Bachelor Degree Project

Distributed Data Management in Internet of Things Networking Environments
- IOTA Tangle and Bitcoin Blockchain Distributed Ledger Technologies

Author: Gerard Ruiz Caparrós
Supervisor: Mauro Caporuscio
Date: 2018-05-23
Subject: Computer Science

Abstract

Distributed ledger technology (DLT) is one of the latest in a long list of digital technologies, which appear to be heading towards a new industrial revolution. DLT has become very popular with the publication of the Bitcoin Blockchain in 2008. However, when we consider its suitability for dynamic networking environments, such as the Internet of Things, issues like transaction fees, scalability, and offline accessibility have not been resolved. The IOTA Foundation has designed the IOTA protocol, which is the data and value transfer layer for the Machine Economy. IOTA protocol uses an alternative blockless Blockchain which claims to solve the previous problems: the Tangle.
This thesis first inquires into the theoretical concepts of both technologies Tangle and Blockchain, to understand them and identify the reasons to be compatible or not with the Internet of Things networking environments. After the analysis, the thesis focuses on the proposed implementation as a solution to address the connectivity issue suffered by the IOTA network. The answer to the problem is the development of a Neighbor Discovery algorithm, which has been designed to fulfill the requirements demanded by the IOTA application.
Dealing with IOTA network setup can be very interesting for the community that is looking for new improvements at each release. Testing the solution in a peer-to-peer specific protocol (PeerSim), with different networking scenarios, allowed us to get valuable and more realistic information. Thus, after analyzing the results, we were able to determine the appropriate IOTA network configuration to build a more reliable and long-lasting network.

Keywords: Distributed Ledger Technology (DLT), Bitcoin, IOTA, peer-to-peer, dynamic networks

Preface

This study is, in fact, the product of a fruitful collective collaboration and could not have been completed without the participation of several people who, in different ways, gave me support during the last academic semester. I do not want to miss this opportunity without expressing my sincere thanks to my thesis supervisor, Professor Mauro Caporuscio, whose comments and advice, without a doubt, enriched this work and, in moments of frustration, allowed me to see rays of light - thank you!
I also would like to make a special mention to Mirko D'Angelo, Master's student at the University of Linnaeus, whose comments on the most technical part of the simulation and advice on the approaches have been very appreciated.
Finally, last but not least, thanks to my family and friends who have been supporting and encouraging me throughout this project.

Contents

1 Introduction

This bachelor thesis aims to give a broad overview of two different Distributed Ledger Technologies, Blockchain, and Tangle, in the Internet of Things (IoT [1]) networking environments. First of all, a literature review was conducted for both technologies to acquaint with their operation and main features. At the same time, all the essential concepts were described in this thesis for the subsequent study of both technologies in the IoT environments. The document makes particular attention to those qualities that are different in Blockchain and Tangle, and analyze their advantages or disadvantages for IoT environments. Since Tangle was explicitly designed for IoT scenarios, IOTA [2] and the Tangle architecture have been emphasized throughout this thesis.

As far as the experimental section is concerned, and on account of the primary stage in which IOTA protocol currently is, critical connectivity issues can be found. Despite all features explained in its whitepaper, there is still a significant lack of them in the software version. In the case of network setup, currently, there is no protocol for automatic neighbor discovering, so the connectivity with other nodes has to be done manually through an exchange of IP addresses between users. Hence, most of the community members continuously experience problems with their nodes synchronization because when they lose connectivity with their neighbors, its node is not able to automatically find another. With the purpose of preventing this situation, the goal of our research was to encourage an automated neighbor discovering, which allow nodes to connect more efficiently and which will prevent the owners from having to ensure that their nodes are still synchronized.

The implementation of the neighbor discovering protocol has been performed in PeerSim, which allowed us to simulate the execution on a peer-to-peer (P2P) network, and therefore, to obtain more realistic data regarding its behavior in a real scenario. After obtaining all necessary data, a new program was implemented to employ a previously defined metric, and measure the success of the solution. Finally, for the visualization and topology analysis of the network, we have used Gephi [3].

1.1 Background

This thesis aims to give an overview of the topic described in the section above. Since they are technologies which are continually being updated with new functionalities, there are some future features, which are not referenced in this thesis. Apart from this, because IOTA is currently in a very primary stage, several main theoretical characteristics (e.g., Automated Neighbor Discovering, Offline Tangle, Automated Snapshotting) are still not available in its latest software version. Regarding the research references, due to the constant development and the recent publication of this technology, it has been hard to find the latest updates in academic papers. Thus, for all proven concepts that are already implemented, information from academic papers was gathered for the thesis. Nevertheless, much of the collected information arise from IOTA's online communities, e.g., IOTA Forum, IOTA Blog, IOTA Slack Channel, Reddit, Discord channel, several explanatory videos and online interviews, where most of the community members, developers, and co-founders participate actively.

[1]Internet of Things (IoT) is a global network of devices that have Internet connectivity, as well as, can acquire information about their surroundings and communicate it among them [1].

[2]IOTA is the approach that incorporates the Tangle technology as a public distributed ledger, at its core. This approach is currently being implemented as a cryptocurrency called iota [2].

[3]Gephi is an open-source visualization and exploration software for all kinds of graphs and networks.

Ledgers have been present in the commerce since ancient time and are used to record all sort of information, mostly capital movements and money [3]. However, the first application of Distributed Ledger Technology (DLT) introduced with the cryptocurrency Bitcoin in 2008 [4]. Since then, DLT has become one of the latest digital technologies appearing in the next industrial revolution. Digital ledgers are a consensus of replicated digital data that is spread across multiple geographies, sites or institutions and are usually public. Records are continuously stored one after the other in the ledger, and the participants from any node of the network can access the recordings shared across that network and own an identical copy of it [3]. DLTs are a very recent technology that has aroused frenzy in banks, companies, governments and other institutions, which are continually working on developing distributed ledgers for different purposes. All this sudden interest is due to the significant advantages concerning data availability, decentralization, and security, that this technology offers to new applications.

In its purest form, DLT represents a unique technology in two different ways. Firstly, it is distributed; third party's validation is replaced in a DLT by a set of cryptographic solutions, which allows a general consensus among the nodes of the network [5]. Secondly, users can deposit on the digital ledger digital assets (e.g., transactions, records, acts), the record of them remains immutable, transparent and resistant to manipulation on account of technology's cryptography and distributed design.

A critical issue that arose was that a distributed ledger could achieve digital identification [4]. In a distributed infrastructure, authentication must be ensured to establish trustful connections between parties on the network. As a ledger, it could store digital identities of humans, organizations or even Internet of Things (IoT) devices. As far as the Internet of Things is concerned, a distributed ledger for Machine-to-Machine (M2M) communication is presented as a worldwide network with millions of devices interconnected. In order to build and manage such a comprehensive database for the Internet of Things, IOTA Foundation was created.

The most well-known DLT used to provide digital identification so far is called Blockchain. However, this solution is not entirely appropriated for huge scenarios, such as IoT networks, due to two significant drawbacks:

- Transaction fees
- Limited scalability

IOTA Foundation hereby developed an alternative technology called Tangle, which solves the previous shortcomings and allows a suitable distributed ledger for IoT devices. Tangle is a very scalable technology, which does not only allow a large number of devices but also its performance is based on scalability. The more transactions are issued, the better the performance [6]. It is mandatory to have a ledger that is scalable since IoT consists of millions of devices which are interconnected and constantly communicating between them, generating massive traffic of transaction over the network.

Moreover, another feature of IoT network is the small value transactions (micro-transactions) that are sent, sometimes with value zero, messages. Hence, if for each transaction the fee has to be paid and the value transferred is minimal, the IoT scenarios would be inconceivable because the cost of issuing transactions would be higher than the quantity sent it.

[4]Digital identification is achieved with a digital identity used by computer systems to represent an external agent (e.g., person, organization, application). The information contained in a digital entity allows assessment, authentication, and make it possible for computers to mediate relationships.

1.2 Related work

As it has been introduced in the previous section, one challenge for this thesis was the search for IOTA information in published articles. Since Tangle is such a new technology and is still in its primary stage, the amount of papers is very scarce. Nevertheless, some articles such as [2], [7] and [8], were used to get a general overview of the Tangle, especially the last version of The Tangle whitepaper. Thus, to gather more information for the research section 2, other information sources were also used. The IOTA Blog publishes articles made by members of the IOTA Foundation, developers and even the founders of IOTA. Some of these articles [9], [10], [6], [11], and [12], include a general overview of the features and specifications of the technology, others instead, are focused on explaining a specific property in detail.

On the other hand, the related work previously done on Bitcoin Blockchain is much more extensive, and there are published articles in practically all fields related to the technology. Section 3 of the thesis was based on several articles to obtain the information, for instance, the Bitcoin whitepaper [13] was used to get the general overview of the protocol and its technology. Other related works used were the literary review [14] written by Conoscenti, M., Vetro, A. and De Martin, J. C about the suitability of blockchain for the IoT, the article [4], written by Decker, C. and Wattenhofer, R which has a complete overview of the technology and information about the network configuration, that was contrasted and complemented with the Bitcoin Developers Guide [15].

Regarding the related work to the problem-solving activity suggested in this thesis, a company called SemkoDev was found, and it is currently dealing with both similar and other problems of IOTA. Their proposed solution is called CarrIOTA Nelson, and it is presented as a node which runs a simple API on top of the IOTA Tangle [16]. This tool was thought to be used with IOTA's software [17] in order to solve the connectivity problem formulated in section 1.3.

1.3 Problem formulation

For this thesis, we wanted to attempt to ascertain the behavior between two distinct DTLs in an IoT networking environment. First of all, we find Bitcoin Blockchain, which despite its reputation, is poorly suitable for IoT applications [14]. Whereas on the other hand, we have IOTA Tangle an innovative project, whose design is wholly focused on IoT scenarios. From the beginning of this project, we wanted to compare and analyze both of them from an IoT networking point of view, to be able to answer the two research questions specified in table 1.1.

RQ1	What are the similarities and differences between IOTA Tangle and Bitcoin Blockchain?
RQ2	What makes IOTA Tangle the appropriate technology for IoT networking environments when Bitcoin Blockchain is not?

Table 1.1: List of research questions

However, the problem formulation has changed along the thesis. In the beginning, and as it was specified in the project plan, the project was more centered on the idea of comparing both technologies with the implementation of a prototype of the DLTs protocols

[5] in Peersim simulator. Nevertheless, after a rigorous literature review, we determined that the comparison between the technologies should be addressed in a more theoretical way. Thence, regarding the problem formulation, we focused all our efforts on designing, implementing and testing a neighbor discovery protocol for the IOTA network setup. The lack of a neighbor discovery protocol is a severe drawback present in the latest IOTA software release that makes the performance of the network worse than the expected by the users.

Dynamic networks are known for the instability in the connection of their nodes. Thus, in order to find a solution for the connectivity of the nodes and achieve an appropriate network performance, the neighbor discovering protocol was necessary. Nowadays, IOTA implementation does not have this protocol available because as Dominik Schiener [19] argues, the old neighbor discovering protocol generated several problems to IOTA, one of the most important was the exponential increase in bandwidth. However, drawing from the premise that any dynamic network needs a protocol that automatically manages the connectivity of all the nodes, the thesis focused on this. To face this problem, a new peer discovering algorithm based on weights was designed. Moreover, different approaches were implemented with the purpose of finding the best solution and achieve an efficient IOTA network setup.

1.4 Motivation

Our initial motivation for the thesis was to compare both technologies; IOTA Tangle and Bitcoin Blockchain, building up a prototype of each technology and test their performance in several Internet of Things networking environments using Peersim. After the research analysis about the properties and operation of each technology, and in addition to the setup process of their respective networks, we realized that because of the IOTA infancy stage, its network did not fulfill the requirements to operate satisfactorily in an IoT networking environment, that was for what it had been designed. Then, we thought that it could be fascinating to work on the neighbor discovery protocol. Because IOTA is an open-source code, this thesis could be useful for future studies or implementations on IOTA, even for other similar applications since we have worked on the network layer.

Due to the technological innovation that IOTA Tangle represents, and because of the lack of information[6], it was thought that this thesis could also be an excellent opportunity to collect the most important information describing how IOTA Tangle works and what makes its operation suitable for IoT networking environments. Furthermore, since currently, the most well-known DLT is Bitcoin Blockchain we were fond of the idea of analyzing this technology and looking for reasons to answer the research questions 1.1. The fact that both technologies were studied is positive since we can compare their behaviors on IoT scenarios and highlight the advantages and disadvantages of each DLT.

As personal motivation, this project represented a great opportunity considering my interest in continuing my Master studies in the field of computer networks and distributed systems. It allowed me to study two current distributed systems and work with specific software tools, such as the distributed network simulator, which could be used throughout my professional career.

[5]Bitcoin and IOTA are the protocols implemented in the Bitcoin Blockchain and IOTA Tangle, respectively [18], [2].

[6]Information is mostly spread in online communities (such as Discord, Slack and Reddit) instead of in research papers.

1.5 Objectives

The objectives to achieve along this project are the following:

O1	Study the features of IOTA Tangle and Bitcoin Blockchain and their performance in IoT environments.
O2	Design a Neighbor Discovering Protocol prototype for the IOTA network setup.
O3	Run several executions with different network scenarios and measure its performance.
O4	Analyze the results and make conclusions about the strengths and weaknesses of the implemented solution for IoT networking environments.

Table 1.2: List of the objectives for the thesis.

In previous section 1.3 the problem formulation was defined and supported with two research questions shown in table 1.1. In this section, the prime objective of the literature review was defined to obtain answers to the research questions. Our expectation for this thesis was to achieve a report which contains enough information to distinguish the differences between both DLTs, and how each of them works. In addition to understanding the reasons for their appropriateness to be run in an IoT environment.

Furthermore, in section 1.3 also appeared the problem formulation of IOTA connectivity, referring to the fact that there is no neighbor discovery protocol. Thus, the other three objectives aimed to design different prototypes with alternative approaches, and study their performance to integrate the best solution in the IOTA network configuration process. Each version had to include an improvement in its design that should lead to an upgrade of its performance. The design evolution was progressive and incremental among the versions; therefore, the performance results of each version should be better than the previous strategies. Nonetheless, as it is seen in section 7, some approaches aimed at improving the network design and nodes integration, could negatively impact the overall performance of the network.

1.6 Scope/Limitation

The first part of the report is focused on the theoretical analysis of the DLTs. Since we are working on distributed data management in IoT networking environments, the thesis showed a particular interest for the IOTA Tangle. Therefore, the content of the IOTA Tangle chapter is more extended, and some processes are explained more in detail. However, there are limitations to the concepts of security, protection against attacks, and the encryption process of transactions. On the other hand, the objective of the Bitcoin chapter is to show to the reader a general idea of how this DLT works, and the main reasons why it is not possible to be adopted in IoT environments. Regarding the limitations, there are the same as in the previous case for the DLT security, and as a consequence of not having deepened so much in the explanation of the processes, a more generalized explanation is observed.

As far as the implementation is concerned, it includes a functional version of the neighbor discovering protocol. Nevertheless, the simulation was limited to a *cycle-based* model instead of an *event-based* model. The difference is that cycle-based model makes it possible to achieve extreme scalability and performance, at the cost of some loss of realism [20]. This gap of realism is there due to the lack of the transport layer in the simulation, and the lack of concurrency.

Regarding the analysis, the current approach is based on a static observation of the network durability along with a timeline. The limitation of this process is the fact of not analyzing the implementation in a dynamic way, where new nodes would be added along the simulation.

