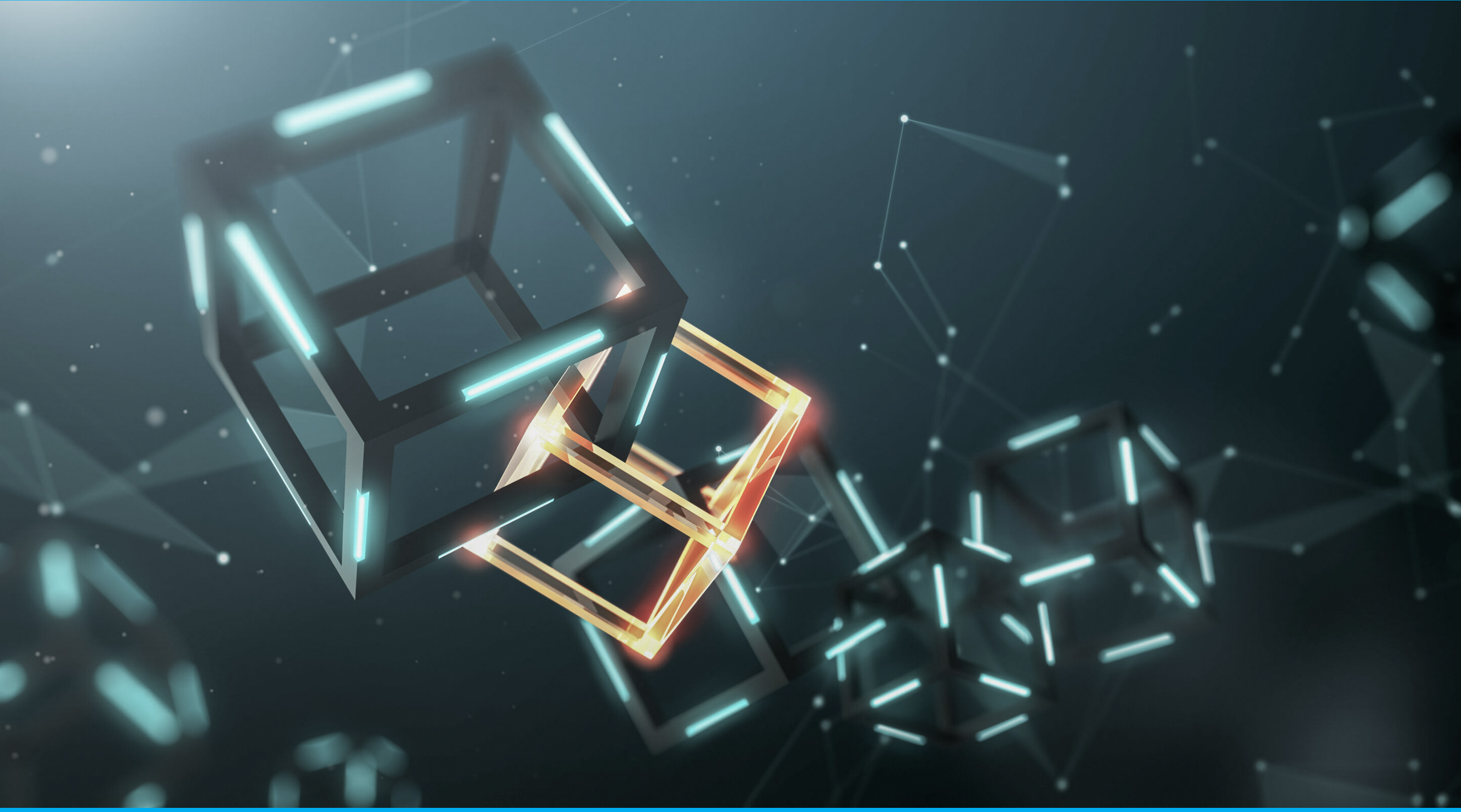


Enabling double-spending detection in a pair-based ledger

Umeer A. Mohammad

Enabling double-spending detection in a pair-based ledger

by

Umeer A. Mohammad

to obtain the degree of Master of Science

in Embedded System at the Delft University of Technology,

to be defended publicly on Tuesday November 26, 2019 at 14:00 PM.

Student number:	4748549	
Project duration:	January 1, 2019 – November 26, 2019	
Thesis committee:	Dr. Stefanie Roos,	TU Delft, Supervisor
	Dr. J. A. Pouwelse,	TU Delft
	Dr. Z. Erkin,	TU Delft

Acknowledgement

The two years of my master program at TU Delft have been an amazing journey that enriched me in so many ways.
I would like to express my sincere gratitude to my supervisor **Dr. Stefanie Roos** for giving me the opportunity to conduct my master thesis on this topic. Thank you, for pointing me in the right directions when I faced obstacles, for all the weekly meetings and the precious feedback.
I would like to thank my fellow batchmates and friends I made at TU Delft, **Shivanand**, **Abhairaj**, **Himansu** and many more for all the good moments we had together.
Special thanks to **James S.**, **Tom W.** and all the co-workers at recyclabox, the company where I have been working along with my study. Thanks for flexibility granted to me, probably without you this amazing experience would not have been possible.
Finally, I would like to thank my **parents**. Your encouragement when the time got rough helped me a lot, thanks for being a constant source of motivation and support.

Thank you all very much!

Abstract

The development of blockchain, along with the creation of Bitcoin, opened the road to the first digital currency that is secure against double-spending without the need for a trusted authority or centralized server. The architecture of traditional blockchain structures like the one used in Bitcoin is, however, not very scalable because of its limiting slow global consensus mechanism.
To address this limitation multiple paths have been explored and one of these is represented by Trustchain. Its architecture is based on a pair-based ledger system; where pairs of nodes register their transactions on entangled blocks stored in local, immutable and ordered chain.
This technology guarantees high transaction throughput but has a big drawback; it is not possible to actively prevent fraudulent operations like double-spending but only to detect them a posteriori.
The goal of this thesis is to research and evaluate mechanisms that can be implemented to reduce the detection time of this fraudulent activity, in a system like Trustchain.
The first strategy involve relying on global dissemination, but this is quite inefficient. The second one uses anonymous auditing and localized dissemination to better perform regarding efficiency, at a small expense in terms of transaction duration.
The performance improvements are confirmed by simulations. If compared to the existing mechanism, the second proposed solutions achieved double-spending detection in just a fraction of the original time at a cost of a small overhead.

Umeer A. Mohammad
Delft, 26 November 2019

Contents

List of Figures

1 Introduction

Transactions between two entities are at the core of society, and the ease of execution and the reliance of these have been always an important factor.
Transactions were initially concluded by exchanging physical material (barter, payment in minerals, metals) and there was no need of a centralized authority to verify or track them. This was possible because of the reliance of the transactions was granted by the physicality of the payment action and the security of its irreversibility.
To cope with the technological shifts in commerce evolution, new forms of payment systems have been created. Some of the added values present in these are improvements in the ease of use and the possibility of performing transaction remotely. This innovation has however introduced hidden costs, one of which is the dependency from centralized authority to register all the transactions.

1.1. Road to Blockchain

Since when transactions were done by exchanging goods, there was a need of having a secure and reliable way to perform them. For example, in situations where transactions were done by exchanging animals, it was necessary to be an expert veterinary to evaluate the conditions and value of the goods. There are many inconveniences and problems linked to the barter exchange, such as difficulty in storing wealth, the indivisibility of goods and lack of a common measure of value.
With the trade of precious metals or minerals, some of the difficulties were solved, but precise balance scales and significant knowledge was still necessary to guarantee the fairness of the transactions.

With time, transactions evolved in a settlement based on currency. Initially, currencies were made of valuables metals (gold, silver, copper), but today's money is known as fiat money. It was introduced as an alternative to the already mentioned commodity money and representative money (can be used to claim a commodity).
Fiat is a Latin word that means “let it be done." Fiat money doesn't have an intrinsic value itself, but its value is attributed and guaranteed by government regulations or laws.
Today's moneys have many complex anti-fraud systems that make it very complicated to counterfeit. Some of them include print with photo-luminescent ink on security thread, laminated paper with watermarks or use of an embedded transparent layer with intricated hologram, like the one in Figure 1.1.

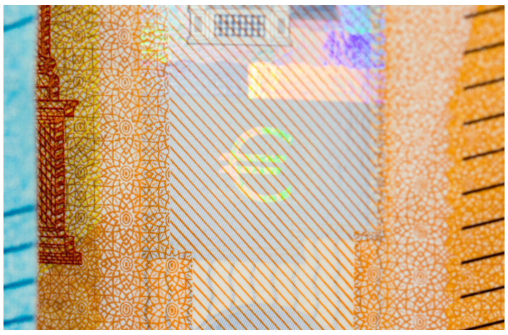


Figure 1.1: Close-up of 50 euro note's hologram.

Modern currencies solved many issues and made performing transactions much simpler than ever before, but they have still many limitations, for example:

- The fact that money is of no value outside the country of issue, and foreign currency exchange is not free of cost.
  Hypothetically, if you keep converting any amount of money you will eventually end up with zero.
- There is a possibility of damaging the paper and consequently losing all its value.
  A fire may burn it; natural events can destroy or disperse it, it may also get eaten by animals.
- It needs to be transported or stored by someone all the time, which makes it a vulnerable target to malicious people/organizations and cause robberies and thefts.
- A major drawback is the ease of money production by the government. There is always a danger of over issue in a situation where the government is in difficulty.
  Emitting in circulation more money creates inflation, which results in the reduction of currency value.
  If this process is iterated the buying power decrease, and more money is necessary to purchase the same service/goods.
  This happened in various countries in recent times: in Russia (1917), in Germany (1919), in China (1944), and so on [22].

With the advent of technology, the world started moving towards digital payment. The developments of digital payments are linked to the beginning of the internet, which can be outlined with the creation of ARPANET, developed by the US during the Cold War and launched at the end of the 1960s.
The modern internet banking system has developed from the banking services over electronic media that were introduced in the early 1980s; there were several trial and errors before the modern concept was discovered.
The first one, was later in the 1990s when The Stanford Federal Credit Union become the first institution to offer online banking services to customers. However, early online payment methods were not very user-friendly, requiring specialised knowledge of data transfer protocol [16].
During that period, the idea of total digital cash was also being introduced by David Chaum's research paper [18]. In 1990, he founded DigiCash in Amsterdam, an electronic cash company to commercialize the ideas in his research, this, unfortunately, did not last long as it was filed for bankruptcy in 1998. "It was hard to get enough merchants to accept it, so that you could get enough consumers to use it, or vice versa," he says [31].

Today, one of the biggest digital currency systems is PayPal, which started in 1999 as an online service to transfer money and become popular thanks to the integration with eBay. PayPal always innovated by introducing new features like making payment available just with an email address, implementation of new currencies, mobile payment app; while always being easy to use and implement in 3rd party applications.
With the growing popularity of e-commerce digital payment became a great demand, with numerous digital payments companies beginning as major players.
PayPal continued to expand and spread, for the fiscal year 2017, PayPal reported earnings annual revenue of US $13.094 billion, an increase of 20.8% over the previous fiscal cycle and a market capitalization of $100 billion.

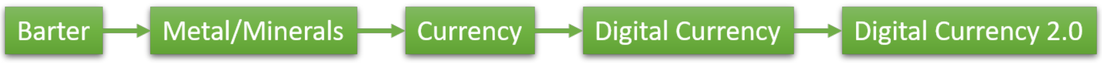


Figure 1.2: The evolution of payment methods.

1.2. Blockchain

One of the main problems of using financial institutions as a trusted third party to perform electronic payments is that these can be influenced by their status and affected by their regulations where these are based. Having a "middle man" that has authority over your assets and on the transactions can lead to problems in situations where this authority is unstable or is located in problematic organizations/states (corrupted and fraudulent). This situation can be improved by cutting any centralized dependency from the equation to perform transactions.

The idea of a decentralized currency using cryptography had been around for many years, but it got a lot of attention for the first time with the launch of Bitcoin [28] in 2009. Bitcoin was introduced by Satoshi Nakamoto [1] as a peer-to-peer version of an electronic payment system, that would allow digital transactions to be sent directly from one individual to another without going through anyone.
Along with Bitcoin many other cryptocurrencies have been conceived in recent years; these differ from the latter by various aspects, from the ease of use to privacy improvement or introduction of additional feature.
One of the most popular is Ethereum, which entire functionality is based on smart contracts. This is also considered by many of the most prominent Bitcoin alternative.
Ripple registered an astonishing growth since it was created back in 2012. This cryptocurrency has especially been appreciated by banks and payment network for its suitability as settlement infrastructure technology.

1.2.1. Scalability Limitation of Blockchain Systems

One of the main limitations of the blockchain architecture used in Bitcoin and many other cryptocurrencies is scalability.
Bitcoin's maximum throughput varies from 3.3 to 7 transactions/sec [17]. With the rise of its popularity and the consequent growth in the currency network size, this limitation can lead to an increase of transaction time and fees.
The most distinct surge reached an all-time high in December 2017 when the transaction fee was close to US$160. This was in concomitance with the period where Bitcoin reached the highest evaluation of more than US$20,000.

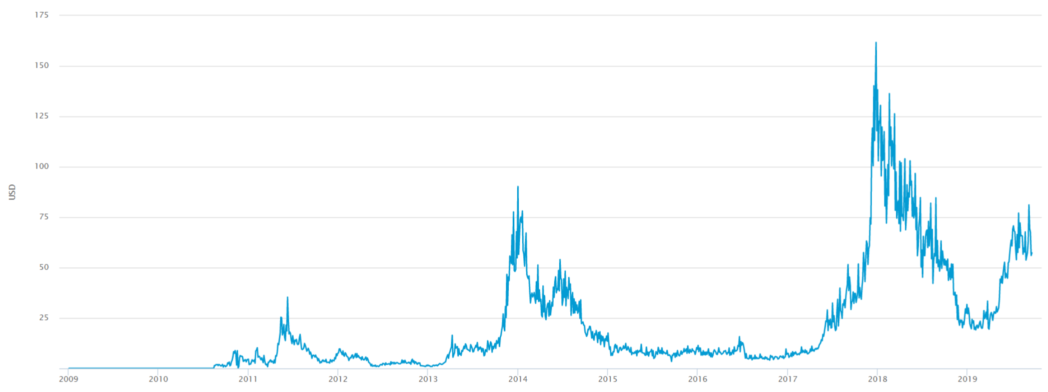
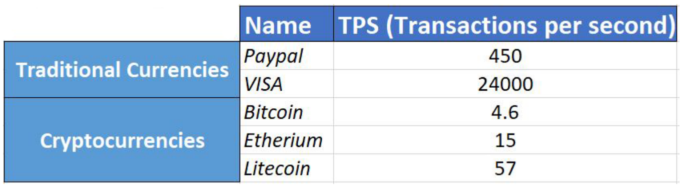


Figure 1.3: Historical transaction fee for bitcoin

[1]Satoshi is just a pseudonym and no one knows the real identity of this individual/team. Craig Steven Wright, an Australian computer scientist, has publicly declared to be part of the team that created bitcoin and to be behind that name. But this is considered by many just a hoax.

Today, the Bitcoin system processes 4.6 transactions per second (TPS), which if compared to traditional currencies is quite low. Paypal handles millions of transactions every day with an average of 193 TPS but up to 450 TPS (registered on Cyber Monday 2015).
VISA, which is often used as a comparison benchmark, does sustainably 1,736 TPS but can handle up to 24,000 TPS, while financial institutions like stock exchanges can accomplish up to eighty thousand TPS.



