Comparison of Underlying Data Structures for Distributed Ledgers

Sebastián Sánchez
Advisor: Nicolás Cardozo

***Abstract*—Cryptocurrencies, such as Bitcoin, have achieved public recognition in the recent past due to their innovative financial proposal. Such innovation has been deemed a consequence of the blockchain technology underlying said cryptocurrencies. Therefore, this technology is now popularized as an underlying technology for distributed systems, more precisely, for distributed ledgers. However, a look inside cryptocurrencies shows there are still a number of challenges to address for their wide acceptance. These challenges have prompted the proposal of different underlying data structures for distributed ledgers different from the blockchain. The aim of this thesis is to address the optimality of different data structures, by assessing if they can respond to some of the technical challenges identified for this technology and if and how they guarantee its fundamental properties. We conduct a quantitative analysis to compare the data structures with respect to the quality attributes of throughput and latency, and we conduct a qualitative analysis to determine whether they satisfy the fundamental properties. These analyses serve as a basis to conclude which data structures would be more suited for use in different real-life scenarios.**

***Index Terms*— blockchain, data structure design and implementation, distributed ledger technology**

I. Context

BLOCKCHAIN is the technology behind Bitcoin. This technology provides an append-only data store of transactions replicated between peers. Since Satoshi Nakamoto set the Bitcoin blockchain into motion in January 2009 [1], many institutions around the world are exploring the applications of blockchains in areas such as supply chain management [2], smart grids [3], and Internet of Things [4].

A blockchain implements a distributed ledger, which can verify and store any kind of transactions. The distributed ledger industry is in its early stages of development and there are different limitations identified for blockchain as a solution for these systems. These limitations include technical issues related to the underlying algorithms and data structures used, ongoing industry thefts and scandals, public perception, government regulation, and mainstream adoption of the technology.

The Bitcoin blockchain prompted a rise in the research of distributed ledger technology. Researchers around the world have proposed different data structures that can underlie distributed ledgers, and which can offer solutions to some of blockchains limitations [5]. The objective of this thesis it to determine whether there exist viable alternatives to the blockchain from the perspective of the underlying data structures used to build distributed ledgers.

In order to do this, we evaluate three options of underlying data structures for distributed ledgers, namely the blockchain, the tangle, and the block-lattice. We conduct a quantitative analysis to compare the data structures with respect to the quality attributes of throughput and latency, and we conduct a qualitative analysis to determine whether they satisfy the fundamental properties of distributed ledgers. These analyses serve as a basis to conclude which data structures would be more suited for use in different real-life scenarios.

The results demonstrate that the best option for users who prioritize throughput is the block-lattice, and the best option for users who prioritize latency is the blockchain. The results also demonstrate that, although the fundamental properties of distributed ledgers were defined for and from the blockchain, the tangle and the block-lattice do a better job guaranteeing these properties because they guarantee equal rights for participants of the system better than the blockchain.

The remainder of this manuscript is structured as follows. Section II presents the research problem. Section III is devoted to the state of the art. Section IV presents distributed ledgers and three of their underlying data structures. Section V shows the details of the implementations of the data structures that were developed for this research. Section VI is devoted to the evaluation of the data structures. Section VII details the results of the evaluation. Section VIII discusses the relevance of the results. Section IX describes the lessons learned while conducting this research. Section X presents the directions for future work and Section XI presents our conclusions.

II. Research Problem

With the recent increase in the popularity of Bitcoin, blockchain has become a prime keyword in the computer science market. The blockchain has risen to become the most popular implementation of distributed ledgers and distributed applications in general. Nonetheless, one of the reasons to use blockchain implementations in distributed systems is centered

S. Sanchez is with the Department of Systems and Computer Engineering, Universidad de los Andes, Bogota, Colombia (e-mail: s.sanchez@uniandes.edu.co).

N. Cardozo is with the Department of Systems and Computer Engineering, Universidad de los Andes, Bogota, Colombia (e-mail: n.cardozo@uniandes.edu.co).

around the properties it offers in terms of security (in the broader and popular use of the term). Blockchain is designed around the concept of trustless systems, which can be of interest for some application domains (e.g., cryptocurrency applications, financial services, and risk management [6]). However, implementations of distributed ledgers that use different data structures have started to appear. This thesis will try to answer whether blockchain's status is justified or if other implementations can replace the blockchain as the most popular implementation of distributed ledgers.

A. *Technical Challenges*

A number of technical challenges related to the blockchain have been identified. Developers have a clear view of these challenges, which has led to discussions, coding of possible solutions and proposals of different answers to these challenges. Some researchers believe the Bitcoin blockchain will remain as the solution with the biggest market value, while others are building new solutions that they claim could overcome these challenges and become the next steps in the evolution of this technology [7], [8].

Melanie Swan, Founder of the Institute for Blockchain Studies and author of the foremost blockchain reference, "Blockchain: Blueprint for a New Economy" [9], identifies a number of technical challenges related to the blockchain from the perspective of a distributed system. These are explained below.

1) *Throughput*

Throughput is defined as the amount of transactions that can be processed per second. The on-chain transaction processing capacity of the Bitcoin network is limited by the average block creation time of 10 minutes and the block size limit of 1 MB. These jointly constrain the network's throughput. This network is processing only one transaction per second (tps) and has a theoretical maximum of 7 tps. This is an issue for the Bitcoin network.

Other transaction processing networks have much higher throughputs such as VISA, which supports 2,000 tps, Twitter, which supports 5,000 tps, and advertising networks, which support more than 100,000 tps [9]. Blockchain's low throughput is a major concern for enterprises that depend on high-performance legacy transaction processing systems. To be ready for pervasive use, distributed ledgers would need to be able to process more transactions per second.

2) *Latency*

Latency is defined as the time it takes from the creation of a transaction until the initial confirmation of it being accepted by the network. Latency is an issue for the Bitcoin network because each Bitcoin transaction block takes 10 minutes to process, meaning it takes at least 10 minutes for a transaction to be confirmed as accepted. As a comparison metric, VISA takes seconds at most to process a transaction [9]. Latency is an issue that needs to be solved if this technology is to compete with payment methods currently in use.

B. *Fundamental Properties*

Satoshi Nakamoto created the Bitcoin blockchain as a peer-to-peer network that could serve as a solution to double-spending; a problem that hindered the development of decentralized digital currency for decades [1]. In the Bitcoin whitepaper, Nakamoto explained,

> "A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they will generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone." [1, p. 1]

After carefully studying blockchain implementations, Xu et al. [10] identified the fundamental properties supported by this family of technological solutions. Any solution targeting the distributed ledger technology should guarantee the five properties below:

1) *Immutability*

In the blockchain context, immutability refers to the fact that no one is able to change the data that is contained in a committed transaction. If a malicious user attempts to alter the information in a transaction, this change will cause a change in the hash of the block, causing the protocol to register it as an inconsistency in the chain.

2) *Non-repudiation*

Blockchain users sign the transactions they initiate with digital signatures, which means that they cannot dispute the authorship or validity of such a transaction.

3) *Integrity*

The blockchain guarantees data integrity, which means it maintains and assures the accuracy and consistency of data. There are mechanisms in place by which all parties in the system can reach a consensus on the accepted truth. The Bitcoin blockchain achieves consensus via an economic measure called Proof of Work (PoW).

4) *Transparency*

On a blockchain, the identity of a user is concealed using cryptography so that linking public addresses to individual users is difficult to achieve. The transparency of a blockchain comes from the fact that the holdings and transactions of each public address are open to viewing. This level of transparency has not existed within financial systems and adds a degree of accountability not existing to date.

5) *Equal Rights*

All of the participants of a blockchain network have equal rights, and there are no privileged users because they all have the same ability to access and manipulate the blockchain.

For a solution to be comparable to a blockchain it must guarantee these five fundamental properties or give a reasoned explanation as to why it does not do so. Researchers designing different solutions similar to blockchains must keep these five properties in mind.

C. *Justification*

Taking into account the technical challenges and the fundamental properties of the blockchain technology, we state that the implementation of distributed ledgers by making use of a blockchain is insufficient. In order to truly realize a distributed ledger, new data structures and algorithms need to be designed. Hereinafter, we present an evaluation of different alternatives for the implementation of distributed ledgers from the perspective of the underlying data structure used to manage them.

The problem we address with this thesis is that of the advantages and disadvantages of the underlying data structures used to implement distributed ledgers. We study different data structures to determine how they compare in terms of the degree to which (1) they respond to some of the technical challenges that have been identified for this technology, and (2) they guarantee the fundamental properties of distributed ledgers.