1.7 Target group

This thesis encompasses different objectives, which are divided between the literary research and the implementation process. As a consequence, the target group is broader since we can find people who are just interested in the analysis of the DLTs and their differences, and others who care about the whole concept. The first target group interest would be limited to reading the first chapters that refer to the theoretical concepts. However, the second target group which includes students, software developers, even members of the IOTA community, could be as much interested in the theoretical concepts as the implementation and analysis process of the neighbor discovery protocol.

IOTA community is very active and interested in the evolution of this technology, for this reason, projects like this thesis can be a great help and inspiration for all those who are working in different ways with IOTA.

1.8 Outline

This thesis is structured along the research objectives mentioned above. After the introduction, chapter 2 covers the data structure and the protocol that make up the Tangle. In chapter 3, the reader gets to know how the Blockchain works differently or similar to the Tangle. For this, it is essential to have understood most of the previous sections. Furthermore, these two first chapters also answer the research questions 1.1 previously defined.

Henceforth, the next chapters cover the implementation process of this thesis. The fourth chapter, 4, shows which steps the problem-solving activity follows in order to find a solution to the problem, previously defined in the problem formulation section. The fifth chapters, 5, explains which tools have been used in the implementation and how each step defined in the previous chapter has been developed.

After the developing process of the implementation, the report focuses on the results obtained. The sixth chapter, 6, shows the results, whereas, the seventh chapter, 7, displays the explanations and conclusions about them.

Finally, the last two chapters refer to the whole process of the thesis. The eighth chapter, 8, holds a general discussion about the results, and if the problem has been solved. Furthermore, it focuses on how this thesis relates to previous works. The last chapter, 9, reveals a conclusion of the findings and suggests a perspective of further work that can be done in the future.

2 IOTA Tangle

2.1 General specifications about Tangle

IOTA Tangle is an unprecedented new open-source distributed ledger that does not use a blockchain. Instead of a global blockchain, there is a directed acyclic graph (DAG) called *Tangle*. The Tangle is a new data structure, which gives rise to unique features such as zero fees, scalability, fast transactions and secure data transfer [6]. Due to this new architecture, the behavior in IOTA works differently in comparison to Bitcoin Blockchain. As such it contains no blocks, no chain and also no miners.

It is important to emphasize that Tangle is made of transactions (tx), which are called *sites*. So, every issued tx by nodes that are connected to the IOTA network constitutes the set of the tangle graph, which is the ledger for storing transactions. Tangle is a consensus structure which instead of requiring miners to perform the proof-of-work and validating blocks of transactions, it achieves consensus because every participant node on the network has to perform consensus validating two previous transactions in the Tangle before they make a transaction [2]. As a result of this consensus, we get the academic vision of the Tangle reflected in figure 2.1.

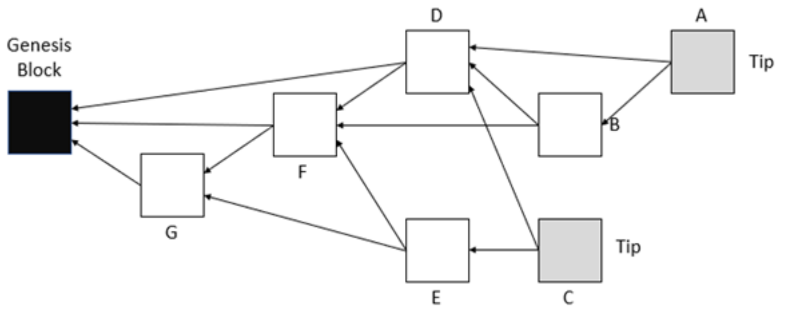


Figure 2.1: IOTA Tangle visualization with the genesis represented with the black square and two tips (*A* and *C*) colored in grey.

The timeline of the previous figure goes from left to right, wherein the beginning of the tangle there is a unique transaction called *genesis*. The importance of the genesis is that all IOTA tokens[7] were created in this first transaction. No additional iotas will ever be created. This leads to a fixed supply of IOTA-tokens of exactly $(3^{33} - 1)/2 =$ 2,779,530,283,277,761 [21]. This exact quantity was chosen for ternary computation [8] efficiency and in an attempt to be sufficient quantity of tokens in IoT scenarios.

The starting point of everything in IOTA is the seed, since it is indispensable to create an account with addresses and its corresponding keys. A seed consists of 81 trytes and is the unique access to the addresses of an account, besides all its funds [22]. For this reason, it is essential to keep it always save. In order to generate a seed safely, there are several seed generators which create a random sequence of 81 characters using only A-Z and the number 9.

IOTA addresses, unlike Bitcoin Blockchain, cannot be used more than once if they have been used for outgoing money transfers. This fact means that addresses do not have

[7]IOTA tokens refer to every "coin" of the iota cryptocurrency.

[8]Because it is not entirely relevant to this thesis, there is not a special mention about the ternary computation IOTA uses. The ternary logic is simply a different numerical system that allows a more efficient and faster calculation instead of the binary one.

any limit on how many incoming tx receive, but as soon as they are used to transfer any quantity of tokens, it will no longer be secure to use them again [23]. The reason why they are not useful anymore is that when the address is used to transfer money, they need to be signed by the owner and this process uses a signature scheme based on Winternitz-One-Time-Signature-Scheme [24], which reveals part of the private key. Therefore, there is the possibility that, due to this vulnerability and through brute-force[9], others may acquire the private key and therefore its funds. Since a considerable quantity of addresses will not be useful after transferring, it is essential to be able to create new addresses. Every address in an account is calculated from the combination of seed + index of the address, where the address index is an integer starting from 0. Address index is unique for each address, so if there is an address with that index attached to the Tangle, it will be attempted with the next index.

Before using any new address, it is highly recommended to attach them to the Tangle, even if iotas can be sent to them. The purpose of attaching an address to Tangle is to prevent the re-use of the addresses private key. When an address is attached to Tangle, it indicates that the address is in use or it will be. If users do not attach addresses in the Tangle, they may be compromised. The next example shows how this could happen: Alice's wallet has address 0 with 500 iotas, and Albert and Giannis want to send her 2500 iotas. Then, Alice generates a new address (address 1) and sends it to them, but she does not attach it to the Tangle. Afterward, she sends 100 to Charles, and because of how system transaction works on IOTA, section 2.1.1, the change is sent to a new transaction in Alice's wallet. As the next address 1 was not attached, which means that is not being used, it is now attached and used for the unspent 400 iotas. Later, Alice sends 100 more iotas to Miquel and, as in the previous case, the unspent 300 iotas are sent to address 2, which is automatically attached. However, because Alice made the transaction from the address 1, its private key is partially revealed, and therefore it becomes unsafe to use it. Nonetheless, Albert and Giannis previously had received address 1 as a destination to their transactions. So, when Albert and Giannis issue their transactions the amount of money will be linked to address 1 which is not safe anymore.

Another remarkable reason to attach addresses is that IOTA's wallet[10] does not create new addresses before the previous one has been attached. The wallets do not store the balances of the addresses by itself, but they request them to the Tangle. It is necessary that they are attached so that the wallet can show the balance of each of them [23]. As a consequence of how wallet works, if there is an attached address which contains iotas, its balance would not be reflected on the wallet because the Tangle does not know about the unattached address.

Attaching an address means publishing the address to the Tangle. When an address is attached to the Tangle, it creates a zero-value tx referencing that address [26]. When a zero-value tx is sent, the next steps to successfully attach the address are the same as those to make a tx, subsection 2.1.1. Thus, after the tx, two tips from the Tangle are chosen and validated, and then the proof-of-work is performed. Finally, the attached transaction is ready to be used.

[9] *"A brute-force attack is a cryptanalytic attack that can, in theory, be used to attempt to decrypt any encrypted data"* [25]

[10] IOTA's wallet is a software program that stores your private and public keys, enabling you to send and receive transactions through the Tangle, as well as monitoring your balance.

2.1.1 How IOTA makes a transaction

From the point of view of IOTA, a tx is the transfer of σ tokens from one address to another. Making a transaction in IOTA is a process which contains 3 main steps [27]:

1. **Bundling & Signing:** Whenever a transaction should be added to the Tangle, a node must first create a bundle. IOTA uses an Unspent Transaction Output (UTXO) scheme, which means that *inputs* (addresses) are needed to transfer tokens from them to the *outputs* (addresses). Furthermore, bundles are atomic, which means that either all tx are accepted by the network, or none [28]. As a consequence of using UTXO, most of the times, to make a transaction of x tokens it is necessary to include several transactions in the bundle.

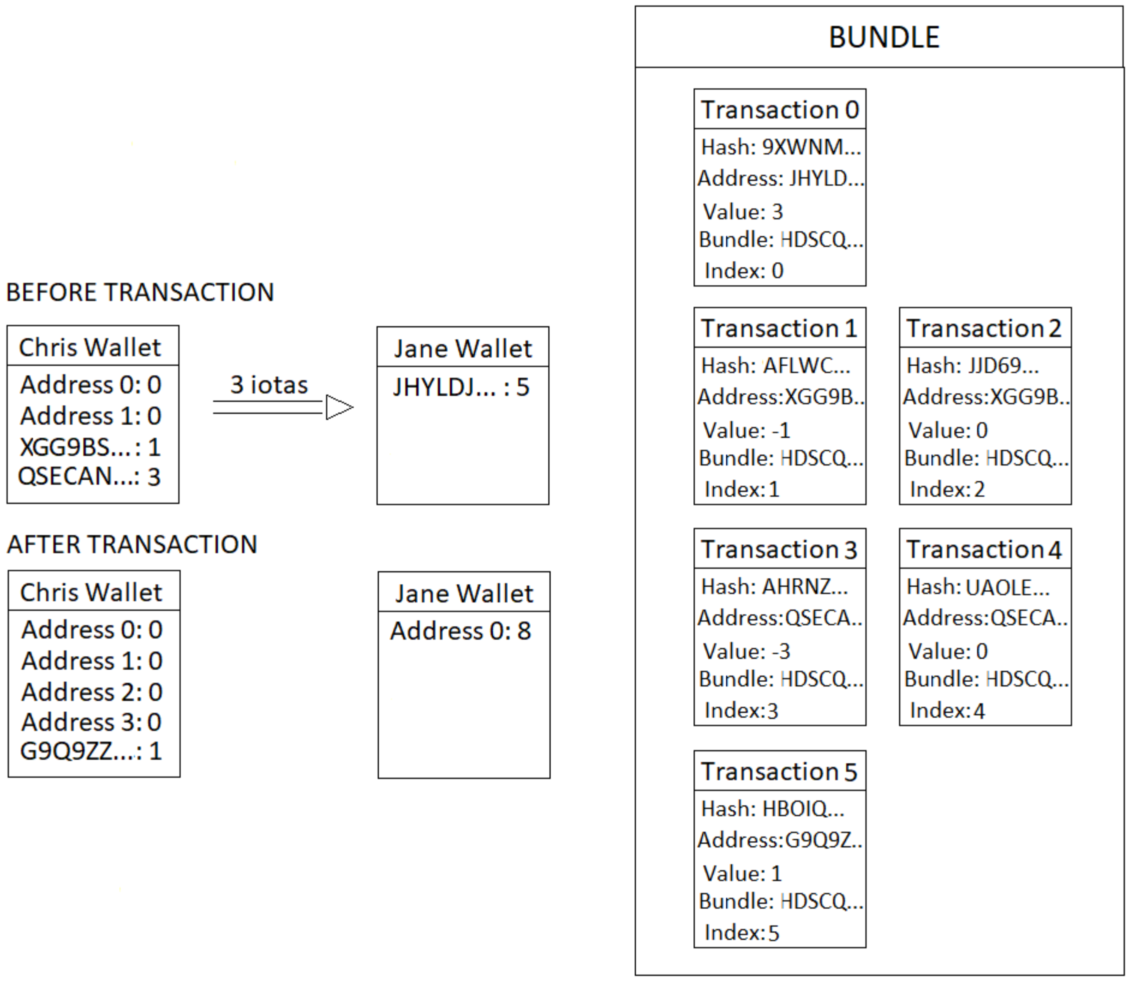


Figure 2.2: In the left part of the picture, it is indicated the state of the wallets after and before the transaction. Meanwhile, in the right part, it is shown the appearance of the resulting bundle.

The figure 2.2 shows the process of issuing a transaction, following UTXO scheme rules. Besides, it has been used as a reference for the next explanation. Typically, in a wallet, there are several addresses, and because the amount of funds in a single address may not be sufficient for a transaction, more than one can be used as inputs. Regarding the outputs, there could also be more than one although in the previous example all tokens were transferred to the same address. In IOTA the only way a node can issue txs is within a bundle. Thus, when a node makes a transaction has to choose which of its addresses are used to transfer and include them in the bundle. Because these txs are included in the bundle as inputs, their values are negative.

After every input there is a tx with a value equal to zero, we call this tx *meta transaction*, and it contains part of the previous transaction signature[11]. IOTA uses asymmetrical cryptography when it sends transactions, and as any other asymmetrical two keys are used: a public key and a private key. Thereby, the authentication of the sender is accomplished by the receiver of the transaction, who checks with the public key if the holder of the paired private key was who issued the transaction.

On the other hand, recipient addresses also have to be included in the bundle as the outputs, even so, because they are receiving tokens their value is positive. Finally, because the amount of money transferred does not correspond to the amount agreed upon in the transfer, all remaining tokens are transferred back to the sender in a last positive transaction to a new address called remainder, and which is controlled by the sender. In order to build an accepted bundle, it must satisfy the condition in which the sum of all included transactions must be 0 [24]. This complicated scheme is applied because addresses may be used as inputs only once since the signature is generated with a signature scheme based on Winternitz-One-Time-Signature-Scheme.

2. **Tip Selection:** Once the bundle has been built, it is time to attach a transaction to the Tangle. However, before a transaction can be included in the Tangle, every new transaction must approve two previous non-confirmed transactions (tips), this process is called validation, and it can be seen more in detail in subsection 2.6. In order to find them, the node performs an algorithm called Tip Selection Strategy, which is also cautiously explained in subsection 2.5.

   After the two previous processes and if everything was successful, the new tx can be included in the Tangle. Since we know that a tx represents a bundle of txs inside it, we can assume that every tx in the Tangle is a bundle of txs. Every tx inside the bundle is connected with each other, but besides, they are also linked with the tips of the tangle. Therefore, every bundle in the Tangle is also connected to the tips to approve, selected by the Tip Selection Strategy.

[11]Depending on the security level used to sign a transaction, it is necessary to issue 0, 1 or 2 meta transaction with the signature [29]. In the previous case, it was used the security of level 2, so 1 meta transactions was sent for each input

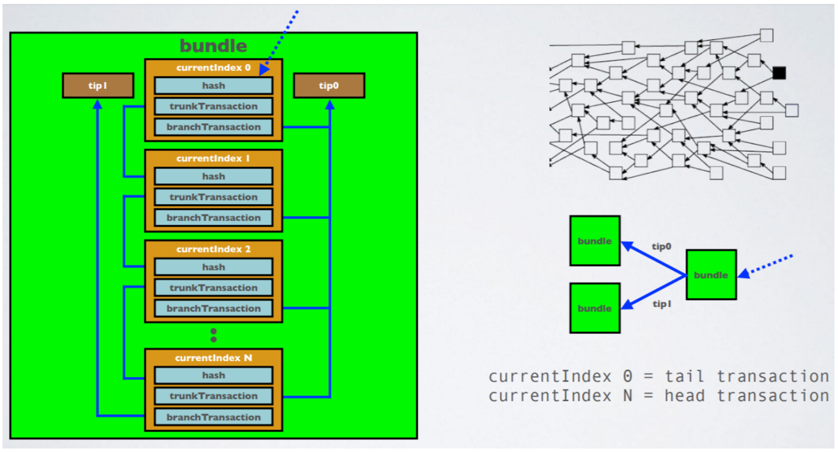


Figure 2.3: On the left, there is a zoom-in snapshot of the bundle with its transactions connections. Meanwhile, on the right part a zoom-out of the bundle connections to the Tangle.