	Name	TPS (Transactions per second)
Traditional Currencies	*Paypal*	450
	VISA	24000
Cryptocurrencies	*Bitcoin*	4.6
	Etherium	15
	Litecoin	57

Figure 1.4: Comparison of TPS for different currencies.

Besides the low TPS, every transaction conducted on the Bitcoin network needs to wait for a confirmation time before can be considered successful.
Every transaction needs to have a couple of confirmation block over it (typically 6). Because a new block is generated every 10 minutes [3] in the best scenario possible the waiting time is around 1 hour. This, however, is an optimistic evaluation as the probability of having a transaction always registered in the very next block is quite low, realistically a Bitcoin transaction can take up to several hours and sometimes even days to be confirmed.

According to Vitalik Buterin (Ethereum founder), in a not too distant future the majority of the demand for network capacity will not come from human-generated transactions but from IoT devices, which potentially will require thousands of TPS. In the "low millions" is reasonable, although billions of TPS is still the "long-term goal." Buterin says [15].

1.2.2. Solutions Strategies

These problems are not only affecting Bitcoin but many other blockchain projects. To allow growth and usage in everyday life, this technology needs to evolve to perform adequately to the market requirements (fast, easy and cheap). In order to address these problems the scientific community started promptly the study of two kinds of solutions.

- **Layerization (Layer 2 Solutions)**
  The first one is to create a layer on top of existing blockchain architecture that acts as a buffer for the latter. This parallel layer will inherit all the benefits of the below structure but will add flexibility to it. Payment channel [32] allows nodes to perform transactions without the need of pushing every time the outcome on the main blockchain framework. This allows to have much higher TPS as the upper layer is not bounded to the previous constraint.
- **Re-Engineering Blockchain Fabric**
  These solutions focus on modifying the blockchain fabric, by altering the consensus methodology or entirely modifying the way how transactions are structured and stored in the network.

Layerization is a very good option as it works on the already well-established blockchain core structure. This solution, however, has some limitations and problematics, for example, the needs for depositing assets on the channel to perform transactions and possible issues when transferring large amounts [5].
For these reasons I prefer to focus the efforts of this work on the second type of solutions, in particular, towards a pair-based blockchain architecture called Trustchain [30].

1.3. Motivation and Problem Statement

Trustchain is a protocol for a shared interconnected data structure with enormous potential, it is designed to record transactions among strangers without central control, and it is able to support high transaction

volumes, all being performed without any fees.
To achieve these performance the consensus system is not based on a global collective agreement, but rather on a faster local consensus and dissemination.

This architecture can be exploited by malicious nodes, as under certain conditions it is possible to perform double-spending attacks [2] and operate for long periods before being identified by the current detection system. Besides being slow, the current mechanism does not scale up well with the network size.
The longer it takes to detect an attack the more complex is the correction required to fix the collateral damage (not part of this thesis).

With this work, I want to approach this weakness with the goal of finding a faster and scalable detection system for it.

1.4. Research Question and Contribution

The goal of this thesis is to find and evaluate mechanisms that reduce the double-spending detection time in a pair-based ledger blockchain architecture and can be implemented in the Trustchain fabric.

These are the contributions that were needed to find an answer to the research objective.

- Initially, the Bitcoin architecture fabric is analyzed to understand the reasons for its inadequacy to scalability. The Trustchain system is then thoroughly analyzed via literature study, online repositories examination and by interviewing members of the team that developed it, with the intent to collect as much information as possible.
- The double-spending problem is then studied with the scope to create a realistic adversarial model operable in Trustchain, that can be used to evaluate the performance of its double-spending detection system.
- A literature study is performed to acquire details on existing pair-based ledger system and on anonymous auditing techniques that can be employed in the contest of this work.
- Based on the knowledge collected with the above steps, two solutions algorithms are designed and detailed designs of these are discussed.
  The first one is based on the total broadcast of every new transaction. And a second solution, much more cost-efficient [3], is based on a two-step mechanism, an initial anonymous auditing scheme to verify the partner's ledger and localized dissemination done post-transaction.
- A simulation of the Trustchain system is designed on OMNeT++, and the original detection system and the two proposed solutions are here composed along with the adversarial model defined before.
- We determine suitable simulations scenarios and network structures that can be used to efficiently assert the performance of the solutions, as well as defining metrics parameters that can be employed to evaluate them.
- Utilizing DAS-4 [4], a supercomputer cluster, all the experiments are executed and its results collected. The current system's performance are compared to the outcomes of the proposed solutions, to draw conclusions.

1.5. Organization

Before describing the studied solution/s, it is necessary to understand why the traditional blockchain is so slow and inadequate to scalability, and how a pair-based architecture systems like Trustchain can help to solve the problem. Details of these argumentation can be found in Chapter 2.
In Chapter 3 are analyzed the fragility aspects of the pair-based ledger system and how malicious nodes can perform double-spending attacks. In Chapter 4 are described as relative work that has been done on this

[2]An attack which involves spending the same single digital token more than once.
[3]The completion of a transaction requires a smaller amount of messages sent across the network.

topic, and in the following Chapter 5 are illustrated in details the two proposed solutions of this document. Chapter 6 contains an in-depth description of the tools and environments used to evaluate the performance of the solutions, and are described all the results obtained from them. Finally, in Chapter 7 conclusions of the research are reported.

2 Background

2.1. Bitcoin

Bitcoin is the most popular cryptocurrency, it is composed of a distributed system operating on a P2P network, where multiple nodes work collectively on a single global ledger. It uses digital signatures to establish the bitcoin's possessions and employees the blockchain architecture to record the history of the transactions. The network registers transactions by connecting them into a continuous chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work [35] [14].

The chain is disseminated via a P2P network to achieve consensus about the history of transactions. Public broadcast of logs lead to achieving transparency that satisfies the needs of trust for the users, however, this is a double-edged sword because has the negative effect of being critical in regards to performance.

The history of transactions is called blockchain or ledger, and it is mantained, recorded and run by "miners" who get rewards in exchange for their service [20].

Why It's Slow

The number of transaction per second in this cryptocurrency (like many others) is very minute in comparison to VISA or PayPal. We described in the introduction how scalability is a very significant concern for cryptocurrencies.

In this section, we are going to examine why the TPS in Bitcoin is low, and the reason why not possible to speed-up this number. The reasoning here conducted on Bitcoin can also be applied to many other cryptocurrencies having a similar architecture.

Bitcoin's block size is set to 1MB (1,048,576 bytes — 4MB with SegWit) and the average transaction size is equal to 380.04 bytes [2].

$$N_{TransactionsPerBlock} = \frac{BlockSize[Bytes]}{AverageTransactionSize[Bytes]} = \frac{1048576}{380.04} \approx 2759.12 \tag{2.1}$$

The current block generation period is 10 minutes, which mean that a new block is added to the chain every 600 seconds.

$$N_{TPS} = \frac{N_{TransactionsPerBlock}}{BlockGenerationTime[s]} = \frac{2759.12}{600} \approx 4.6 \tag{2.2}$$

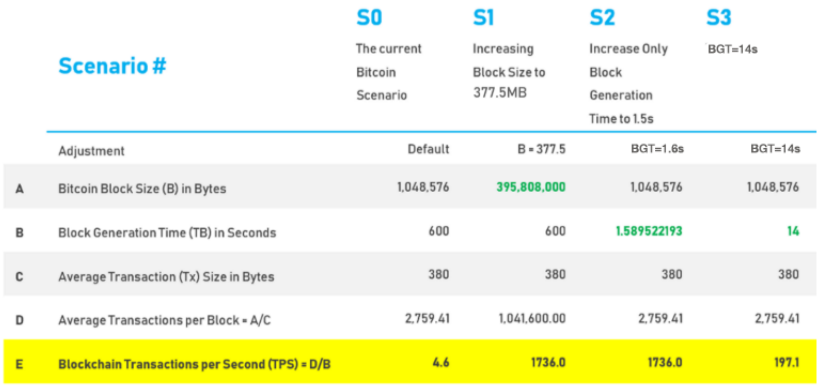
This means that globally there can be at most 4.6 transactions per second processed, which is nowhere comparable to the today's market competitors. However, Bitcoin can hypothetically increase the TPS by adjusting two variables.
One of them is the block size which is hardcoded to 1 MB. The second one is the block generation time (BGT), which can be modified.

BGT Variation

The BGT is linked to the complexity of the hashing puzzle. Reducing the complexity of the puzzle should then increase TPS.
To match Visa's performance, a new block needs to be generated every 1.6 seconds. This is however not possible due to the slower blocks propagation system.
The Karlsruhe Institute of Technology measures Bitcoin parameres, and the average block dissemination time reported on Jan 2019, was of 13989.42 milliseconds to reach 99% of the network [1].
The BGT needs to be higher than this time, if it does not, a new block will be generated before the previous one reaches the majority of the nodes; The more close BGT is to this number the more eminent is the probability of creating forks, orphan blocks or even vulnerability issues. If we reduced BGT to 14 seconds the increase of performance is of 188x times, reaching 864 TPS, but this result would still be quite far from VISA moreover it will introduces a lot of risks.

Block Size Variation

Let's now imagine that the block size is modifiable to any value. The first idea would be to make it larger so that it can accommodate more transactions.
However, increasing the dimension of the block will also affect the dissemination time.
Assume we increase the block size to 10MB, the dissemination time will also increase to a factor x10, from 14s to 140s. Consequently, BGT should also be increased of the same factor, and this will annul any TPS gain.
The only way to decouple these two factors is to increase the bandwidth between the nodes that compose the Bitcoin network, decreasing consequently the dissemination time. This however is complicated to achieve because of the nature of P2P networks, where each peer is independent and different.
In the Figure 2.1, it is possible to see all the situations described before and a hypothetical case where the block size is 377 MB.



	Scenario #	S0 The current Bitcoin Scenario	S1 Increasing Block Size to 377.5MB	S2 Increase Only Block Generation Time to 1.5s	S3 BGT=14s
	Adjustment	Default	B = 377.5	BGT=1.6s	BGT=14s
A	Bitcoin Block Size (B) in Bytes	1,048,576	395,808,000	1,048,576	1,048,576
B	Block Generation Time (TB) in Seconds	600	600	1.589522193	14
C	Average Transaction (Tx) Size in Bytes	380	380	380	380
D	Average Transactions per Block = A/C	2,759.41	1,041,600.00	2,759.41	2,759.41
E	Blockchain Transactions per Second (TPS) = D/B	4.6	1736.0	1736.0	197.1

Figure 2.1: The different scenarios for increasing TPS. Only solution 1 and 2 can obtain Visa's performance, whoever these are not implementable. Source: TowardsDataScience.com

2.2. Trustchain

Trustchain [30] has been designed as a non-blocking transaction system between agents that supports multiple simultaneous trades. This system is highly scalable while maintaining stable performance independently of the numbers of active users, which is not possible to achieve in a traditional blockchain structure where there are a unique shared ledger and a global consensus system.
These properties are possible thanks to it's fully distributed architecture.

Architetture

In contrast to Bitcoin or Ethereum, where there is a global consensus system on a unique universal ledger for all nodes, in Trustchain every agent own their data by keeping track of transactions where they were involved in. Consistency of the chain is guaranteed by a local consensus system. This creates the basis for its scalable

performance.
The system is made of individuals that have a interconnected ledger (also defined as chian) and rules that define how these are constructed.
A transaction between two parties creates a new element or block in both of the participant's ledger, this new component contains information like:

- **Transaction Data**
  This field reports the value that is exchanged between the two parties.
- **Previous Block Information**
  Two fields contain the hash values of the last block in the chains of the two concerned nodes.
- **Previous Block Information 2**
  A pair of sequence numbers $s \in \mathbb{Z}$, are also being registered. These define in which position a block is located in the agent's chain.
- **Public keys**
  Public keys of the nodes participating are also being registered.
- **Signature**
  Both agents sign the block using their private key, and thanks to the previous fields it is possible to cryptography prove the block ownership to them.

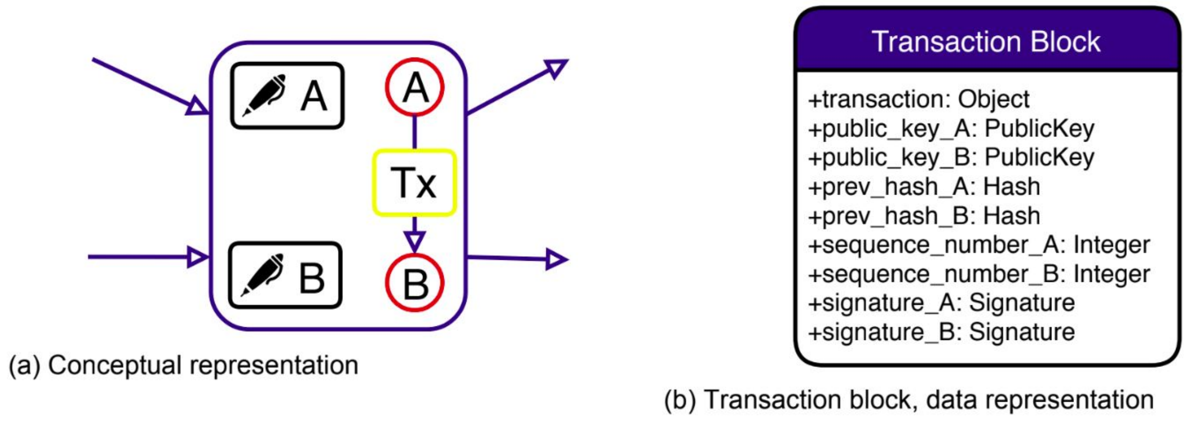


Figure 2.2: Anatomy of a block belonging to a Trustchian ledger. Source: Creation of TU Delft Blockchain Lab

Every node in the network starts with an initial transaction in his ledger called genesis block, the sequence number of this special block is one, and any subsequent block created on top of this will increase this number by one unit.

The system is based on the idea that before signing a transaction both parties will agree on a value, this makes tampering easy to detect.
Another aspect that contributes to the security of the system, is the fact that ledgers elements are temporally ordered and cryptographically interconnected. This not only makes complicated to alter the history of transactions but, also gives the system participants the possibility to check the correctness of transactions themselves, enabling them to locally reach consensus with out the need of consensus on a global level.

As dissimilarity to a Bitcoin's block, where multiple transactions are registered in a single block to enable high throughput[1], here a block holds at most one transaction. The generation of a new block is not compute-intensive and can be created almost instantly; This system can be considered fully mobile-friendly.

[1]Due to the complex and challenging block generation procedure, a new block is generated approx. every 10 min. By storing multiple transactions on a single block it is possible to increase the transaction throughput.