In particular, to address this problem we will follow these four steps:

1. Implement distributed data structures in a way that allows for their comparison in perspective of the two technical challenges identified for this technology.
2. Quantitatively evaluate and compare the data structures in terms of the two technical challenges.
3. Classify the different data structures according to their impact on the two technical challenges identified for this technology.
4. Qualitatively evaluate and compare the data structures in terms of the five fundamental properties of distributed ledgers.

III. State of the Art

Before we present our evaluation study, we describe the current state of the art in the blockchain technology. This section is focused in the implementation of blockchains, rather than their use in other systems.

A. *Methodology*

The methodology used to gather the state of the art for this research follows the systematic review process described by Petersen et al. [11], and is informed by guidelines for a systematic literature review described by Kitchenham [12]. The steps followed are (1) the definition of the research area, (2) the search for papers, and (3) the screening for relevant papers.

1) *Definition of the Research Area*

The first stage of the process is the definition of the research area. The research area is defined by two topics:

1. *Evaluation of the blockchain* - rigorous scientific evaluation and validation of the blockchain as a distributed ledger.
2. *Distributed Ledgers and their Underlying Data Structures* – alternative data structures that can underlie distributed ledgers.

2) *Search for Papers*

The second stage of the systematic review is the search for papers. The search for papers includes scientific databases and technical text books, but it also needs to include working papers (e.g., white papers), considering the fact that the blockchain industry is still in the early stages of development. The chosen databases for the search are (1) ACM Digital Library, (2) IEEE Xplore, and (3) Springer Link, because of the high quality of the papers published in these scientific databases. The terms used in the search string are different according to the two topics that were described before, and they are shown below:

1. *Evaluation of the blockchain:* evaluati* and blockchain*
2. *Distributed Ledgers and their Underlying Data Structures*
    a. *Distributed Ledger Technology:* distributed and ledger* and technolog*
    b. *Underlying Data Structures for Distributed Ledgers:* (blockchain* and alternative*) or (blockchain* and substitute*)

The interest in the blockchain technology has been increasing since the idea was first introduced in January 2009. In 2015, Yli-Huumo et al. [13] did a systematic mapping study to describe the state of research on the blockchain technology and found that the cumulative number of papers written about blockchain from a computer science perspective increased from 2012 until 2015, when their study was published. This trend can be seen in Figure 1. Yli-Huumo et al. provide some recommendations on future research directions of the blockchain technology. One of their recommendations is for more studies to be conducted on the scalability issues of blockchain, which this thesis does by analyzing throughput and latency.

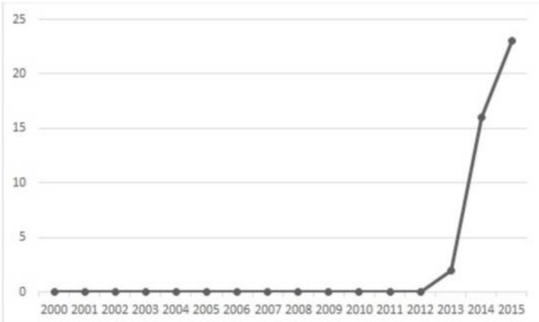

Figure 1. Increasing number of publications related to blockchain per year. Taken from [13].

Dabbagh, Sookhak and Safa [14] conducted a bibliometric study of blockchain-related publications indexed by Web of Science from 2013 to 2018. Figure 2 shows a generally growing trend in the number of blockchain papers indexed by Web of Science per year.

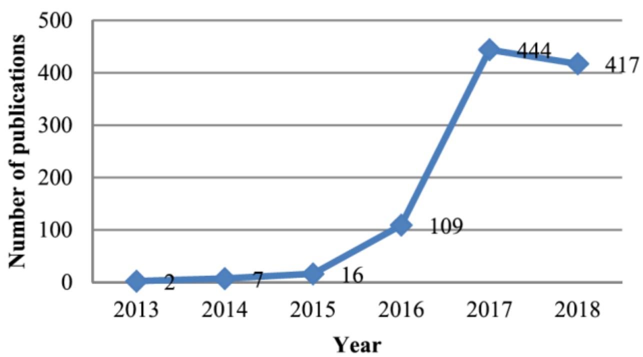

Figure 2. Number of blockchain papers indexed by Web of Science per year. Taken from [14].

3) *Screening of Papers for Inclusion and Exclusion*

The last stage of the process is the screening of relevant papers. In this stage, some of the papers that were found had to be discarded because they were not necessarily related to the research topics. This process is described in detail in the following section.

4) *Search and Selection Results*

The search and selection results for each of the three research topics are presented below.

1. *Evaluation of the blockchain:* 6 papers were found after the search. 2 were selected and 4 were excluded. Some of the papers that were excluded evaluated blockchains in specific settings, such as business process execution [15] and cloud [16], but what was needed was a general evaluation of blockchain.

2. *Distributed Ledgers and their Underlying Data Structures*
    a. *Distributed Ledger Technology:* 7 papers were found after the search. 5 were selected and 2 were excluded. The papers that were excluded were two literary reviews of distributed ledgers from a business perspective, and therefore were not related to the research topic.
    b. *Underlying Data Structures for Distributed Ledgers:* 7 papers were found after the search. 4 were selected and 3 were excluded. The papers that were excluded presented solutions that made use of blockchains, which meant they did not present other alternatives to this data structure.

B. *Evaluation of the blockchain*

It is important to have an understanding of what blockchains can and cannot do. The following describes evaluations of blockchains aimed at answering these questions.

Ahn Dihn et al. [17] argue that it is important to understand what a blockchain can offer, especially with respect to its data processing capabilities. This group of researchers introduce BLOCKBENCH, a benchmarking framework that helps understand the performance of private blockchains against data processing workloads. Ahn Dihn et al. conduct a comprehensive evaluation of three blockchain systems based on BLOCKBENCH, namely Ethereum, Parity, and Hyperledger Fabric. The results demonstrate several trade-offs in the design space, as well as big performance gaps between blockchain and database systems.

Xu et al. [10] propose a taxonomy that enables the classification and comparison of blockchains. Their taxonomy is informed by academic literature, books, government and technical reports, documents of industrial blockchain products, and developer forums and wikis. Additionally, they use an investigation for the Australian government of the use of blockchains in various use cases and their experience from implementing proof-of-concept blockchain-based systems. This taxonomy aims to assist with the design and assessment of the impact of blockchains on software architectures and helps with considerations about the quality attributes of blockchain-based systems. The authors analyze architectural design issues for blockchain-based systems in terms of three categories: the level of (de)centralization, the support for client storage and computation, and the blockchain infrastructural configuration. It is worth remembering that this group of researchers were responsible for identifying the five fundamental properties of distributed ledgers.

According to "Stack Overflow's annual Developer Survey for 2019" [18], which "is the largest and most comprehensive survey of people who code around the world" [18], and which was taken by nearly 90,000 developers from over 170 countries; when asked what they primarily believe about the blockchain technology, respondents of the survey are largely optimistic about its broad usefulness. However, this optimism is largely concentrated among young, less experienced developers. The more experienced a respondent is, the more likely he is to say the blockchain technology is an irresponsible use of resources.

The papers described in this section present a general view of the capabilities and limitations of blockchains. They were essential in defining the properties and metrics that are analyzed and measured in the experiments conducted in this research.

IV. Distributed Ledgers and their Underlying Data Structures

In this section we present three data structures used to implement distributed ledgers. We focus on the definition of the structures in this section. The following section presents our implementation for each of them.

Before going further, let us properly define ledgers and distributed ledgers. A *ledger* is an accounting book of final entry, in which business transactions are recorded. A *distributed ledger* is a consensus of replicated, shared, and synchronized digital data where there is no central administrator or centralized data storage [19]. To ensure replication across nodes is undertaken, distributed ledgers require a peer-to-peer network and a consensus algorithm [20]. Distributed ledgers make use of data structures to store the transactions.

As a means of comparison between the different distributed ledgers, we use a common scenario of transaction interactions for all three of them. Let us take a scenario of transactions taking place between three actors: Alice, Bob, and Carl. At the start of the system all actors have 400 units. Suppose all transactions are of 100 units, happening in the following sequence:

1. Alice → Bob
2. Alice → Carl
3. Carl → Bob
4. Bob → Alice

We explain how each of the ledgers' data structures change with this transaction sequence.

A. *The blockchain*

When Satoshi Nakamoto set the Bitcoin blockchain into motion in 2009, he introduced the concept of a proof of work-based blockchain to allow an agreement on the order of transactions [1]. The blockchain is the first credible solution to the double-spending problem, which for decades hindered the development of decentralized digital currency.