Previous figure 2.3 introduces two new concepts, which are represented in the *trunkTransaction* and *branchTransaction* fields. The objective of these fields is, what in computer science is known as a pointer. The branchTransactions is in charge of attaching the bundle txs to a tip, in this case, tip 0, but except for the last tx (*head transaction*) which its branchTransaction field points to the tip 1. However, trunkTransaction which also acts as a pointer, it is responsible for connecting every tx with the tx below. Again, except for the head tx, which its field points to the tip 0. All these unions between txs are made by referencing the unique hash[12] of each tx. Moreover, the strategy used for linking bundles is similar, but in this case, all references to tips are done to the hash of the first tx in the bundle, as it can be observed in the dashed arrow of the image.

With all these connections, IOTA achieves its compromise of having all txs interconnected between them along the Tangle. This feature is crucial for the validation process, which requires to check several linked txs through the Tangle.

3. **Proof of Work:** The last step before a node issues a tx, is the performing of a Proof of Work (PoW). Once the bundle is constructed, signed and the tips are added to the bundle, the PoW has to be done for each transaction in the bundle. Every transaction in a bundle requires a *nonce* to be accepted by the tangle network.

   The proof of work is based on the execution of an algorithm to solve a computationally hard puzzle, but which is easy to verify it later. IOTA's PoW is directly comparable with Hashcash[13] due to the fact that they share the same goal of preventing spam, although IOTA's PoW is also used for preventing Sybil attacks[14].

   The IOTA team designed their cryptographic hash function called Curl, which is used to perform the PoW [24]. To perform a PoW, we need to define the *Minimum Weight Magnitude* (MWM) which is the minimum number of trailing zeros in the output of

[12] A hash is the unique fixed-length output that a hash function generates from an arbitrary length input. [30]

[13] *"Hashcash was originally proposed as a mechanism to throttle systematic abuse of un-metered internet resources such as email, and anonymous remailers in May 1997"* [31]

[14] *"Sybil attack is an attack in which the identities of the node are subverted, and a large number of pseudonymous identities is produced to gain the access of the network" [32].* Moreover, it is considered one of the most damaging attacks in P2P overlay network.

the hash function Curl, or more informally known as the difficulty of the PoW. Currently, in IOTA main network, it is set to 14 and each increment of the MWM, on average, triple the PoW difficulty [33]. The PoW is a complex mathematical process, but because it is not the main point of this thesis its explanation is delimited to the fundamental performance. The Pow seeks to find a particular value for a counter that, in combination with the transaction data, achieves a nonce from the Curl hash algorithm with an MWM of trailing zeros. The nonce is the result of the Curl hash and the counter increases in each execution until at some point the nonce matches the MWM, and the PoW is completed.

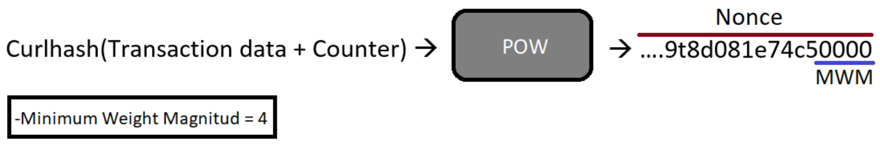

Curlhash(Transaction data + Counter) →
POW
→ 9t8d081e74c50000
Nonce
MWM
-Minimum Weight Magnitud = 4

Figure 2.4: This figure illustrates the PoW process with the input of the PoW, and how after the process the nonce is achieved with an MWM = 4.

After the successful performance of the PoW, the nonce found has to be included in each transaction of the bundle. Finally, after updating all new data, the transaction is ready to be broadcasted through the network and wait for it to be approved by someone else.

2.2 Network Setup

A node starting up for the first time has its database empty. It needs to get all transactions in the Tangle, and this process can take some days. The method to accomplish this task consists of constantly listening for broadcast, and when the node receives new transactions from its neighbors, it adds them to the Tangle. The IRI [15] process listens on *udp_receiver_port* and *tcp_receiver_port* to accept transactions from its peers. Each transaction passes through a validation process, subsection 2.6, before being added to the ledger. The validation process consists of several steps in which it is ensured that the transaction is valid by checking fields such as [35]:

- An acceptable timestamp
- An acceptable value field
- A valid signature
- A valid nonce

Every node is directly connected to other peers which are continually broadcasting txs from others. Hence, every node is receiving and broadcasting txs at all times. However, when one node receives a tx that it did not have in its ledger, it will validate the tx and increment the counter numberOfNewTransactions. Since the tx is considered new, it will also be placed in the *broadcastQueue* to be later sent to the node neighbors. Thereby, the node is solidifying a view of the tangle with all txs received.

[15] IOTA Reference Implementation (IRI) is the only core client available and has implemented all necessary core functionality for participating as a full node in the network. It exposes an API that makes it possible to access all required functionality and utilizes IRI for making transactions. [34]

IOTA txs are always referencing two others: tx1 and tx2 (trunk and branch fields). So, when one node receives a tx, it also checks if either tx1 or tx2 is or is not in the Tangle of our node. If it is not, then the transaction hash of tx1 or tx2 is added to the set of transactions to request, *transactionsToRequest* queue. At some point, the node will take a look at the queue and requests to get more details about the tx from its neighbors. When any peer receives a request from another node, it looks through its copy of the Tangle and checks if it has been previously validated. If it was the case, the peer answers the request by sending the transaction data back to the node.

During the synchronization process, the transactionsToRequest queue increases and only decrease as the ledger of the node approach consensus with the ledger of the other peers. Finally, once the database of the node is completely restored, it has to be verified that the node is synchronized. Currently, the consensus is achieved if the last Coordinator transaction, called latestMilestone is approving all the transactions in the Tangle, which means that all of them become confirmed [36], it can be seen in section 2.3.

However, there is a faster alternative to achieve an updated database, which consists of downloading a copy of the previous Tangle status and replace it with the empty database that the node had at the beginning. After the replacement, the node still needs to update the last transactions, even though it is faster than updating all of them. However, the approach for getting the missing transactions is the same as before, listen for new transactions and request trunk and branch transactions to keep fulfilling the Tangle until it becomes synchronized. It has to be stressed that the time to get all missing transactions and become a synchronized node also depends on the performance of the neighbors. Thus, even if a node does not issue a transaction, so it does not need to share any tx to approve its txs, it still has incentives to participate if it wants to keep being a node of the network [2].

It is essential that nodes have reliable neighbors to get as many transactions as possible to stay synchronized. In order to keep beneficial neighbors, every node collect data about its neighbors performance. For example, they notice when any of their neighbors do not have proper behavior, or they detect if a neighbor has not been broadcasting transactions for several hours. Since the connections are bidirectional, nodes should ask about its neighbors poor performance and advise them before removing them from the iota.ini configuration file.

The iota.ini file is used for the startup of a node [37]. All neighbors and other parameters have to be specified there, and because this process is done manually, there may also be some notes indicating the availability of each neighbor and trust level. It is essential to be as much available as possible, and if at any time the node is turned off, previously all neighbors should be warned of it.

2.3 IOTA Network Consensus

Since IOTA Tange is a DLT, public consensus is one of the most important qualities that must be ensured, in order to be useful as a payment method. In IOTA, the consensus is reached when a tx is considered to be confirmed. A confirmed transaction is accepted into the public consensus and is very unlikely to be removed from it. There are two strategies to achieve consensus in the Tangle [38]; the Coordinador which is the one currently implemented, and the distributed approach which is the suggested in the whitepaper but still not available.

The Coordinator (Coo) is an entity controlled by the IOTA Foundation, which issues special txs without any value every two minutes, called *milestones*. Currently, milestones set the general direction for the Tangle growth, and also do some checkpointing. Transactions (in)directly referenced by milestones are considered confirmed. The Coo is a temporary measure to help bootstrap the network and protect it against attacks during its infancy stage, and until there are sufficient nodes in the network that it becomes unnecessary. Although there is the figure of the Coo, as the Founder of IOTA, David Sønstebø, says [10], it does not mean that IOTA ledger is centralized at any point. IOTA ledger is entirely decentralized since every node has to validate every transaction issued to check that consensus rules are not being ignored. The IOTA Foundation has not yet specified a deadline to remove the Coo from the system. However, what they said is that is not going to disappear suddenly, it will be a gradual process [39].

On the other hand, the future alternative for the Coo approach is the distributed solution, which gives a probabilistic answer. Along the consensus process, we get an academic vision of the Tangle like the one reflected in the figure 2.5.

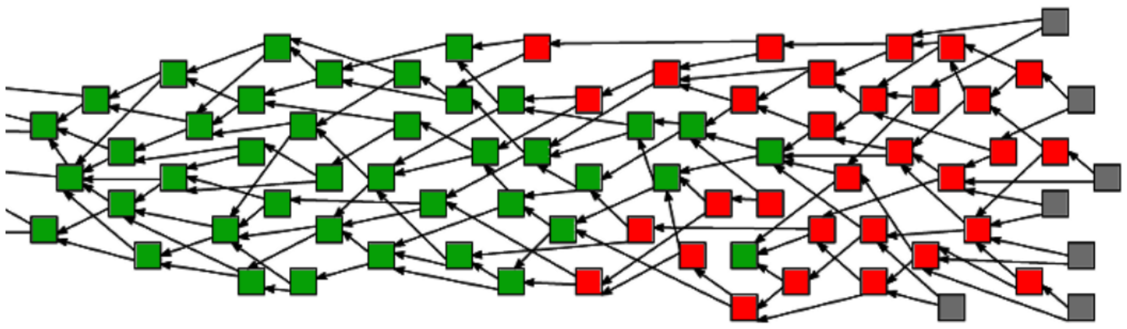


Figure 2.5: Represents the incoming transaction flow where green squares are confirmed transactions, red block are still uncertain of their fully acceptance, and grey blocks are unconfirmed.

Transactions in the Tangle have different states according to the number of subsequent txs that reference it. Every tx can be referenced directly or indirectly. Direct references mean that between two txs there is an edge, which joins them, whereas an indirect reference implies that there is a path of edges, which goes from one transaction to another, crossing other txs along the way. If we observe the figure 2.5, it can be seen that there are three different sorts of transactions status. A transaction is confirmed (green square) when it is indirectly referenced by all tips, this means that there is a path leading to it from each tip of the Tangle. Red colored squares indicate that they have been approved directly or indirectly[16] by at least two other transactions, and the grey ones have not yet been approved by anyone. As a transaction receive additional approvals, it is accepted by the system with more confidence.

The goal of any transaction in the system is to become green and be confirmed and accepted by the whole network. However, IOTA allows the nodes to establish with which probability of confirmation the txs become confirmed. The method to determine the confirmation level of a tx is based on the execution of Markov Chain Monte Carlo (MCMC) algorithm N times, MCMC is explained in the subsection 2.5. Therefore, the probability of being confirmed is M of N, M being the number of times the MCMC selectes a tip that has a direct path to the tx [6]. It is considered to stress that very high levels of confirmation such as 100% or 99% are hard to achieve because there could always be some malicious tips which are not following the protocol.

[16]Transaction A is indirectly approved by transaction B when there is a directed path of at least length two from A to B.

As more and more transactions are issued, the depth[17] of the transactions in the Tangle increases, as well as their probability of confirmation. Therefore, since nodes confirm their txs when they achieve a particular level of confirmation and this is directly related with the size of the Tangle, it is observed that the faster the Tangle grows, the quicker the transactions will be confirmed. Furthermore, since consensus in IOTA is parallelized, unlike Bitcoin where it is sequential, the network is able to grow and scale with the number of transactions [6]. The consensus is reached at some point when a certain percentage of the nodes in the network has the transaction confirmed.

2.4 Nodes

IOTA considering the different capabilities and requirements of nodes provides software for three types of nodes: full-node, light-node, permanode.

2.4.1 Full Nodes

When a new IOTA node wants to join the network, IOTA documents suggest that it has to be connected with 7-9 neighbors, joining more than 9 neighbors could get the node into trouble due to a high network load. Although the suggestions, IOTA protocol has a condition where the health of the node is verified, checking that the node at least has between 4-9 neighbors [40]. To obtain a successful connection between two nodes, both of them must exchange their IPs and ports, and add the other node address to the iota.ini file [11]. Moreover, both peers have to set up the connection with the same protocol *TCP* or *UDP* (e.g., tcp://ip-address:15600 or udp://ip-address:14600).

However, before the node tries to connect with other peers, it is important to note that it is required a static IP address to use IOTA [11]. If a node is using a dynamic address, it might be changing around every 8h, and it would be necessary to relay it to every neighbor. Since the process of peers connectivity is manually done, broadcasting new addresses would not be adequate for the users, nor for the network performance.

In comparison with Bitcoin, IOTA does not use an automated neighbor discovery method, so users who want to join the network can find partners in places such as; Reddit forums, the nodesharing channel on the official IOTA slack and in the forum HelloIOTA. In the beginnings, IOTA had a neighbor discovery protocol to face the dynamism of the network. Nevertheless, after gathering some performance data, developers determined that the automatic discovery was causing more problems than benefits [19]. Some of these problems were the exponential increase of the bandwidth, the synchronization decrease or the increased vulnerability of the network against attacks.

2.4.2 Lightweight Nodes

Light node is the second node option, and the main difference regarding full nodes is that they are directly connected to a public node[18]. Hence, clients are not interacting with their node, but because they have to set up a connection with a public server, it will be requested when a transaction is issued.

Regular users who want better performance and support the IOTA network set up a full node. However, a significant number of users use the light wallet and become light nodes. As it was referred above, currently running full nodes is a bit more of a hassle,

[17]Depth is the length of the longest reverse-oriented path to some tip.

[18]They are 24/7 full nodes which are hosted by volunteers from the IOTA community, and there are some web pages with a list of available nodes and their current status.

since it is required to manually find 7-9 reliable full nodes and be permanently monitoring the synchronization of the nodes.

On the other hand, if a light node is being used it is necessary to interact with a third party, often referred to as *light wallet provider*, in order to be able to issue and receive transactions. However, all sensitive functions such as hashing and signing happen on the client side. Thereby, it is ensured that even if the light node interacts with a provider, its seed and the private key never leave the wallet[41], so there is no chance that someone can steal token from the wallet.

The process of issuing a transaction from a light wallet is utterly different from what it would be done in a full node. In light node case, the wallet sends an API request (*getTransactionsToApprove*) to the public node to get the two tips that needs to validate – since it is required to attach its new transaction to the Tangle. Then, some proof-of-work needs to be done; the API has a command *attachToTangle* to perform the PoW in the node provider side. However, most of the public nodes have this API command disabled because it is a fairly compute-intensive task. Full nodes are in charge of keeping up the tangle handling their transactions and light nodes transactions, but if they also have to do the PoW of each light node connected to them, there could be severe backlogs in the Tangle.