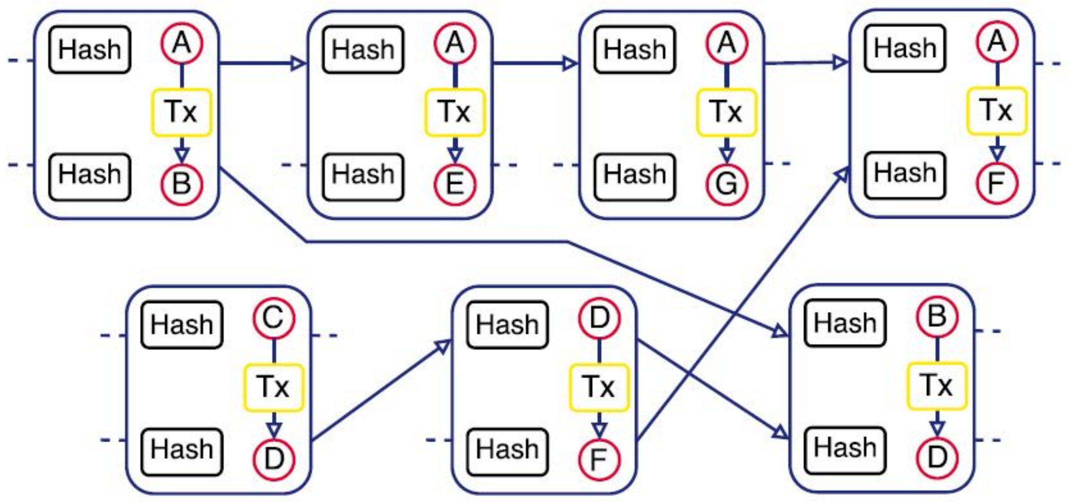


Figure 2.3: Every block in Trustchain is linked to previous transactions of both participants ledgers. Source: Creation of TU Delft Blockchain Lab

2.2.1. Transaction Dynamics

In this section are explained all the steps that two agents have to follow to perform a transaction. A transaction can be divided into 5 messages/steps.

1. Transaction Request
2. Chain Request
3. Chain Replay Message
4. Transaction Confirmation
5. Dissemination

Transaction Request

This is the first phase of the transaction. A node that wants to send a value to another node in the network has to create a message called "Transaction Request" or "half-block", and transmit it to the latter to initiate the process.
This message contains information like the value that he desires to transfer, the newly calculated sequence number, the hash of the last block in his ledger, his public key and the signature on these pieces of information. The recipient of this message will proceed with the next phase if he is available, in case he is not, it will replay back with a "Busy" message. The busy message indicates that a node is already engaged in another transaction or it is not willing to proceed with the affair.

Chain Request

As the name suggests, this message is employed to request a ledger from a node. This message does not contain any transaction-related information.

Chain Replay

Upon receiving a "Chain Request", a node has to reply using the "Chain Replay" message. Similarly to the request, this message does not include any transaction-related information, but it carries all the blocks contained in the sender ledger (starting with the "genesis block" [2]).

[2]The genesis block is the first block to be created in every node's ledger.

Transaction Confirmation

When the message containing the ledger reaches the node waiting for it, is analyzed and if there are no issues with it, a "Transaction Confirmation" or "full-block" message is generated and sent back.
This message contains all the knowledge that was present in the half-block plus additional information, like new the sequence number of the other party, the hash of his last transaction, his public key and finally his signature on all the information of the message. This full block is also registered in his ledger before being transmitted.
The node receiving this message, after performing an inspection of it, it proceeds to store its content in his ledger.

Dissemination

After adding the newly created block to their ledgers, both parties disseminate the event to their neighbour nodes. According to the latest implementation of this architecture in a emulation system designed on Tribler [11], the dissemination is done up to a maximum of 20 nodes.
The dissemination message is quite similar to the full-block in terms of content.
When this message is received by a node, it's content is compared to the previously collected knowledge[3] to check if there are a discrepancy with the transaction.

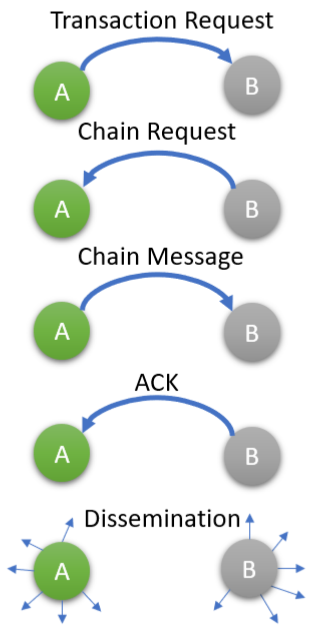


Figure 2.4: Graphical representation of all the steps involved in a Trustchain transaction.

2.2.2. Problems with Non-Global Consensus System

The superior performance of the Trustchain, in contrast to systems where there is a single global ledger, comes with a cost to pay.
By not having a global consensus system it is possible to obtain better scalability, thanks to the ability to perform multiple transactions in parallel, but the drawback is that it becomes difficult to actively prevent frauds such as double-spending. The only defence is the fact that the fraud can be eventually detected after it is committed.

In the following chapter, this problem is analyzed thoroughly in order to comprehend how to designed an effective detection system.

[3]Knowledge collected from previous transaction or dissemination happened in the network.

3 Problem Statement

The absence of an unique ledger and not reliance on global consensus, can be exploited by malicious nodes. An evil node can perform two colliding transactions with two different partners, the result of this operation will create in the system two blocks with identical sequence number of the attacking agent. The evil node will keep only the second transaction in his chain, by hiding the first one as if it never happened.
In Trustchain this attack is detectable thanks to the block's sequence numbers, but it can take a while before this happen.

3.1. Double-Spending

A double-spending attack allows the attacker to utilize resources already consumed before, allowing him to fraud two or more node with the same resource set.

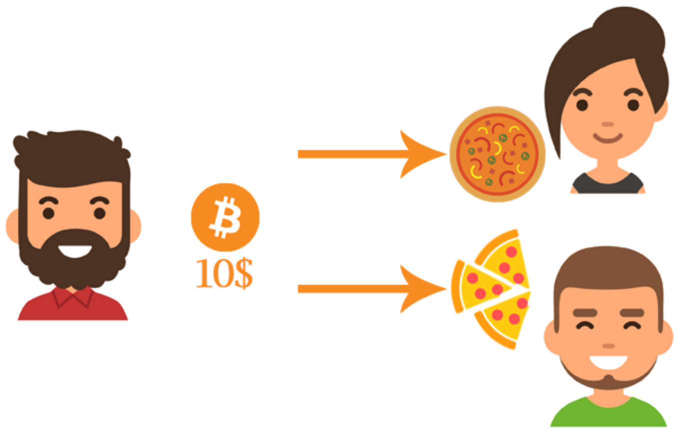


Figure 3.1: Double-spending a virtual coin.

Let's consider a real-world example to better understand the problem.
You decide to go to a restaurant to order a pizza worth 10 euro, and you pay with a paper note that goes into the shop's cash desk. Now, you can not spend that note somewhere else that day. This constriction is easy to understand in the real world, but in Trustchain things are a bit different, here it is possible to double-spend your 10 euro somewhere else multiple times before being caught by a cashier, which might take quite some time.

This is attainable because of the incapability of the second cashier to perform a complete check on all the transactions done by the buyer. The adversarial node can easily possess multiple ledgers, and use a ledger where an expense is not registered to perform a transaction, doing in such a way a double-spending. The retroactive detection system can take a while to detected this instance, and this can result in chain corrections increasingly complicated.

3.2. Double-Spending Attack Strategies

The dynamic with which an double-spending attack can be performed can vary a lot, and consequently with that the detection time also changes.
Here are some examples of attack strategies that an evil node can perform.

- Performing one malicious transaction and then a correct one. In this case, after completing the first transaction, the evil node does not save the details about it and starts the second one.
- Performing one malicious transaction and then a correct one, but do not disseminate it at the end.
- Performing one malicious transaction and then many normal in row.
- Performing multiple malicious transactions in a row.
- Just do a malicious transaction and go offline.
- Many others...

The best way for an evil node to increase the probability of not being detected is to limit the information dissemination at the end of the malicious transaction, and this can be achieved on his side by not performing ulterior transactions with other nodes.
However, a double-spending attack can be considered successful only if the evil node performs a second transaction where he spends coins that he is not supposed to have, and more importantly creates a block with an already used sequence number. Hence for all the above reason I believe that the best attack strategy is the second one in the list.

The timing between the two transactions should be as short as possible, by performing transactions quickly, one after another the evil node does not allow time for the information to spread. Imagine a scenario where the evil nodes waits days or even weeks before performing the second transaction, here the potential target might get sufficient information to detect the evil node.

Because of the fact that information dissemination takes time to cover the network, an evil node can exploit this facts, and conduct transactions with nodes that are far away from each other. By doing this the two transactions dissemination will take longer to reach the opposite node's area, resulting in longer longevity for the evil node before being detected.
This type of attack requires extra knowledge regarding the nodes network's positions and dissemination dynamics, information that can be quite challenging to obtain.

In the case of transactions initiated by other nodes in the interest of the evil node, there are two strategies that can be performed. The evil node can simply ignore the request or engage in the transaction. The best option is clearly the first one because it limits the exposure of the evil node's chains to the network and so the risk of being detected.

3.3. Detection of a Double-Spending

A node can detect double spending during two instances, before and after the evil transaction is performed.

Proactive Detection
It is possible to proactively detect a double spending during the chain verification phase, hence before the transaction is registered. During this phase a issue can be identified if and only if the node performing the check has previously collected information regarding the malicious node's transaction. This data can be collected via a previous transaction with the evil node itself, or more likely as a result of receiving dissemination messages from the network.

During this phase, a node has to compare the received information with previously collected knowledge. There are two main conditions that can lead to a failure of the verification:

- The node performing the verification has information about a transaction in which the partner was involved, but there is no information about that event in the chain sent by him.
  In this scenario, the evil node is hiding his last transaction, by not presenting the last block in his chain.
- The local knowledge regarding the partner transaction differs from what he is claiming in the chain.
  The evil node has overridden one of the blocks in his chain with another transaction and now it is attempting to perform another double-spending.

Retroactive Detection
The second way to detect an evil node is by analyzing the dissemination information a posteriori of a transaction. During this phase, the network is broadcasted transaction's information. Hence, by comparing these information with previously collected knowledge, it is possible to detect possible duplicate of sequence numbers, which is an unremarkable trace of a double spending event.

3.4. Probability of Detecting Contrasting Information

The historical knowledge collected by every node that is created thanks to transactions dissemination is the key that enables detection of double-spending.
It was not easy to find details regarding the system employed to detect these attacks, in the original paper [30] there is no clear specification but just an indication that dissemination is utilised.
By interviewing some of the PhD students and developers working on Trustchain, I got to know details regarding the emulator's architecture developed on the Tribler network. Due to the impossibility in obtaining more information on this matter, the dissemination specification employed in the emulator are used as if representative of the Trustchian system.
As already expressed in Section 2.2.1, the dissemination reaches at most a total of 40 nodes [1], more specifically randomly selected nodes directly connected to the agents performing the transaction. In case of transactions involving evil nodes, where the latter avoid any information dissemination, this number is reduced to half.

Having small dissemination area affect the probability of reaching nodes that have contrasting information in the network. Furthermore, if the neighbour nodes are offline during dissemination this can significantly decrease.

$$n = \text{Number of nodes reached during dissemination} \tag{3.1}$$

$$P(n,t) = \text{Probability of detecting double spending in a given time } t \tag{3.2}$$

$$P(n,t) > P(n-1,t) \tag{3.3}$$

In reality, this probability is affected by many other parameters, but for the scope of this demonstration will simplify to just the number to nodes disseminated.

[1] Up to 20 nodes per peer.

4 Related Work

To formulate a solution, analysis of existing approaches addressing this and similar problem was required. Some of these concepts that helped to understand the problem and form the solution are described in this section.

4.1. State Machine Replication

The problem that we are facing with double-spending attacks resembles the problem of state machine replication (SMR).

Service replication is a technique in which copies of the "service" are deployed on multiple servers instead of just a single one. This is typically done to increase performance and/or provide fault tolerance. The improvement in the performance of the service can be achieved thanks to the existence of multiple points providing the same service. Fault tolerance can be provided thanks to the redundancy in the system, since if a fraction of servers fail there are replicas that can still provide the same service.

Replication of database is the core difficulty behind the SMR, and early works like the Epidemic Algorithms by Demers et al. [12] were addressing these difficulties. One of these is the Anti-Entropy system, here every site contacts another instance at random and both exchange the content of their database to resolve any differences/conflicts between the two. This is extremely reliable, but can not be used too frequently as it requires a considerable amount of time to examine the databases.
This concept was later generalized as gossiping [36], and describes a situation wherein a distributed a system randomly chosen pairs of agents communicate to share knowledge.

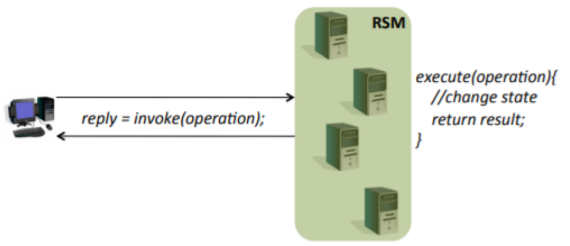


Figure 4.1: The basic programming model of a replicated state machine Source: [13]

In this work we adopt a variation of the information dissemination scheme described in this section, to build the verification of the ledger and the dissemination phase.

4.2. Pair-Based Ledger Architecture

In parallel with blockchain development, a variety of accounting systems based on the pairwise system was developed. Here transactions are initially only registered in the two participant ledgers, and later through the employment of dissemination this is spread to other agents.
In contrast to traditional blockchain architecture which enforces global consensus, a pairwise system is capable to offer much higher transaction throughput. However, by dropping this requirement, consistency can not be guaranteed, as there is no insurance on the information that agents are bartering.

BarterCast [26] is a system that works on this premise. It is based on a maxflow calculation executed on a flow graph created and shared by agents in a BitTorrent network. The goal of this system is to detect and successively ignore lazy-riders, or in other words, nodes that download content from the torrent without sharing it back.
The graphs structure utilized in this system is composed as follows, vertexes are represented by nodes and the edges by the bandwidth transferred by downloading or uploading content form the network.
BarterCast lacks a tamper-proof scheme, in fact, it is possible to forge and share custom graphs to manipulate the reputation of a node.
Trustchain was built on the basis of this system, and accomplish the anti-tampering scheme thanks to the entanglement of previous transaction hashes.

IOTA is a popular distributed ledger that works without a global consensus thanks to the Tangle [34]. Users of this system can perform transactions much faster and with no transaction fees.
Every new transaction is linked with two previous approved ones. The process of approving a transaction implies that its history is verified, this makes sure that there is no illegitimate token created. Proof of work is utilized to limit the rate of transactions, to avoid spamming.

Both Trustchain and IOTA works by using the directed acyclic graph (DAG) to store transaction information. The key difference from a blockchain ledger is that instead of operating and agreeing on a single chain, nodes can create new multiple branches in the DAG, being able to achieve in such a way a much higher overall throughput. In the Figure 4.2 are represented abstractions of these two typologies.

The drawback of DAG ledgers is also the reason why it is so fast, consistency can not be guaranteed and this can lead to conflicts in the transactions.
The key to solving this problem is to have nodes to perform a verification before any transaction, check the information from the transaction partner (even against their will), and when the transaction is concluded making sure to disseminate the information properly in the network. In this thesis, we work on finding efficient and effective strategies that can help to achieve this.