As a data structure, the blockchain, shown in Figure 3, is an ordered list of blocks where each block contains a list of transactions. Each block in the blockchain is “chained” back to the previous block by means of a hash of the previous block. This way, the historical transactions in the blockchain may not be deleted or altered without invalidating the chain of hashes.

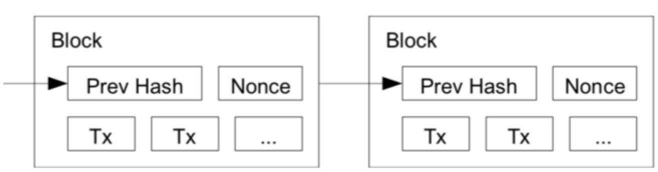

Figure 3. The blockchain. Image taken from [1].

Nakamoto explains that the steps to run the network are the following:

1. New transactions are broadcast to all nodes.
2. Each node collects new transactions into a block.
3. Each node works on finding a difficult proof-of-work for its block.
4. When a node finds a proof-of-work, it broadcasts the block to all nodes.
5. Nodes add the block to their copy of the blockchain only if all transactions in it are valid and not already spent.
6. Nodes express their acceptance of the block by working on creating the next block in the chain, using the hash of the accepted block as the previous hash.

Nodes always consider the longest chain to be the correct one and will keep working on extending it.

1) *True Example of the blockchain*

As the transactions in the scenario are submitted to the network, they are added to a block which is then appended to the blockchain. After the four transactions in the scenario are submitted to the network, all of the nodes in the network would have the same copy of the blockchain and it would have the structure shown in Figure 4.

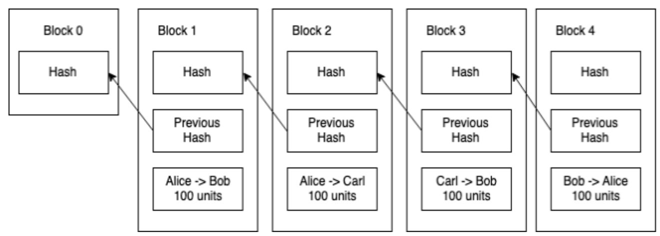

Figure 4. True example for the blockchain.

B. *The tangle*

IOTA is a new a cryptocurrency designed for the Internet of Things industry. Sergei Popov introduces the main feature of this cryptocurrency: the *tangle* [7]. The tangle, shown in Figure 5, is a directed acyclic graph for storing transactions and, according to the group behind IOTA, “succeeds the blockchain as its next evolutionary step” [7, p. 1].

To clarify the terminology, vertices are the transactions represented on the tangle graph and nodes are entities that issue and validate transactions. The transactions issued by nodes are the vertex set of the tangle graph and the edge set is obtained in the following way: when a new transaction arrives, it must approve one or two previous transactions. These approvals are represented by directed edges from the new transaction to the previous transactions. If there is not a directed edge between transaction A and transaction B, but there is a directed path of length at least two from A to B, it is said that A indirectly approves B.

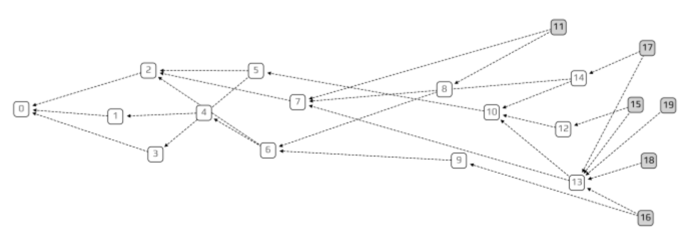

Figure 5. The tangle. Image taken from [21].

Popov explains that to issue a transaction, a node does the following:

1. The node chooses one or two other transactions to approve according to a tip selection algorithm (tips are unapproved transactions in the tangle graph).
2. The node checks if the transactions are conflicting and does not approve conflicting transactions.
3. For a node to issue a valid transaction, the node must solve a cryptographic puzzle similar to those in the Bitcoin blockchain. This is achieved by finding a nonce (an arbitrary number used only once in a cryptographic communication) such that the hash of that nonce concatenated with some data from the approved transaction has a particular form.

The tangle may contain conflicting transactions. However, in this case, the nodes need to decide which transactions will become orphaned. Orphaned transactions are not approved by incoming transactions and therefore do not achieve a high level of confirmation confidence which is a measure of a transaction's level of acceptance by the rest of the tangle. The confirmation confidence of a transaction is computed as follows:

1. Run the tip selection algorithm 100 times.
2. Count how many of those tips approve the transaction, and call it X.
3. The confirmation confidence of the transaction is "X percent".

In other words, the confidence of a transaction is the percentage of tips which approve it.

In the same way as a block in a blockchain cannot be modified without invalidating all the subsequent blocks in the chain, a transaction in the tangle cannot be modified without invalidating all the transactions that approve that transaction.

1) True Example of the tangle

As the transactions in the scenario are submitted to the network, they are added to the tangle as a vertex of the directed acyclic graph. After the four transactions in the scenario are submitted to the network, all of the nodes in the network would have the same copy of the tangle and it would have the structure shown in Figure 6. Notice that transactions can directly approve one or two transactions as can be seen by the directed edges connecting the transactions. These edges go from the transaction that is approving to the transaction that is being approved.

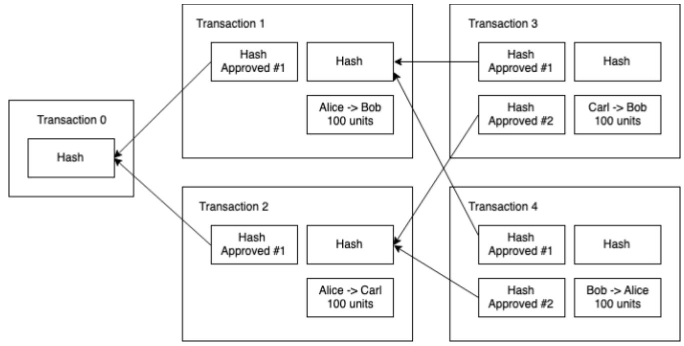

Figure 6. True example for the tangle.

C. The block-lattice

The block-lattice, shown in Figure 7, is the data structure used to store the information for the Nano cryptocurrency [8]. In the block-lattice, each account has its own blockchain (account-chain) which stores the account's transaction history. Every node in the network stores a ledger composed of its account-chain and a copy of the account-chains of all the other nodes in the network.

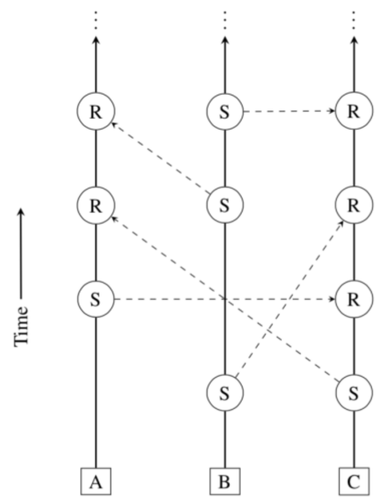

Figure 7. The block-lattice. Image taken from [8].

Each account-chain can only be updated by the account's owner, which allows each account-chain to be updated asynchronously to the rest of the block-lattice. Every transfer of funds requires the creation of a send block (S) on the sender's account-chain and the creation of a receive block (R) on a receiver's account-chain. The transfer is completed only when both blocks are acknowledged and accepted by the network after being broadcast by the respective account-chain owners.

1) True example of the block-lattice

After the four transactions in the scenario are submitted to the network, all of the nodes in the network would have the same copy of the block-lattice and it would have the structure shown in Figure 8. To aid in the understanding of the way this data structure works, the blocks in the figure are colored such that a send block and its corresponding receive block have the same color.

