentralization, and security. Therefore, the need of a scalable structure within the Blockchain ecosystem is crucial because the usability of the Blockchain platform at a large scale won’t be possible without it. Adopting the

sharding concept in the Blockchain ecosystem could considerably improve the network throughput. For instance, the sharding concepts gives the ability to divide network into different sectors so that processing of transactions could be done in parallel. However, some fundamental question may arise. In [36], we have come with some essential questions:

1. How can security be maintained within a shard?
2. How can the fundamental property of a Blockchain, which is decentralization, be maintained within a collection of nodes?
3. How can fairness be maintained within the whole Blockchain ecosystem after the network has been sectored into small portions?
4. How can scalability and accuracy be assured during inter-shard communication?

These questions should be seriously taken into account due to the fact they are essential and fundamental to the core concept the Blockchain technology. For instance, the usability of the Blockchain networks could produce some technical concerns if those questions are not taken into consideration to ensure the robustness of the Blockchain platform.

3.3.1 Sharding Security Maintainability

As we have seen with the off-chain and side-chain techniques, isolating the part the network could be risky. For instance, some security parameters could be violated putting the whole Blockchain system in danger. Therefore, maintaining security withing a sharded Blockchain network requires some careful analysis. For instance, in [37], they illustrated a security issue called ”shard takeover attacks”. In this type of attack, some malicious nodes may try to attack some validator nodes such that they could get the control of a specific shard to prevent the executions of valid transactions [37]. To prevent this type of attacks, some specific measures need to be taken. For instance, in the Ethereum Blockchain network, they use a method called ”random sampling”. In this randomized method, validator nodes are chosen within a specific interval of time such that malicious nodes

would not have the possibility to be part of this randomized set of validator nodes [37]. Therefore, a random creation of shards could be a way for sharding security. Most importantly, to fight against any adversaries within a specific shard variant, the BFT consensus would be run to protect the network.

3.3.2 Maintainability of Decentralization within a Sharded Blockchain Network

The decentralization is one of the core property of the Blockchain ecosystem and any kind of distributed system. For instance, the Blockchain network eliminates the sense of centralized system. Therefore, making sure that the sharded Blockchain follows the same properties could be crucial. For instance, since each shard in the network needs to maintain the same properties, the network should be evenly partitioned. In [14], they have enabled a sharding procedure that is making sure the computing power proportionally distributed across each shard. In addition, the maintainability of decentralization by ensuring that each shard within the sharded network would have appropriate connections to other shards in the network. This aspect is to maintain inter-shard communication by preventing shards isolation. For instance, in [15], they introduced the notion ”cross-shard” transactions in which a specific transaction input could have different outputs in other shards creating shard ”interoperability”. From that perspective, the decentralization aspect of the sharded Blockchain could be maintained because the shards are interrelated at some extend.

3.3.3 Maintainability of Fairness within a sharded Blockchain Ecosystem

The consensus level is the central authority to ensure fairness within a decentralized platform such us the Blockchain network.The main goal of sharding the Blockchain network is the possibility to reduce the overhead of the consensus level. The most appropriate consensus algorithm that should be used within a sharded Blockchain is BFT based consensus algorithm because they could be run within a small subsets of nodes. BFT based consensus because it could be used to eliminate any kind of $Byzantine$ behaviors. More description is given about this type of consensus in **Section 1.2.4**.

3.3.4 Scalability and Accuracy Within a Shard

The combination of scalability and accuracy could create a dilemma within the sharded Blockchain network. In other words, scalability without accuracy would be meaningless. As a result, the atomicity of transactions in a sharded platform could be very essential for the consistency of the information across the network. In a traditional Blockchain network, the execution of a specific transaction uses validators across the whole network. This type of execution is putting the burden at the consensus level by negatively impacting the network efficiency in terms of throughput. As we have explained sharding the Blockchain network is a possibility to randomly partition the Blockchain network such that multiple transactions could be run in parallel within each specific shard that enables the network to considerably scale. One may ask how accuracy could be maintained? Since multiple transactions have been executed simultaneously. For instance, the shards in the network must be disjoint because a set of nodes in specific shard must differ from other shards. This property enables the shards to have a disjoint ledgers for the executions of their transactions [15]. These disjoint ledgers would sequentially recorded in the global state such that the network could keep track of occurrences within the sharded platform.

3.4 Conceptual Understanding of Sharding

Sharding could be done at different levels. The partitioning of levels should appropriately done so that the network could efficiency scale. The sharding conceptualization needs to make sure that those main components are structured maintaining the Blockchain network properties.

3.4.1 Computation

The computation at each validator node depends to the storage at that node. For instance, the validation of a specific transaction at a validator node requires the evaluation of previous transactions. This process is required because the validity of the transaction depends on it. Therefore, sharding the computation power within a sharded ecosystem could be very crucial for the scalability in the

network. in [14], they have used a sharding concept to ensure proportional distribution of computation across the sharded network. Doing so, Elastico could scale up its transactional throughput in accordance with its computational capacity [15]. We have given more details about Elastico protocol in **Section 1.6.1**.

3.4.2 Storage

Within a Blockchain setting the ledger represents the storage at each node. Therefore partitioning the Blockchain corresponds to partitioning the global ledger. Each shard within the sharded Blockchain network must have its individual ledger to keep track of transactional procedures within its ecosystem. In [16], when a node joins a shard, the node needs to update its storage by downloading the ledger in this shard. This is to make sure that new transactional procedures could be validated. In global, sharding at the storage level could be fundamental since the storage is included at each node. For instance, to enable efficient storage procedures, in [16], RapidChain adopts a strategy in which nodes only acquire a sufficient amount of information to guaranty the integrity of the network.

3.4.3 Communication

The communication is the interaction of nodes within the Blockchain networks. As we have been explaining sharding allows parallel execution of transactions. For instance, each transaction would be executed within its specific shard reducing its propagation across the network. As a result of that, the communication cost through the network could be sufficiently reduced permitting the network to scale. In [16], they have used particular approach that is the following: a nodes gathers information about its neighboring nodes which have the shortest path to other nodes within the network. Therefore, when a node tries to communicate through the sharded network, it would try to minimize the cost as much as possible using the shortest path. Our concept of shortest path could be fairly related to this principle at some extent.

3.5 Sharding Consensus Algorithms and Functionalities

BFT based consensus algorithms are mainly used within a shared ecosystem. The evolution of the Blockchain technology has triggered the evolution of its consensus algorithms. These evolution efforts aim the scalability of the Blockchain ecosystem.

3.5.1 BFT

The BFT needs certain requirements to be met such at the sharded could satisfy the BFT properties. Those fundamental properties are the following [12]:

- **Agreement (Consistent):** Two validator nodes should not have a disagreement on the validations of transaction.
- **Validity (Correctness):** Validator nodes should not validate invalid transactions.
- **Termination (Liveness):** Validator nodes should be informed about eventual transactions within the network.

These properties should be maintained such that BFT consensus could efficiently operates. Most importantly, most of the BFT based consensus algorithm could function within asynchronous systems using replicated state machine [38]. For instance, each entity (node) in the network would have a copy of the state machine for the execution of each transaction in the sharded network. In [39], they found that BFT protocol relies most likely on the authenticity of nodes in the network rather than the number of processes that need to be executed at these nodes. Therefore, the main objective of the BFT is to discard any malicious attempt in network. The core foundation of the BFT protocol could be found in **Section 1.2.4**.