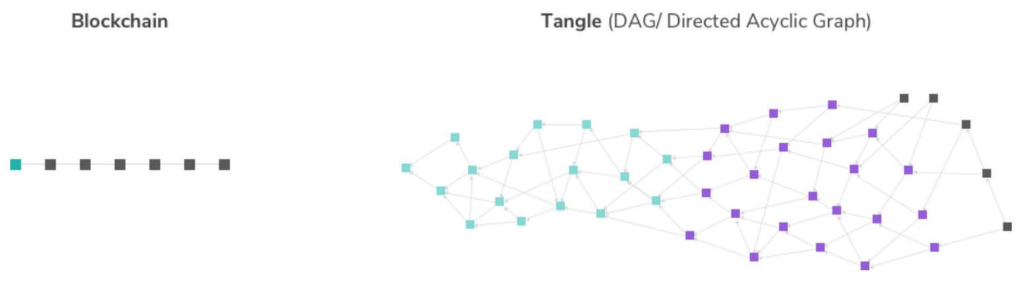


Figure 4.2: Differences between a blockchain and DAG type ledger. Source: IOTA.org

4.3. Anonymous Auditing Systems

Anonymization can be used to guarantee privacy to a node in the network [24], but can also be used as a tool to enhance security.
Such use case is adopted in the Roos et al. malicious peers eviction mechanism for the super-p2p sys-

tems [23]. This mechanism mitigates the effect of routing table infiltration as well as the subsequent denial-of-service attack in super-P2P networks.
The system is constituted of two stages.

- **Proactive Detection**
  It is based on the anonymous auditing scheme introduced by Singh et al. [37], and enforces a limit on the number of connections that peers can establish (size of the routing table). It consists of a quorum that employs a set of anonymizer nodes to query the backpointer set of a node, to determine whether to keep or eliminate that node from the personal routing table.

- **Reactive Detection**
  The reactive stage aims to expel malicious peers by performing a denial-of-service attack. The decision is made by constructing a quorum on the trustworthiness of the suspected peer.

Although the employment of a set of anonymizer helps to detect malicious nodes, it also introduces a new point of vulnerability that was not existing before. The selected set of anonymizer nodes can be composed of malicious nodes.
These can either cooperate with the queried peer, allowing an evil peer to appear benign, or can just drop the requests with the result of excelling an honest node.
From the evaluation section of [23], it is clear how varying the parameters of the anonymized quorum affects the outcome. However, in general, the prospect of removing an honest node from a routing table is way smaller than for a malicious peer.

In this work, we implement a system inspired by the proactive system of this mechanism, and we evaluate how the presence of evil nodes in an anonymization nodes set can drive the outcome to different results.

5
Proposed Solutions

5.1. Broadcast Dissemination

To improve the double-spending detection time it is necessary to increase the probability of detecting contrasting information in a given time. Consequently on the basis of what we discovered in Section 3.4, by increasing the number of nodes disseminated this probability also increase.

The first of the solutions is base on this concept. In this solution, the dissemination is extended to all the nodes present in the network. Through total dissemination, all the nodes will be reached by the dissemination of a transaction two times, or at least one time in case of transactions with evil nodes.

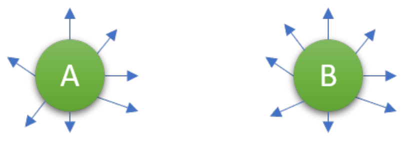


Figure 5.1: In the "broadcast dissemination" solution, the nodes performing a transaction disseminate the its information to the entire network.

How Dissemination Works

The broadcast of the transactions is not done by sending messages directly to the addressee, but rather by sending the information to all the neighbour and expecting them to do the same on their own over and over. Opportune algorithms are in place to avoid loop; A loop is wherever a set of nodes in a particular disposition keeps exchanging the same message again and again.
A direct message approach is not possible since nodes don't have the global imagine of the network (who is in the network and to whom is connected to).

Solution Limitations

If on one side this method seems to improve the detection time on the other side it has limitation in regards to scalability. Complete dissemination is maybe feasible in a network with a couple of hundreds of nodes and unfrequent transactions, but in a large system disseminating to all the nodes can lead to problems.

- **Network Congestion**
  Every pair of nodes transacting creates and send messages to all the nodes, these will travel across all the edges of the network. If multiple transactions are happening concurrently there is a risk of clogging the system. The bandwidth of the connections that connects the nodes is limited and because it's a P2P network there are no regulations on its minimum size.

- **Data Storage**
  Having a broader local knowledge is advantageous for the fact that a larger number of blocks can be verified, but this comes with a cost. Storing all the transactions happening in the network will require larger and larger storage, and chain/block verification procedure will take longer [1].

5.2. Anonymization and Selective Localized Dissemination

In this section are described the architecture design choice and communication protocols behind the anonymous auditing and the selective dissemination.
This approach has some likeness to the limited broadcast dissemination, currently employed in Trustchain, in particularly with regards of how transactions populate ledgers.

5.2.1. The Key Idea

The concept behind the solution relies on being able to ask and check a node's ledger in anonymity, this is not only a deterrent mechanism, as it can discourage malicious behaviours, but can also help to detect double-spending.
Having the identity of the requester node hidden during the chain verification phase is the key idea, because an adversarial node that has cheated will not be able to craft and send back a custom chain [2] that can deceive the receiver, or at least not with a sustainable probability of success.
Imagine being an evil node and receiving a request to view your ledger. Because these requests are anonymized, you don't know who is demanding your chain, and the possibility of performing transactions with multiple ledgers became difficult and risky; sending the wrong ledger to the right node can lead to being detected and punished [3].
Anonymization allows nodes in the network to disguise their identity and perform checking.

5.2.2. Anonymizer Nodes

The function of this node is to be the relay of messages for nodes that want to stay anonymous to the recipient, in such a way that the recipient will not get to know how is really asking for information, and so will not be able to send back tailored customized replies for him.

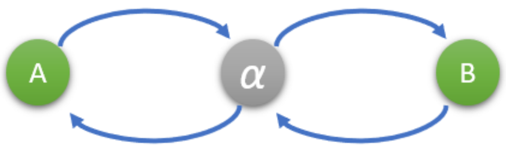


Figure 5.2: How communication is relayed via the anonymizer node. Alfa is representing the anonymizer node.

Relying on a single node to anonymize messages has the same risk as not using one, because the anonymizer node itself can be malicious and/or even under the control of the node you are trying to communicate with. Using a single node opens the doors to vulnerability such as eclipse look-alike attack [21] in which basically a set of evil nodes can prevent a node from being well connected to a network, by eclipsing him and leading him to take biased decisions.
To avoid this the best solution is to use a larger auditing system, consisting of multiple queries performed across several anonymizer nodes selected in the network.
The role of the anonymizer node can be played by anyone except for the node from which we are trying to hide from.

[1] Fetching information from a larger database takes typically more time.
[2] The adversarial node can have two or multiple nodes with all the transaction that can be switched depending on the requester identity.
[3] The procedures post-detection are not part of this work.

How Anonymity is Created

When a node receives a transaction request it initiates the auditing procedure, and so it contacts multiple anonymizer nodes. These will autonomously query the partner, verify the chain and forward the reply back to the requester. All the replies are collected and verified before deciding whether to proceed with the transaction or not.
Nodes not only receive chain requests as a consequence of starting a transaction, but also as a result of the dissemination phase conducted by other nodes in the network (explained in details later). Because of this, it is impossible for a node to exactly determine the reason of the received chain request, therefore, creating a more deeper effective anonymization.

Anonymizer Nodes Selection

The selection of these nodes is very important. There can't be deterministic roles regarding how these nodes are being chosen, because this can be easily exploited by an malicious node that has control on multiple nodes. Therefore the selection is performed uniformly at randomly from a list of potential candidates that every node keeps updated. This list is kept refreshed by a gossiping protocol conducted by every node, instantiated when a node joins the network for the first time and repeated later.

A node participates in the gossiping to be part of other node's potential anonymizer because motivated by a sense of reciprocity for the community. Similarly to what happens in torrents, where people keep their pirated files in seeding despite the risk and the bandwidth consumption. A node chooses to be an anonymizer so that when it is his time to perform a transaction it can make use of this service.

The number of the selected anonymizer can be randomly resolved from an interval so that it varies between consecutive transactions. Having a constant pool size can be exploited by adversarial nodes.

To all the randomly selected anonymizer is sent a message with the target details. Upon receiving this message an anonymizer has to reply with an acknowledgement in order to join the auditing. The purpose of this message is to distinguish anonymizers that are simply offline from the ones that have not yet received a reply from the target (or are malicious). Once the transaction is concluded, all the nodes that were offline are purged from the list of potential anonymiser candidate, these will be able to join again the list by performing gossiping.

Malicious Anonymizers

Evil anonymizer nodes can not modify the reply of the chain as signature are employed on the messages content, but have the capability to collude the questioned node or to simply drop the request messages.

Let's suppose that the K is the number of anonymizers that a node employs for the auditing, and L as a threshold number of these anonymizers that have to replies correctly for the transaction to proceed.
The probability that a set of selected anonymizer nodes is malicious and collude successfully with a potential evil node is quite low [4].
To pass the auditing without the risk of being detected, at least L nodes in the selected set of nodes should collude and reveal the real identity of the requester. We don't need all K nodes to be colluding as the evil node can afford to not reply to at most $K - L$ nodes without any repercussion.
The probability of this event is calculable as:

$$P(E) = \prod_{i=0}^{L-1} \frac{N_{ColludingAnonymizers} - i}{N_{TotalAnonymizers} - i} = \frac{N_{C.A.}}{N_{T.A.}} \times \frac{N_{C.A.} - 1}{N_{T.A.} - 1} \times ... \times \frac{N_{C.A.} - (L-1)}{N_{T.A.} - (L-1)}, K \leq N_{C.A.} \leq N_{T.A.} \tag{5.1}$$

If we compute the probability for a network of 10000 nodes of which 5% are evil anonymizer that wants to collude, and we select $K = 15$ and $L = 10$. In the best case [5], the probability of selecting L colluding anonymizers is only $\sim 8,925 \times 10^{-12}\%$, or also almost as one every a trillion of transactions.
If we increase K or L or the solution is employed in larger networks this probability becomes even smaller.

[4]Anonymizer nodes are chosen uniformly random from a pool of peers which has a size much larger than K.
[5]An adversarial node does not have an idea of what are the selected K and L values by the transaction partner, and we are not considering the probability of this guess in the equation.

In the case where more than $K - L$ nodes that have been selected are evil and decide to drop the anonymization request, the audited node will be considered evil regardless of its nature, because it will gives the perception that it is not willing to provide the chain for verification. This scenario is a bit more challenging to figure mathematically, therefore it will be analyzed more in-depth later in the evaluation section.
However, in these false-positive cases (when a benign node is reported as evil), the system will eventually find that there is no reason to punish this node. So these attacks do not produce real damage other than making losing time to the transacting parties and creating overhead on the resolution of the double-spending claim. The resolution procedure for double-spending attacks (post-detection operations) is out of the scope of this thesis.

Problematics similar to the one addressed here can also be found in [23], where they have been analysed more thoroughly.

Quantity of Anonymizers

As introduced before, it is important to have multiple anonymizers, but how many is enough? Well, the right answer is that it depends because it is in relation to the number of evil nodes in the system. If we have no malicious node this number can even be one because we can trust everyone, but if we have any evil node this number of nodes matters.
We don't know how many evil nodes there are in a network, so we can't have a prior fixed number.
Having this too small can increase the risk of selecting evil nodes, however, the whole transaction will be quick as you don't have to wait for replies from many nodes. On the other hand, a big number is more secure but it will have a larger cost to pay in terms of time that is lost before a transaction can be concluded.

Auditing Unlinkability

A fundamental factor is the timing with which chain requests are sent to the anonymizer and ultimately the target. These requests should look as random as possible to the target node and not linkable to each other. For this reason, is important to scatter these request randomly in time.
Imagine a situation where a node contacts another one to start a transaction, and just an instant after it gets flooded by transaction requests from different nodes. It would be easy for him to understand that those requests are all from the same node, and so preparing a custom reply is not that challenging anymore.
If we randomly distributed in time the auditing process, the targeted node will have a hard time finding a correlation between the requests, as these will starts to blend and appear more like to nodes that are asking for ledgers during their dissemination phase (more info on this later).
The drawback of an auditing well spread in time is that this will considerably increase the time before a transaction can be completed.

Taking The Decision

When a node receives all the results from the anonymizer nodes, it needs to decide whether to proceed with the transaction or not.
Every time a node receives a reply, this is verified and stored in a table. This process is repeated until the node gets at least L replies (or until a timeout triggers) before confronting the results [6] and taking a decision.
If there are at least L verified replies and they are all reporting the same content, the auditing can be considered successful and the transaction can continue. In any other case, the transaction is annulled and the audited peer is reported as malicious.

5.2.3. Transaction Protocol

In this section are explained all the steps that are part of a transaction using anonymizer nodes.
A transaction event is executed by performing these phases:

1. Transaction request

[6] An evil node can provide to some anonymizer a ledger in which are missing the last n transactions. The verification of the individual chain might result successful, therefore is necessary to compare all the outcomes among each other.

2. Chain verification
3. Confirmation
4. Dissemination

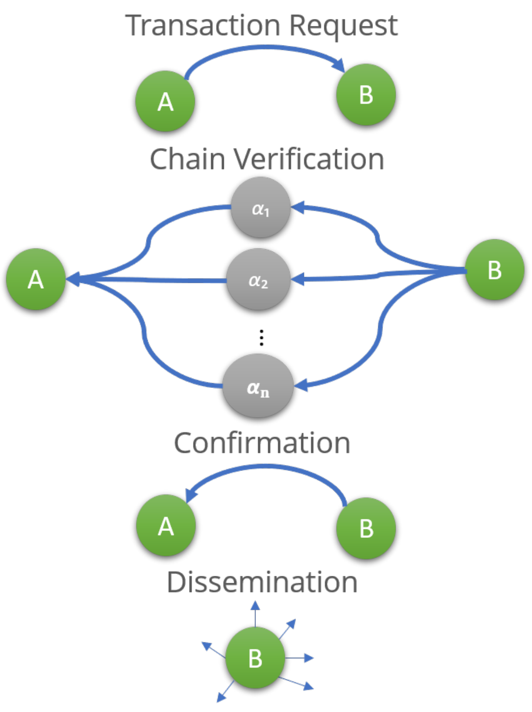


Figure 5.3: The steps involved in a transaction using anonymiser nodes.

Transaction Request

When a node in the network wants to start a transaction with another one, the node that wants to send money needs to create a "Transaction Request" message.
This message contains the identity of the node itself and the details regarding the transaction amount, along with parameters that can be used to validate the message such as a signed hash of the message itself, however, these parameters are identically to the one in the Section 2.2.1.
This message needs to be sent to the node that has to receive the amount. The recipient will reply with a "Busy" notification message if he is busy in another transaction or perhaps not willing to perform the affair (evil node). If the recipient is available and incline to transact it will initiate the next phase, chain verification protocol.

Chain Verification

The recipient will now randomly selects nodes that will audit the transaction partner's ledger. This operation is requested through special messages sent to the inquired anonymizer nodes, these contain the nominative of the target. As described in the previous section these nodes will contact the target, and request its chain before sending it back to the commissioner.
At this point, the transaction recipient will analyze the replies from the anonymizers, and based on the outcome decide how to behave.

Confirmation

If there are no issues with the received ledger the node will compute the transaction and store the newly generated block in his chain (procedure similar to the one actuated in Trustchain), at this point it will create a

direct connection with the node that started the transaction and replies to him with a transaction confirmation message.
This message contains details regarding the new block as well as security-related parameters to validate the reply once received. (These will not be taken in consideration in this paper.)

Dissemination

Once the confirmation message is sent the node will initialize the last phase of the transaction, called dissemination. This phase is the most important with regard to detecting double-spending, because it will help to distribute the newly created transaction in the network, so that it is possible to find issues by spotting differing information in the system.
The node will now contact all the nodes in the partner chain [7] using an anonymizer node, and ask them for their chain in order to verify it, and once that has been done it will send back to them his recently transaction.