Finally, since one of the primary purposes of PoW is spam preventing [33]. If public nodes were responsible for running it, light nodes would be able to spam their public nodes with loads of transactions to process. Therefore, when clients use the IOTA wallet app in light node mode, their nodes will do the PoW locally, and when the tx is ready, they will request the public node to broadcast it to the other peers of the network.

2.4.3 Permanodes

Since snapshots[19] exist, full nodes do not have a complete copy of the Tangle. They store the snapshot and all subsequent transactions in order to be able to validate new transactions [12]. Because there may be some applications that need access to the full raw data of the ledger, Permanodes were created to store permanently and securely the whole Tangle history, from the beginning. As regards the other features, they have the same than full nodes.

As it can be seen in the next comparative table 2.3, permanodes fulfill all the possible characteristics of a node, whereas the others meet some of them.

[19]A snapshot is a method to reduce the size of the Tangle database by removing all transactions from the Tangle, leaving only a record of addresses with corresponding balances.

	Full Nodes	Light Nodes	Permanodes
Stores the whole Tangle	✗	✗	✓
Stores the Tangle since the last snapshot	✓	✗	✓
Finds neighbors & communicates with them	✓	✗	✓
Bundling & Signing	✓	✓	✓
Tip selection	✓	✗	✓
Validation	✓	✗	✓
PoW	✓	✓	✓
Attaching the address to the Tangle	✓	✗	✓

Table 2.3: Comparison of the most remarkable characteristics among the three types of nodes

2.5 Tip Selection Strategy

As it was introduced in the subsection 2.1, each incoming transaction needs to approve two previous tips[20]. At this point, it is essential to present the concept of *transaction rate (λ)*. This new property shows the flow load of the Tangle that can vary depending on the node workload, the number of issued transaction, or the network delay. Since some periods are busier than others concerning the number of txs, IOTA Foundation used a Poisson point process to model how the flow of transactions arrives [7]. In order to simplify the Tangle analysis, we only need to know is that on average the Poisson point process is constant and this value is set by (λ). The next figures can help us to understand the concept and compare the tip selection in two tangles with a high and low flow.

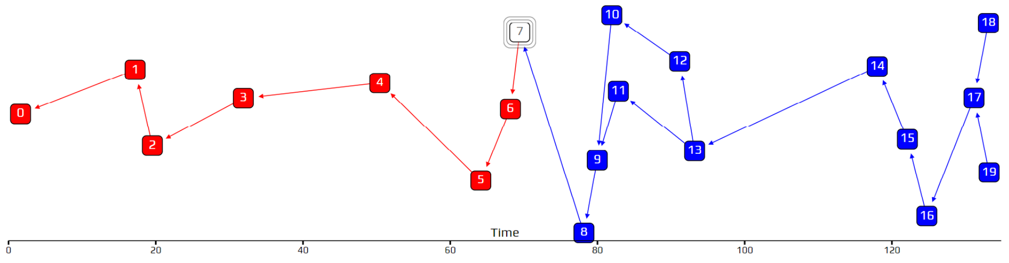


Figure 2.6: Illustrates a low flow tangle simulation with 20 transactions and a λ=0.1.

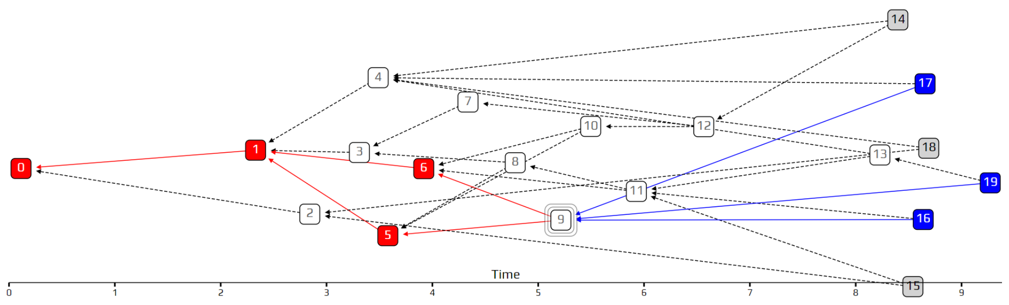


Figure 2.7: Illustrates a high flow tangle simulation with 20 transactions and a λ=2.1.

The figures 2.6 and 2.7 show two different scenarios, in the first case due to the fact that the flow of transactions is so small the typical number of tips become 1. Therefore,

[20]In the Tangle there is always at least one tip.

the tip selection is quite easy because of the fact that the new tx only has one previous tip to approve or two as a maximum. However, in the second case, the transfer rate is higher, and because λ is almost 2, we can see on average that 20 transactions were issued in 10-time units.

The strategy for choosing the next two tips to approve is crucial and is the key of the Tangle. However, before the explanation goes further, two new concepts need to be clarified. Firstly, we have to introduce the delay of a transaction *(h)*, which represents the time a transaction requires to be prepared (PoW) and propagated through the nodes of the network. Secondly, a reference is made to the term *cumulative weight*, which is used to denote how important a transaction is. For the following scenarios, two simplifying assumptions were made, about the delay of the txs we assume that *h*=1, as well as the weight of each transaction which is also equal to 1.

The algorithm of the *unweighted random walk* execution consists of placing a "walker" on the genesis transaction and starting a process of choice. For each transaction, the walker has to decide randomly one of the transactions that reference it to keep walking. In the end, the walker ends its path in the next tip to reference by the new transaction. Since the process of election among the transactions is random, all transactions have the same probability. However, due to this feature, tangle could suffer from "lazy" tips. Lazy tips are those transactions which do not want to update the latest state of the Tangle, and as a result, they confirm their txs based on old data, instead of the most recent one. Considering that this is not the expected behavior for tangle performance, IOTA Foundation developed another version called: *the weighted random walk*. In the next figure 2.8, it can be appreciated the lazy tips problem.

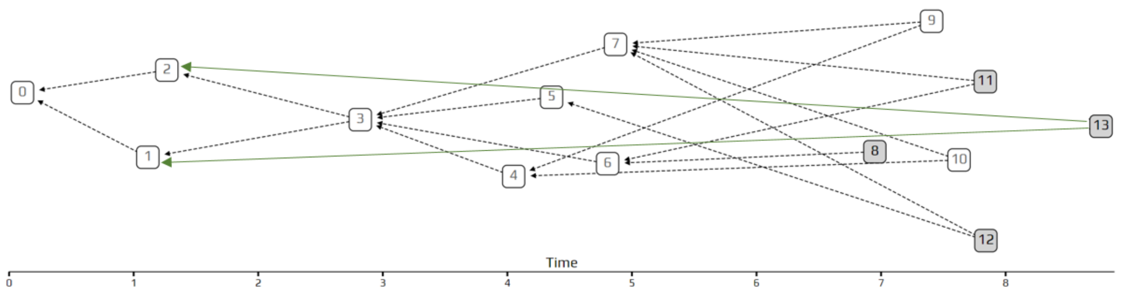


Figure 2.8: The transaction with ID 13 is a lazy tip because it is approving old transactions (1, 2) instead of any of the current tips (9, 11, 8, 10 or 12).

The weighted random walk is a better version of the algorithm that deals with lazy tips. Its approach uses the previously introduced concept of cumulative weight, which is the own weight of a transaction plus the sum of the weight of all txs that approve it, directly or indirectly. With this new attribute, not all txs have the same probability to be chosen, therefore because lazy tips have a meager cumulative weight, their odds of being selected as the next referenced tip plummeted. Nevertheless, the value of the probabilities varies depending on the importance of the cumulative weight (α). If α is a high number, the cumulative weight would be significant, and therefore, the odds would apply to the transaction with highest cumulative weight as the next step on the walk. Consequently, this model leaves a considerable number of txs unapproved along the Tangle. Nonetheless, a low alpha means that the cumulative weight is not relevant and the resulting scenario would be the same as in the unweighted random walk with the lazy tips. The ideal value for α is still a research topic in IOTA.

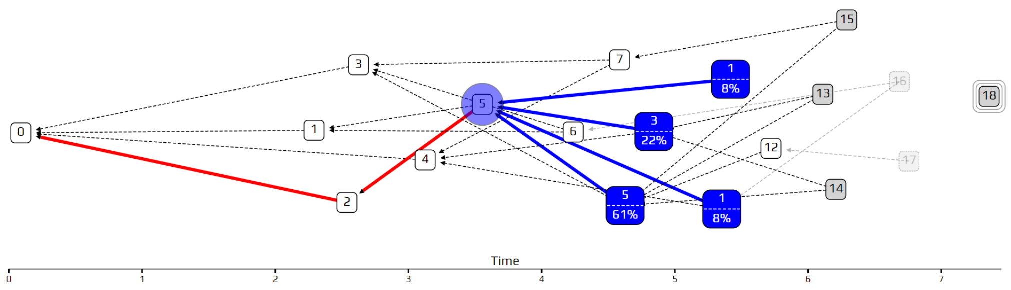


Figure 2.9: It is shown how the MCMC, based on the weight of txs, assigns a probability to the txs that can be chosen as the next tx in the path of the tip selection process.

As it can be observed in figure 2.9, the method that decides which is the probability of the transactions at each step in a random walk is the Markov Chain Monte Carlo (MCMC) algorithm. Besides, it is also used for determining if a tx is confirmed in the consensus process.

2.6 Transactions Validation

In order to add new transactions to the Tangle, two previous transactions need to be validated. The validation process is done after the new transaction (*ntx*) is selected by the tip selection algorithm. To validate the still tip transaction it has to be ensured that it did the PoW. Moreover, it also has to be checked that the selected tips only references previous valid transactions, so that there are no anomalies with the balances of the accounts, as it could be a negative value. To be sure of this, *ntx* has to list and check all the transactions approved directly and indirectly by the selected tip, all the way back to the genesis[21]. Validation is a high time-consuming process and computing resources but indispensable for transactions to be confirmed. Since IOTA network is asynchronous, the nodes of the network do not have to see the same transactions in their copies of the Tangle [2]. This feature gives us a scenario where transactions arrive at the nodes in different order, imagine a simple example with three different accounts A, B, C and two transactions Tx1 and Tx2, where:

· Tx1: A $\rightarrow$ (5) $\rightarrow$ B

· Tx2: B $\rightarrow$ (5) $\rightarrow$ C

However, the order of the transactions arrival in a particular node is first the Tx2 and then Tx1. Because the balance of B before Tx1 was 0, Tx2 would not be validated by other transactions in the Tangle because B account would have a negative value. Nonetheless, the next example describes the consequences of a lousy validation process. Eventually, someone issues another transaction (Tx3) so when the selection tip algorithm is executed, Tx2 is the selected tip to be approved. If Tx3 does not follow IOTA protocol and approve it, even if it is not valid, Tx3 will never be approved by others because it also became invalid when it validated Tx2. Therefore, although now Tx2 would appear attached to the Tangle, there is still uncertain about their full acceptance by a relevant amount of tips, because as Tx2 and Tx3 are not valid, they will never become confirmed.

One of the most well-known reasons to find conflicting transactions in the Tangle is because of a double-spend. This problem appears when there are two approved transactions in the Tangle which transfer tokens from the same account but there are not enough

[21]Actually it not until the genesis, there is a variable *maxDepth* in IOTA software which establishes the limit.

funds to supply both operations. Considering that there is enough to keep one of them, what tx reminds in the Tangle?

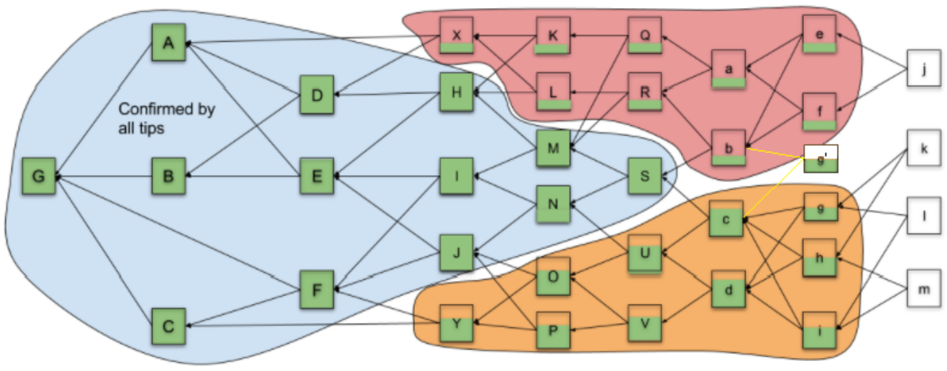


Figure 2.10: Transaction X and Y are conflicting transactions (e.g., double-spend) and the orange and red area show which transactions approve directly or indirectly each of them.

In order to answer the previous question, figure 2.10 is used as a reference for the explanation. If we observer the Tangle, when txs *a*, *b*, *c*, and *d* were tips, *X* and *Y* where confirmed with a ratio of 50%, so both of them had the same probabilities to become confirmed. However, in the next wave of tips *g'* tried to become attached approving *b* and *c* but then the conflict was detected. Therefore, new tips were selected randomly and this time *g* was attached correctly. Because the previous attachment *g* confirmed *Y*, as the Tangle progresses there will be more probabilities that new transactions become attached to the orange area. In the end, because any future transaction cannot approve *j* and *k* due to the fact that they are conflicting, the red part of the tangle + *j* will become orphan. Regarding the transactions in the red area, it will be necessary to reattach them to the Tangle.

3 Bitcoin

3.1 Network Setup

This section deals with the Bitcoin network configuration, how new nodes discover other peers and how the connection between nodes is made. First of all, Bitcoin network is a P2P network which runs over TCP protocol and has a random topology, where each node peers with other random nodes. The network changes over time and is quite dynamic due to nodes are constantly entering and leaving. New nodes can join at any time, and there is not an explicit way to leave the network. Hence, if a node has not been heard in three hours, which is the duration hardcoded in the protocol [42], other nodes start to forget it. A bitcoin client has several ways to discover new IP addresses for the network setup:

1. Nodes broadcast their addresses to other nodes.
2. Nodes query DNS seeds to find new IP addresses.
3. Addresses are provided to the nodes as command line arguments.
4. Nodes read a text file provided on the startup to know new addresses
5. Nodes use addresses hardcoded into the software.
6. Nodes read the database where their store addresses for the startup.
7. Nodes receive the callback addresses from other nodes.

3.1.1 Peer Discovering

When a new bitcoin client starts for the first time, the node does not know the IP address of any other active full node. In order to discover IP addresses, the node query one or more DNS seeds [22] that are stored in the *chainparams.cpp* file of the Bitcoin implementation. Typically, the response of the DNS seed to the request would be one or more resource record with IP addresses of full nodes which may accept new connections. Nevertheless, if the request does not succeed a failure event is triggered. It contains seed nodes, which are several dozens of hardcoded IP addresses that were available at the time of the last software release [43]. These nodes are only contacted if no other discovery mechanism works.

Once the node has been connected to the bitcoin network, it can update its database sending *getaddr*[23] to its neighbors and receiving *addr*[24] messages with new addresses. However, nodes can also receive addr without previously having asked for them. Bitcoin nodes advertise addresses to a certain number of peers when they relay addresses with an *addr*, when they broadcast their address to all their neighbors, and when a connection is made [44]. Meanwhile, nodes with all this exchange of messages are continually updating their database with the most recent addresses for future startups.

Peers often leave the network or change its IP address, and the databases become outdated. Frequently when an old node wants to reconnect to the network, it checks its

[22]A DNS seed is a server that returns the IP addresses of the full nodes to the requests that are received during the peer discovery.