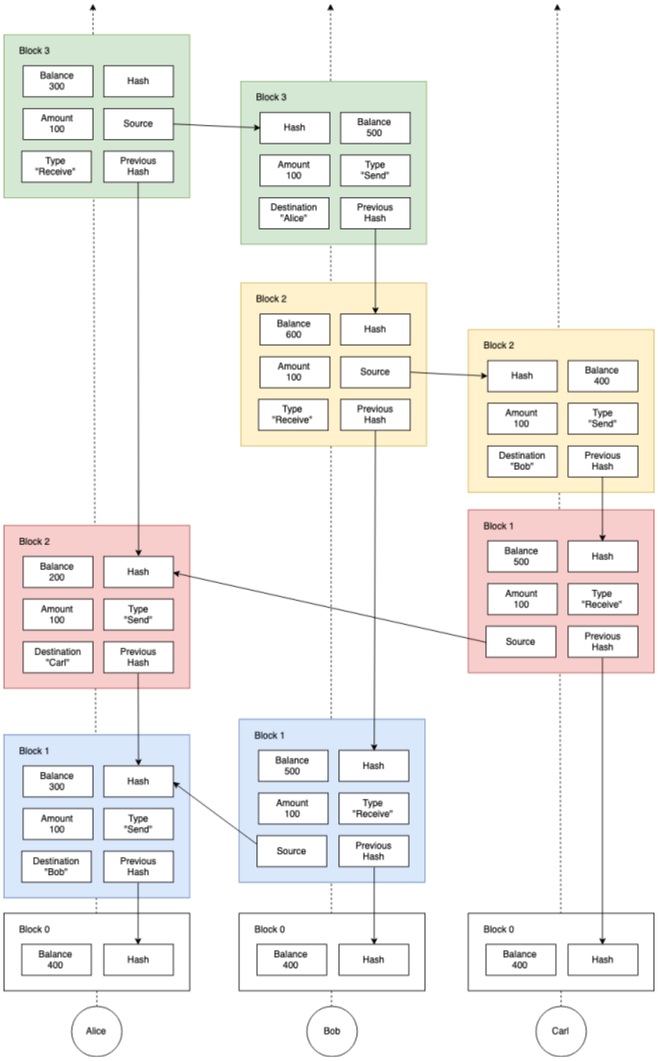

Figure 8. True example for the block-lattice.

When one of the actors in the scenario chooses to send units to another actor, he adds a send block to his account-chain and broadcasts this block to the network. The other participants in the network add this block to the account-chain of the actor who initiated the transaction.

To complete a transaction, the recipient of the units adds a receive block to his account-chain and broadcasts this block to the network. The other participants in the network add this block to the account-chain of the actor who received the transaction.

V. Ledger Data Structure Implementations

This section shows the implementation of the three data structures underlying distributed ledgers introduced in the previous section, namely the blockchain, the tangle, and the block-lattice. To implement the data structures, we use the Go programming language[1] in order to take advantage of its distributed programming characteristics.[2]

A. Go in a Nutshell

After studying the advantages and disadvantages of different programming languages, the language selected for the implementation of the data structures is Go [22]. Go, also referred to as Golang, is a statically typed, compiled programming language designed at Google. Some of its features include memory safety, garbage collection, structural typing, and CSP-style concurrency. This language is selected because its concurrency mechanisms make it easy to write programs that get the most out of multicore and networked machines. It has two main built-in facilities for writing concurrent and distributed programs, goroutines and channels, which are describe in more detail below.

1) Goroutines

Goroutines [23] are functions that run concurrently with other functions. They can be thought of as light weight threads as the cost of creating a goroutine is much less than the cost of creating a thread. Goroutines are a few KB in stack size and the stack can grow and shrink according to the needs of the application whereas, with threads, the stack size has to be specified and is fixed.

The language abstraction `go f(x,y,z)` starts a new goroutine running the `f(x,y,z)` function. The evaluation of `f`, `x`, `y`, and `z` happens in the currently executing goroutine (the one calling the go abstraction), and the execution of `f` happens in the new goroutine.

The code snippet shown in Figure 9 presents a go package with two functions `main` and `say`. Note in this snippet that the first call to the `say` function takes places in an independent go routine, while the second call happens in the currently executing go routine. The output when executing the main function is presented in Figure 10.

```
package main

import (
  "fmt"
  "time"
)

func say(s string) {
  for i := 0; i < 5; i++ {
    time.Sleep(100 * time.Millisecond)
    fmt.Println(s)
  }
}

func main() {
  go say("hello")
  say("world")
}
```

Figure 9. Code snippet for Goroutines

[1] https://golang.org/

[2] The implementation of the data structures and benchmark results are available at: https://github.com/FLAGlab/ssanchez59_Blockchain.

```
hello
world
world
hello
hello
world
world
hello
hello
world

Program exited.
```

Figure 10. Output for Goroutines code snippet in Figure 9.

Notice the interleaving between the prints of “hello” and “world”. This interleaving reflects the goroutines running concurrently.

2) *Channels*

Goroutines communicate using channels [24]. Channels prevent race conditions from happening when accessing shared memory using goroutines, and they allow goroutines to synchronize without explicit locks or condition variables. Channels can be conceived as communication buffers allowing to read and write multiple values.

Channels send and receive values with the channel operator: `<-` whose functioning is show in Figure 11.

```
//create a channel
ch := make(chan int)

// Send v to channel ch
ch <- v

// Receive from ch, and assign value to v
v := <-ch
```

Figure 11. How Go Channels work.

The code snippet shown in Figure 12 sums the numbers in a slice (a dynamically-sized array in Go). It distributes the work between two goroutines to provide data parallelism. Once both goroutines have completed their computation, it calculates the final result. The output when executing the main function is presented in Figure 13.

```
package main

import "fmt"

func sum(s []int, c chan int) {
  sum := 0
  for _, v := range s {
    sum += v
  }
  c <- sum // send sum to c
}

func main() {
  s := []int{7, 2, 8, -9, 4, 0}

  c := make(chan int)
  go sum(s[:len(s)/2], c)
  go sum(s[len(s)/2:], c)
  x, y := <-c, <-c // receive from c

  fmt.Println(x, y, x+y)
}
```

Figure 12. Code snippet for Go Channels.

```
-5 17 12

Program exited.
```

Figure 13. Output for Go Channels code snippet in Figure 12.

In the setting of this thesis, read and write operations have to be performed in an asynchronous manner because nodes in the network have to be able to receive and send the shared data structures. Goroutines and channels help to achieve this effectively.

B. *Peer-to-Peer Communication*

In a distributed setting, nodes in the network must communicate with each other without the need for a central server. To do this, we use a peer-to-peer (p2p) networking stack called noise [25]. Noise is an easy-to-use networking stack for developing decentralized applications written in Go and developed by the Perlin Team [26].

When a node runs a Go program using noise, it acts as a peer to which other nodes can connect, and nodes can connect and disconnect from the network without causing a disruption to the communication. Noise also allows nodes to broadcast messages over their network of peers.

C. *Data Structure Implementation*

The blocks in the IOTA tangle and the Nano block-lattice have a granularity of one transaction. In order to enable a comparison, the data structures implemented in this thesis are comprised of items which record one transaction each. This marks a clear difference from the way in which the Bitcoin blockchain is implemented since the blocks in that system hold the data for various transactions.

1) *Blockchain Implementation*

Our Go implementation of the blockchain uses the implementation presented by Coral Health [27] as a starting point. The blockchain consists of a series of blocks which each have an index, a timestamp, a hash, and the hash of the previous block as shown in Figure 14.

```
// Block represents each transaction in the
// blockchain
type Block struct {
  Index       int
  Timestamp   string
  Transaction string
  Hash        string
  PrevHash    string
  Signature   string
  TimeSent    time.Time
}

// Blockchain is a series of validated Blocks
type Blockchain struct {
  Blocks     []Block
  State      map[string]int
  Difficulty int
}
```

Figure 14. Basic code for the blockchain.

a) Specific Functions for the blockchain

`generateBlock` creates a new block to be added to the blockchain with the necessary transaction information inside it. Figure 15 shows the code for this function.

```
// create a new block using previous block's
// hash
func generateBlock(oldBlock network.Block,
Transaction string, address string, timeSent
time.Time) network.Block {

  var newBlock network.Block

  t := time.Now()
  newBlock.Timestamp = time.Unix(0,
t.UnixNano()).String()

  newBlock.Index = oldBlock.Index + 1
  newBlock.Transaction = Transaction
  newBlock.PrevHash = oldBlock.Hash
  newBlock.Signature = address
  newBlock.TimeSent = timeSent

  newBlock.Hash = calculateHash(newBlock)

  return newBlock
}
```

Figure 15. `generateBlock` function.

`calculateHash` uses sha256 to hash a block's raw data and return a string which is the hash for the block. Figure 16 shows the code for this function.

```
// SHA256 hashing
func calculateHash(block network.Block) string {
  record := strconv.Itoa(block.Index) +
block.Timestamp + block.Transaction +
block.PrevHash + block.Signature
  h := sha256.New()
  h.Write([]byte(record))
  hashed := h.Sum(nil)
  return hex.EncodeToString(hashed)
}
```

Figure 16. `calculateHash` function.

`isBlockValid` checks that the chain of hashes of the blockchain is consistent. It does this by: (1) checking that the index of a new block is one more than the index of the previous block, (2) checking that the reference to the previous block is correct, and (3) verifying the hash for a new block. Figure 17 shows the code for this function.

```
// Make sure block is valid by checking index,
// comparing the hash of the previous block,
// and veryfying hash
func isBlockValid(newBlock, oldBlock
network.Block) bool {
  if oldBlock.Index+1 != newBlock.Index {
    return false
  }

  if oldBlock.Hash != newBlock.PrevHash {
    return false
  }

  if calculateHash(newBlock) != newBlock.Hash {
    return false
  }

  return true
}
```

Figure 17. `isBlockValid` function.