3.5.2 PBFT

The $PBFT$ is built on top of the BFT. We could say that an amelioration of the BFT protocol. In [7], the $PBFT$ is an optimization of the BFT algorithm that could also operate within an

asynchronous network setting. This enhancement of the BFT protocol has introduced some of the following phases [40]: a pre-prepared phase in which a designated "leader" in the network send transactional requests to other nodes within a sharded. The second phase is the prepared phase in which nodes in the sharded network agree on set of transactions. The last phase is the commit in which nodes within which nodes in the shard commit transactions to their storage. All these interactions could take up $O(N^2)$ where N is the number of nodes in the network [40]. One of the reason that $PBFT$ protocol is used within a sharded ecosystem is that it allows pipelined execution [40][16].

3.5.3 HotStuff

$HotStuff$ is an improved version of $PBFT$ because it has added some procedural phases to $PBFT$ consensus [10]. Therefore, it could have more advantages over the previous BFT protocol. The $HotStuff$ consensus protocol enables the ability to simplify the BFT properties we described in **Section 3.5.1**. $HotStuff$ protocol decouples those parameters such that executions of processes could be done in parallel increase the sharded network throughput by reducing the latency [11].

3.5.4 LibraBFT

$LibraBFT$ consensus is an extension of the $HotStuff$ algorithm. They have incorporated the following functionalities [11]: proofs of safety, liveness, and optimistic responsiveness. The $LibraBFT$ created an innovative way to execute transaction at the validator nodes. For instance, the consensus can take some robust measures such that validator nodes can be resistant to $Byzantine$ errors by collectively approving blocks instead transaction stream [11] within a sharded ecosystem. The most important factor is that the protocol could be resistant against some critical security issues such as a denial-of-service and sybil attacks within a specific shard in the network. Such a protocol could beneficial within sharded ecosystem because its robustness creating an efficient and secure Blockchain platform.

CHAPTER 4
NETWORK OPTIMIZATION MODEL

For the implementation of the network structure, we have modeled a directed Blockchain network with n nodes and M edges to minimize the Communication Cost per Transaction (CCPT) within a shard. The shard would have a collection of specific nodes that sartorially close to each other in the Blockchain network. The nodes in the network would have a proof size such that the nodes in the network are assumably honest because it is a permissioned Blockchain platform.

When a transaction moves through network, the nodes in the network would try to validate the transaction. The network would have the origin (supply node) and destination node (demand node) during the transaction. The other nodes in the network would play the role of transshipment nodes. The transshipment nodes that would be traversed by the transaction flow are supposed to have small proof sizes (pz). The edges (arcs) in the network would have the same communication cost. Since the objective is to minimize the CCPT within the network, the transshipment nodes that would be traversed would be included along with their proof sizes on validating the transactions and would determine the trajectory of the transaction. The trajectory of the transaction would create the transaction path. The optimization model would try to determine the minimum cost to transact from one node to another node by applying some constraints that we would define accordingly.

We would assign random values as proof sizes to the transshipment nodes within the sharded network. These proof sizes could be considered as the validation costs for the transshipment nodes. By randomly assigning the proof sizes to the nodes, we are trying to simulate how a transaction is transmitted across the network. With our network model, we are conceptually evaluating how transactions are manipulated within a Blockchain ecosystem. The random property would permit unbiased results which give estimates of the transaction cost calculations within the Blockchain network. To accomplish this implementation, we formulate an optimization problem whose resolution could permit the satisfaction of these properties.

4.1 Notations

We represent a Blockchain network as a directed graph $G = (V, E)$ where $V = \{V_1, V_2, ..., V_n\}$ is the set of nodes in the network and $E \subseteq V \times V$ is the set of directed arcs. Let $TX = \{tx_1, tx_2, \ldots, tx_m\}$ be an independent set of transactions (i.e., tx_{i+1} is initiated after tx_i is completed.) performed in the network during a given time interval T. Let $V_{s_i}, V_{d_i} \in V, s_i \neq d_i$ be the source and destination nodes for the transaction tx_i, $1 \leq i \leq m$. A given transaction tx_i elaborates a communication between the two nodes, V_{s_i} and V_{d_i} in the network. We assume the proof size of a node may dynamically change from a transaction to another transaction. For the transaction tx_i, let $pz(i, j)$ be the proof size of node V_j. We will have $pz(i, s_i) = pz(i, d_i) = 0$ because the source and destination nodes would not validate themselves during transactional procedures. The proof size assignment to the nodes except V_{s_i} and V_{d_i} in the network would be randomized so that we could unbiasedly evaluate the propagation of the flow across the network from V_{s_i} to V_{d_i}. Figure 4.1 illustrates the proof size assignment to the network.

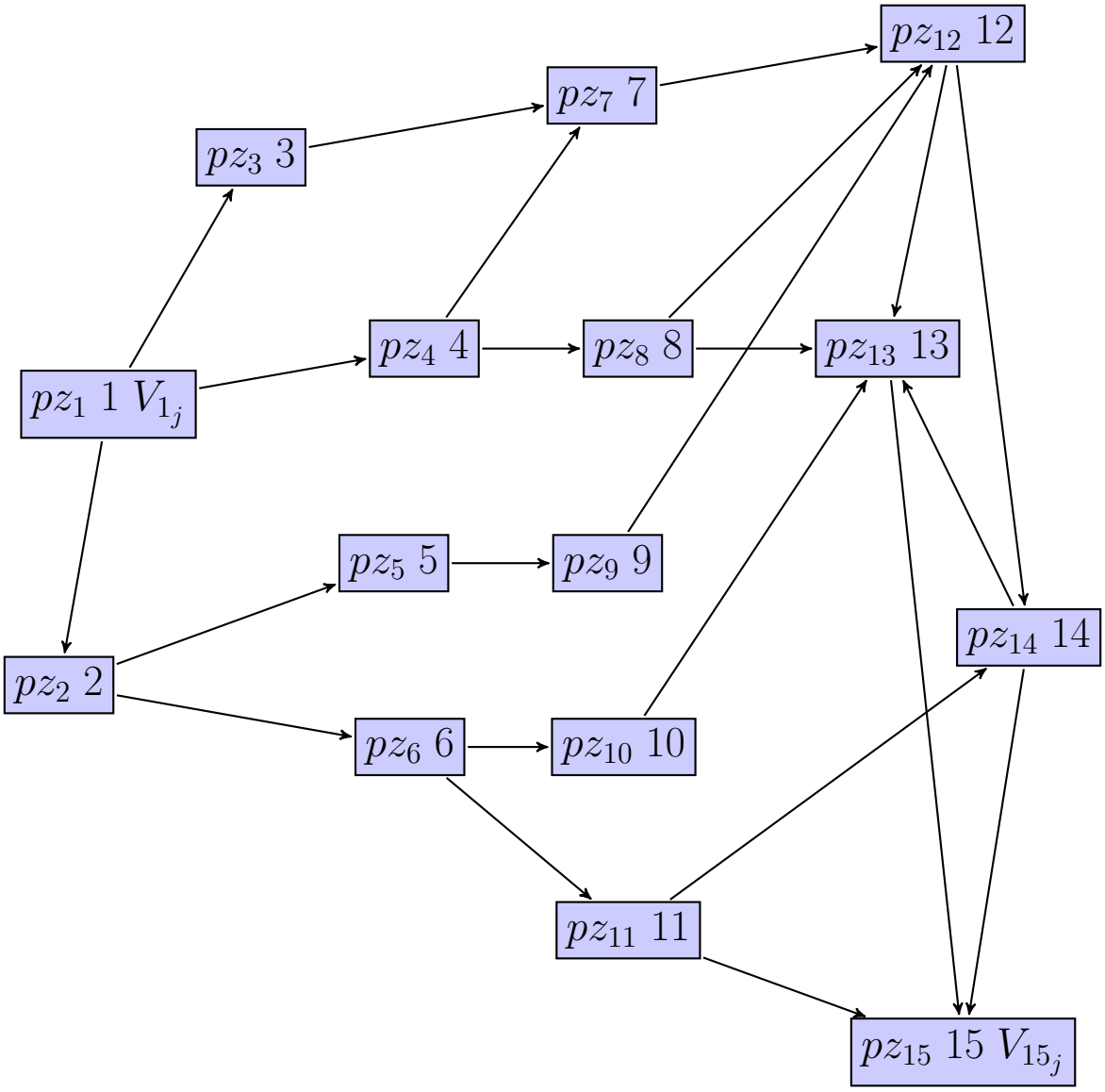

Figure 4.1: Network Structure with Proof Sizing Representation: pz_i would be randomly assigned to the nodes during stimulation of the $CCPT$. V_1 and V_{15} are the supply($source$) and demand($destination$) nodes respectively.