[23]When a node receive a *getaddr*, it selects a maximum of 2500 addresses with a timestamp in the last 3 hours.

[24]Each *addr* contains 10 addresses or less.

database trying to find reliable known addresses, but sometimes this process takes time due to their availability. Hence, after a reasonable time, if there are no results in the search, the program starts looking up for new addresses using DNS seeds [43]. In case of not having results using this method, the seed nodes are used.

Bitcoin Core[25] also offers a manual alternative with several command-line connection options, including the possibility of getting a list of peers from the IP address of a specific node. As soon as the node knows new IP addresses, it specifies with what addresses it wants to establish a new connection, using the *–connect* command. This command automatically executes a function called *OpenNetworkConnection()*, which processes the connection establishment [45].

3.1.2 Connecting to Peers

Once the peer discovering process has been accomplished successfully and the node has reliable addresses, the connection to a peer is made by sending a *version*[26] message to the address of the node with which it wants to establish a connection [18]. *Version* messages contain some parameters of the sender such as the version number, block and the current time. Afterward, when the remote node receives the *version* message, it replies with the same message with its information. At that moment, both nodes are already connected, and the sender can send to the remote node *getaddr* and *addr* messages to get more addresses of other peers.

In order to check the status of the connections, nodes send a message after some time of inactivity. If the node does not receive any reply message after some time, the connection between them will be closed.

3.2 General Overview Bitcoin Blockchain

The blockchain is a decentralized, distributed, immutable public ledger in a heterogeneous P2P network. In blockchain, multiple transactions are stored in blocks, and every block is sequentially connected to each other "chained". Every block has a variable amount of txs although the maximum capacity of the block is 10MB. A newly issued tx is broadcasted from its source node to all the other peers, therefore when a node hears about a new tx it has to validate it. Since blockchain also uses the Unspent Transaction Output (UTXO) scheme, the validation of transactions is similar to IOTA's process; it consists of using the copy of the blockchain that every node has, and executing some cryptographic functions implemented in the protocol. These functions validate that the transactions are formatted correctly and the balances are available to be spent. If the node proves that the transaction is valid, it is added to the Mempool[27] of the node, and it is relayed to all its neighbors that do the whole shebang again. Otherwise, if it is invalid, the transaction is discarded, and the node does not broadcast anything.

Theoretically, any full node can add valid tx to a new block and mine it. Mining is the process of getting tx from the Mempool, adding them into a new block and doing a PoW. Bitcoin uses the hashcash proof-of-work function, which its primary purpose is to avoid tx spam, in addition to saving the history of transactions in a way which is computationally impractical to be modified by another entity. Moreover, the proof-of-work has been intentionally designed to be resource-intensive and challenging so that the time between

[25]It is the name of the software used for the implementation of bitcoin.

[26]The version message provides information about the sender to the receiver, and until this message is not sent, any other message will be accepted.

[27]The Mempool is a list of txs that have been validated and are pending to be added in a block.

new blocks remains steady, around 10 min. Nevertheless, technology has been improved since 2008, and every year new computers are more powerful. Consequently, Bitcoin has to increase the difficulty of the PoW to keep the interval of time between them. The difficulty is encoded as a target, which is essentially a 256-bit number determined via consensus by all nodes of the network [4]. Since block hashes are produced by SHA-256, they are also a string of 256 bits, so if the hash of a block is numerically smaller than the target, the block candidate is a valid block. Unlike IOTA, nodes do not have to do PoW to send a tx. Thus, since PoW is a resource-intensive activity and it is not mandatory for nodes, there has to be a reward for those who do it. Currently, the participants in the bitcoin network are divided into two roles: the full nodes and the miners. The only difference between them is that while the nodes are limited to validate and broadcast the transactions, the miners also add txs into a block and perform the PoW. In exchange for their computational work, miners receive a reward plus the fees of the transactions added in the block. The current reward is 12.5 BTC, but because the protocol is halving them periodically, the amount is diminishing until 2140 where it will reach the value 0 [46]. At that moment, it is still uncertain what could happen, but as David Sønstebø - founder of IOTA said [47], *"Fees will grow in accordance with a decrease of mining rewards"*, which is not very convenient for IoT networking environment.

Once a block is mined, the miner has to broadcast it all over the network to be validated by the all the other nodes. Hence, whenever a node receives the last block, it has to validate it by checking the Bitcoin consensus rules[28]. Here there are several examples of consensus rules, although there are many more:

· Blocks may only create a certain number of bitcoins.

· Transactions must have correct signatures for the bitcoins being spent.

· Transactions/blocks must be in the correct data format.

· Within a single block chain, a transaction output cannot be double-spent.

At some point, every node will have validated the last block published in the network. When this happens, it is said that the block has been mined at a depth of 1. The validation of subsequent blocks will increase the depth of the block. As in IOTA's case, to avoid conflicting transactions, such as double spending, txs are not confirmed until a certain number of blocks depth. Nevertheless, when the block is deep enough, it becomes confirmed keeping the consensus of the system.

The miners are all time mining, trying to be the first to complete the mining of a block and obtain the corresponding reward. However, due to the difficulty of mining has increased to the point where slow miners could take centuries to generate a block, the mining pools appeared. Mining pools is an approach for pooling the resources of miners and have more computational power to create the block. The reward in a mining pool is split among all the participant according to the computational work they contributed to finding the block.

When a miner receives one block, it checks if it is valid and the acceptance is shown when blocks start working on generating the next block. A reference between two blocks, as it can be seen in figure 3.1, is established by including the hash of the previous block in the current block [13]. Therefore, when the current block is finally mined, it becomes

[28]The consensus rules are a set of rules that every full node has to consider for the validation of a block and its transactions.

sealed to the previous one. Any change in the previous block would alter its hash, and likewise the subsequent block would notice that it is different from what it has, because it has been modified.

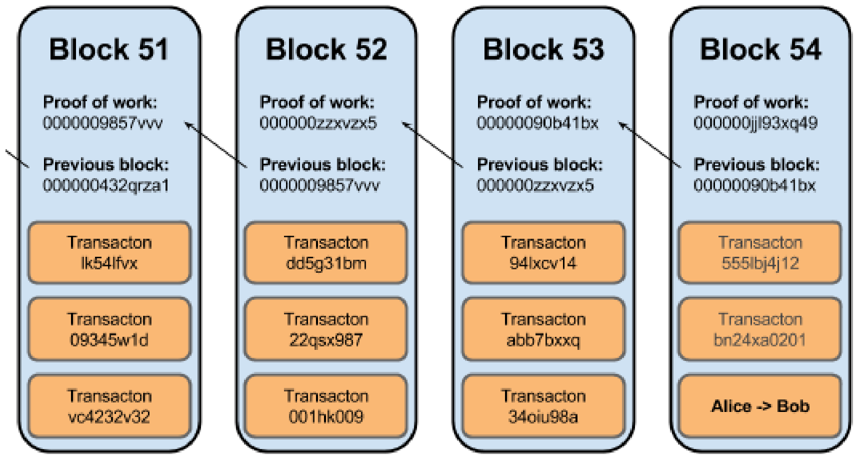


Figure 3.1: Every block is chained to the previous one by including a reference to the proof-of-work resulting hash.

Due to the latency of the network, the blockchain can be split into two or more chains, it is known as *blockchain fork*. A fork occurs when more than one miner, at the same time, find different versions of the next block. Since each of them broadcasts its version, some nodes receive one before the other. During a blockchain fork nodes in the network do not agree on which block is the blockchain head. Hence the system is no longer consistent [4]. A blockchain fork may be prolonged by adding new blocks on their respective blockchain heads. In the end, one branch will be longer than the others, and then the nodes that were not following that branch will switch to the longest one. Every node when receives different block versions saves all of them until the fork is solved, in case it has to change its branch. At the moment that one chain is longer than the others, the fork is resolved, and the ledger replicas are consistent about the blockchain head. The blocks discarded after the fork resolution are called *orphan blocks*. The transactions included in the orphan blocks are returned to the mempools unless they have already been included in a block of the new chain.

4 Method

The method implemented in this thesis deals with the still present problem in the last IOTA Reference Implementation release, the neighbor discovery algorithm for the automatic connectivity of nodes in the network. IOTA Foundation presents its IOTA network as *"an independent peer-to-peer network with a first-user, friend-to-friend, network structure"* [37]. However, the idea behind IOTA Tangle was built thinking of IoT scenarios, where hundreds of thousands of nodes are interconnected, and the network is highly dynamic. Therefore, due to the qualities that IoT scenarios include, the current limitations of network connectivity, and besides the complaints that the IOTA community and users show to find reliable nodes to be connected. We decided to work on the next problem-solving activity to study a possible neighbor discovery implementation for the IOTA network, and help the IOTA Foundation in the adaptability process of the IOTA network with the purpose of IOTA Tangle.

4.1 Neighbor Discovery Algorithm

The method used for this research project is a controlled experiment using a peer-to-peer network simulator, called Peersim. It has allowed us to obtain more realistic results, with the purpose of performing an analysis that is as similar as possible to the real IOTA network.

Implementing an efficient algorithm is essential to obtain better results. Because this algorithm has been designed to improve IOTA network connectivity, a wrong solution would imply an inappropriate performance to the IOTA application layer. Therefore, we designed four versions to compare them and corroborate which approach has better results and is more suitable for the network performance. The most relevant implementations are the Weighted, the Greedy and the e-Greedy neighbor discovery, since they base the selection process on the weight of each node, whereas the other version consists of a completely random solution where there is not any rule for the selecting neighbor process.

An IoT network is made up of different types of devices (nodes), and each of them offers different qualities [29] to the network. The goal of every node is to reach the best performance, which regarding the IOTA application would mean the best capacity for processing transactions and distributing them over the network. In order to achieve it, nodes need to be connected to other peers which also have high performance to be continually receiving new transactions, otherwise, the node will have nothing to approve and broadcast. Furthermore, IOTA is in charge of keeping the nodes active in a way that if a node becomes "too lazy" and it does not propagate transactions, it will be dropped by its neighbors [2]. Thus, when it comes to selecting a neighbor, it is essential to enhance the quality of the node to choose, as the three main neighbor discovery versions do. It is also important to take into account the interest in having long-term neighbors that constantly provide new transactions to keep the local Tangle updated.

Nevertheless, although it is vital to be connected with high-quality nodes, the application level restricts the amount of neighbor that a node should have. As specified by the IOTA documentation, they recommend to the users to have their node connected to 7-9 neighbors to keep the node synchronized. Because we are designing this protocol for the IOTA application, we took into consideration this requirement in the Weighted version, and we have implemented the algorithm limiting the output number of connections of the

[29]The quality of a node could be measured by attributes such as its computational capacity or battery level.

nodes with the amount specified by IOTA. However, in the Greedy and e-Greedy versions, this limitation was dismissed to focus on obtaining a connected network with the highest durability[30].

4.1.1 Protocols of the Neighbor Discovery Algorithm

The execution of the neighbor discovery protocol includes several other steps to get the IOTA network with all its nodes indirectly interconnected. The process for the neighbor discovery is based on three main steps, which in the Peersim language are divided into two sub-protocols, as it can be seen in image 5.1. Each of this steps has a primary task in the process. As far as the simulation is concerned, the first step is to build a network with a certain number of nodes. Every peer has the same protocols as the rest, they and their protocols are entirely identical, that is why their instances are created by cloning. Once all nodes are created, all protocols used in the simulation are also declared, and they are ready to be initialized. In our simulation case, the only protocol initialized is the one which performs the wiring of the static overlay network. In the beginning, the simulation starts with a set of nodes which are randomly connected with *k* number of nodes. The first step of the algorithm itself consists of executing the linkable sub-protocol: **the Newscast**. After its execution, every peer in the system knows about a set of nodes (neighbors) and has a list of their addresses and the last connection time-stamp. In order to distribute the content and build a new topology, every node selects randomly one of its neighbors to exchange their neighbor lists. Then, after exchanging the list, nodes merge both lists and build a new one with all the latest connections. Thus, at each cycle, the nodes expand the vision of their overlay network, until reaching the maximum number of connections per node, and at the same time update the network with the most recent connections.

After expanding the overlay network for every node, the second sub-protocol is **the Neighbor Selection**. It is executed for every node to choose a subset of relevant neighbors. The selection sub-protocol is the core of the neighbor discovery protocol since is at this point that the process is divided into the four versions:

- Random Neighbor Discovery Algorithm
- Weighted Neighbor Discovery Algorithm
- Greedy Neighbor Discovery Algorithm
- e-Greedy Neighbor Discovery Algorithm

The random version of the protocol selects a certain number of neighbors randomly. However, in the other versions, because every node was initialized with a quality parameter, the selection process is based on this quality, selecting the number of neighbors in a descending order of priority.

If at this point the network is drawn, it would be observed that the requisites established by the IOTA application are not fulfilled in the network. The selecting protocol does not guarantee the requirements of an indirect graph nor a balance between the in-degree and out-degree of the nodes. However, with the purpose of solving this problem, there was a filter included in the same sub-protocol that was designed to achieve the previous requisites. The difference between quality-based versions is in the strategy they

[30]The durability of the network in this study is understood as the number of simulation cycles that the network remains a connected graph and therefore is functional for IOTA.

follow to perform the filtering. The weighted version was designed thinking about satisfying all the features of IOTA network. Hence, after the weighted filtering, the network was undirected, and the maximum number of neighbors was limited, as in the real network, to a maximum of 9 neighbors. Nevertheless, this version had a drawback about the minimum number of neighbors; there was no control in the case that a node did not have any node available in the list of selected to set a connection. This situation is more likely to happen when the filtering is executed in the last nodes of the network because a considerable amount of the network already have the connections established and the variety of available nodes is lower. If a node can not find any neighbor that accepts connections, it will not be part of the network.

The fact of not limiting the maximum number of connections in the weighted version could cause that the network becomes a complete graph. Thus, in order to cope with the weighted version issue and at the same time keep a limit on the maxim amount of connections, Greedy and e-Greedy versions were created. Their main goal and difference regarding the previous version is to include all nodes in the setup of the network. However, to be able to do this, the maximum number of connections that a node can reach is more flexible than the previous limitations of IOTA. Theoretically, since all nodes select their neighbors based on their quality, those with the highest quality will be more requested than the others. In the real world, this hypothetical case would make sense for big companies that have sufficient resources to manage computers with large computational capacity. These companies could offer a service that in exchange for accepting a huge amount of transactions they would be rewarded somehow.

Therefore, in these two new versions, preference is given to those nodes with higher quality, whereas those with less quality are forced to establish a minimum of connections to eradicate the loss of nodes during the configuration of the network. In the previous case, where in-degree could be equal to zero, not only IOTA would not accept that network for its application, but also there would be synchronization problems since they would not receive any income transaction to add to their Tangle.

The difference between Greedy and e-Greedy was the attempt to decrease the requests to the most quality nodes and diversify the connections among the selected nodes that have a lower quality than the top 9. Greedy is the version that potentially benefits the quality nodes, while e-Greedy seeks the balance between the selected nodes.

The selection algorithms choose a subset of neighbors from its overlay network. The number of selected neighbors is defined at the beginning of the simulation, and depending on its value, the resulting network could be different because the amount of quality nodes increases correlatively to the number of the selected. A high quantity of selected neighbors makes the nodes have a wide variety of possibilities to choose those neighbors that have more quality. On the other hand, if the selection is rigorous, in versions such as the Random and the e-Greedy where the design is beneficial for those with less quality, their probabilities would be zero because the rigidity in the selection process would not allow them to participate as candidates for neighbors.