A powerful evil node can create many dummy transactions in his chain, made with many nodes under his control, just for the scope of slowing down the dissemination phase for his victims. To avoid this scenario, during this phase nodes are picked in random order, instead of contacting them sequentially.

The dissemination is slightly smarter than the one performed in Section 2.2.1 because it utilizes the locality of reference to intelligently distribute the information in the network where it is really necessary.
The locality of reference in computer science is the tendency of a processor to access the same region of memory in a defined period of time [39]. Transactions in the real world also obey an similar trend [29], people regularly buy grocery from the same place, buy stuff from the same website and have interaction with the same people because we are a sedentary organism and we have a semi-constant routine in our life, behaviours which leads to low-entropy transaction history.

Figure 5.4: Network of people which shape and form is catheterized by human behaviours.

5.2.4. System Improvements

Compared to the previous solutions this architecture has these benefits:

[7] Nodes that has done transaction with him.

- **Dissemination Efficiency**
  Most of the transactions in the chain will most likely to be conducted with the same set of nodes and this means that the dissemination process will be quick because these will be contacted only once.

- **Detection Speed**
  The new dissemination phase has a restricted announcement area, focused on nodes that are most prone to detect anomalies. This in combination with the previous point allows quick detection of contrasting information in the system.

- **Network Load**
  Compare to the global dissemination here the dissemination is done only by one side and on a restricted area of nodes, this means that the total network footprint and load are minor and contained to only where is most effective.

- **Offline Reliance**
  Thanks to the ledgers and transactions information dissemination, caused by the anonymous auditing and the dissemination phase, it is possible to detect anomalies even when the perpetrators are offline. This characteristic is comparable to the one obtained with global dissemination, but without having to pay the cost for it.

5.2.5. System Limitations

This solution is much more efficient in comparison to the previous one but it has some limitation:

- **Variation of Dissemination Effectiveness**
  The detection of a double-spending has a high probability of happening in a short amount of time in instances where a node has done many transactions before starting to misbehave. This because having a large variety of nodes where the benign node can disseminate, reduce the possibility of finding offline or not working nodes.
  Let's think an extreme scenario where the first transaction of an evil node is a double-spending; Here the victim can't disseminate to anyone and the second node will be able to disseminate solely to the first one (and detect the attack), but only if he is online and working.
  Therefore the effectiveness of the dissemination is highly influenced by the number and status of the nodes in the evil node chain.

  A possible solution to this problem is to persist with the dissemination with nodes that are not available until it is possible to reach them; This, however, is a very time-consuming operation and can limit the scalability of the architecture.
  Alternatively, it is possible to implement a solution similar to the watchtowers, conceptualized within the original Lightning Network paper [33]. Design network entities that stay online 24/7, and has the task to be the realtor of chain request and dissemination for those who aren't able to stay constantly online.

- **Scalability Limitation**
  The biggest limiting factor of this solution concerning the scalability is to be found in the dissemination phase.
  The dissemination process considers all the unique nodes in the partner chain (nodes with whom he has transacted in past) and contacts all of them one by one. This operation is doable in a reasonable amount of time if there are just dozens of nodes, but it becomes challenging if the number starts to increase.
  Imagine if the partner ledger contains transaction with thousands of different nodes, reaching them and waiting for a reply not only will take a lot of time but also network bandwidth. Finding nodes that have done so many transactions is totally reasonable, considering that this system has to store information for an indefinitely amount of time.
  With time performing transaction will become safer, but the drawback is that it will take longer and longer to complete a transaction.

A potential solution to solve this problem is to limit the dissemination to only recently transacted nodes, by defining thresholds (the time when transactions occur is registered in the chains' blocks). There is a need for two thresholds, the first one will define how old a transaction can be to still be used, and the second one the minimum amount of nodes that are opportune to disseminate before taking in consideration the first threshold. The second condition is designed to avoid exploitation of the first threshold by nodes that do transactions rarely.
Disseminating to more nodes is beneficial, but the added security gain starts to decrease. The reason for the diminishing returns is due to the fact that the probability of finding a node with contrasting information in dissemination, is not linearly dependant on the number of nodes reached.
To understand why, we need to analyze how this probability works. Will call $P(C)$ the probability of finding contrasting information, and this can be defined as:

$$P(C) = 1 - P(none) = 1 - (\frac{N_{tot} - N_{alfa}}{N_{tot}})^{N_{tot}} \tag{5.2}$$

where,

$$N_{tot} = \text{Total number of nodes reached}, N_{tot} > 0 \tag{5.3}$$

$$N_{alfa} = \text{Number of online nodes with contrasting information}, N_{alfa} > 0, N_{alfa} \subset N_{tot} \tag{5.4}$$

N_{alfa} can be written as a constant multiplying N_{tot}, which therefore can be simplified with the denominator. Because of this, the second argument of the function can be seen as an exponential of a fraction, therefore it is possible to write this:

$$P(C) \approx 1 - (\frac{1}{\gamma})^n \tag{5.5}$$

If we plot the function we obtain the graph in Figure 5.5. From this graph, it is visible how after a certain number of nodes disseminated the increase in probability starts to become insignificant.

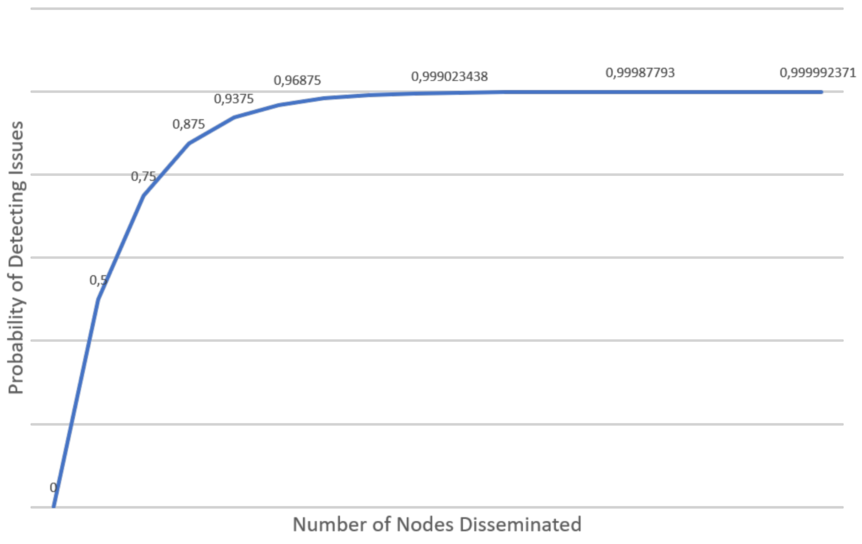


Figure 5.5: Graph showing the law of diminishing return.

6 Performance Evaluation

In order the evaluate the performance of the proposed solutions against the current system, we developed three simulations.
Simulations were chosen over emulations because of the faster development time, the ease with which it is possible to run experiments in different scenarios and the ability to run the systems on larger networks. These characteristics are a consequence of the fact that simulations are not replicating all the internal mechanisms of the system that is reproduced.
Emulations, on the other hand, are more detailed and can offer results that are more close to the one obtainable by a real implementation.

In the designed simulations there has been implemented a version of Trustchain without the usage of hashing or signature. The aspect that we are analyzing this in this research does not rely on nor are (directly) affected by the use of those components. By not implementing them helped to speed-up development and achieve faster and larger simulations run[1].

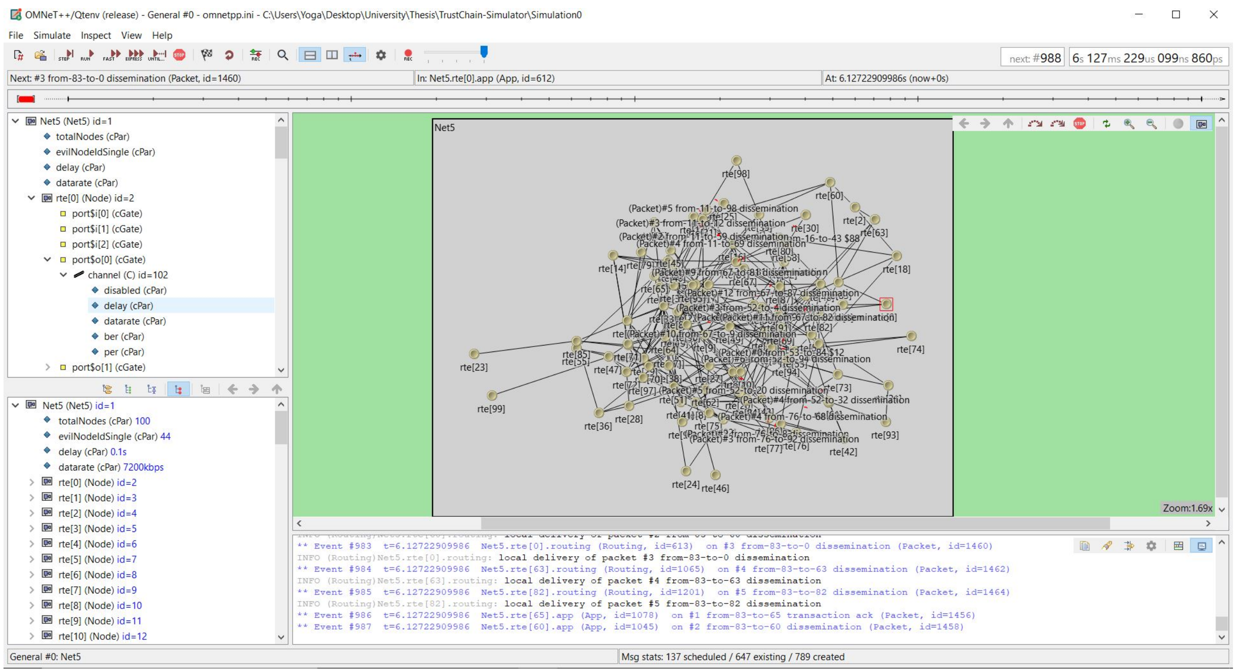
All the files and programs referred in this section can be found on the thesis's GitHub page: `https://github.com/Tribler/trustchain-simulator`.

6.1. Evaluation Environment

OMNeT++

Simulations have been developed using the OMNeT++ framework, a component-based C++ simulation environment.
The reason behind its adoption was guided by many factors, like its ease of use and its popularity (which makes troubleshooting easier). Another major advantage that OMNeT++ offers is the fact that it can execute simulations with a detailed UI or just via command line. This feature is particularly helpful during initial development, to easily debug problematics, thanks to its visual representation of nodes/edges and messages transferring.
This framework has also a powerful profiling tool that allows efficient network's messages analysis.

[1]The term "Run" can be used to indicate a specific execution of a experiment. When evaluating a system we execute multiple runs for every experiment (every run has a different random seed).



```
OMNeT++/Qtenv (release) - General #0 - omnetpp.ini - C:\Users\Yoga\Desktop\University\Thesis\TrustChain-Simulator\Simulation0
Next: #3 from-83-to-0 dissemination (Packet, id=1460)   In: Net5.rte[0].app (App, id=612)   At: 6.12722909986s (now+0s)
next: #988   6s 127ms 229us 099ns 860ps

Net5 (Net5) id=1
    totalNodes (cPar)
    evilNodeIdSingle (cPar)
    delay (cPar)
    datarate (cPar)
  rte[0] (Node) id=2
      port$i[0] (cGate)
      port$i[1] (cGate)
      port$i[2] (cGate)
      port$o[0] (cGate)
        channel (C) id=102
          disabled (cPar)
          delay (cPar)
          datarate (cPar)
          ber (cPar)
          per (cPar)
      port$o[1] (cGate)

Net5 (Net5) id=1
    totalNodes (cPar) 100
    evilNodeIdSingle (cPar) 44
    delay (cPar) 0.1s
    datarate (cPar) 7200kbps
  rte[0] (Node) id=2
  rte[1] (Node) id=3
  rte[2] (Node) id=4
  rte[3] (Node) id=5
  rte[4] (Node) id=6
  rte[5] (Node) id=7
  rte[6] (Node) id=8
  rte[7] (Node) id=9
  rte[8] (Node) id=10
  rte[9] (Node) id=11
  rte[10] (Node) id=12

** Event #983  t=6.12722909986  Net5.rte[0].routing (Routing, id=613)  on #3 from-83-to-0 dissemination (Packet, id=1460)
INFO (Routing)Net5.rte[0].routing: local delivery of packet #3 from-83-to-0 dissemination
** Event #984  t=6.12722909986  Net5.rte[63].routing (Routing, id=1065)  on #4 from-83-to-63 dissemination (Packet, id=1462)
INFO (Routing)Net5.rte[63].routing: local delivery of packet #4 from-83-to-63 dissemination
** Event #985  t=6.12722909986  Net5.rte[82].routing (Routing, id=1201)  on #5 from-83-to-82 dissemination (Packet, id=1464)
INFO (Routing)Net5.rte[82].routing: local delivery of packet #5 from-83-to-82 dissemination
** Event #986  t=6.12722909986  Net5.rte[65].app (App, id=1078)  on #1 from-83-to-65 transaction ack (Packet, id=1456)
** Event #987  t=6.12722909986  Net5.rte[60].app (App, id=1045)  on #2 from-83-to-60 dissemination (Packet, id=1458)

General #0: Net5   Msg stats: 137 scheduled / 647 existing / 789 created
```

Figure 6.1: User interface of the OMNeT++ framework. In the image, there is in execution a simulation with a network with 100 nodes.

DAS-4

DAS-4 is a network of supercomputers across the Netherlands, a joint effort of multiple universities and research institutes.
Every computational unit in the cluster has a processor with two or four cores, with memory sizes that vary from 24GB to 48GB of RAM. Some of the nodes have also a graphics card. The operating system running on the whole architecture is CentOS Linux. Special head-nodes are utilized to load up the program and schedule their execution, there is one of them per each physical location of the nodes.
Following the development of the OMNeT++ simulations and its tuned up, all its configurations were loaded to a head node in DAS-4, and then thanks to scripts, automatically loaded into server's execution queue.
The outcome of the simulations have been parsed and processed automatically by programs written respectively in Java and R.

Unfortunately because of the incompatibility of the core architecture of OMNeT++, it was not possible to utilize the clustering capability of DAS, but only a single node per each run. This limited the maximum simulations network size.

6.2. Simulations

There are in total three simulation systems designed for this research and are defined as follow.

- **Simulation0 / Solution0**
  This system is an implementation of the solution currently used in the Trustchain architecture emulated on Tribler.

- **Simulation1 / Solution1**
  In this simulation is implemented the first proposed solution, the total broadcast system.

- **Simulation2 / Solution2**
  Here is implemented the anonymization and selective localized dissemination scheme.

6.3. Adversary Model

After the design and testing of three simulations mentioned in the previous section, the next step was to evaluate their performance, to do so it was necessary to have nodes with malicious behaviour.
In this section, we are going to define the adversary model. This model is created on the base of the reasoning that we did in Section 3.2.

These are the assumptions made on the adversary model performing double-spending:

- Any node in the network can be evil.
- There can be multiple nodes assuming evil behaviour in a simulation.
- Nodes have a non-adaptive behaviour. During a simulation, the status of a node (benign or malevolent) is determined during initialization and it is constant for the duration of the execution.
- The evil nodes selection can be done manually or uniformly at randomly in the simulation setup.
- Adversarial nodes do not collude and nor have the power to control other nodes in the network.
- The only goal of the adversarial node is to perform double-spending as defined in Section 3.2.
  DOS (denial of service) [27] or other types of attacks are not being considered in this thesis.
- A node assumes malicious behaviours only after the triggering point, before that it acts just like a normal node. The triggering point is defined by a number of normal transactions completed before double-spending. This feature is used to assert the effects of attacks caused by nodes having transaction records of different length (prior to the attack).