4.2 Mixed Integer Programming Model

To realize our implementation, we would define a directed graph that would represent our Blockchain network. As defined in the previous section, we have V as a set of vertices and E a set of arcs in the network. Therefore, the graph notation could be illustrated as $G = (V, E)$. We would have

a supply vector $S = (S_i)$ for V, and fixed cost and proof sizes of two connected nodes would be represented as f_{ij} and pz_i, pz_j for $(i, j) \in E$ respectively. Using these representations, we need the model to find a set of nodes with small proof sizes within the network that minimizes the communication cost per transaction $(CCPT)$ at the same time ensuring the shortest path to the destination.

We have defined the transaction trajectory variables as binary variables. Transshipment nodes that would be used for the transactions would be represented by 1 and the other nodes that do not participate in the transaction procedures would be represented by 0. Therefore, this model could be defined as a mixed-integer programming model. The communication cost C_{ij} for traversing a specific arc could be calculated using the summation of half of the outbound and the inbound nodes proof sizes making sure that the flow is perfectly conserved during the transaction adding the fix cost at each arc. We have decided to use a fix cost at each arc because the purpose of this evaluation is to test the influence of the proof size at each validator node. Therefore, we could have the following constraints:

$$min \sum_{(i,j)\in E} C_{ij}x_{ij} \tag{4.1}$$

$$s.t.$$

$$\sum_{(i,j)\in E} x_{ij} - \sum_{(j,i)\in E} x_{ji} = S_i \tag{4.2}$$

$$C_{ij} = (\frac{1}{2}) * (pz_i + pz_j) + f_{ij} \tag{4.3}$$

$$x_{ij} \in \{0, 1\} \quad \forall \quad (i, j) \in E, \quad f_{ij} = 1 \quad \forall \quad (i, j) \in E \tag{4.4}$$

In our implementation, the model would try to minimize the objective (1) using the flow conservation constraint (2) and the cost of traversing an arc constraint (3). As additional notation, we

use the following notion for the supply vector S_i for a transaction tx_k.

$$S_i = \begin{cases} 1 & \text{if } i = s_k \\ -1 & \text{if } i = d_k \\ 0 & \text{otherwise} \end{cases} \tag{4.5}$$

The supply and demand nodes would be randomly chosen to minimize the transaction cost within the network.

4.3 Network Model Extension with Multiple Transactions

4.3.1 Dummy Arcs

The network model we have created for transactional processes is the first phase of our modeling. The extension of our work is the possibility to extend the model to handle multiple transactions at same time within the sharded network. Let us define the following network to illustrate our conception.

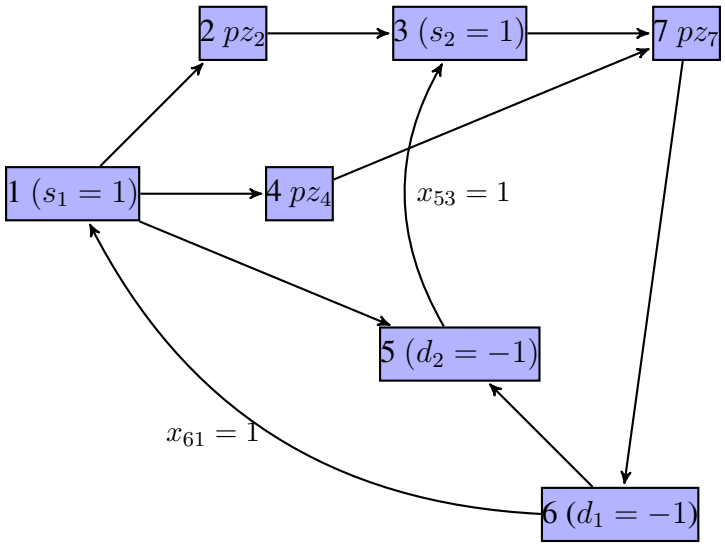

Figure 4.2: Network Extension with Multiple Transactions

In order to handle multiple transactions some particular procedures need to be taken. We could observe from **Figure 4.2** we have the following set of transactions tx_1 (s_1, d_1), and $tx_2(s_2, d_2)$

with their respective sources, and destinations. Therefore, to ensure that those transactions could be performed, we must introduce dummy arcs. For instance, we have x_{16}, and x_{35} such that $x_{ij} \leq cap_{ij}$ where cap_{ij} is the capacity at each arc. For the dummy arcs, we have $cap_{ij} = 1$. Adding these dummy arcs would ensure that the transactions could be sent to the right destinations ensuring that the flow is conserved at each arc.

4.3.2 Dummy Nodes

The executions of multiple transactions could correctly performed by using dummy nodes. Since the nodes that the flow would go through would play the role of transshipment nodes, some restrictions need to be introduced such that a certain number of transactions could traverse a specific transshipment node. In order to enable this principle, we could use dummy nodes to integrate these restrictions. For instance, when the number of transactions exceed what is required for a specific transshipment node, the surplus transactions would be broadcast to the defined dummy nodes. The following illustration is showing how this could be handled.

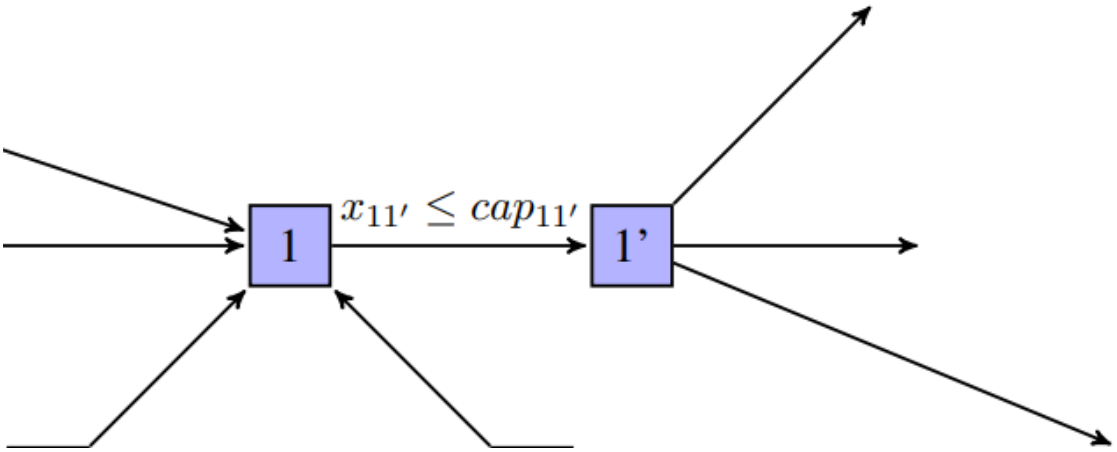

Figure 4.3: The Concept of Dummy Node

With this representation, let us suppose that the capacity of the arc coming from node 1 is $cap_{ij} = 2$. This will limit the maximum number of transactions that can be processed at a given time to 2.