Designing a network discovery algorithm for the IOTA network was one of our objectives for this thesis, however finding a single solution for all cases is a very complicated task. In order to get some conclusions about the performance of the four different versions, once the neighbor discovering and the IOTA network configuration were completed, it was needed to check which one had designed a better IOTA network. Each IOTA network was the results of an experiment in one of the neighbor discovery versions. For the experiments on the simulation we fixed some values for every execution of the protocols, those which were fixed are: the number of simulations (50), the number of cycles

(40), the number of neighbors for the filter (9) and the number of initial connections per node (2). Moreover, the solutions were tested by modifying the value of the network size, the maximum number of neighbors per node and the number of nodes selected.

The number of simulations for each experiment had to be a considerable value to obtain a wide variety of samples and make the average of all of them. The number of cycles was the duration of every simulation, and allowed the overlay network of the nodes to become larger and to be updated throughout the cycles. The number of neighbor connections was set to 9 to satisfy the requirement of the IOTA application. However, this value for the Random and Weighted versions represent the highest degree of a node, whereas for the Greedy and e-Greedy is the minimum amount of connection required for a node. Finally, the last parameter represents the number of initial connections for each node in the network, which was not relevant since during the simulation every node discover new neighbors.

4.2 Analytic Program

In order to prove the quality of each network, an analytic program was built to test which networks offered better results. The approach used for the analysis was based on the durability of the network, so the quality of the nodes was used as a property that decreases proportionally to the number of transactions validated and broadcasted over the network. However, because all nodes receive all transactions, the number of validated transactions is the same for all nodes, so its value was not significant for the equation. Hence, the resulting simplified equation for the quality of the overall network was determined by the following equation:

$$O_v = \sum_{i=0}^{m}(q_i - \cancel{\sum_{j=0}^{n}}(j * \sum_{k=0}^{p}(\sigma * k))) \qquad (1)$$

In the equation, *m* represents the number of nodes in the network, *q* is their quality, σ is the cost of each edge, and *p* refers to the number of neighbors of a given node.

If we observe the state of the network in a temporary line, we see that its value is reduced as the quality of the nodes is diminished by the cost of the edges. During an execution, nodes are disappearing when they do not have quality and until the moment where the network is no longer useful because it has become incomplete. At this point, the execution is over, and the program reveals the durability of a particular scenario executed in a specific neighbor discovery algorithm. Therefore, the most important values to determine which algorithm has the best performance in the analysis are; the O_v from the beginning until the end of the execution and the number of rounds that the network was able to run the IOTA application.

4.3 Reliability and Validity

As far as the reliability is concerned, the methodology that has been used in the thesis for data collection was performed in the same way for each scenario. For each experiment, a random seed was used to modify the behavior of the simulation, so the results were different for each of them. However, since Peersim is a pseudorandom program as long as the executions are made with the same seed and same parameters, the results will be identical in all the simulations. Furthermore, the analytic program was also the same for all versions, and because the results did not depend on the execution time, the efficiency of the algorithm did not affect the results. Thence, because all methods used for data collecting has been automated there are no chances to introduce an error only in some executions, the methodology correctness is atomic.

On the other hand, the importance of including valid conclusions and results in the thesis was taken into consideration along all the steps and sections of the report. In order to construct validity in the interpretation of the results, the analysis and conclusion sections were limited to build their affirmations and explanations from the results that had been previously achieved. For instance, the performance comparison among the algorithms was based on the final results and not on the expected results before the analysis.

The validity of the results is a crucial factor to obtain adequate results from the real algorithm behavior, and thereby be able to perform a successful analysis and conclusions. In this thesis, because each execution has a random factor in its behavior, the results of a simulation could not correspond to the actual behavior of the algorithm. Thus, to grant validity to the results obtained during the experimentation phase, 50 samples of results were obtained for each scenario, and the average of the result was calculated to observe the performance of the algorithm in a general way.

Finally, about the external validity, it should be considered that the proposed solutions were designed following the requirements, specified in the method section 4. For this reason, the validity of the network performance with the real IOTA application would be compromised by other external factors not mentioned in this project or not considered during the implementation.

5 Implementation

For this thesis, as it has been introduced in the previous section, the core of the implementation was a java program to simulate the neighbor discovery protocol in a P2P network in Peersim. Furthermore, an analytic program was also designed to test the fours versions of P2P networks previously built in the simulator.

5.1 Tools

1. **Eclipse:** It is the software used as a development environment to program the simulation and the analytic program, both of them were programmed in Java language. The integration of Peersim to Eclipse was done by downloading the PeerSim package from Sourceforge and adding the three main libraries to the eclipse project.

2. **Peersim:** It is an open source P2P systems simulator that was used to build the neighbor discovering algorithm. Peersim consists of a set of libraries with useful classes and protocols that helped us to build the networks. PeerSim is a simulation engine in which we can write simulations, collect results and analyze them. The engine takes care of the experiment, while the programmer takes care of the logic of the interactions among the elements of the scenario. The simulation is specified in a configuration file that is crucial because it contains the characteristics of the network; declaration of all protocols used in the simulation, protocols initialization and their parameters, and even sometimes observers that get results from the simulation.

3. **External Libraries:** The analytic program uses a java graph library called JGraphT that provides mathematical graph-theory objects and algorithms. Furthermore, it supports a great variety of graphs including the undirected, such as the IOTA network. In our program, we used this library to check until when the graph remains connected, with the function *isGraphConnected()*.

4. **Gephi:** It is an open-source network analysis and visualization software package written in java. It provides lots of options for adequate visualization of the network, distributions and network statistics (e.g., average degree, network diameter, in-degree and out-degree for each node). This software has been used in this thesis to get a precise visualization of the IOTA network with all nodes labeled, a list with all the neighbors of each node, and the in-degree and out-degree of every node.

5.2 PeerSim Implementation

The goal of the PeerSim implementation was to build an appropriate distributed P2P network to run the IOTA application on top of it. Peersim was used in this thesis due to the significant advantages offered by its libraries for the design of the network. The implementation of our solution includes two of its classes which helped to build the initial phase of the network, they are the *WireKOut* which takes the Linkable protocol and adds random connections to the nodes, and the *Newscast* which is the linkable protocol, and it was used as a topology manager. The Linkable protocol class, in this case Newscast, represents the information about the link layer stored by a node. Hence, whenever it is necessary to consult the degree of a node (*int degree()*), obtain a certain neighbor (*Node getNeighbor(int i)*) or add a new one (*boolean addNeighbor(Node node)*, it is necessary to instantiate the class calling the corresponding function.

In order to achieve the main objective of this implementation, several protocols where used through the whole process. The subsequent figure 5.1 shows the layered architecture that each node follows and the corresponding protocol in each of them.

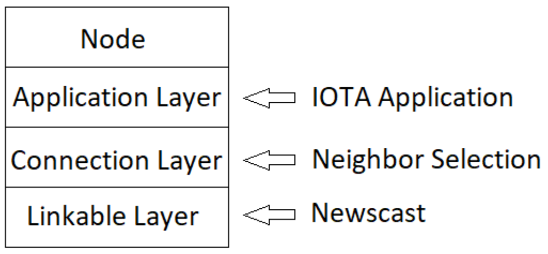


Figure 5.1: Layered protocol architecture of the nodes.

The bottom layer is the linkable layer which is responsible for the network connections, it is probably the most used protocol since any query or modification to other peers implies instantiating it to execute the query. On top of the linkable layer, there is the connection layer which is focused on the logic of the connection to guarantee the requirements demanded by the last layer of the architecture. The top layer is the application that runs on top of each node, in this case, the IOTA application contains the tangle and the whole system involved in the exchange of transactions.

The design of this implementation was thought to offer a high level of adaptability. Since a layered architecture was used, it allowed us to choose between the Random, Weighted, Greedy or e-Greedy neighbor selection, indistinctly, without affecting the other layers or requiring significant changes in their implementations.

5.2.1 Configuration File

The configuration file is a plain ASCII text file composed of key-value pairs. The first thing to note are the key names, some of them refer to global properties, while some others refer to single component instances. For instance, *simulation.cycles* is global, but *protocol.lnk.xxx* defines parameter *xxx* of protocol *lnk*.

Each component in the config file has a name, such as *lnk*. In our implementation, there are three types of components: protocols, initializers, and observers. In the example, *lnk* denote the name of the linkable protocol, whereas *xxx* define a parameter and assign a value. This instruction *protocol.lnk.cache 50* has been extracted from the source code, and we can observe how the value 50 is assigned to the cache parameter of the linkable protocol.

The structure of the configure file used in our implementation is formed by:

– simulation.cycles X

– network.size 100

– network.node example.gerard.IOTANode

– protocol.lnk example.newscast.SimpleNewscast

– protocol.exn example.gerard.NeighborsSelection (Random or Weighted)

– init.rnd WireKOut

– control.nl example.gerard.NodesQualityList

– control.tp example.gerard.TopologyIOTAObserver

The order of execution is from top to bottom, so the first instructions we find are global properties about the cycle-driven simulations and the network. The third instruction in the list indicates that the network is using a customized class (*IOTANode*) for the nodes of the network since it was necessary for our implementation. Afterward, there is an initializer of the components previously declared, in this case, is the linkable protocol with the initial state of the network. It is important to clarify that initializers are executed only at the beginning of the execution.

Finally, after finishing the simulation, the edges were printed in the .csv files for the network topology visualization in the Gephi software. Furthermore, the quality of the nodes was also saved to be later imported into the analytic program. These tasks were done in the simulation by the observers *control.tp example.gerard.TopologyIOTAObserver* and *control.nl example.gerard.NodesQualityList*, respectively.

5.2.2 Newscast

Newscast is the linkable protocol used to deal with the connections of every node of the IOTA network. Every node from the beginning of the simulation has a view of a set of nodes which build an overlay network. The first set of nodes is randomly chosen by the *WireKOut* initilizer after the linkable protocol is declared. Afterwards, when the Newscast protocol is executed in the first cycle, since Newscast is an epidemic content distribution, every node starts increasing its set of nodes and likewise its overlay network.

Regarding the epidemic distribution, Newscast was implemented in a way where every node selects a random neighbor and executes the *void merge(Node thisNode, SimpleNewscast peer, Node peerNode)* function. Nodes have a list of descriptors that contains the address of each neighbor, and a time-stamp with the time of the connection. When the previous function is called, *thisNode* and *peerNode* merge their list of descriptors. If both nodes have the same neighbor, the node with the highest time-stamp (the newest) is saved in the list, whereas the other is dropped. As a result, both nodes have expanded their overlay network and increase their odds to find better neighbors for the selection process.

The maximum number of nodes (k) that are exchanged can be modified in the configuration file, depending on the size of the global network. This number represents the maximum amount of neighbors at this stage of the network. However, if k has a high value and the network is in the initial phase, the number of merged neighbors would be lower than k.

5.2.3 Neighbor Selection

Neighbor Selection is the connection protocol and the core of the whole Neighbor Selection protocol for IOTA. Thereon, every node has run the Newscast protocol and expanded their list of neighbors. The protocol declaration in the configuration file includes two parameters: *maxneighbors* and *maxiota*. Both of them set the limit of the selected neighbors, but each of them is used for different use cases. This protocol is divided into two steps; first, every version selects, in different ways, several neighbors which are added to the selected neighbors list (*selectedNeighbors*). Then, after the selection process, the random version will have a different list than the others versions. However, the difference between the other three version is in the way how the *selectedNeighbors* are filtered into the final set of neighbors, which will be saved in the *IOTAneighbors* list. In order to prove if the suggested strategies had a good performance, and get some results to compare them, it was necessary to implement four classes, one for each version:

- **Random Neighbor Selection:** The first class was not based on any specific strategy for the selection process. It is performed by each node and consists in running a loop until the *maxneighbors* iteration is reached, adding at each iteration a random neighbor in the *selectedNeighbors* list, which in the end contains all selected neighbors. After the selection process, because the connections were set up randomly, the resulting network would not be undirected, which is mandatory for the application. Hence, it was necessary to filter the results and save them in *IOTAneighbors*, which contained the final IOTA network connections. The filter is a loop through the *selectedNeighbors* while the *IOTAneighbors* size is lower than the *maxiota*, which was initialized in the config file with a value of 9. Then, at each iteration the node query one of its selected neighbors to check two conditions: if the node is on its neighbor *IOTAneighbors* list and if the list has not reached the maximum capacity (*maxiota*). If the node is in the list, means that both were already connected, otherwise if it is not and the size of *IOTAneighbors* is less than *maxiota*, the node has to add its selected neighbor to the *IOTAneighbors* and call the function *void setIOTAneighbors(int IOTAnodeID)* to advise the neighbor to do the same on its list. Since both peers are added to each other, the algorithm ensures that the established connections are bidirectional.

- **Weighted Neighbor Selection:** On the other hand, there is the weighted version and one of the reasons why the *IOTANode* class was needed for the implementation. All nodes have a quality parameter assigned when they are created, and its value is a random number between 1-10. This attribute is the key to the selection process because it is based on the quality of the node. A node which runs this selection, as in the previous case, runs a loop through the *selectedNeighbors* checking all its selected neighbors and sorting them according to their quality. Then the filter is applied, the procedure is the same as before with the difference that now they were sorted by quality before being added to the *IOTAneighbors*.

- **Greedy Neighbor Selection:** The fourth version is based on the selection approach of the Weighted version but with a different filter to improve it and get better results. Since this version is more flexible, the difference with the previous case, is the omission of the *IOTAneighbors* list size verification. Thus, the problem on the minimum amount of connections per node was solved because without the previous condition every node establishes its connections with all *IOTAneighbors*. Nonetheless, the number of connections of the highest quality nodes increased considerably.

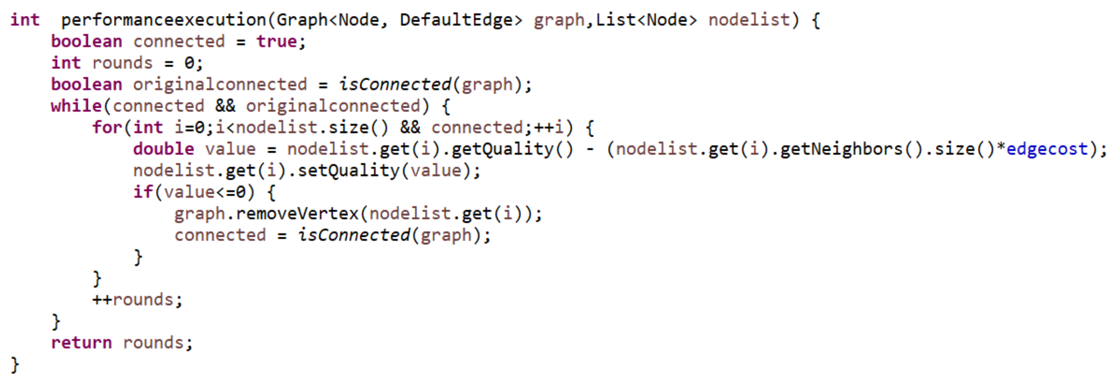
- **e-Greedy Neighbor Selection:** Finally, the last version goal is to spread the connections among the network instead of just focusing on the most quality nodes, while at the same time it keeps including all nodes in the network. In order to achieve that behavior, the previous Greedy filter was modified introducing a probabilistic factor. The idea behind this approach was that with a 70% of probabilities nodes would select one of the first *maxiota* nodes in the *selectedNeighbors*. However, the other 30% of the time, the selection would be made to any of the nodes after the top 9. Hence, with a 0.7 factor the algorithm is ensuring the preferential selection to the nodes with more quality, and with the other 0.3 it gives an opportunity to those less important.