These are the assumptions made on the malicious anonymizer nodes: (FOR THE **SIMULATION2** ONLY)

- There can exist malicious anonymizer nodes, these nodes accept to be part of the auditing but then drop the chain lookup demand, as if the audited peer never replayed back to a chain request.
- The probability for an anonymizer node to be evil in auditing request is decided a prirori as a simulation's configuration parameter. This probability remains constant for all the nodes for the duration of the simulation.
- Malicious anonymizers do not collude and nor can control other nodes in the network.

6.4. Performance Metrics

To evaluate the performance of different architectures it is necessary to identify metrics, that can be measured and confronted on a large variety of simulation runs. For this research, we identified two parameters.

Double-spending Detection Time

Right after the first part of double-spending is completed [2], the simulator starts a timer that is stopped when the evil node is detected. This timer will report in such a way how long it takes to detect an attack.
We decided to start measuring the time right after the first malevolent transaction because, by not registering it, the evil node has already committed an infraction that will lead soon or later to double-spending.

Type of Detection

This measurement can provide a ratio between the number of runs in which the attack is detected early (during the initial chain verification) and later during dissemination.
It is not as significant as the fist metrics, but can provide a bit more information on how the solutions perform.

[2]A double-spending attack is composed of two transactions. A first one not registered and a second one where the value is spent.

Number of Messages

The number of messages (or bandwidth consumption) can be used to determine the efficiency of the proposed solutions. This metrics will be calculated mathematically and not measured using simulations.

Let's determine approximately how many messages are sent across the network with all thee solutions in a transaction.

$$\#\_Messages_{Solution0} \approx 44 \tag{6.1}$$

$$\#\_Messages_{Solution1} \approx 4 + 2 \times N_{Netowrk'sEdges} \tag{6.2}$$

$$\#\_Messages_{Solution2} \approx 2 + 5 \times N_{Anonymizers} + 5 \times N_{NodesDisseminated} \tag{6.3}$$

To understand the difference, let's suppose we perform a transaction in a network of 5k nodes (5k edges), and for the Solution2 we employ 10 anonymizer nodes and disseminate to 50 nodes. With these assumptions, we obtain that Solution0 uses 44 messages, Solution1 a whopping 10k messages and Solution2 only ~ 300 messages.
Even if we consider the fact that in Solution2 some of the messages are larger (as they containing the chain) and we include that anonymizer nodes employ a low-frequency gossiping, the general load on the network remain significantly smaller if compared to Solution1. With a larger networks, this difference becomes more and more prominent.

6.5. Solutions Evaluation

6.5.1. Evaluation Structure

It is important to evaluate the performance of the systems in different situations, we can accomplish this by modifying the parameters of the simulations.
We evaluated the three simulations under several circumstances divided into several sets. Every set is also composed of multiple experiments. Sets are differentiated by changes of a parameter and the experiments by its variation.

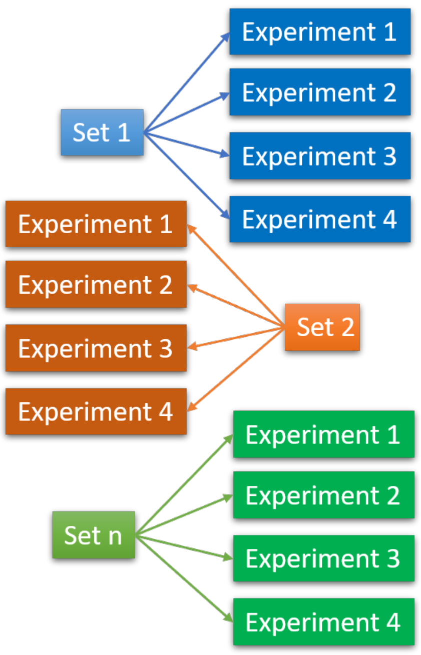


Figure 6.2: Simulations have been evaluated across different network/nodes configurations. These can be divided in sets of multiple experiments.

The sets can be divided into three main categories based on the configuration changes.

- **Communication Effects**
  Variation of simulator's parameters that affect the way nodes communicate with each other. Such as transmission speed and delay.
- **Nodes Behaviours Effects**
  Modifications of nodes behavioural related parameters. For example, the frequency with which a node begins a new transaction, or the number of benign transactions before the evil agent perform a double-spending attack.
- **Network Topology Effects**
  Changes that affect the network structure used in the simulations. For example, but not limited to, the number of nodes, as well as the number of evil nodes, the topology of the network.

6.5.2. Evaluation Parameters

Now, are explained in details all the sets utilized to evaluate the three solutions.

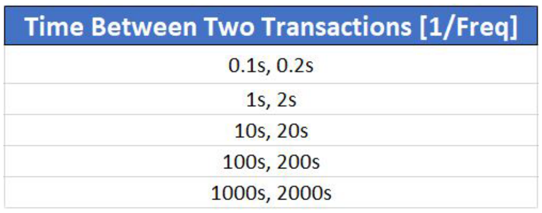
- **Number of Nodes**
  Utilizing the Erdos-Renyi model (explained in details in Section 6.6.1), we created different networks having various sizes. We evaluated the solutions using these to see how the size of the network affects the results. There have been created networks with the following sizes 10, 20, 50,100, 500, 1000, 10000 nodes.
- **Number of Evil Nodes**
  To analyze how the systems behave in situations where there are multiple evil nodes, we run simulations by varying the number of these. We started from one, double it to two, 10 and 20. According to our adversary model, multiple evil nodes do not cooperate but perform attacks individually. Any solution should stand against these un-coordinated multiple attacks without any problem.
- **Bandwidth**
  The proposed solutions will operate in a P2P network, where there are no boundaries regarding the

bandwidth between the nodes. Therefore there can be networks with very low transmission speed and others with higher. We evaluated the system with these speeds: 10kbps, 100kbps, 1000kbps, 10000kbps, 5000000kbps and 1000000kbps to examine the solutions behaviours.

- **Transmission Latency**
  The latency is another parameters that along with bandwidth can vary a lot. We evaluated the solutions with networks having 1ms, 10ms, 100ms, 1000ms and 10000ms communication delay.

- **Evil Initial Harmless Transactions**
  Evil nodes can attack right after joining the network or can do it after a while. To find what happens we experimented different scenarios.
  With evil nodes that misbehave right after a single transaction, 2, 3, 8, 16 and up to 32 transactions. Misbehaving right away it is not possible as at least a transaction [3] is needed to be able to double-spending.

- **Interaction Frequency**
  The frequency with which nodes interacts can influence the way information is spread in the network, and therefore also the detection time. We vary the frequency of transactions that every node does from a very high value up to a small one, to see how this relates to the detection time of the varies simulations. To avid that all the nodes start a transaction at the same time, we defined ranges of time of which every node uniformly select its own. These can be found in Figure 6.3



Time Between Two Transactions [1/Freq]
0.1s, 0.2s
1s, 2s
10s, 20s
100s, 200s
1000s, 2000s

Figure 6.3: Periods of time between two consecutive transactions used to evaluate the proposed solutions.

- **Average Path-Length**
  Utilizing the Watts-Strogatz model (explained in details in Section 6.6.2) we created networks having different average path lengths. A highly connected graph allows an easier spread of information because to reach any other node in the network there are fewer hops to across. We generated networks having an average path length of 1, 2, 5, 10, 100.

- **Variation of Anonymizer Nodes** (FOR THE **SIMULATION2** ONLY)
  As we saw in Section 5.2.2 the anonymizer nodes selection process can be problematic if not done properly. To evaluate how the solutions reacts to different quantities of anonymizer nodes, we varied its value and run multiple experiments. Starting with just a single anonymizer, to 10, 100 and up the max available [4].

- **Evil Anonymizers Effects** (FOR THE **SIMULATION2** ONLY)
  In Section 5.2.2 we considered how the selection of parameters K and L plays an important role in the anonymized auditing mechanism. With this set of experiments, we want to establish the effects that evil anonymizer nodes can produce in relation of different amount of evil nodes, K and L values.
  The simulations will count the number of transactions in which a node is reported malicious because of evil anonymizers, and will suspend simulation after reaching 10000 transactions so that a percentage can be calculated. There is no malicious node trying to perform double-spending in these simulations as we want the results clear out of this variability.
  We evaluated the solution with a concentration of 5%, 30%, 50% and 80% of evil anonymizer nodes

[3]The transaction of which the evil node does not store the details in the chain.

[4]The max value is not always equal to the number of nodes in the network(-2). But it depends on the size of the list of anonymizers that every node keeps updated thanks to the auditing process.

(peers that drop the anonymization request). K was constant at 30 nodes [5], but L has been varied from 30%, 50%, 80% and 100% of K.

The complete list of parameters used for the sets can be found in the appendix.
These configurations have been chosen because resembling real-life networking situations or were producing interesting outcomes during simulations. Among all the default values chose for the experiment the following are particularly worth to mention.

- **Transmission Speed**
  It has been chosen to utilise a transmission speed of $7.2Mbps$ for all the simulation sets that were not varying this parameter, this value resembles the global average internet speed [9][10].

- **Transmission Latency**
  The latency is a network measurement that is quite volatile, we opted to use $100ms$ as default value after analysing statistical reports of different international networking companies [6][7][8].

6.6. Networks Topologies

One of the most important factor while evaluating the performance of a system is the environment in which this is experimented. In our case, this is denoted by the network structure in which nodes are running the different solutions.
The network structure plays a very influent role in the results of the system. For example, a highly connected network will disseminate messages quickly, where a more disassociated one will take longer.
We evaluated the solutions utilizing various networks, generated with Erdos-Renyi [19] and Watts-Strogatz [38] models.

6.6.1. Erdos Renyi Model

In graph theory, Erdos-Renyi [19] is the most popular random graphs generation model. There are two variants of this model, $G(n,M)$ and $G(n,p)$. In this work, we utilized the latter one.
In the first one, a graph is chosen uniformly random from all the possible graphs having n nodes and M edges.
In $G(n,p)$, a graph is generated by creating edges at randomly, each link has a probability p of existing and connecting two nodes.

$$\text{Average number of edges} = \binom{n}{2}p \tag{6.4}$$

By varying p we can modify the degree of connection of the nodes of the graphs.

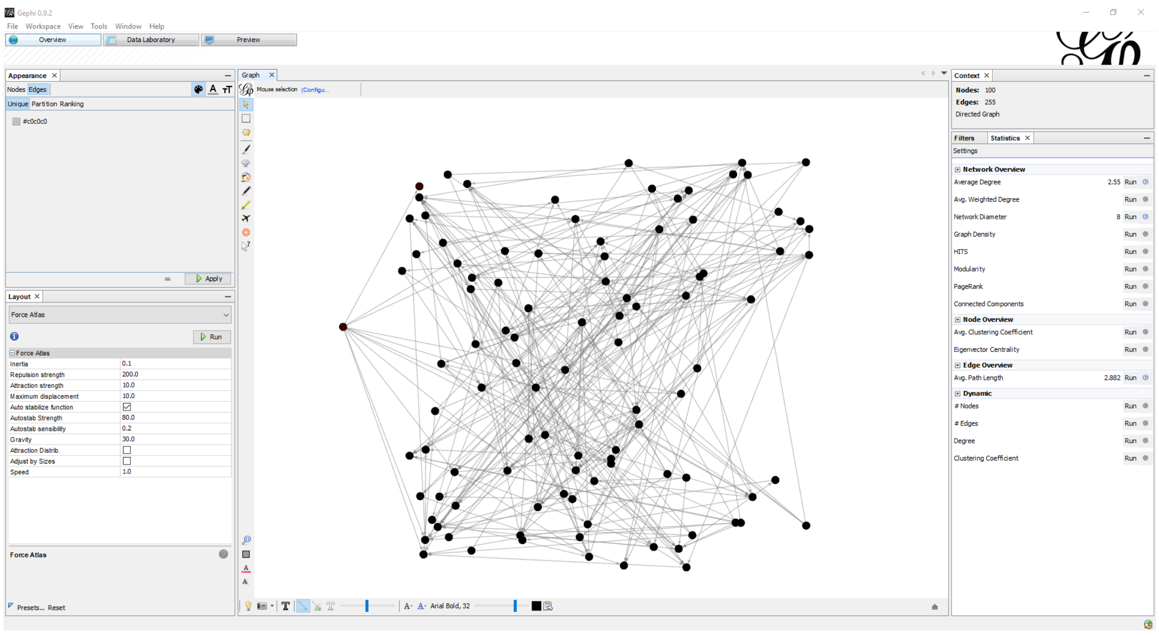
- $p = 0$
  With this value, we obtain graphs that are not connected, composed only of isolated nodes.

- $p > \frac{(1+\varepsilon)ln(n)}{n}$
  By increasing the probability over this threshold we can achieve almost surely connected graphs, with no isolated vertices.

- $p = 1$
  All the edges in the network exist, therefore we obtain fully connected graphs. Graphs in which every node is connected with an edge to every other node.

To generate the networks we utilized a program called Gephi, with the value of p approximately equal to $\frac{ln(n)}{n}$.
We created networks of various sizes, as described in the Section 6.5.2.

[5]We evaluated with 30 anonymizer nodes as this number that is large enough to allow L to be small as 30%, and still be able to have a sufficiently large pool. A larger amount of anonymizer will decrease the probability of a false-positive report, but this will not resemble a real-world application. More anonymizer also mean more time required for a transaction competition.



Force Atlas

Force Atlas	
Inertia	0.1
Repulsion strength	200.0
Attraction strength	10.0
Maximum displacement	10.0
Auto stabilize function	☑
Autostab Strength	80.0
Autostab sensibility	0.2
Gravity	30.0
Attraction Distrib.	☐
Adjust by Sizes	☐
Speed	1.0

Nodes: 100
Edges: 255
Directed Graph

Network Overview		
Average Degree	2.55	Run
Avg. Weighted Degree		Run
Network Diameter	8	Run
Graph Density		Run
HITS		Run
Modularity		Run
PageRank		Run
Connected Components		Run
Node Overview		
Avg. Clustering Coefficient		Run
Eigenvector Centrality		Run
Edge Overview		
Avg. Path Length	2.882	Run
Dynamic		
# Nodes		Run
# Edges		Run
Degree		Run
Clustering Coefficient		Run

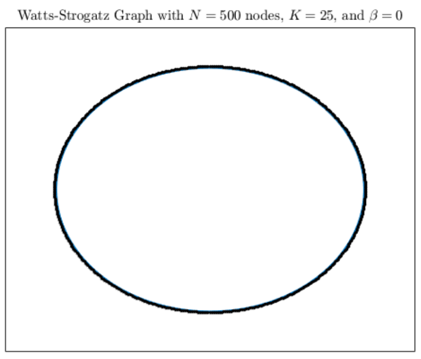
Figure 6.4: Creation of a 100 nodes graph utilizing Gephi and the Erdos-Renyi model.