4.4 Shortest Path Consideration

The main objective of the network model is to minimize the cost of sending a flow from a source node to a destination node. Most importantly, since the arc cost in the whole graph is the same, if a specific flow utilizes nodes in the network that have appropriate proof sizes, the calculation of the communication cost would be the association of f_{ij} and $pz(i,j)$ at the nodes. Therefore, minimizing the $CCPT$, the optimization model is minimizing the number of nodes and arcs that need to be processed from the source node to the destination node. As a result of this modeling, the $CCPT$ could be minimized such that set of participating nodes could be found from those nodes. **Figure 4.4** and **4.5** would illustrate the shortest path concept.

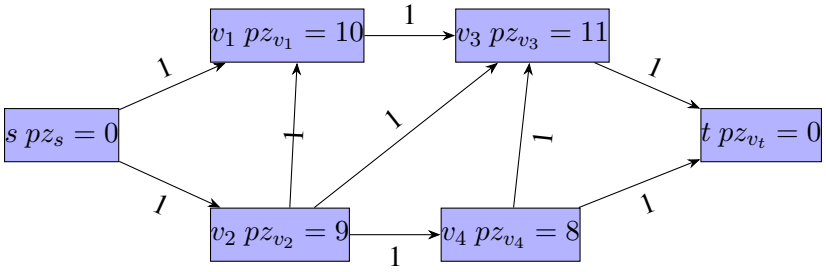

Figure 4.4: Shortest Path Illustration from s to t

After the computations of the cost each arc using **equation 4.3**, we could obtain the following graph.

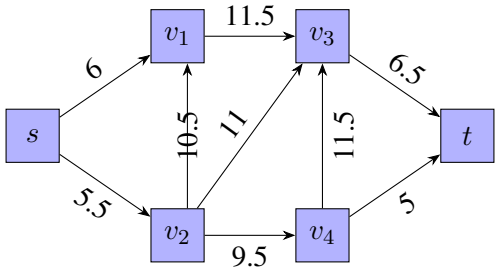

Figure 4.5: Computation of Transaction Costs at Nodes

Applying the network model we have the following shortest path.

Source	Dest	path	$CCPT$
s	t	s $\rightarrow v_2 \rightarrow v_4 \rightarrow t$	20

Table 4.1: Network Flow Shortest Path

4.5 Sharded Network Within the Network

Since the network would be sharded, the model needs to maintain the same properties within each shard to ensure the model effectiveness. Therefore, subsets of the network would be evaluated according to their structures. We have modeled our Blockchain network to make sure that all of the nodes within a shard are connected. Also, the shards in the network need to be connected because transactions could be done from shard to shard maintaining the network properties.

CHAPTER 5
IMPLEMENTATION AND RESULT

5.1 Implementation

For the implementation, we have used our optimization model to evaluate the sharding concept. We evaluate the model using the ILOG CPLEX Optimization Studio software. We use the Cplex solver to perform the stimulation.

5.1.1 Model Components

Using Cplex Optimization Studio, we developed a complex, yet elegant optimization model to test the sharding concept to simulate the concept of the propagation transactions across the Blockchain network. The model consists of two different files. For instance, one of the file is the data file incorporates the network nodes. Since it is a network structure, the nodes in the network would have some connections to other nodes. The nodes in the network are not considered to be part of a complete graph data structure. Rather, they are part of an incomplete graph. The nodes and their respective connections have been randomly generated by a script written by us in the Python programming language. Implementing the model using a complete graph will be left to future research. The other file that we have is the implementation file of our optimization model. Most importantly, Within this file we have written the logic to shard the network with proportional sizes. Each shard have a subsets of nodes connected by a subsets of arcs. The constraints that would be applied to the main network structure would be the same for each shard. We have defined the constraints in **Section 4.2**. We have used a range of nodes to be able to partition our network into shards. The partition is making sure that each shard would have a disjoint set of nodes. This property is very important because it could be the fundamental principle behind the concept of sharding. Most importantly, since the sharding principle is the possibility to partition the network

such that nodes could sectorially close to each other. We have decided to use a partition method that ensures this property. This concept could be fundamental to the minimization of the $CCPT$. Along with the shortest path and lowest cost, we have implemented sharding to further limit the total distance travelled by a transaction within a specific shard. By doing this, the network will expend the lowest energy possible when traversing the nodes for a transaction, further strengthening the argument behind the sharding concept.

5.1.2 Constraints

As if with every model, we have set a specific set of rules for the simulation of the Blockchain network. For this specific model, we have only implemented three constraints required for a transaction to traverse the sharded blockchain to reach its destination node. The first constraint is the objective constraint and the cost and the path constraint. This constraint calculates the total cost of traversing a sharded network from the source node to the destination node, all the while taking into account the proof size of each validator node traversed and the cost of the arc traversed, which are all the same in our model because we want to evaluate the impact of the proof size at each validator node. This is also called the objective constraint due to the fact that this constraint is the constraint being minimized to find the optimal solution for each shard, and the whole network. The second constraint is called the flow conservation constraint. This ensures the conservation of a specific from th source to the destination during a specific transactional procedure. It is implemented using the difference of sums of the constrained transaction path to find the truly minimized solution to the path taken by the model. The third constraint is the calculation of the proof size used during a transaction. This calculation is done in a specific manner because it considers half of the inbound and outbound proof sizes of the validator nodes. The combination those three constraints would ensure the optimization model to find an optimal solution. Most importantly, the model would find the shortest path from the source to the destination within the sharded network structure.

5.1.3 Sharding Implementation

The tricky part for the model was not figuring out the correct constraints, but figuring out how to split an already randomly generated Blockchain network into smaller parts or shards. We had to perform sharding all the while maintaining connectivity between the nodes in a respective shard. Using simple probability, we were able to deduce that a shard will most likely maintain connectivity if each node in the total network had at least seven connections to other nodes. When randomly generating our network, we made sure that the algorithm maintained at least seven different connections from each node to other nodes. By satisfying this rule, we were able to generate many different shards with different respective nodes within those shards and still maintain connectivity between the nodes. While performing the sharding, we created two different sets of nodes using different ranges such that we could have disjoint subsets of nodes. This helped us ensure the uniqueness of a shard and that no shard shared a duplicate node between them. Once sharding was complete, we perform some stimulation of the main network, and each specific shard to obtain some experimental results.