2) Tangle implementation

The company in charge of introducing the tangle created a visual simulation written in React and D3.js [21]. This visualization is used as a guide for the implementation of the tangle in Go.

The tangle is defined as a Directed Acyclic Graph (DAG), where nodes represent blocks, each containing exactly one transaction. Edges represent verification links between transactions. Each transaction is identified by an index, as shown in Figure 18.

```
type Transaction struct {
  Index      int
  Operation  string
  TimeInt    int64
  TimeString string
  Weight     int
  CumWeight  int
  Signature  string
  Hash       string
  HashApp1   string
  HashApp2   string
  TimeSent   time.Time
}

type Link struct {
  Target int
  Source int
}

// Tangle is a DAG of Transactions
type DAG struct {
  Transactions []Transaction
  Links        []Link
  Lambda       float64
  Alpha        float32
  H            int64
  TipSelection string
  State        map[string]int
}
```

Figure 18. Basic code for the tangle.

a) Specific Functions for the tangle

`generateTransaction` creates a new transaction to be added to the tangle with the necessary transaction information inside of it. Figure 19 shows the code for this function.

```
// create a new Transaction using previous
// Transactions index
func generateTransaction(lastTransaction
network.Transaction, Operation string, address
string, timeSent time.Time) network.Transaction
{

  var newTransaction network.Transaction

  newTransaction.Index =
lastTransaction.Index+1
  newTransaction.Operation = Operation

  now := time.Now()
  newTransaction.TimeInt = now.UnixNano() /
1000000
  newTransaction.TimeString = time.Unix(0,
now.UnixNano()).String()
  newTransaction.Weight = 1
  newTransaction.Signature = address
  newTransaction.Hash =
calculateHash(newTransaction)

  newTransaction.TimeSent = timeSent

  return newTransaction
}
```

Figure 19. `generateTransaction` function.

`generateLink` creates a new link, from a source (approving) transaction to a target (approved) transaction, to be added to the tangle. Figure 20 shows the code for this function.

```
func generateLink(target network.Transaction,
source network.Transaction) network.Link {

  var newLink network.Link

  newLink.Target = target.Index
  newLink.Source = source.Index

  return newLink

}
```

Figure 20. `generateLink` function.

3) *Block-lattice Implementation*

The whitepaper "Nano: A Feeless Distributed Cryptocurrency Network" [8] is used as a starting point for the implementation of the block-lattice with Go.

The block-lattice is made of account-chains that store blocks. These blocks have an index, a block type i.e., send and receive, the *amount* being sent/received in the transaction, and the resulting *balance* in the account after the transaction has been completed. The main structure of a block is shown in Figure 21.

```
// Block represents each transaction
// in the block-lattice
type Block struct {
  Index     int
  Balance   int
  Type      string
  Amount    int
  Hash      string
  Source    string
  Previous  string
  Signature string
  TimeSent  time.Time
  Sender    string
  Receiver  string
}
```

Figure 21. Basic code for the block-lattice

a) Specific Functions for the block-lattice

A specific function for the block-lattice is `generateBlock`. This function creates a new block to be added to the block-lattice with the necessary information inside it. Notice that if it is a send block, the amount being transferred is deducted from the account's balance, and if it is a receive block the amount being transferred is added to the account's balance. Figure 22 shows the code for this function.

```
// create a new block
func generateBlock(oldBlock network.Block,
typeOfBlock string, amount int, signature
string, hash string, timeSent time.Time, sender
string, receiver string) network.Block {

  var newBlock network.Block

  newBlock.Index = oldBlock.Index + 1
  newBlock.Previous = oldBlock.Hash

  if typeOfBlock == "send" {
    newBlock.Balance = oldBlock.Balance -
amount

  } else {
    newBlock.Balance = oldBlock.Balance +
amount
  }

  newBlock.Type = typeOfBlock
  newBlock.Amount = amount

  if hash == " " {
    newBlock.Hash = calculateHash(newBlock)
  } else {
    newBlock.Hash = hash
  }

  newBlock.Signature = signature
  newBlock.TimeSent = timeSent

  newBlock.Sender = sender
  newBlock.Receiver = receiver

  return newBlock
}
```

Figure 22. `generateBlock` function.

VI. EVALUATION OF DATA STRUCTURES

The next steps after implementing the data structures is to evaluate and classify them with respect to the characteristics presented in section II. The purpose of this classification is to help developers and practitioners decide which data structure to use depending on the specific characteristics of their application domain and according to the fundamental properties they wish to guarantee and the quality attributes they want to prioritize.

A. *Design of Evaluation*

1) *Quantitative Evaluation and Comparison*

The key metrics that are evaluated for the blockchain, the tangle, and the block-lattice are throughput and latency.

- Throughput: The number of transactions processed per second.
- Latency: The time it takes from the creation of a transaction until the initial confirmation of it being accepted by the network.

To measure these metrics, transactions are submitted to the network by a node which stores the transactions in its copy of the data structure and then shares this information with its peers.

Each node is deployed in a different virtual machine instance. The instances are setup in a Google Cloud Platform cluster, where each instance is an `n1-standard-1` instance that has one virtual CPU, 3.75GB of RAM, 10 GB of hard drive, and runs a Linux operating System. The experiments that are conducted in this research resemble the experiments conducted by Ahn Dihn et al. [17] in their comparative study of three blockchain systems (namely Ethereum, Parity, and Hyperledger Fabric).

To measure throughput, one node submits as many transactions as it can for one second. Each throughput experiment is repeated five times and the median and standard deviation for the five runs are reported. To measure latency one node submits transactions to the network and, for each transaction, the receiving node(s) subtract the initiation timestamp from the completion timestamp. The median and standard deviation are reported. The initiation timestamp for transactions using the block-lattice is taken when the send block is created. The completion timestamp for transactions using the block-lattice is taken after the receive block is confirmed.

2) *Classification of Data Structures*

After gathering and analyzing the quantitative information for each of the data structures, we classify them according to their impact on the quality attributes of throughput and latency. The method of classification that is used mirrors the method used by Xu et al. in their “Taxonomy of Blockchain-Based Systems” [10]. Xu et al. use a classification scheme to describe the impact of architectural decisions over a group of quality attributes by stating whether each option was less favorable (⊕), neutral (⊕⊕), or more favorable (⊕⊕⊕) with respect to each attribute in a comparison established between the different options that are available.

3) *Analysis of Fundamental Properties*

To determine if and how the three data structures satisfy the fundamental properties of distributed ledgers, the documentation for each of these data structures is analyzed. The result of this analysis is a description of the manner in which each data structure guarantees each property. In case a data structure does not guarantee a certain property there is an explanation of why this is the case.

VII. RESULTS OF EVALUATION

A. *Results of Quantitative Analysis*

After conducting the quantitative analysis, the following conclusions can be made regarding the degree to which the different data structures satisfy important quality attributes of distributed ledgers.[3]

1) *Throughput*

The data structure with the best throughput by a large margin is the block-lattice. Figure 23 shows that the difference in throughput is very large between the block-lattice and the other two data structures. The throughput for the block-lattice is about ten times larger than that of the blockchain, and the

[3] All results and data are available in Appendix A to C.

throughput for the blockchain is about 6 times larger than that of the tangle.

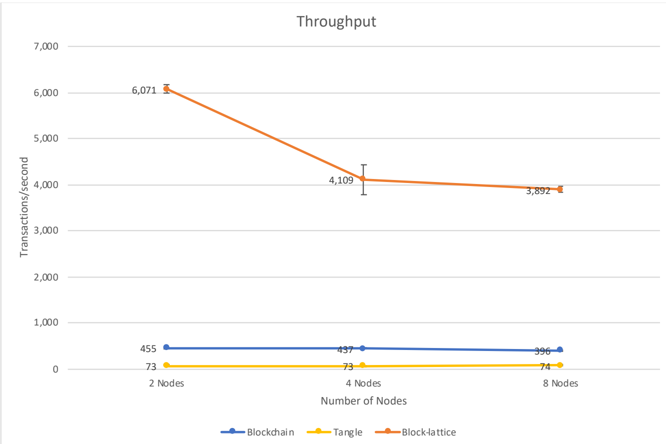

Figure 23. Throughput for the three data structures.

2) *Latency*

Latency refers to the time it takes from the creation of a transaction until the initial confirmation of it being accepted by the network. This time may vary depending on the size of the data structure. For this reason, it is measured for different transactions as each data structure continues to grow (transaction number: 50, 100, 200, 400, and 800).