5.3 Analytic Program

The last part of the implementation has been designed to analyze the resulting IOTA networks of each of the previous algorithm versions. In order to measure the quality of the overall network, we defined the O_v of the network as a metric, and it was calculated with the previous equation 1. Furthermore, the number of cycles in which the network was active, was also calculated as a metric to measure the durability of the network. Therefore, as it can be seen in section 7, the analysis of the results has been done comparing the O_v and the network durability among all versions.

Since the analytic program is independent of the simulation program, all data previously calculated was needed to perform the analysis of the networks. All the essential data of the simulation was exported by the observers in two .csv files for each version. The *IOTAgraphTopologyVersion* file contained the source and target of the edges to generate the graph. On the other hand, *NodesQualityListVersion* file included a list of all nodes in the IOTA network with their corresponding quality. The observers in the config file were declared to export this data at the end of each simulation. However, every simulation was run 50 times to get the average of the results. Thus, every time that the analytic problem was executed it had to import 50*2 .cvs files and calculate the two metric average of the 50 samples.

The analysis of the networks was performed in a static way where at each iteration the quality of the nodes decreased proportionally to the number of edges multiplied by the cost of each of them.



```java
int  performanceexecution(Graph<Node, DefaultEdge> graph,List<Node> nodelist) {
    boolean connected = true;
    int rounds = 0;
    boolean originalconnected = isConnected(graph);
    while(connected && originalconnected) {
        for(int i=0;i<nodelist.size() && connected;++i) {
            double value = nodelist.get(i).getQuality() - (nodelist.get(i).getNeighbors().size()*edgecost);
            nodelist.get(i).setQuality(value);
            if(value<=0) {
                graph.removeVertex(nodelist.get(i));
                connected = isConnected(graph);
            }
        }
        ++rounds;
    }
    return rounds;
}
```

Figure 5.2: Code of the analytic procedure to calculate O_v and IOTA network durability.

As it can be observed in figure 5.2, the program has been implemented using an external java library called JGraphT to check whether the graph was connected or not, using the function *boolean isConnected(Graph<V,E> graph)*. At the time that the graph became incomplete, it returned the number of cycles in which it was complete.

6 Results

This section shows the results of the experiments performed on the different versions of the neighbor discovery algorithms. In order to get enough information for the analysis, three scenarios were designed to force the behavior of the algorithms and observe their performance concerning the durability. Each scenario reveals essential information for the analysis, the cycles of durability can be seen in figures 6.1, 6.3 and 6.5, whereas figures 6.2 and 6.4 indicate the O_v of the network before the execution.

The first experiment was thought to observe the changes of the behavior of the algorithms if the size of the network (1.000 nodes) remained static, while the size of selected nodes increased in each simulation. For this scenario, the maximum size of the overlay network that could be reached was fixed to 70 peers; therefore, the maximum amount of selected neighbors is 70.

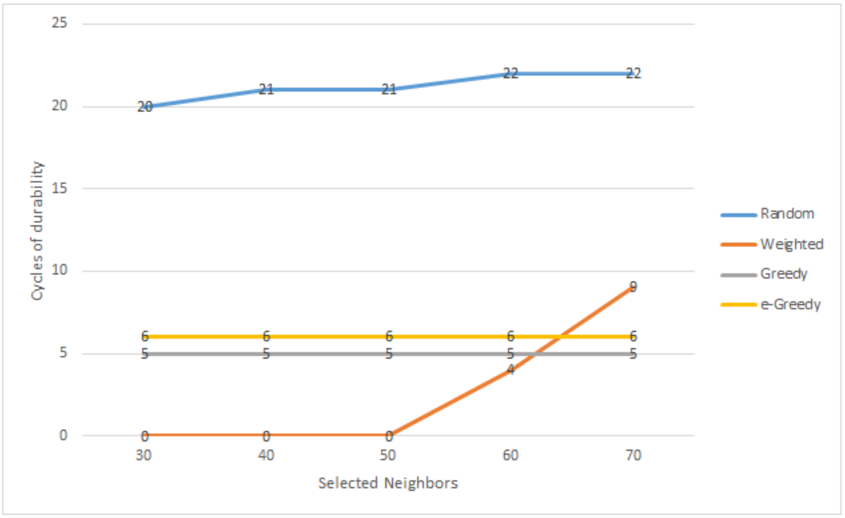


Figure 6.1: Durability cycles of each algorithm according to the number of selected neighbors in a network of 1.000 nodes.

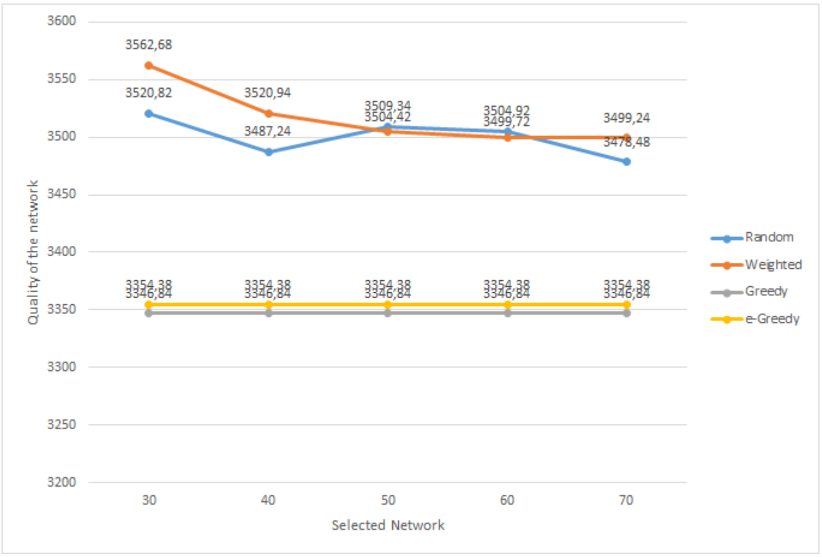


Figure 6.2: O_v of the networks designed by each algorithm and based on the parameters of the first scenario.

The second scenario was designed to see how the algorithms would face the growth of the network and verify its durability through several simulations, each of them includes 500 additional nodes to the network. This scenario fixed the overlay network size to 50 nodes while the number of selected neighbors was limited to 20 nodes.

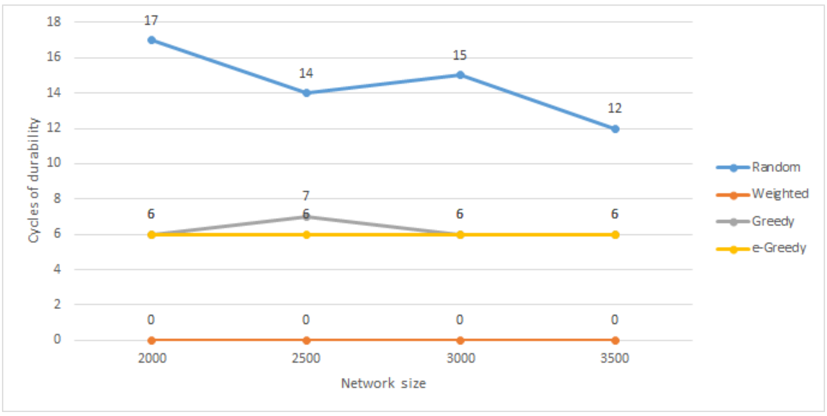


Figure 6.3: Durability cycles of each algorithm according to the size of the network and a maximum of 20 selected neighbors.

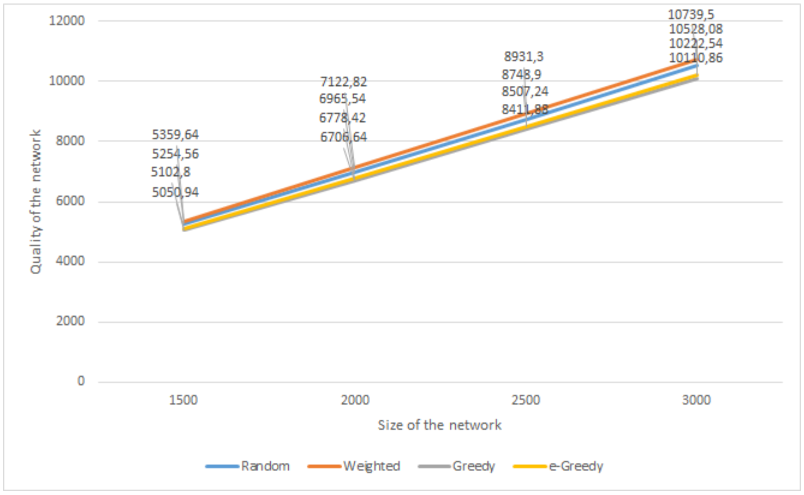


Figure 6.4: O_v of the networks designed by each algorithm and based on the parameters of the second scenario.

Finally, the last scenario was designed mainly to compare the three proposed algorithms for the neighbor discovery protocol. In this case, the values which are changing through the simulations are the size of the network as well as the number of selected neighbors. The reasons for setting the values of the parameters can be seen in detail in the next section 7. Furthermore, as an exception, the *maxiota* value, seen in section 5.2.3, had to be changed to be able to simulate a scenario with these characteristics. Regarding the size of the network, it was fixed to 1.500 nodes.

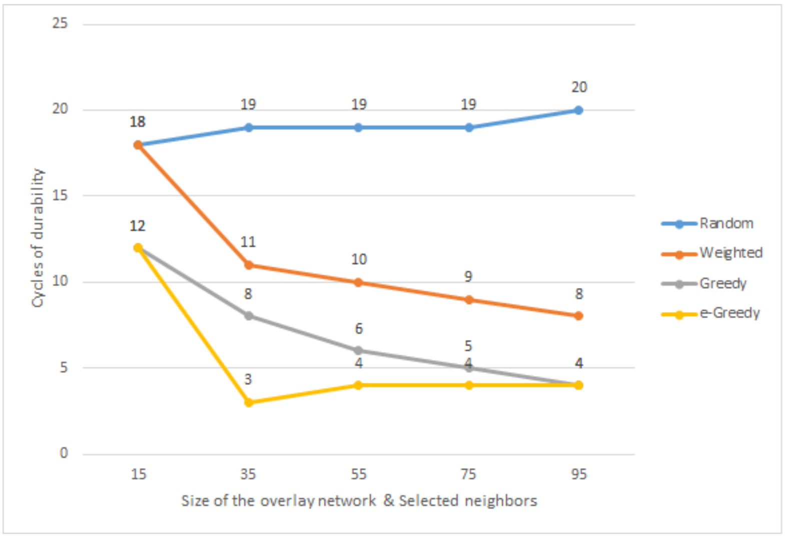


Figure 6.5: Durability cycles of each algorithm according to the size of the overlay network and the number of selected neighbors in a network of 1.500 nodes.

7 Analysis

This section deals with the analysis of the results to verify if the algorithms provide a solution to the problem formulated, and understand the reasons for the performance of the algorithms in the three different scenarios. The following argumentations are mainly based on the results data, but also include information about the implementation of the algorithms. Thence, before starting this section, it is highly recommendable to understand how they work. The analysis is done in the same order in which the scenarios have been presented in the previous section 6. Nonetheless, references are constantly made among the scenarios to coherently compare and analyze all of them.

7.1 Analysis of the First Scenario

The first scenario, as it can be seen in figure 6.1, shows a considerable difference between the performance of the random version and the others. In the first executions, the weighted algorithm had a terrible performance, but suddenly it started to increase until reaching better values than Greedy and e-Greedy versions. These results are due to the weighted version starts to build connected graphs as the number of selected neighbors increments, and it discovers most of the nodes in its overlay network. After the performance have been raising, the number of selected neighbors is 70 or higher. However, since the size of the overlay network is limited to 70, there could not be other selected nodes, and therefore, the performance experiments a constant level of durability. The reason for the growth of the performance in the weighted version is that as the number of selected neighbors increases, the selection algorithm has a broader offer and more probabilities to find a neighbor that still accepts connections. Nevertheless, when the amount of selected neighbors is low, the probabilities are reduced for those nodes that are the last to execute the selection process.

On the other hand, Greedy and e-Greedy versions show a stabilized evolution throughout the simulation. The reason for this continuous trend is because they are affected by the *maxiota* value, seen in section 5.2.3, while the selected neighbor is depreciated. The fact of increasing the number of selected neighbors do not cause any effect on the selection process of the algorithms since the amount of selected neighbors (*maxiota*) is based on the most quality nodes of the set. Because *maxiota* is always lower than the amount of selected neighbors, the top *maxiota* priority nodes of the overlay network are the same in each simulation. These arguments can be corroborated observing the image 6.2, wherein each simulation can be seen that both algorithms have precisely the same O_v because the networks were built with the same neighbors connections in all simulations.

As it has been introduced, surprisingly the random version reached high-performance values. The reasons for this results are hard to define since the behavior is completely random in the selection process. Therefore, the third scenario was designed, among other reasons, to inquire more about the random behavior and make an appropriate analysis. However, due to the connections are randomly established, during the data analysis, poor stability on the results has been observed in the random implementation. The cycles durability in each simulation was very varied, even in some cases, the network was not connected from the beginning.

7.2 Analysis of the Second Scenario

The second scenario studies the algorithms according to an increment of the network size. As it can be seen in figure 6.3, the performance of the weighted version is terrible, although the reasons for these results have been explained in the previous case. In this case, the overlay network was formed by 50 nodes, whereas the number of selected neighbor was 20. However, this difference was too big so the algorithm could not obtain, on average, any complete network throughout the executions with different network sizes. With the purpose of analyzing better the performance of this algorithm, next scenario includes much more relevant results for it.

As far as Greedy and e-Gredy are concerned, their results are very similar since the number of rounds is expressed as an integer which is not entirely accurate, and the variability between simulation makes their results coincide several times. Regarding their performance, it is favorable that they kept the same amount of cycles, specially if the size of the network has increased. The random version instead, despite having better performance than the other versions, its performance experienced a decline. As it can be appreciated in figure 6.4, the O_v enhance as the network size rises. Although at the beginning all versions built networks with similar quality, but in the end, it can be observed a slight division between the random and weighted versions, and the Greedy and e-Greedy. The reason is that Greedy and e-Greedy, as a consequence of including all nodes in the network, their nodes have far more connections, whereas the other versions have a maximum of *maxiota* per node. Thus, since the performance is calculated with the equation 1 that decreases the quality according to the number of edges, the difference between algorithms is reasonable in big networks.

7.3 Analysis of the Third Scenario

Finally, the third scenario was designed after observing the results in the two previous cases, and the idea was to compare the behavior of the four versions at the same time. In order to get the weighted version working on connected networks, the difference between the selected neighbors and the overlay network size should be as small as possible, as we have seen in 6.1. Furthermore, because Greedy and e-Greedy have had similar results, it was thought to compare them from the beginning, where there is no difference between them, and check their evolution. The initial stage for both versions happens when the *maxiota* has the same value as the selected neighbors. On the other hand, there was the random version which has had outstanding performance. Thus, since the selection of nodes is random, we thought about minimizing the options of nodes to choose. Therefore, the overlay network size and the selected neighbors would have the same value, so even if the random version chooses the nodes randomly, all of them would be selected. Moreover, with this equality between parameters, at the same time, the weighted version was endowed with its maximum potential.