6.6.2. Watts-Strogatz Model

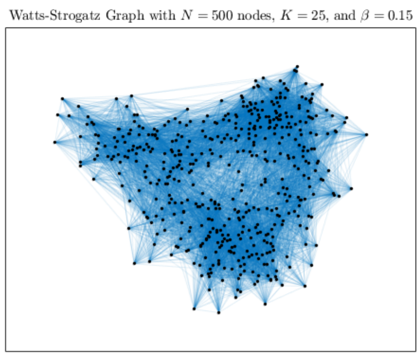
To evaluate the solutions with different average path-lengths we utilized graphs based on the Watts-Strogatz model [38]. This model is a random graph that has small-world network properties, such as clustering and short average path-length.
Let's consider N as the number of nodes that we want in the graph, the mean degree as $2K$ and a special parameters β that satisfies $0 \leq \beta \leq 1$, with these components creating a Watts-Strogatz can be done with the following two steps.

- **1) Ring Lattice**
  Creation of a ring lattice with N nodes having each $2K$ connections. Each node is connected on either sides to its K nearest neighbours.

- **2) Randomization**
  For each connection in the graph, rewrite the reached node with probability β. The substitute edge can not be a self-loop or a duplicate.

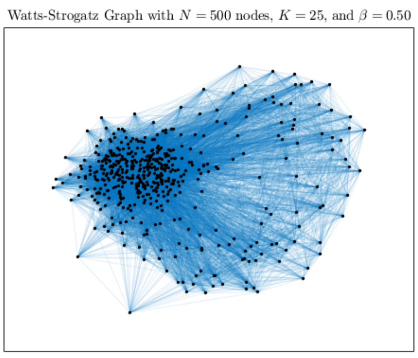

Watts-Strogatz Graph with $N = 500$ nodes, $K = 25$, and $\beta = 0$

(a) Graph with $\beta = 0$

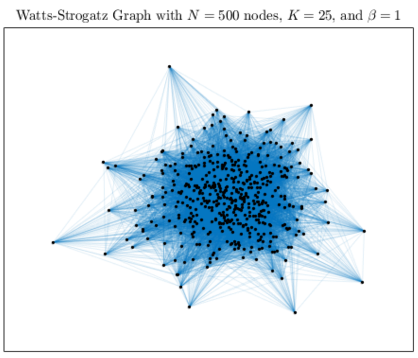

Watts-Strogatz Graph with $N = 500$ nodes, $K = 25$, and $\beta = 0.15$

(b) Graph with $\beta = 0.15$

Figure 6.5: Realizations of graphs based on Watts Strogatz model with small β values - Source Mathworks.com


Watts-Strogatz Graph with $N = 500$ nodes, $K = 25$, and $\beta = 0.50$

(a) Graph with $\beta = 0.5$


Watts-Strogatz Graph with $N = 500$ nodes, $K = 25$, and $\beta = 1$

(b) Graph with $\beta = 1$

Figure 6.6: Realizations of graphs based on Watts Strogatz model with large β values - Source Mathworks.com

As visible from the Figure 6.5 and Figure 6.6, the variation of the β changes the structure of the graph. From an initial ring structure with a small β, we can generate a completely random graph by increasing to its maximum value of 1.0. This rewires all the initial edges to new nodes.

By manipulating the values of K and β it is possible to generate graphs with various average path-lengths (APL).

- $\beta = 0$
  APL = $\frac{N}{2K} >> 1$

- $0 \leq \beta \leq 1$
  The APL decreases very quickly with increasingly values of β, quickly reaching its limiting value.

- $\beta = 1$
  APL = $\frac{lnN}{lnK}$

We employed this property to create the networks utilized for an evaluation set described in Section 6.5.2.

6.6.3. Tribler's Network Structure

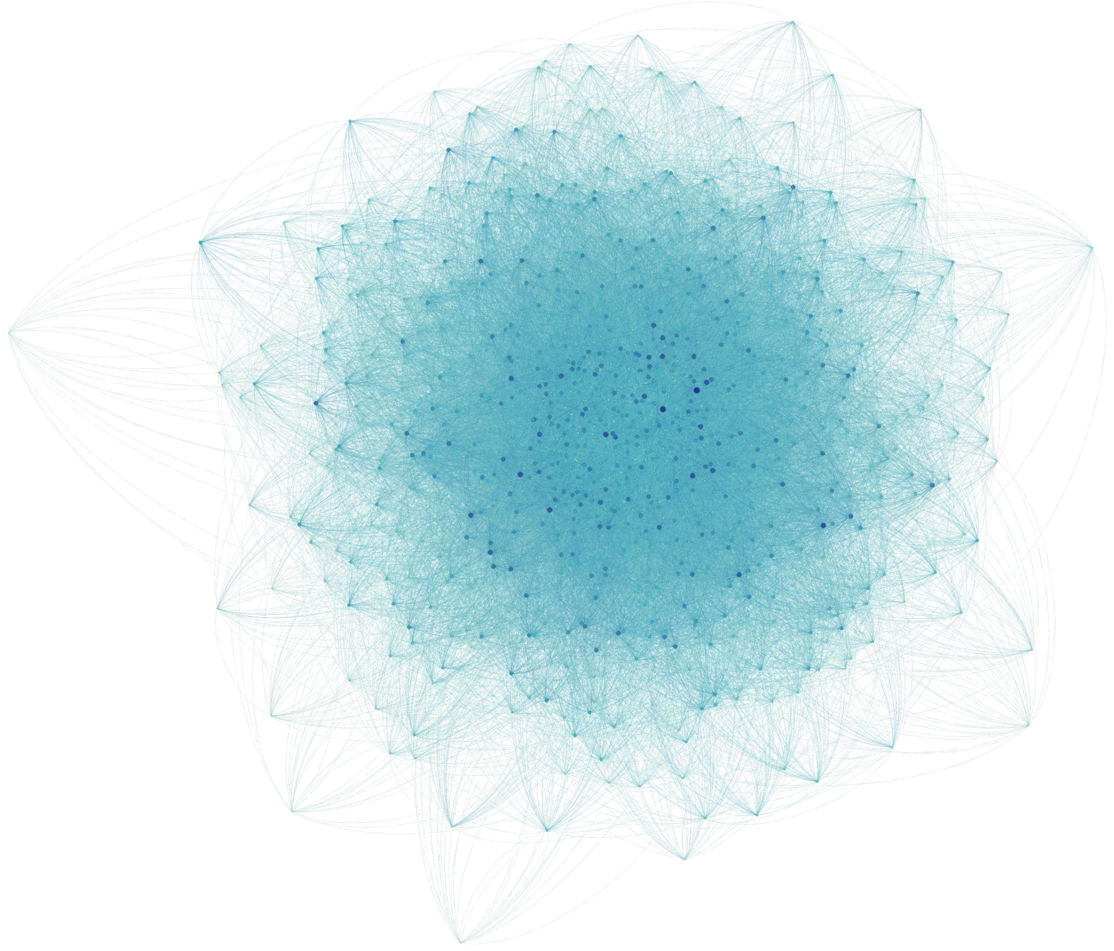


Figure 6.7: The structure of the network where Tribler operates. The darker nodes have a higher degree of connection. Representation created with Gephi.

The current emulation system for Trustchian operates on the network used by Tribler. Thanks to the help of Bulat N. (a PhD student working on Trustchain) I was able to have access to a model of a section of this network, a model composed of 999 nodes and over 29000 edges. It would have been best to have a complete model as more representative of the real network [6], however this would have been challenging to use [7]. And even if this model was copying the whole network, there are still problematics with it as the Tribler network is dynamic, therefore taking an image at different times will result in different models.

We evaluated all the solutions on this network (visible in Figure 6.7) to evaluate their performance in a pseudo-real environment.

6.7. Results

In this section, we are going to examine the results collected from more than 30000 evaluation runs, explained in Section 6.5.2. We will analyze the differences or/and the trends of simulations outputs, with the scope of the assessing the three solutions performance.
For ease of readability, the results are grouped by Communication Specification Effects, Nodes Behaviours Effects and Network Topology Effects.

[6]A section of a network might not have the same properties as the original one. Imagine having a section composed of just nodes that have a low degree of connection.
[7]Due to computational limitation the simulator is not able to handle large networks.

6.7.1. Statistically Significant Differences

Some statistics knowledge is needed to understand the analysis of results, for this reason in this section we describe some of the notions utilized.
Executing multiple [8] runs per experiment and then calculating the mean value (Formula 6.5) does not provide precise information on how an experiment will always perform, but rather an approximate value.
Imagine executing just three runs of a experiment, there is no guarantee that the average of this small sample is equal to the mean value of the population, especially if there is a lot of variances [9] in the results.

$$\overline{X} = \frac{\sum Xi}{n} \tag{6.5}$$

By increasing the sample size (hypothetically to infinite runs) will get that at every new iteration the average of the detection times calculated up to that point will get close and close to the population's mean value. However, these two values are asymptotic because it is not possible to perform infinite runs.

The solution is to express the calculated average along with an error. Researchers usually choose alpha levels of 5% or lower which translates to confidence levels of 95% or greater. This means every experiment's result will be represented as the average value followed by an upper and lower margin, as can be seen in Figure 6.8. The higher is the number of runs the small this margin will be around the centre value.

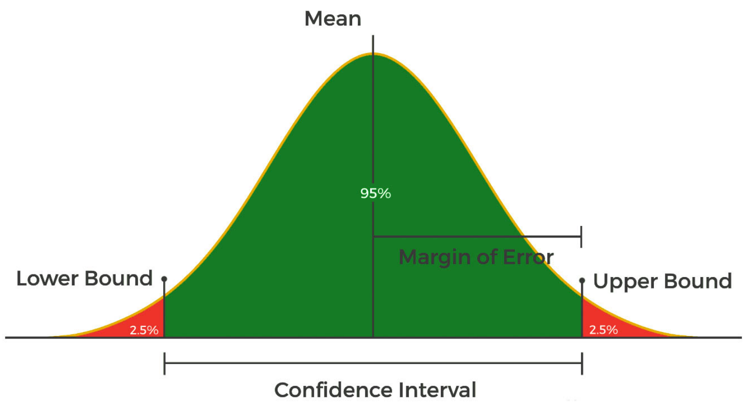


Figure 6.8: The confidence interval is delimited by the mean value and the upper and lower bounds.

To calculate the margin of error you need to use the following formula:

$$\text{Margin of Error} = Z \times \frac{\sigma}{\sqrt{n}} \tag{6.6}$$

Where Z is the Z-Score, n is the sample size (number of runs) and σ is the standard deviation and it is defined as:

$$\sigma = \sqrt{\frac{\sum(\overline{X} - Xi)^2}{n-1}} \tag{6.7}$$

With the margin of error calculated, to get the upper bound it is just necessary to sum the average to the error. And for the lower bound it is necessary to subtract the error.

[8]For this study, we executed ~100 runs for each experiment.
[9] Having high variance in a data set means that its values vary quite a lot around the mean value. On the other extent, if the variance is zero all the values are equal.

To utilize this formula the results of the simulations should have a normal distribution or being a set large enough (n>30). With some pilot runs found that it does not follow a normal distribution (the collected dataset failed the Kolmogorov-Smirnov test [25]). Therefore we opted to run a large number of simulations per each experiments (100 runs).
Alternatively, it is also possible to utilize T-Student but this provides less precise results.

To state that two results are statistically significantly different, these should have a difference between the two mean values greater than the sum of their margins of error. This property can also be used to affirm (or not) the existence of trends between multiple values.

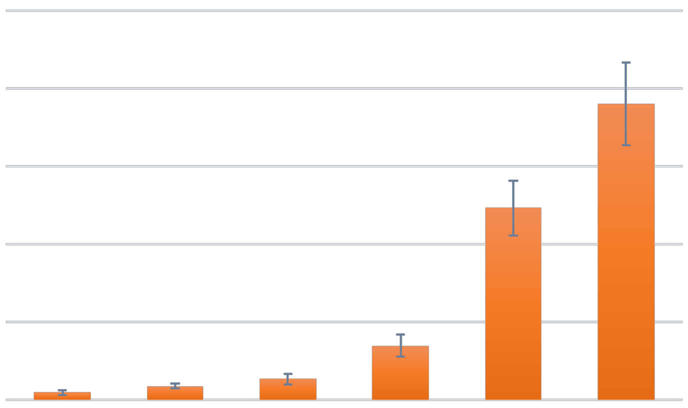


Figure 6.9: A generic data set characterized by an upward trend.

For example, if we look at the Figure 6.9 we can affirm that these results have statistically significant differences from one to another, and additionally state that there is an increasing trend in the data set.

6.7.2. Communication Effects

Bandwidth Variation

With the evaluation set that was analysing the influence of the transmission speed, we obtained the results in the Figure 6.10 and Figure 6.11.
From the plots, it is visible how all three simulations results are dependent on the bandwidth of the network's edges. The larger is the bandwidth the faster is the detection of the double-spending.

After the initial two experiments at 10kbps and 100kbps where the the influence is significant, the effect seems to disappear. A possible explanation for this is due to the existence of network congestion present only at low speed, that would queue up messages and slow the dissemination.

In this evaluation set, it was not possible to execute the experiment in which we have the bandwidth at 10kbps for the Simulation 2, due to limited computational of power. Neither my workstation nor the DAS−4 server were able to complete the executions of any run relative to this experiment.

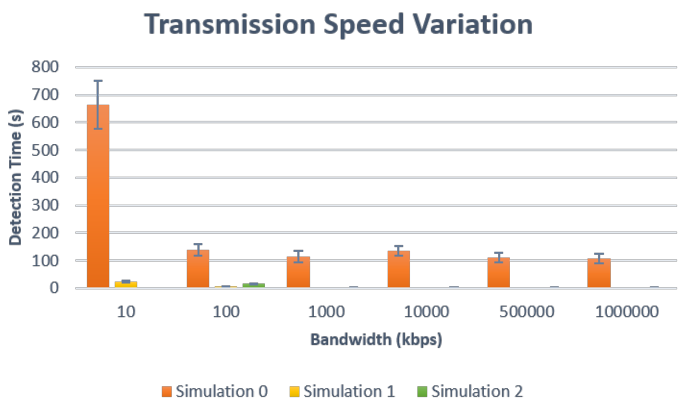


Figure 6.10: Results from transmission speed variation experiments.

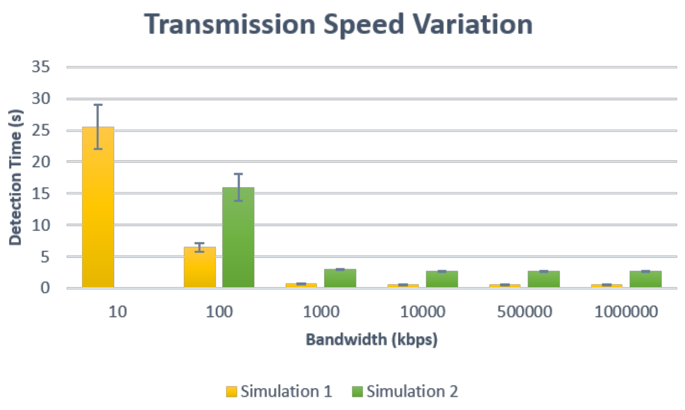
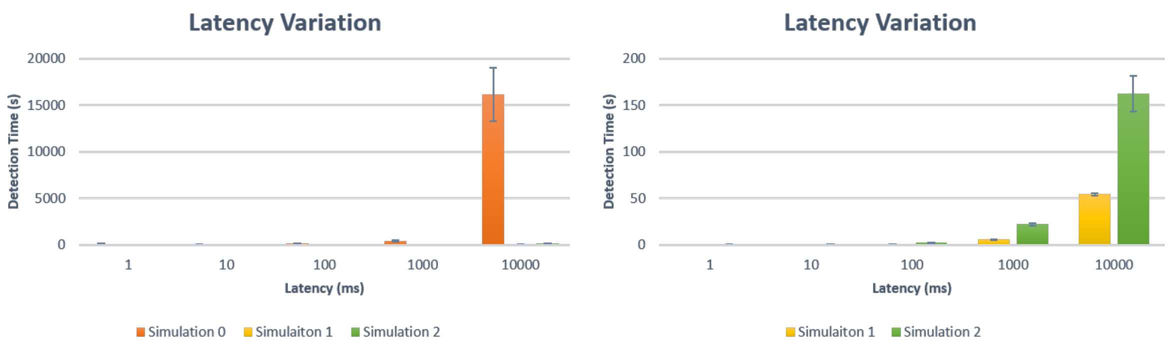


Figure 6.11: Results from transmission speed variation experiments. Partial results.