The data structure with the best latency is the blockchain. Figure 24 (network with 8 nodes), Figure 25 (network with 4 nodes), and Figure 26 (network with 2 nodes) show that from transaction number 100 onwards, the blockchain has a better latency than the block-lattice and the tangle. The latency for the blockchain holds steady below 30 milliseconds for all of the experiments that were performed. The latency for the block-lattice has an increasing trend as the data structure continues to grow; starting from around 40 milliseconds for transaction number 100 to around 200 milliseconds for transaction 800. The latency for the tangle for transaction number 100 and above is in the order of seconds and massively exceeds that of the other two data structures. The experimental data shows that median latency is more-or-less independent of the network sizes that were chosen.

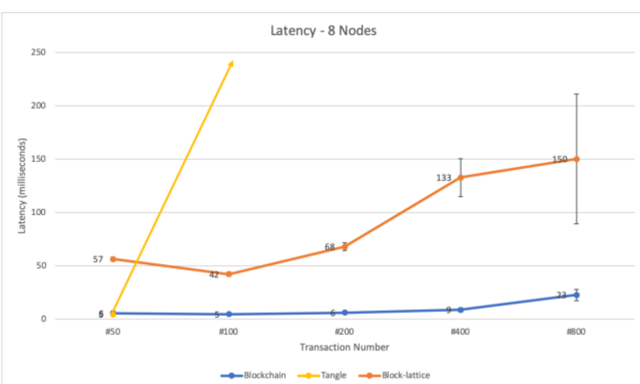

Figure 24. Latency for the three data structures on a network with 8 nodes.

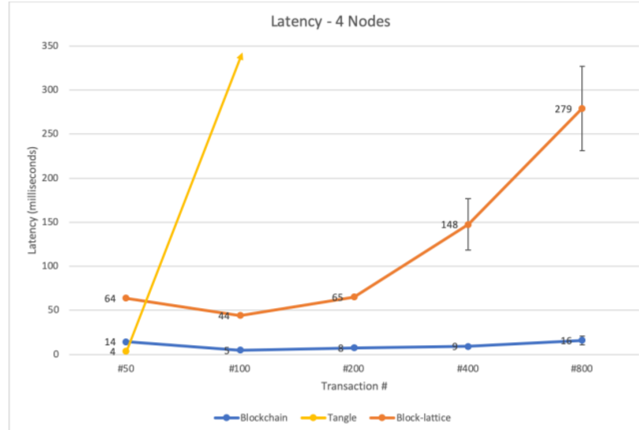

Figure 25. Latency for the three data structures on a network with 4 nodes.

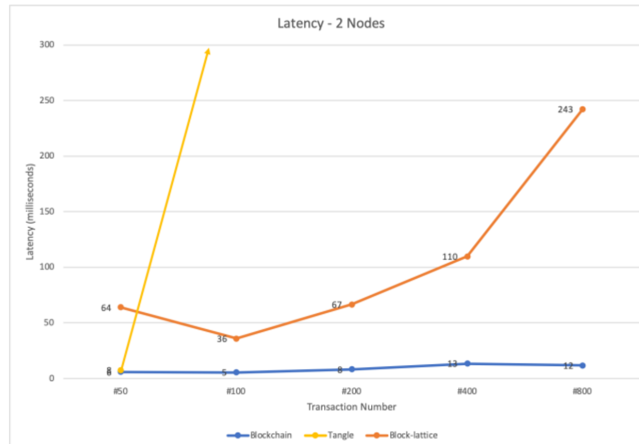

Figure 26. Latency for the three data structures on a network with 2 nodes.

B. *Classification of Data Structures*

After gathering and analyzing the quantitative information about the two metrics for each of the data structures, they are classified as Table I.

TABLE I. CLASSIFICATION ACCORDING TO IMPACT ON QUALITY ATTRIBUTES.

Design Decision - Data Structure		
(□: Less favorable, □□: Neutral, □□□: More favorable)		
Option	**Impact**	
	Quality Attributes	
	Throughput	**Latency**
Blockchain	□	□□□
Tangle	□	□
Block-lattice	□□□	□□

C. *Results of Qualitative Analysis*

1) *The blockchain*

In section II - Research Problem, the five fundamental properties of distributed ledgers are defined together with an explanation of the manner in which blockchains guarantee these properties. However, a more detailed study of blockchains opens room to make further comments about the degree to which this data structure guarantees equal rights for participants in the network.

a) Equal Rights

In the Bitcoin blockchain system, there are two distinct types of participants, those who issue transactions, and those who approve transactions, known as miners. According to Popov, the mathematician behind the tangle, “the design of this system creates unavoidable discrimination of some participants”[7, p. 1]. LeMahieau, the researcher who proposes the block-lattice, explains this in more detail:

> “Bitcoin, and other cryptocurrencies, function by achieving consensus on their global ledgers in order to verify legitimate transactions while resisting malicious actors. Bitcoin achieves consensus via an economic measure called Proof of Work (PoW). In a PoW system participants compete to compute a number, called a nonce, such that the hash of the entire block is in a target range. This valid range is inversely proportional to the cumulative computation power of the entire Bitcoin network in order to maintain a consistent average time taken to find a valid nonce. The finder of a valid nonce is then allowed to add the block to the blockchain; therefore, those who exhaust more computational resources to compute a nonce play a greater role in the state of the blockchain. PoW provides resistance against a Sybil attack, where an entity behaves as multiple entities to gain additional power in a decentralized system, and also greatly reduces race conditions that inherently exist while accessing a global data-structure.”[8, p. 1]

Since the other fundamental properties are defined for and from the blockchain, this data structure satisfies them all.

2) *The tangle*

a) Immutability

The tangle guarantees that no one is able to change the data that is contained in a committed transaction by using hashes. If a malicious user attempts to alter the information in a transaction, this will cause a change in the hashes of the tangle, causing the protocol to register it an inconsistency in the data structure. “For a node to issue a valid transaction, the node must solve a cryptographic puzzle similar to those in the Bitcoin blockchain. This is achieved by finding a nonce such that the hash of that nonce concatenated with some data from the approved transaction has a particular form” [7, p. 3].

b) Non-repudiation

The tangle users sign the transactions they initiate with digital signatures, which means that they cannot dispute the authorship or validity of such a transaction.

c) Integrity

The tangle guarantees data integrity, which means it maintains and assures the accuracy and consistency of data. There are mechanisms in place by which all parties in the system can reach a consensus on the accepted truth. When using the tangle, consensus is achieved via a cumulative Proof of Work (PoW) of stacked transactions.

d) Transparency

On the tangle, the identity of a user is concealed using cryptography so that linking public addresses to individual users is difficult to achieve. The transparency of a tangle comes from the fact that the transactions of each public address are open to viewing.

e) Equal Rights

All of the participants of a tangle network have equal rights, and there are no privileged users because they all have the same ability to access and manipulate the tangle. With respect to the approval of transactions, “the main idea of the tangle is the following: to issue a transaction, users must work to approve other transactions. Therefore, users who issue a transaction are contributing to the network's security.”[7, p. 2]

3) *The block-lattice*

a) Immutability

According to the Nano whitepaper, the cryptocurrency which uses the block-lattice architecture, “a send block is immutable once confirmed. Once broadcasted to the network, funds are immediately deducted from the balance of the sender’s account and wait as pending until the receiving party signs a block to accept these funds. Pending funds should not be considered awaiting confirmation, as they are as good as spent from the sender’s account and the sender cannot revoke the transaction” [8, p. 3].

b) Non-repudiation

Block-lattice users sign the transactions they initiate with digital signatures, which means that they cannot dispute the authorship or validity of such a transaction.

c) Integrity

The block-lattice guarantees data integrity, which means it maintains and assures the accuracy and consistency of data. There are mechanisms in place by which all parties in the system can reach a consensus on the accepted truth. When using the block-lattice, consensus is achieved via a balanced-weighted vote on conflicting transactions.

d) Transparency

On a block-lattice, the identity of a user is concealed using cryptography so that linking public addresses to individual users is difficult to achieve. The transparency of a block-lattice comes from the fact that the holdings and transactions of each public address are open to viewing.

e) Equal Rights

All of the participants of a block-lattice network have equal rights, and there are no privileged users because they all have the same ability to access and manipulate their own account-chain. Regarding transaction verification, “each individual user provides the computational power for the verification of their own transactions, meaning entire network is not required to update the overall ledger together in massive blocks.” [28]

VIII. Relevance of Results

A. *Relevance of Results of Quantitative Analysis*

The quantitative analysis demonstrates there is a trade-off between throughput and latency.