At this point, as it can be observed in figure 6.5, the first simulation was performed with the size of the network, selected neighbors and *maxiota* with the same value. Logically the results for the Greedys were the same because there was no difference in their behaviors. However, the random and weighted version due to their selection algorithm chooses the same neighbors, and their performance results were better than Greedys. The reasons for the better results could be affected by several factors, from the number of nodes that have participated in the network, to the number of edges per node. Is then reasonable that there were two groups of initial results, the random and weighted group which contain networks with fewer edges because their networks do not guarantee the integration of all the nodes, neither a minimum amount of connections. Whereas on the other group, the Greedys set up networks with all nodes interconnected and *maxiota* as the minimum amount of connections per node. Regarding the results of these two groups, as the metrics have been established to measure the durability performance, the networks which do not guarantee all nodes participation present more cycles of durability than the other versions.

The following simulations with an increase in the size of the overlay network and the selected neighbors reveal a dispersion of the results among versions. The random version maintains its performance throughout the simulations of the scenario. Nevertheless, the other algorithms experiment a decline on their durability as the parameters increase. The performance reduction among versions is understandable since in each iteration there are more nodes to select and the behavior of each version acts differently. At this point, when the *maxiota* has no longer the same value as the other parameters, the Greedys show different results between them. The analysis extracted from the second simulation of this scenario indicates that those versions based on a quality selection present lower performance. It is even lower if the algorithms force the integrity of all nodes with a minimum amount of connections, and as the worst case when the selection is forced with nodes which have curtailed levels of quality. Nevertheless, in the last simulations Greedy's performance values tend to unite, this tendency can also be observed in figure 6.1 and 6.3. The reason is that, unlike the second iteration of this scenario, the network is bigger and there are more nodes with the highest values of priority. Thus, even though the e-Greedy bases its selection 30% of the times on the selected neighbors, they will be nodes with the same priority as the top *maxiota* nodes.

7.4 General Analysis and Conclusions

To sum up, although the performance results obtained have not been as we had expected, the information extracted from the analysis and the new hypothesis is favorable for a future continuity of the project. Regarding this study, it should be noted that along the thesis several solutions were obtained to answer the problem formulated. Nevertheless, the objective of this thesis was not only to implement a solution, but also to find the solution with the best performance. Thus, the answer to the quality requirement of the problem formulated are all the relevant qualities of each implementation, extracted from the performance analysis of the executions in the scenarios.

One of the most remarkable conclusions of the analysis is the difference between the networks. The first group is created by the random and weighted algorithms, and it is characterized by not establishing a minimum number of connections for the nodes, and therefore, by not including all nodes in the network designs. On the other hand, the networks built by the Greedys algorithms guarantee a minimum number of connections per node, so all of them are included in the resulting network. Nonetheless, the fact of including all nodes in the network notably increases the number of edges in the most priority nodes. Such is the value of the number of edges that the performance of the most priority nodes is reduced faster than the performance of those nodes with less quality and fewer connections. Thence, even though a more in-depth investigation is needed to make reliable affirmations, the analysis has revealed that a connection overflow to the nodes with the highest priority is counterproductive to the durability of the network.

8 Discussion

The proposed problem for this thesis arises after the IOTA research, where it is discovered the connectivity problem that IOTA network is facing. Then, we have worked on the implementation of several algorithms to design an appropriate neighbor discovering protocol for the IOTA application. The primary goals of the algorithms are to create an undirected network where all nodes have a decent amount of connections to keep nodes synchronized.

The Greedy and e-Greedy neighbor discovery algorithms comply with the objectives defined for them. Although comparing the durability data among the other versions do not show that these versions provide the best results. Nonetheless, as it is argued in the analysis, after analyzing the results, it is observed that the comparison is made between two groups of algorithms with different properties in their networks. Thus, to be able to compare their performance, it should be studied in detail in further research taking into account the analysis that has been done observing the results of the algorithms performance.

In general, this thesis has been a very satisfactory project since it has found a problem in a booming technology for DLTs, and which is still in a development phase. After finding the problem, there was a long process of learning about the work environment for the development, design and implementation of the proposed solutions, running experiments to gather data about their performances, analyzing those and drawing conclusions about the results of the algorithms. During this process, the problem has been studied and faced with several solutions. Even though due to the temporary limitation to work on this project, and despite the motivation to keep improving the design of the solution, the implementation phase has been limited to what is shown in this report. However, since the project deals with very recent technologies the information provided and solutions implemented in this project could be beneficial for future research.

9 Conclusion and Future Work

This thesis consists mainly of two parts; the first one deals with literary research about IOTA Tangle features and how it works. This section includes a general overview of the technology as well as more detailed descriptions of some essential concepts. Along the explanation, there are several comparison and references to Bitcoin Blockchain, which is the other DLT explained after IOTA. The findings from the research are based on answering the research questions of this project, and they can be seen in table 1.1.

The second part of the project focuses on solving the problem formulated about IOTA network connectivity. The proposed solution is the implementation of a neighbor discovery protocol for the IOTA application. In order to find the best solution for the requirements of the application, the proposed solution consists of three algorithms with different neighbor selection processes. The findings are based on the analysis performed on the data collected from the executions of each version in various scenarios. Regarding the performance obtained from the experiments, it is different from what we have expected. However, since the algorithms were executed in specific scenarios, the previous analysis has allowed us to study the causes of their performance, as it has been seen in section 7.

The thesis has focused on improving the connectivity of the IOTA network because it is a problem that currently has direct repercussions on the users. In addition, due to the properties of dynamic networks, this is a problem that affects the correct functioning of the system, and in the long run, it is essential to find a solution for the system to work correctly. In order to formulate a possible solution, and besides being efficient, several proposals have been implemented to automatically establish interconnections between the nodes of the network.

Through this project, we have managed to implement four fully functional network configuration designs in a distributed system. Subsequently, a differentiation between two groups of algorithms has been observed through the analysis of performance results. The results show that the algorithms that do not integrate all the nodes in the network offer better results than those that guarantee integrity, probably because the number of nodes and connections between them is considerably lower. Furthermore, it is observed that focusing radically on the most priority nodes reduces the network durability instead of increasing it.

If we had continued working on this project, we would have kept studying the maximum number of connections to ensure optimum performance of the network, depending on the quality of the nodes. In addition, a comparison strategy should be defined for the two types of scenarios we have found. Currently, the comparison between them is conditioned by the difference in the number of nodes and interconnections.

Moreover, we could improve the implementation of the project by developing a more realistic simulation with an *event-based* model on Peersim. The level of realism in the simulation is what gives more reliable results and therefore more accurate conclusions. After achieving a new implementation, we could face the process of getting the results more in detail. Currently, the metric used to measure the network performance is obtained from static observations of the networks. A fascinating idea would be to analyze better the performance and update this process into a dynamic scenario where new nodes would join and leave the network throughout the simulation.

Finally, after improving these previous steps would be the time to run several executions, get new results, analyze them and make conclusions such as how many neighbors should be chosen, which networks size has better results or how the dynamism affects to the network.

References

[1] B. Sanou, *Measuring the Information Society Report 2015*. International Telecommunication Union, 2015.

[2] S. Popov, "The Tangle (white paper)," Feb. 2018.

[3] M. Walport, *Distributed ledger technology: Beyond block-chain*. Government Office for Science. Retrieved, 2016.

[4] C. Decker, R. Wattenhofer, "Information Propagation in the Bitcoin Network," *13-th IEEE International Conference on Peer-to-Peer Computing*, pp. 1–10, 2013.

[5] A. Pinna, W. Ruttenberg, "Occasional Paper Series Distributed ledger technologies in securities post-trading Revolution or evolution?" *European Central Bank*, Apr. 2016.

[6] D. Schiener. (2017, May.) A Primer on IOTA. Last accessed on 02/05/18. [Online]. Available: https://blog.iota.org/a-primer-on-iota-with-presentation-e0a6eb2cc621

[7] S. Popov, O. Saa and P. Finardi, "Equilibria in the tangle," Mar. 2018.

[8] M. Divya and B.B. Nagaveni, "Iota-next generation blockchain," *International Journal Of Engineering And Computer Science*, vol. 7, pp. 23 823–23 826, Apr. 2018.

[9] I. Fiedler. (2017, Mar. 5) Properties of a Token-Based Machine Economy. Last accessed on 02/04/18. [Online]. Available: https://medium.com/@IngoFiedler_96862/properties-of-a-token-based-machine-economy-5022e5041e56

[10] D. Sønstebø. (2017, Jun.) The Transparency Compendium. Last accessed on 01/05/18. [Online]. Available: https://blog.iota.org/the-transparency-compendium-26aa5bb8e260

[11] D. Schiener. (2016, Dec.) The IOTA GUI is here! Last accessed on 07/05/18. [Online]. Available: https://blog.iota.org/the-iota-gui-is-here-e1ae80e462d7

[12] D. Sønstebø. (2017, Mar.) IOTA Development Roadmap. Last accessed on 08/05/18. [Online]. Available: https://blog.iota.org/iota-development-roadmap-74741f37ed01

[13] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system (white paper)."

[14] M. Conoscenti, A. Vetro, and J.C. De Martin, "Blockchain for the Internet of Things: A systematic literature review," *IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA)*, pp. 1–6, Nov-Dec. 2016.

[15] Bitcoin. (2018) Developer Documentation. Last accessed on 28/04/18. [Online]. Available: https://bitcoin.org/en/developer-documentation

[16] SemkoDev team. (2018) IOTA finance manager. Last accessed on 15/03/18. [Online]. Available: http://www.carriota.com/

[17] SemkoDev. (2018) CarrIOTA Nelson. Last accessed on 15/03/18. [Online]. Available: https://github.com/SemkoDev/nelson.cli#running-nelson

[18] A. Gervais, H. Ritzdorf, G.O. Karame and S.Capkun, "Tampering with the Delivery of Blocks and Transactions in Bitcoin," *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 692–705, 2015.

[19] D. Schiener. (2016, Sept.) Discussion: Removing peer discovery. Last accessed on 02/04/18. [Online]. Available: https://forum.iota.org/t/discussion-removing-peer-discovery/939

[20] G. P. Jesi, "PeerSim HOWTO: Build a new protocol for the PeerSim 1.0 simulator," Dec. 2005, last accessed on 10/05/18.

[21] IOTA Support community. An introduction to IOTA. Last accessed on 17/04/18. [Online]. Available: https://www.iotasupport.com/whatisiota.shtml

[22] D. Schiener. Seeds and accounts. Last accessed on 17/04/18. [Online]. Available: https://domschiener.gitbooks.io/iota-guide/chapter1/seeds-private-keys-and-addresses.html

[23] IOTA Support community. How addresses are used in iota. Last accessed on 19/04/18. [Online]. Available: https://www.iotasupport.com/how-addresses-are-used-in-IOTA.shtml

[24] E. Heilman, N. Narula, T. Dryja and M. Virza. (2017, Sep. 7) IOTA Vulnerability Report: Cryptanalysis of the Curl Hash Function Enabling Practical Signature Forgery Attacks on the IOTA Cryptocurrency. Last accessed on 25/04/18. [Online]. Available: https://github.com/mit-dci/tangled-curl/blob/master/vuln-iota.md

[25] C. Paar, J. Pelzl and B. Preneel, *Understanding Cryptography: A Textbook for Students and Practitioners*. Springer, 2010.

[26] Dr. C. van Raaltenpark, "IOTA tutorial 5: Snapshot and Attach to tangle," Dec. 5 2017, last accessed on 20/04/18. [Online]. Available: https://www.youtube.com/watch?v=U4vF0_Imz_0

[27] D. Schiener. Making a Transaction. Last accessed on 22/04/18. [Online]. Available: https://domschiener.gitbooks.io/iota-guide/chapter1/how-to-make-a-transaction.html

[28] ——. Bundles. Last accessed on 22/04/18. [Online]. Available: https://domschiener.gitbooks.io/iota-guide/content/chapter1/bundles.html

[29] IOTA Developers. Glossary of Terms. Last accessed on 24/04/18. [Online]. Available: https://docs.iota.org/introduction/other-stuff/glossary

[30] "Properties of cryptographic hash functions," Ph.D. dissertation, Comenius University in Bratislava, 2008.

[31] A. Back, "Hashcash - a denial of service counter-measure," Aug. 2002.

[32] A. M. Bhise, S. D. Kamble, "Review on Detection and Migitation of Sybil attack in the network," *International Conference on Information SecurityPrivacy(ICISP2015)*, pp. 395–401, Dec. 2015.

[33] IOTA Developers. PoW on the Tangle. Last accessed on 08/05/18. [Online]. Available: https://docs.iota.org/introduction/tangle/proof-of-work

[34] D. Schiener. What is IRI. Last accessed on 28/04/18. [Online]. Available: https://domschiener.gitbooks.io/iota-guide/core-and-libraries/iri.html

[35] G. Rogozinski. (2018, Mar.) IOTA Reference Implementation. Last accessed on 28/04/18. [Online]. Available: https://github.com/iotaledger/iri/blob/dev/src/main/java/com/iota/iri/network/Node.java#L252

[36] IOTA Developers. getNodeInfo. Last accessed on 29/04/18. [Online]. Available: https://iota.readme.io/reference

[37] ——. IOTA Reference Implementation - Installing. Last accessed on 29/04/18. [Online]. Available: https://github.com/iotaledger/iri

[38] ——. Consensus on the Tangle. Last accessed on 31/04/18. [Online]. Available: https://docs.iota.org/introduction/tangle/consensus

[39] D. Schiener. (2018, Jun.) IOTA Foundation - Ask Us Anything. Last accessed on 02/05/18. [Online]. Available: https://www.reddit.com/r/Iota/comments/7orp03/iota_foundation_ask_us_anything_january_7th/dsbqsjb/

[40] IOTA Developers. (2017, Feb.) iotaledger/cli-app/lib/commands/health.js. Last accessed on 07/05/18. [Online]. Available: https://github.com/iotaledger/cli-app/blob/master/lib/commands/health.js#L27

[41] ——. Light vs. Full Node. Last accessed on 08/05/18. [Online]. Available: https://iota.readme.io/docs/light-vs-full-node

[42] A. Narayanan, J. Bonneau, E. Felten, A. Miller and S. Goldfeder, *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction.* Princeton University Press, Jul. 2016.

[43] Bitcoin Developers. Peer discovery. Last accessed on 04/05/18. [Online]. Available: https://bitcoin.org/en/developer-guide#peer-discovery

[44] Bitcoin Community. Satoshi Client Node Discovery. Last accessed on 05/05/18. [Online]. Available: https://en.bitcoin.it/wiki/Satoshi_Client_Node_Discovery

[45] Bitcoin Developers. bitcoin/src/net.cpp. Last accessed on 04/05/18. [Online]. Available: https://github.com/bitcoin/bitcoin/blob/master/src/net.cpp#L1745

[46] K. Kaşkaloğlu, "Near Zero Bitcoin Transaction Fees Cannot Last Forever," *The International Conference on Digital Security and Forensics (DigitalSec2014)*, pp. 91–99, Jun. 2014.

[47] J. Okonya. (2016, Oct. 25) What Happens When Bitcoin Mining Rewards Diminish To Zero? Last accessed on 06/05/18. [Online]. Available: https://coinidol.com/what-happens-when-bitcoin-mining-rewards-diminish-to-zero/