5.2 Result

5.2.1 Result in the Complete Network

To ensure an illustrative experiment result, we performed some stimulation on the whole set of nodes such that we could do a comparison on the minimization of the $CCPT$. During the evaluation on the main network, we have observed some satisfactory results. For instance, with random proof size assignment at validator nodes, the model was able to find the shortest path from the source to the destination minimizing the $CCPT$. As reminder, the purpose of the proof sizes at each validator is the stimulation of the amount of computation that needs to be done in real Blockchain network setup. We have explained the logic behind this principle in **Section 1.7.1**. For the evaluation of our network, we have decided to use 203 nodes. Performing some tests on these nodes, we have randomly chosen the supply(source) and demand (destination) nodes according to

the network model we have defined in **Section 4.2**. We obtain the following result in the main network:

Sender Node	Receiver Node	Transaction Trajectory	$CCPT$
26	166	$26 \rightarrow 32 \rightarrow 37 \rightarrow 43 \rightarrow 48 \rightarrow 52 \rightarrow 58 \rightarrow 61 \rightarrow 67 \rightarrow 77 \rightarrow 83 \rightarrow 83 \rightarrow 86 \rightarrow 93 \rightarrow 97 \rightarrow 102 \rightarrow 106 \rightarrow 112 \rightarrow 116 \rightarrow 122 \rightarrow 127 \rightarrow 132 \rightarrow 137 \rightarrow 143 \rightarrow 143 \rightarrow 147 \rightarrow 152 \rightarrow 157 \rightarrow 161 \rightarrow 166$	991

Table 5.1: Flow Stimulation in the Main Network

With some observation, we release that is the shortest path from the source to the destination with the lowest $CCPT$.

5.2.2 Results within the Shards

After doing the test in the main network with all the nodes, we have done some tests within each shard to make some observation. To test the sharding concept, we divided the network into four shards of approximate number of nodes such that we could unbiasedly perform our experimentation. Most importantly, we needed to make sure that the nodes within each specific shard are connected. Therefore, with the appropriate connection within each individual we could perform some stimulation to get some results on the transaction flow.

Sender Node	Receiver Node	Transaction Trajectory	$CCPT$
41	6	$6 \rightarrow 12 \rightarrow 17 \rightarrow 23 \rightarrow 28 \rightarrow 32 \rightarrow 37 \rightarrow 41$	391

Table 5.2: Flow Stimulation: Shard

Sender Node	Receiver Node	Transaction Trajectory	$CCPT$
80	96	$80 \rightarrow 83 \rightarrow 86 \rightarrow 93 \rightarrow 96$	138

Table 5.3: Flow Stimulation: Shard1

We could observe that the above stimulation that the $CCPT$ could be minimized within a specific shard. Now, let's perform the stimulation for shard2 and shard3 using the same approach.

Sender Node	Receiver Node	Transaction Trajectory	$CCPT$
107	134	107 $\rightarrow$ 112 $\rightarrow$ 118 $\rightarrow$ 122 $\rightarrow$ 127 $\rightarrow$ 132 $\rightarrow$ 134	210.5

Table 5.4: Flow Stimulation: Shard2

Sender Node	Receiver Node	Transaction Trajectory	$CCPT$
192	153	192 $\rightarrow$ 186 $\rightarrow$ 182 $\rightarrow$ 177 $\rightarrow$ 172 $\rightarrow$ 167 $\rightarrow$ 161 $\rightarrow$ 157 $\rightarrow$ 153	191

Table 5.5: Flow Stimulation: Shard3

Given that all the nodes in each shard are disjoint, the optimization model ensures that a path could not be found from one shard to another shard. Allowing that would imply a cross-shard communication which could introduce another level of complexity. This research is heavily focused on the impact of sharding the network to get some optimal results. We have observed the calculation of the $CCPT$ within the four shards. In a real Blockchain setup this process could be run in parallel, ensuring the executions of multiple transactions at the same time such that the network could scale in terms of throughout.

CHAPTER 6
CONCLUSION AND FUTURE WORK

6.1 Conclusion

The purpose of this research was to understand the fundamental principle behind the concept of sharding within the Blockchain network ecosystem. With a good understanding this concept, we were able to design a mathematical optimization network model to perform some simulations on the structure of network. These simulations have given some results on how the transactions follow shortest paths to get to the destinations minimizing the $CCPT$. Most importantly, during our tests, the transactions would traverse validator nodes with minimal proof sizes to get to their destinations. This experimental results illustrate our assumption that about the proof sizing concept at validator nodes. Within a practical Blockchain setup, the assumption is that the transactions flow would try to use validator nodes that would require less amount of computing power such that the network could scale.

Another perspective of this research is to illustrate the current usability of the Blockchain technology in different societal domain. In **Chapter 2**, we have exposed some practical applications such as in cryptocurrencies, smart contracts, and distributed storage systems. These applications are showing how the Blockchain ecosystem could be used to enable the interaction of different businesses.

6.2 Future Work

We have created mathematical optimization network model to perform our experimentation. We have implemented this network model using some optimization software such as ILOG CPLEX Optimization Studio, and GAMS. The results we have obtained could be considered as conceptual results. For future implementations, we are going to perform our experiment on a real network

ecosystem. For instance, we are going to configure a network of virtual instances in which virtual instance would be considered as nodes within a Blockchain network. The network would be partitioned into shards such that we could perform some parallel processing on each shard. For the proof sizing stimulation, the nodes that would be used as validator nodes would have some amount of computation such that we could evaluate the propagation of the information across a specific shard. The configuration of this setup would be done using Mininet.

BIBLIOGRAPHY

[1] S. Nakamoto, ”Bitcoin: A peer-to-peer electronic cash system”, 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[2] Buterin, V, ” A next generation smart contract decentralized application platform”, 2014. [Online]. Available: https://cryptorating.eu/whitepapers/Ethereum/Ethereum_white_paper.pdf

[3] Libra, ”An Introduction to Libra”, 2019. [Online]. Available: https://libra.org/en-US/wp-content/uploads/sites/23/2019/06/LibraWhitePaper_en_US.pdf

[4]] W. Wang, D. T. Hoang, P. Hu, Z. Xiong, D. Niyato, P. Wang, Y. Wen, and D. I. Kim, ”A survey on consensus mechanisms and mining strategy management in blockchain networks,” IEEE Access, vol. 7, pp. 22328–22370, 2018.

[5] Buterin, V., and V. Griffith. 2017. Casper the Friendly Finality Gadget. CoRR abs/1710.09437. https://arxiv.org/abs/1710.09437

[6] L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. ACM Transactions on Programming Languages and Systems, 4(3):382-401, July 1982.

[7] M. Castro, B. Liskov, ”Practical Byzantine Fault Tolerance”, 3rd OSDI, 1999.

[8] C. Cachin. Architecture of the Hyperledger blockchain fabric. In Workshop on Distributed Cryptocurrencies and Consensus Ledgers, 2016.

[9] https://tendermint.com/

[10] M. Yin, D. Malkhi, M. K. Reiterand, G. G. Gueta, and I. Abraham, “HotStuff: BFT consensus in the lens of blockchain,” 2019. http://arxiv.org/abs/1803.05069v4

[11] Mathieu Baudet, Avery Ching, Andrey Chursin, George Danezis, François Garillot, Zekun Li, Dahlia Malkhi, Oded Naor, Dmitri Perelman, and Alberto Sonnino. 2019. State Machine Replication in the Libra Blockchain. Technical Report. Calibra. https:

//developers.libra.org/docs/state-machine-replication-paper

[12] Z. Ren and Z. Erkin, "A scale-out blockchain for value transfer with sponteneous shardin g," CoRR, vol. abs/1801.02531, 2018. [Online]. Available: http://arxiv.org/abs/1801.02531

[13] Popov, S. The Tangle. [Online]. Available: https://assets.ctfassets.net/r1dr6vzfxhev/2t4uxvsIqk0EUau6g2sw0g/45eae33637ca92f85dd9f4a3a218e1ec/iota1_4_3.pdf

[14] L. Luu, V. Narayanan, C. Zhang, K. Baweija, S. Gilbert, and P. Saxena. A secure sharding protocol for open blockchains. In CCS, 2016.