If users wish to prioritize throughput, the best option is to use the block-lattice. The reason for the high throughput for the block-lattice is believed to be the ease with which users can update their own account-chain when they submit a transaction. The reason for the lower throughput for the blockchain is believed to be the process that has to be carried out to add blocks to the blockchain. The reason the tangle has the lowest throughput is believed to be the long process that goes into deciding how to add new transactions to the directed acyclic graph (DAG).

If users wish to prioritize latency, the best option is to use the blockchain. The low latency for the blockchain is believed to be a consequence of the fact that nodes can accept the longest proof-of-work chain as the latest state of the system. The higher latency for the block-lattice is believed to arise from the fact that both a receive transaction and a send transaction have to be broadcasted for a transaction to be completed. The reason the tangle has the highest latency is also believed to be the long process that goes into deciding how to add new transactions to the DAG.

Users may choose to use the block-lattice in a setting which requires high throughput such as a machine-to-machine micropayment system in the Internet of Things industry. Users may choose to use the blockchain in a setting which requires low latency such as real-life payments systems, where users do not want to wait a long time for a transaction to be finalized and become irreversible.

B. *Relevance of Results of Qualitative Analysis*

The qualitative analysis shows that the tangle and the block-lattice guarantee the five fundamental properties defined by Xu et al. for distributed ledgers [10]. This analysis shows that there is skepticism regarding the claim of equal rights for the participants in a blockchain system, which stems from the fact that a blockchain network relies on miners to aggregate valid transactions into blocks and append them to the blockchain. The tangle and the block-lattice guarantee equal rights for participants by making the users who submit a transaction participate in the validation of previous transactions, in the case of the tangle, or validation of their own transaction, in the case of the block-lattice.

Users may choose to use the tangle or the block-lattice in a permission-less public setting, such as peer-to-peer payment networks, where all users should be able to join the network and submit and validate transactions. Users may choose to use the blockchain in a permissioned setting where one or more authorities act as a gate for participation and not all users expect to have the same permissions as others. The blockchain may be more suitable for regulated industries, such as banks, for example, where permissions may include permission to join the network, permission to initiate transactions, and permissions to validate transactions.

C. *Threats to Validity*

The implementation of the data structures developed for this thesis does not precisely reflect all of the intricacies of the real implementations of the Bitcoin blockchain, the IOTA tangle, or the Nano block-lattice. This is because the real implementations have a degree of complexity that could not be replicated in the time frame of this research and because we implemented the data structures so that they would have the same level of granularity, which is not the case in real life. However, this thesis does represent a step forward in the study and comparison of different alternatives for underlying data structures of distributed ledgers because it captures the essence of three data structures and analyzes their fundamental functioning.

As mentioned in section V. Ledger Data Structure Implementations, in order to enable a comparison, the data structures implemented in this thesis are comprised of items which record one transaction each. This marks a clear difference from the way in which the Bitcoin blockchain is implemented since the blocks in that system hold the data for various transactions. For this reason, the throughput and latency data reported for the blockchain implemented for this thesis do not reflect the values for these quality attributes for the Bitcoin Blockchain. The Bitcoin blockchain has a theoretical maximum of 7 tps and a latency limited by the block generation time of 10 minutes. In a comparison with the data structures that were implemented, these would be the worst transaction processing metrics.

The experiments conducted in this research did not thoroughly tackle one part of the blockchain scalability problem which refers to how a large number of nodes affects throughput and latency. However, the experiments that were run do show a trend that should hold true for a greater number of nodes.

IX. Lessons Learned

As explained before, to measure throughput and latency one node submits transactions to the network. We also conducted experiments in which multiple nodes submit transactions to the network. This experiment failed for the blockchain and the tangle because nodes only rewrite these data structures if the incoming data structure is larger than the one in memory, which is not guaranteed in a scenario in which multiple nodes are submitting transactions at the same time. This experiment succeeded for the block-lattice because nodes can update their data structure by appending incoming transactions to the relevant account-chains regardless of the size of the account-chains. This is an advantage of the block-lattice because, in its case, the size of the data structure is not a blocking factor when processing transactions.

In this study, we also looked into the problem of the size of the blockchain: the blockchain is 25 GB and grew by 14 GB in the last year. So, it already takes a long time to download (e.g., 1 day). The Bitcoin community calls the size problem ‘bloat’. This motivates centralization, because it takes resources to run the full node, and only about 7,000 servers worldwide do in fact run full Bitcoind nodes [9, p. 82]. However, we were unable to recreate a scenario where size would be a differentiating factor. The data we collected (shown in Appendix C) suggests the tangle would be a less favorable option with regard to disc usage. This is due to the fact that information about vertices and edges for a large graph would have to be stored.

X. Future Work

The implementations of the data structures developed for this thesis can be refined to better reflect the complexity of the Bitcoin blockchain, the IOTA tangle, and the Nano block-lattice. This may allow a more robust comparison of these data structures to be conducted.

Another analysis that could be conducted is testing whether a relaxation of some of the five fundamental properties could lead to an improvement in the technical challenges identified for the blockchain. As an example, Xu et al. discuss transparency in their analysis of different blockchain configurations, “using a public blockchain results in better information transparency and auditability, but sacrifices performance” [10, p. 248]. Another fundamental property that could be adjusted to improve some of the quality attributes of distributed ledgers is equal rights. This fundamental property could be adjusted by setting up permissioned ledgers where one or more authorities act as gate keepers for participation. These authorities can decide who has permission to join the network, who has permission to initiate transactions, and who has permission to mine. Xu et al. explain that “there are often trade-offs between permissioned and permission-less blockchains including transaction processing rate” [10, p. 245].

The problem of the blockchain bloat and the size of these data structures in general is still open to discussion. Different researchers have proposed solutions to this problem either by proposing changes to existing data structures or proposing different data structures that would not have this problem. It is important to continue research into this problem as it is linked to a historical problem of cryptocurrencies which is the trade-off between scaling and decentralization. “As cryptocurrencies scale, their blockchains increase in size, causing increasing burden for participants to verify balances and data on the blockchain. These cryptocurrencies tend towards inaccessibility and more centralized validation with usage” [29, p. 1].

There are other options of underlying data structures that are being proposed and it would be interesting to compare them as well. Among these are the Hedera Hashgraph [30] proposed by professor Leemon Baird and the Avalanche Consensus Protocol [31] promoted by professor Emin Gün Sirer of Cornell University. These data structures are not included in this evaluation because they have a higher complexity and their implementation requires more time/resources than the ones available to complete this thesis.

XI. Conclusions

Research on distributed ledger technology has risen in the recent past and various alternatives of underlying structures for this technology have been proposed. The aim of this thesis is to address the optimality of different alternatives, by determining if they can respond to some of the technical challenges identified for blockchains and if and how they guarantee the fundamental properties of distributed legers.

Quantitative and qualitative analyses allowed us to determine which data structures would be more suited for use in different real-life scenarios. It will be some time before distributed ledger technology is adopted for widespread use, but this research represents a step in the right direction when it comes to paving the way to making this a reality.

References

[1] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system.” 2009.

[2] S. Chen, R. Shi, Z. Ren, J. Yan, Y. Shi, and J. Zhang, “A Blockchain-Based Supply Chain Quality Management Framework,” in *Proceedings - 14th IEEE International Conference on E-Business Engineering, ICEBE 2017 - Including 13th Workshop on Service-Oriented Applications, Integration and Collaboration, SOAIC 207*, 2017, pp. 172–176.

[3] A. Goranovic, M. Meisel, L. Fotiadis, S. Wilker, A. Treytl, and T. Sauter, “Blockchain applications in microgrids: An overview of current projects and concepts,” in *Proceedings IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society*, 2017, vol. 2017-January, pp. 6153–6158.

[4] K. Yeow, A. Gani, R. W. Ahmad, J. J. P. C. Rodrigues, and K. Ko, “Decentralized Consensus for Edge-Centric Internet of Things: A Review, Taxonomy, and Research Issues,” *IEEE Access*, vol. 6, pp. 1513–1524, Dec. 2017.

[5] N. Kannengießer, S. Lins, T. Dehling, and A. Sunyaev, “What Does Not Fit Can be Made to Fit! Trade-Offs in Distributed Ledger Technology Designs,” in *Proceedings of the 52nd Hawaii International Conference on System Sciences*, 2019.

[6] The Blockdata Team, “Blockdata.” [Online]. Available: https://www.blockdata.tech/.

[7] S. Popov, “The Tangle,” 2018.

[8] C. Lemahieu, “Nano: A Feeless Distributed Cryptocurrency Network,” *White Pap.*, 2018.

[9] M. Swan, *Blockchain : blueprint for a new economy*. Sebastopol, Calif.: O’Reilly Media, 2015.