[15] E. Kokoris Kogias, P. S. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. A. Ford, "Omniledger: A secure, scale-out, decentralized ledger via sharding," in iEEE Symposium on Security a nd Privacy (SP), San Francisco, CA, May 2018, pp. 583-593.

[16] M. Zamani, M. Movahedi, and M. Raykova, "RapidChain: Scaling blockchain via full sharding," IACR Cryptol. ePrint Arch., Tech. Rep. 2018/460, 2018. [Online]. Available: https://eprint.iacr.org/2018/460

[17] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, and Emin Gün. On scaling decentralized blockchains.

[18] "Whisper protocol v.5," https://github.com/ethereum/go-ethereum/wiki/Whisper-Overview, accessed: 08/24/2019.

[19] https://github.com/telehash/telehash.github.io. access: 08/24/2019.

[20] Ethereum Foundation, "Ethereum wire protocol v.5," https://github.com/ethereum/wiki/wiki/Ethereum-Wire-Protocol, accessed: 08/24/2019.

[21] P. Maymounkov and D. Mazières, "Kademlia: A peer-to-peer information system based on the xor metric," in Peer-to-Peer Systems: First International Workshop, Cambridge, MA, Mar. 2002, pp. 53–65.

[22] Siddhartha Sen and Michael J. Freedman. Commensal cuckoo: secure group partitioning for large-scale services. ACM SIGOPS Operating Systems Review, 46(1):33–39, 2012.

[23] King, S., Nadal, S.: Ppcoin: Peer-to-peer crypto-currency with proof-of-stake (2012).

[24] Jabbari, A., Kaminsky, P., 2018. Blockchain and Supply Chain Management. http://www.mhi.org/downloads/learning/cicmhe/blockchain-and-supply-chain-management.pdf

[25] https://www.hyperledger.org/wp-content/uploads/2019/02/Hyperledger_CaseStudy_Walmart_Printable_V4.pdf

[26] https://cointelegraph.com/news/does-crypto-always-mean-decentralizat

[27] https://medium.com/coinmonks/blockchain-scaling-30c9e1b7db1b

[28] Gervais, A., Karame, G.O., Wüst, K., Glykantzis, V., Ritzdorf, H., Capkun, S.: On the security and performance of proof of work blockchains. In: ACM CCS, pp. 3–16 (2016)

[29] M. Conti, C. Lal, S. Ruj et al., "A survey on security and privacy issues of bitcoin," arXiv preprint arXiv:1706.00916, 2017

[30] https://litecoin.org/

[31] Yonatan Sompolinsky and Aviv Zohar. Secure high-rate transaction processing in Bitcoin. In Financial Cryptography and Data Security - 19th International Conference, FC 2015, pages 507–527, 2015.

[32] I. Eyal, A. E. Gencer, E. G. Sirer, and R. van Renesse. (Oct. 7, 2015). "Bitcoin-NG: A scalable blockchain protocol." [Online]. Available: http://arxiv.org/abs/1510.02037

[33] POON, J., AND DRYJA, T. The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments, Jan. 2016.

[34] A. Back et al. (2014, Oct.). "Enabling blockchain innovations with pegged sidechains," Tech. Rep. [Online]. Available: http://www.blockstream.com/sidechains.pdf

[35] https://en.bitcoinwiki.org/wiki/Sidechain

[36] Yigo, A., AND Dehari, H. Scalable Sharded Blockchain Network, Apr. 2019. Avail-

ble: https://github.com/Nouldine/ShardedBlockchainNetwork/blob/master/Paper/ScalableShardedBlockchainNetwork.pdf

[37] https://github.com/ethereum/wiki/wiki/Sharding-FAQ

[38] F. Sun and P. Duan. (Sep. 2014). Solving Byzantine Problems in Synchronized Systems Using Bitcoin. [Online]. Available: https://allquantor.at/blockchainbib/pdf/sun2014solving.pdf

[39] A. Miller and J. J. LaViola Jr. Anonymous Byzantine Consensus from Moderately-Hard Puzzles: A Model for Bitcoin, 2014.

[40] Towards Scaling Blockchain Systems via Sharding, Proceedings of International Conference on Management of Data, pages 123–140, 2019.

[41] M Antonopoulos. 2017. Mastering Bitcoin - Programming the Open Blockchain. OReilly Media (2017). [Online]. Available: https://github.com/bitcoinbook/bitcoinbook

[42] "Bitcoin Wallet Generator". https://www.bitaddress.org/bitaddress.org-v3.3.0-SHA256-dec17c07685e1870960903d8f58090475b25af946fe95a734f88408html

[43] Eskandari, Shayan (2015) Real-world Deployability and Usability of Bitcoin. Masters thesis, Concordia University.

[44] Infostealer.Cointbit https://www.symantec.com/security-center/writeup/2011-061615-3651-99

[45] https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki

[46] https://github.com/minium/Bitcoin-Spec/blob/master/Bitcoin.pdf [47] Peethi K.(2017, September 2017). How does Ethereum work, anyway? [Blog post]. Retrieved from https://medium.com/@preethikasireddy/how-does-ethereum-work-anyway-22d1df506369

[49] "Guide on Ethereum Wallets: Mobile, Web, Desktop, Hardware". COINTELEGRAPH.

Available: https://cointelegraph.com/ethereum-for-beginners/ethereum-wallets

[50] Hyperledger Whitepaper

[51] An Introduction to Hyperledger

[52] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. Hyperledger fabric: A distributed operating system for permissioned blockchains. arXivpreprint arXiv:1801.10228, 2018.

[53] Benet, J.: IPFS - content addressed, versioned, P2P file system. CoRRabs/1407.3561 (2014), http://arxiv.org/abs/1407.3561.

[54] Protocol Labs (2017). Filecion: A Decentralized Storage Network.[Online]. Available: https://filecoin.io/filecoin.pdf

[55] https://en.bitcoin.it/wiki/Transaction

Appendices

APPENDIX A

IMPLEMENTATION OF NETWORK MODEL IN IBM CPLEX

This is the implementation of the network model in IBM CPLEX.

```
1
2
3  // Number of nodes in the main network
4  int NumNodes = ...;
5  range Nodes = 1..NumNodes;
6
7  // Get the supply (positive) and demand (negative)
8  // at each node
9  int SupDem[ Nodes ] = ...;
10
11 // Create a record to hold information about each arc
12 tuple arc {
13
14   key int fromnode;
15   key int tonode;
16   float cost;
17 }
18
19 // create a tuple for the proof size of object
20 tuple proof_size {
21   int p_1;
22 }
23
24 // Get the set of arcs
25 {arc} Arcs = ...;
```

```
26 //SupDem = [ Node ];
27 float ArcsCost[ Nodes ][ Nodes ];
28 proof_size proof[ Nodes ];
29
30 range shard = 1..50;
31 range shard_1  = 51..100;
32 range shard_2 = 101..151;
33 range shard_3 = 152..203;
34 {arc} shard_arcs;
35 {arc} shard_arcs_1;
36 {arc} shard_arcs_2;
37 {arc} shard_arcs_3;
38
39 int SupDem_shard[ shard ] = ...;
40 int SupDem_shard_1[ shard_1 ] = ...;
41 int SupDem_shard_2[ shard_2 ] = ...;
42 int SupDem_shard_3[ shard_3 ] = ...;
43
44 execute {
45
46   // Compute the cost to traverse
47   // a specific validator node
48   function ComputeCost( proof_1, proof_2 ) {
49      return ( 0.5 ) * ( proof_1.p_1 + proof_2.p_1 + 1 );
50   }
51
52   // Randomly assign proof size
53   // to nodes
54   for( var i in  Nodes ) {
55     proof[ i ].p_1 = Opl.rand(100);
56   }
57
58    // Store the processing cost
```

```
59   // at the inbound and outbound costs in a 2D array.
60   // The cost would be calculated at each connected
61   // node in the network. If there is not
62   // a connection between the nodes, the cost would be zero
63  for( var n in Arcs) {
64    ArcsCost[ n.fromnode ][ n.tonode ] = ComputeCost( proof[ n.fromnode ],
         proof[ n.tonode ] );
65  }
66
67  // Creation of the first
68  // shard arcs from its
69  // disjoint set of nodes
70  for( var i in shard ) {
71    for( n in Arcs ) {
72      if( i == n.fromnode ) {
73         shard_arcs.add( i, n.tonode, n.cost );
74
75      if( i == n.tonode ) {
76        shard_arcs.add( fromnode, i, n.cost );
77      }
78    }
79   }
80    }
81
82    // Create of the second
83    // shard from its disjoint
84    // set of nodes
85    for( var i in shard_1 ) {
86
87    for( n in Arcs ) {
88      if( i == n.fromnode ) {
89         shard_arcs_1.add( i, n.tonode, n.cost );
90
```

```
91      if( i == n.tonode ) {
92        shard_arcs_1.add( fromnode, i, n.cost );
93      }
94     }
95    }
96     }
97
98     // Create of the third
99     // shard shard from its
100    // disjoint set of nodes
101    for( var i in shard_2 ) {
102
103    for( n in Arcs ) {
104      if( i == n.fromnode ) {
105         shard_arcs_2.add( i, n.tonode, n.cost );
106
107      if( i == n.tonode ) {
108        shard_arcs_2.add( fromnode, i, n.cost );
109      }
110    }
111   }
112    }
113
114    // Creation of the Fourth shard
115    // from its disjoint set of nodes
116    for( var i in shard_3 ) {
117    for( n in Arcs ) {
118      if( i == n.fromnode ) {
119         shard_arcs_3.add( i, n.tonode, n.cost );
120      if( i == n.tonode ) {
121        shard_arcs_3.add( fromnode, i, n.cost );
122      }
123    }
```

```
124   }
125
126    }
127
128    // Since source and destination would randomly
129    // chosen, this function is making there
130    // within the right interval
131    function CheckBoundaries( low_bound, upper_bound ) {
132
133         var rand_value = Opl.rand(upper_bound);
134
135         while( rand_value < low_bound || rand_value > upper_bound ){
136
137            rand_value = Opl.rand(upper_bound);
138         }
139         return rand_value;
140    }
141
142    // Check the supply and demand are not the same
143    // Otherwise, do re-computation of one of them
144    function CheckSupplyDemand( supply_node, demand_node, low_bound,
          upper_bound ) {
145
146       var require_node = supply_node;
147       while( supply_node == demand_node ) {
148
149          require_node = CheckBoundaries( low_bound, upper_bound );
150       }
151       return require_node;
152    }
153
154    // Main Network
155  var supply_node = Opl.rand(NumNodes);
```

```
156
157     // Shard_ 0
158   var supply_node_shard= Opl.rand( 50 );
159
160   // Supply for Shard_1
161   var supply_node_shard_1 = CheckBoundaries( 51, 100 );
162
163
164   // Supply for Shard_2
165   var supply_node_shard_2 = CheckBoundaries( 101, 151 );
166
167
168   // Supply  for Shard_3
169   var supply_node_shard_3 = CheckBoundaries( 152, 203 );
170
171
172   // Demand for the main network
173   var demand_node = Opl.rand(NumNodes);
174
175   while( supply_node == demand_node ) {
176       demand_node_shard = Opl.rand(NumNodes);
177   }
178
179   // Demand for Shard_0
180   var demand_node_shard = Opl.rand(50);
181
182   while( demand_node_shard == supply_node_shard ) {
183       demand_node_shard = Opl.rand(50);
184   }
185
186   // Demand for Shard_1
187   var demand_node_shard_1 = CheckBoundaries(51, 100);
188   if( demand_node_shard_1 == supply_node_shard_1 ) {
```

```
189     demand_node_shard_1 = CheckSupplyDemand( demand_node_shard_1,
            supply_node_shard_1, 51, 100 );
190   }
191
192   // Demand  for Shard_2
193   var demand_node_shard_2 = CheckBoundaries(101, 151) ;
194
195   while( demand_node_shard_2 == supply_node_shard_2 || Opl.abs(
          supply_node_shard_2 - demand_node_shard_2 ) < 7 ) {
196     demand_node_shard_2 = CheckSupplyDemand( demand_node_shard_2,
            supply_node_shard_2, 101, 151 );
197   }
198
199   // Demand for Shard_3
200   var demand_node_shard_3 = CheckBoundaries(152, 203) ;
201   if( demand_node_shard_3 == supply_node_shard_3 ) {
202     demand_node_shard_3 = CheckSupplyDemand( demand_node_shard_1,
            supply_node_shard_1, 152, 203 );
203   }
204
205   // Main network
206   SupDem[ supply_node ] = 1;
207   SupDem[ demand_node ] = -1;
208
209   ArcsCost[ supply_node ][ supply_node ] == 0;
210   ArcsCost[ demand_node ][ demand_node ] == 0;
211
212   // shard_0
213   SupDem_shard[ supply_node_shard ] = 1;
214   SupDem_shard[ demand_node_shard ] = -1;
215
216   ArcsCost[ supply_node ][ supply_node_shard ] == 0;
217   ArcsCost[ demand_node ][ demand_node_shard ] == 0;
```

```
218
219   // shard_1
220   SupDem_shard_1[ supply_node_shard_1 ] = 1;
221   SupDem_shard_1[ demand_node_shard_1 ] = -1;
222
223   ArcsCost[ supply_node_shard_1 ][ supply_node_shard_1 ] == 0;
224   ArcsCost[ demand_node_shard_1 ][ demand_node_shard_1 ] == 0;
225
226   // shard_2
227   SupDem_shard_2[ supply_node_shard_2 ] = 1;
228   SupDem_shard_2[ demand_node_shard_2 ] = -1;
229
230   ArcsCost[ supply_node_shard_2 ][ supply_node_shard_2 ] == 0;
231   ArcsCost[ demand_node_shard_2 ][ demand_node_shard_2 ] == 0;
232
233   // shard_3
234   SupDem_shard_3[ supply_node_shard_3 ] = 1;
235   SupDem_shard_3[ demand_node_shard_3 ] = -1;
236
237   ArcsCost[ supply_node_shard_3 ][ supply_node_shard_3 ] == 0;
238   ArcsCost[ demand_node_shard_3 ][ demand_node_shard_3 ] == 0;
239 }
240
241
242 /*
243 dvar boolean path[ Arcs ];
244 dexpr float CCPT = sum( < i, j, fcost > in Arcs ) ArcsCost[ i ][ j ] * path[ <
        i, j, fcost > ] * fcost;
245
246 minimize CCPT;
247
248 subject to {
249
```

```
250   // Preserve flows at each node.
251   forall( i in Nodes )
252     ctNodeFlow:
253       sum( < i, j, fcost > in Arcs ) path[ < i, j, fcost  > ]
254      -  sum( < j, i, fcost > in Arcs ) path[ < j, i, fcost  > ] == SupDem[ i
            ];
255 };
256 */
257
258 /*
259 dvar boolean path[ shard_arcs  ];
260 dexpr float CCPT_1 = sum( < i, j, fcost > in shard_arcs ) ArcsCost[ i ][ j ] *
      path[ < i, j, fcost > ] * fcost;
261
262 minimize CCPT_1;
263
264 subject to {
265
266   // Preserve flows at each node.
267   forall( i in shard )
268     ctNodeFlow:
269       sum( < i, j, fcost > in shard_arcs ) path[ < i, j, fcost  > ]
270      -  sum( < j, i, fcost > in shard_arcs ) path[ < j, i, fcost  > ] ==
            SupDem_shard[ i ];
271
272 };
273 */
274
275 /*
276 dvar boolean path[ shard_arcs_1  ];
277 dexpr float CCPT_1 = sum( < i, j, fcost > in shard_arcs_1 ) ArcsCost[ i ][ j ]
      * path[ < i, j, fcost > ] * fcost;
278
```

```
279 minimize CCPT_1;
280
281 subject to {
282
283   // Preserve flows at each node.
284   forall( i in shard_1 )
285     ctNodeFlow:
286       sum( < i, j, fcost > in shard_arcs_1 ) path[ < i, j, fcost  > ]
287      -  sum( < j, i, fcost > in shard_arcs_1 ) path[ < j, i, fcost  > ] ==
           SupDem_shard_1[ i ];
288
289 };
290 */
291
292 /*
293 dvar boolean path[ shard_arcs_2  ];
294 dexpr float CCPT_2 = sum( < i, j, fcost > in shard_arcs_2 ) ArcsCost[ i ][ j ]
      * path[ < i, j, fcost > ] * fcost;
295
296 minimize CCPT_2;
297
298 subject to {
299
300   // Preserve flows at each node.
301   forall( i in shard_2 )
302     ctNodeFlow:
303       sum( < i, j, fcost > in shard_arcs_2 ) path[ < i, j, fcost  > ]
304      -  sum( < j, i, fcost > in shard_arcs_2 ) path[ < j, i, fcost  > ] ==
           SupDem_shard_2[ i ];
305
306 };
307 */
308
```

```
309
310 dvar boolean path[ shard_arcs_3  ];
311 dexpr float CCPT_3 = sum( < i, j, fcost > in shard_arcs_3 ) ArcsCost[ i ][ j ]
        * path[ < i, j, fcost > ] * fcost;
312
313 minimize CCPT_3;
314
315 subject to {
316
317   // Preserve flows at each node.
318   forall( i in shard_3 )
319     ctNodeFlow:
320       sum( < i, j, fcost > in shard_arcs_3 ) path[ < i, j, fcost  > ]
321      -  sum( < j, i, fcost > in shard_arcs_3 ) path[ < j, i, fcost  > ] ==
           SupDem_shard_3[ i ];
322
323 };
324
325
326 main {
327
328     var source = new IloOplModelSource("Sharding.mod");
329     var cplex =  new IloCplex;
330     var def  = new IloOplModelDefinition(source);
331     var opl  = new IloOplModel(def, cplex);
332     var data = new IloOplDataSource("ShardingImplementationFiles/Sharding.dat"
          );
333
334     opl.addDataSource(data);
335     opl.generate();
336
337     if( cplex.solve() ) {
338
```

```
339         writeln("OBJ = " + cplex.getObjValue() );
340      }
341      else {
342
343         writeln("No Solution");
344      }
345
346      var opl_0 = new IloOplModel( def, cplex );
347      var data_0  = new IloOplDataSource("shard_0.dat");
348      opl_0.addDataSource(data_0);
349      opl_0.generate();
350
351      if( cplex.solve() ) {
352
353         writeln("OBJ = " + cplex.getObjValue() );
354      }
355      else {
356
357         writeln("No Solution");
358      }
359
360      var opl_1 = new IloOplModel( def, cplex );
361      var data_1  = new IloOplDataSource("shard_1.dat");
362      opl_1.addDataSource(data_1);
363      opl_1.generate();
364
365      if( cplex.solve() ) {
366
367         writeln("OBJ = " + cplex.getObjValue() );
368      }
369      else {
370
371         writeln("No Solution");
```

```
372     }
373
374     var opl_2 = new IloOplModel( def, cplex );
375     var data_2  = new IloOplDataSource("shard_2.dat");
376     opl_2.addDataSource(data_2);
377     opl_2.generate();
378
379     if( cplex.solve() ) {
380
381        writeln("OBJ = " + cplex.getObjValue() );
382     }
383     else {
384
385        writeln("No Solution");
386     }
387
388     var opl_3 = new IloOplModel( def, cplex );
389     var data_3  = new IloOplDataSource("shard_3.dat");
390     opl_3.addDataSource(data_3);
391     opl_3.generate();
392
393     if( cplex.solve() ) {
394
395        writeln("OBJ = " + cplex.getObjValue() );
396     }
397     else {
398
399        writeln("No Solution");
400     }
401
402 }
```