[10] X. Xu *et al.*, “A Taxonomy of Blockchain-Based Systems for Architecture Design,” in *2017 IEEE International Conference on Software Architecture (ICSA)*, 2017, pp. 243–252.

[11] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, “Systematic Mapping Studies in Software Engineering,” in *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, 2008, pp. 68–77.

[12] B. Kitchenham, “Procedures for Performing Systematic Reviews,” Department of Computer Science, Keele University, UK, 2004.

[13] J. Yli-Huumo, D. Ko, S. Choi, S. Park, and K. Smolander, “Where Is Current Research on Blockchain Technology?—A Systematic Review,” *PLoS One*, vol. 11, no. 10, pp. 1–27, 2016.

[14] M. Dabbagh, M. Sookhak, and N. Safa, “The Evolution of Blockchain: A Bibliometric Study,” *IEEE Access*, vol. PP, p. 1, 2019.

[15] P. Rimba, A. B. Tran, I. Weber, M. Staples, A. Ponomarev, and X. Xu, “Quantifying the Cost of

Distrust: Comparing Blockchain and Cloud Services for Business Process Execution," *Inf. Syst. Front.*, Aug. 2018.
[16] I. Singh and S.-W. Lee, "Comparative Requirements Analysis for the Feasibility of Blockchain for Secure Cloud," in *Requirements Engineering for Internet of Things*, 2018, pp. 57–72.
[17] T. T. A. Dinh, R. Liu, M. Zhang, G. Chen, B. C. Ooi, and J. Wang, "Untangling Blockchain: A Data Processing View of Blockchain Systems," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 7, pp. 1366–1385, Jul. 2018.
[18] Stack Overflow, "Stack Overflow's Annual Developer Survey for 2019," 2019.
[19] UK Government - Office for Science, "Distributed Ledger Technology: beyond block chain," London, 2016.
[20] D. Mingxiao, M. Xiaofeng, Z. Zhe, W. Xiangwei, and C. Qijun, "A review on consensus algorithm of blockchain," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2017, pp. 2567–2572.
[21] The Iota Team, "Tangle visualization." [Online]. Available: https://github.com/iotaledger/iotavisualization.
[22] "The Go Programming Language - Documentation." [Online]. Available: https://golang.org/doc/.
[23] "A Tour of Go - Goroutines." [Online]. Available: https://tour.golang.org/concurrency/1.
[24] "A Tour of Go - Channels." [Online]. Available: https://tour.golang.org/concurrency/2.
[25] The Perlin Team, "The Noise Book." [Online]. Available: https://perlin-network.github.io/noise/noise.html.
[26] The Perlin Team, "Perlin." [Online]. Available: https://www.perlin.net/.
[27] Coral Health, "Code a simple P2P blockchain in Go!," *Medium*, 2018. [Online]. Available: https://medium.com/@mycoralhealth/code-a-simple-p2p-blockchain-in-go-46662601f417.
[28] The Nano Team, "About Nano." [Online]. Available: https://nano.org/en/about/.
[29] I. Meckler, E. Shapiro, and) Labs, "Coda: Decentralized cryptocurrency at scale," 2018.
[30] L. Baird, "The SWIRLDS Hashgraph Consensus Algorithm: Fair, Fast, Byzantine Fault Tolerance," 2016.
[31] Team Rocket, "Snowflake to Avalanche: A Novel Metastable Consensus Protocol Family for Cryptocurrencies," 2018.

XII. ANNEXES

A. *Throughput Data*

TABLE II. THROUGHPUT DATA FOR THE BLOCKCHAIN. IN TRANSACTIONS/SECOND.

Data Structure		Number of Nodes		
	Run #	2 Nodes	4 Nodes	8 Nodes
Blockchain	Run 1	456	418	405
	Run 2	455	440	396
	Run 3	453	440	403
	Run 4	450	435	394
	Run 5	457	437	394
	Average	454	434	398
	Median	455	437	396
	StdDev	3	9	5

TABLE III. THROUGHPUT DATA FOR THE TANGLE. IN TRANSACTIONS/SECOND.

Data Structure		Number of Nodes		
	Run #	2 Nodes	4 Nodes	8 Nodes
Tangle	Run 1	72	78	74
	Run 2	65	80	72
	Run 3	80	73	76
	Run 4	73	68	60
	Run 5	82	73	82
	Average	74	74	73
	Median	73	73	74
	StdDev	7	5	8

TABLE IV. THROUGHPUT DATA FOR THE BLOCK-LATTICE. IN TRANSACTIONS/SECOND.

Data Structure		Number of Nodes		
	Run #	2 Nodes	4 Nodes	8 Nodes
Block-lattice	Run 1	6,197	4,216	3,933
	Run 2	6,061	4,109	3,920
	Run 3	6,208	3,603	3,788
	Run 4	6,071	4,020	3,810
	Run 5	6,005	4,470	3,892
	Average	6,108	4,084	3,869
	Median	6,071	4,109	3,892
	StdDev	90	317	66

B. Latency Data

1) 8 Nodes

TABLE V. LATENCY DATA FOR THE BLOCKCHAIN. 8 NODES. IN MILLISECONDS.

Data Structure		Transaction #				
	Node #	#50	#100	#200	#400	#800
Blockchain	Node 2	6	4	4	9	14
	Node 3	6	5	7	8	22
	Node 4	5	4	6	8	21
	Node 5	4	5	7	10	23
	Node 6	8	5	6	10	24
	Node 7	3	5	6	12	24
	Node 8	6	6	7	6	31
	Average	6	5	6	9	23
	Median	6	5	6	9	23
	StdDev	1	1	1	2	5

TABLE VI. LATENCY DATA FOR THE TANGLE. 8 NODES. IN MILLISECONDS.

Data Structure		Transaction #
	Node #	#50
Tangle	Node 2	5
	Node 3	5
	Node 4	5
	Node 5	5
	Node 6	4
	Node 7	4
	Node 8	5
	Average	5
	Median	5
	StdDev	0

TABLE VII. LATENCY DATA FOR THE BLOCK-LATTICE. 8 NODES. IN MILLISECONDS.

Data Structure		Transaction #				
	Node #	#50	#100	#200	#400	#800
Block-lattice	Node 1	57	42	69	107	150
	Node 3	55	41	70	128	150
	Node 4	56	41	68	155	265
	Node 5	59	45	71	133	193
	Node 6	59	45	61	120	280
	Node 7	58	44	65	152	150
	Node 8	56	42	66	147	130
	Average	57	43	67	135	188
	Median	57	42	68	133	150
	StdDev	2	2	4	18	61

2) 4 Nodes

TABLE VIII. LATENCY DATA FOR THE BLOCKCHAIN. 4 NODES. IN MILLISECONDS.

Data Structure		Transaction #				
	Node #	#50	#100	#200	#400	#800
Blockchain	Node 2	14	7	8	8	9
	Node 3	16	5	10	9	16
	Node 4	14	4	6	9	19
	Average	15	5	8	9	15
	Median	14	5	8	9	16
	StdDev	1	1	2	0	5

TABLE IX. LATENCY DATA FOR THE TANGLE. 4 NODES. IN MILLISECONDS.

Data Structure		Transaction #
	Node #	#50
Tangle	Node 2	3
	Node 3	4
	Node 4	4
	Average	4
	Median	4
	StdDev	0

TABLE X. LATENCY DATA FOR THE BLOCK-LATTICE. 4 NODES. IN MILLISECONDS.

Data Structure		Transaction #				
	Node #	#50	#100	#200	#400	#800
Block-lattice	Node 1	65	46	67	124	263
	Node 3	63	44	65	148	279
	Node 4	64	44	65	182	353
	Average	64	45	66	151	299
	Median	64	44	65	148	279
	StdDev	1	1	1	29	48

3) 2 Nodes

TABLE XI. LATENCY DATA FOR THE DATA STRUCTURES. 2 NODES. IN MILLISEDONDS

Data Structure		Transaction #				
	Node #	#50	#100	#200	#400	#800
Blockchain	Node 2	6	5	8	13	12
Tangle	Node 2	8				
Block-lattice	Node 1	64	36	67	110	243

C. Size Data

TABLE XII. SIZE DATA FOR THE BLOCKCHAIN

Blockchain	Size (bytes)
Blockchain	40
[]Block	24
Block	112

TABLE XIII. SIZE DATA FOR THE TANGLE

Tangle	Size (bytes)
Tangle	96
[]Transaction	24
[]Link	24
Transaction	160
Link	16

TABLE XIV. SIZE DATA FOR THE BLOCK-LATTICE

Block-lattice	Size (bytes)
Block-lattice	8
[]Cube	24
Cube	160