APPENDIX B

IMPLEMENTATION OF NETWORK MODEL IN GAMS

This is the implementation of the network model in GAMS.

```
Set
  n 'nodes',
  Arcs( n, n ) 'Set of arcs in the network'
;
Alias
  ( n, i, j, k );
Parameter
  proof_size(n),
  supply(n),
  ArcCost(n, n)
;
*$Include parameters_1.inc
*$Include test_node11.inc

$Include parameters_1.inc
Arcs( i, j ) = no;
Arcs( i, j ) = yes$( fcost( i, j ) Gt 0 );
proof_size( n ) = Uniformint( 1, 100 );
*$( Supply( n ) ne 1 and Supply( n ) ne -1  );

Binary Variable
  path( n, n );
Free Variable
  CCPT
;
```

```
Equations
  Objective,
  Balance(i)
;
Objective.. CCPT =E= sum((i, j)$( Arcs( i, j ) and fcost( i, j ) gt 0 ),
   ArcCost(i, j) * path( i, j ) ) + sum((i, j)$Arcs( i,  j), fcost(i, j) *
   path( i, j));
Balance( i ).. sum(j$( Arcs( i, j ) ), path( i, j )) - sum(k$(Arcs(k, i)),
   path(k,i)) =E= supply( i );
Model
  Shard /all/;
Supply( i ) = 0;
Display
  proof_size;
Display
  fcost;
Display
  Arcs;
$Ontext
Display
  ArcCost
;
$Offtext
Set
     iter /1*10/,
  used_nodes(n),
  unused_nodes(n) /n1*n10/,
  shard1(n),
  shard2(n),
  shard3(n)
;
scalar rando, count, s_index, d_index, shardLength, shardLength2, shardLength3
   ;
```

```
*Try the optimization for different supply and demand nodes
Loop( iter,
    s_index = uniformint(1, 15);
    rando = uniformint(1, 15);
    d_index = rando$(s_index ne rando);
    supply(i)$(ord(i) = s_index) = 1;
    supply(i)$(ord(i) = d_index) = -1;
*   ArcCost(i, i)$( ord(i) = s_index ) = 0;
*   ArcCost(i, i)$( ord(i) = d_index ) = 0;
    proof_size( i )$( ord(i) = s_index ) = 0;
    proof_size( i )$( ord(i) = d_index ) = 0;
    ArcCost(i, j) = ( 1 / 2 ) * ( proof_size(i) + proof_size(j) );
    Solve Shard using mip minimizing CCPT;
    Display ArcCost;
    display s_index, d_index;
    display supply;
    proof_size( n ) = Uniformint( 1, 100 );
    supply(i) = 0;
);
```

PRACTICABILITY OF BLOCKCHAIN TECHNOLOGY AND SCALABLE BLOCKCHAIN NETWORK: SHARDING

Approved by:

Date Approved: ,