Latency Variation

Similarly to the bandwidth variation set, all three solutions are influenced by latency variations. In particular the solution 0, which reported an increment of 38 times going from the experiment at 1s to 10s, where the others of two just 9 times. Simulation 1 provides the fastest detection time but seems to scale linearly with the increase of latency. Where simulation 2 incremental trend is a bit irregular, particularly at lower latency, as can be seen in the Figure 6.13.



(a) Simulations 0, 1 and 2

(b) Simulations 1 and 2

Figure 6.12: Results from latency variation experiments.

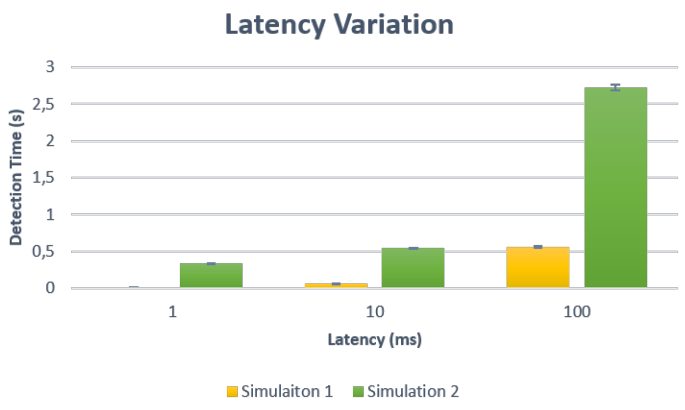


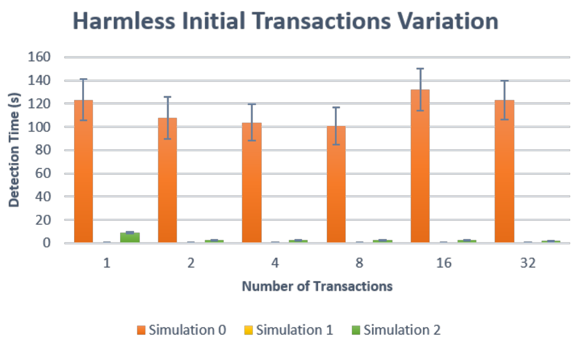
Figure 6.13: Focus on the results of simulations 1 and 2 on the first three simulations experiments.(Latency Variation Experiment)

6.7.3. Nodes Behaviours Effects

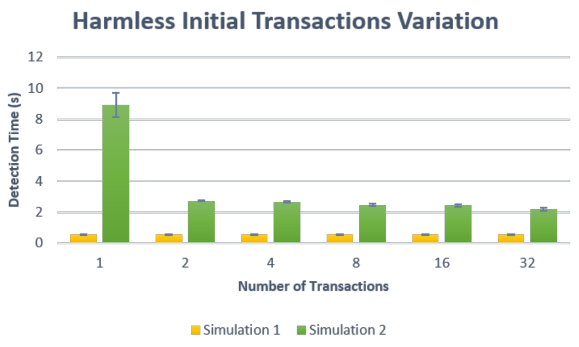
Harmless Initial Transactions Variation

In this evaluations experiment, solution 0 and 1 are not reacting to the variation of the number of the harmless transaction before the evil one.
But solution 2 seems to be sensible on this matter, in fact, in the first two experiments (one and two transactions) the variation is more than 226%. The effect however seems reduced significantly as the number increase.

Simulation 2 display this behaviour because of the way dissemination is performed. In this solution, the dissemination is solely performed with nodes with which the transaction partner has transacted in past. Therefore if there are many transactions in the partner chain the detection will be faster, as it is more likely to find nodes having contrasting information.



(a) Simulations 0, 1 and 2



(b) Simulations 1 and 2

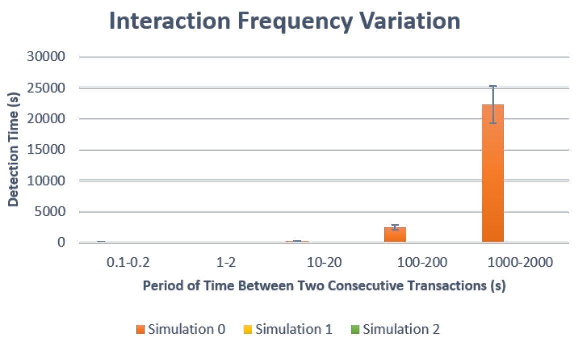
Figure 6.14: Results from number of harmless initial transactions variations experiments.

Simulation 1 is the fastest one, but solution 2 is just behind that with just a four times longer detection time. This is followed by the slowest solution 0 with a whopping 260 times slower.

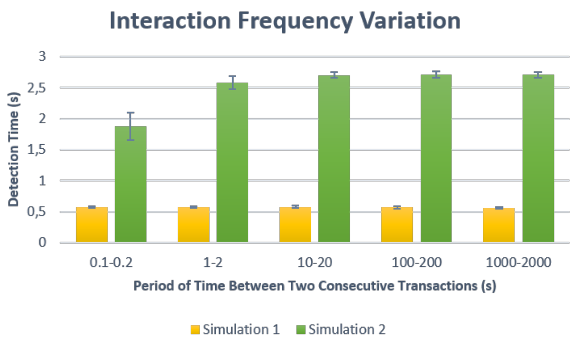
Interaction Frequency Variation

By varying the frequency of interaction we indirectly also affect the speed of dissemination. But if we increase it too much we face a bottleneck effect, higher frequency means that there are a lot of messages being exchanged, and these can cause network cough and consequently slow down all the transactions.
In this evaluation, we started with a high transaction rate (every 0.1 to 0.2 seconds between consecutive trans-

actions) and gradually decrease it by every experiment. The outcome of this evaluation set is displayed in the Figure 6.15.



(a) Simulations 0, 1 and 2



(b) Simulations 1 and 2

Figure 6.15: Results from variation of frequency of interaction.

Similarly to what happened with the "Number of Nodes Variation" set of experiments, solution 1 seems indifferent to the variation of the parameters. Where the other two does, with solution 0 particularly sensitive to this change, having a total variation between all the experiments of more than 20000% (compared to just 44% of the solution 2).

Variation of Anonymizer Nodes (For the Simulation2 only.)

Anonymizer nodes selection plays a very important role in solution 2, during a transaction besides hiding the identity of a node they also performs chain verification.
The employment of more anonymizer can cost time, but can also help to detect the attack early. This effect is visible from the result illustrated in the Figure 6.16.

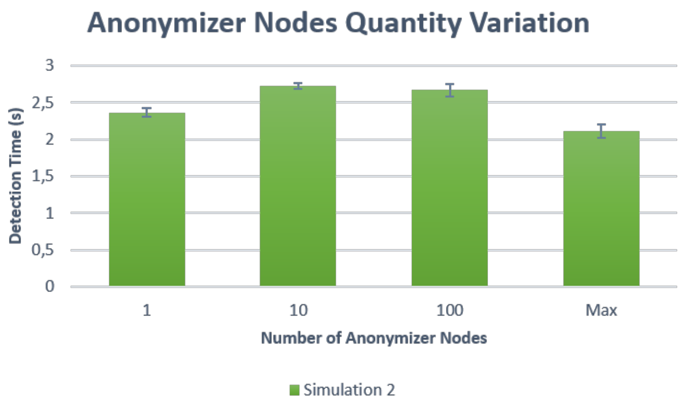


Figure 6.16: Results from variation of quantity of anonymizer nodes.

When one anonymizer node is being employed the dissemination phase is executed promptly if compared when ten nodes are utilized [10].

If we increase the number of anonymizer to the maximum amount possible the system will be very secure, this because there is a very high probability [11] that one of the nodes employed for the anonymization will detect the double-spending.

[10]A node employing ten anonymiser needs to wait for a reply by all of them before proceeding to complete the transaction.
[11]In some instance this can also be equal one, when a node has in his list of anonymizer nodes all the nodes in the network.

By analysing the types of detection of these experiments we validated our previous reasoning, these are illustrated in the Figure 6.17. In the experiment with maximum number of anonymizer possible, 100% of the runs resulted in detections performed by anonymizer nodes, and happened before the transaction was even completed.

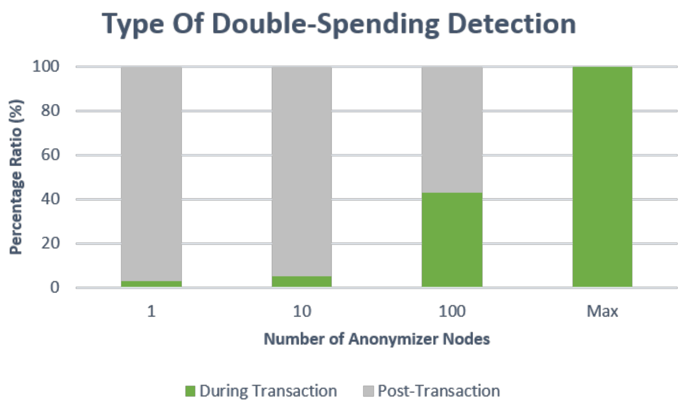
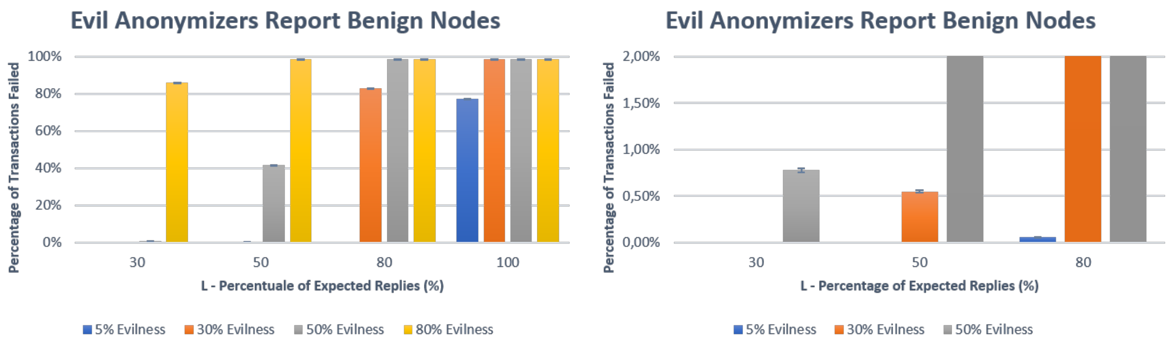


Figure 6.17: Types of double-spending detection for the anonymizer nodes quantity variation experiments.

The extreme cases (with one anonymizer and the maximum amount) are impractical be used in a real implementation, but it is important analyze them too.

Evil Anonymizers Effects (For the Simulation2 only.)

The anonymization mechanism, used in the partner ledger auditing, can be exploited and used by evil nodes. During the auditing, if a sufficient number of nodes drop the anonymization request the queried node will appear as evil independently of its real nature.
In Section 5.2.2 it is possible to read the reason behind this vulnerability.



(a) All the simulations results. (b) A close up of the results.

Figure 6.18: Effects of evil anonymizers in relation to L (minimum number of anonymizer replies required for a successful auditing).

After running multiple simulations to analyze the problem, it has been found that for a low percentage of evil anonymizer and L (replies required) smaller than 80% of K (total anonymizer reached), the transactions in which a peer is reported as evil is less than 0.5% of the total transactions. Even better are the results in the case where we have 5% of evil nodes (realistic assumption), in fact, here the transactions reported are at most 0.06% (L 80%). However, we also measured no reported transactions if L is smaller than 50%.

In conclusion, it has been determined that in environments where there is a small amount to evil nodes the auditing system is reliable and able to overcome these kinds of malicious activities. However, it is possible to reduce L (or increase K) to compensate for a high concentration of evil nodes and still obtain reasonable performance.

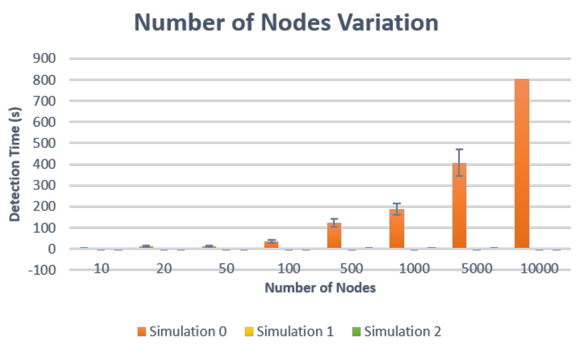
6.7.4. Network Topology Effects

Number of Nodes Variation

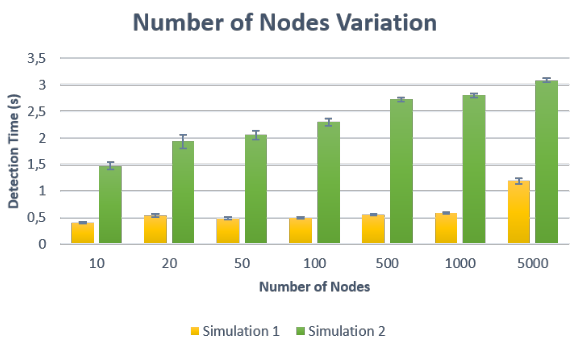
In this evaluation set not all the experiments were concluded at the moment of submitting the paper. Evaluations of simulations 1 and 2 with 10000 nodes are missing.

From the results displayed in Figure 6.19, it is clear that solutions 0 and 2 are significantly dependent on the number of nodes present in the network. However, it is important to note that the trend of solutions 2 seems to slow down and asset to a value, where for solution 0 the detection time keeps increasing.
Solution 1, on the other hand, is constant around ~0,5s for networks smaller than 5000 nodes, but for larger ones it starts to increase too.

This behaviour is because complete dissemination reaches all the nodes in a definite amount of time, where the partial dissemination (employed in solution 0 and 2) doesn't. For solution 0 the detection of the attack relies on future transactions between the disseminated nodes, which should occur between nodes holding contrasting information, in a large network the probability for this to happen is low. Solution 2 has a slight advantage as initially it is relied on the anonymizer nodes, the confined dissemination area and only later to future transactions.



(a) Simulations 0, 1 and 2

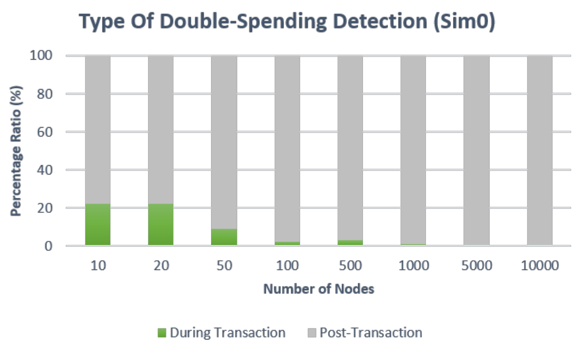


(b) Simulations 1 and 2

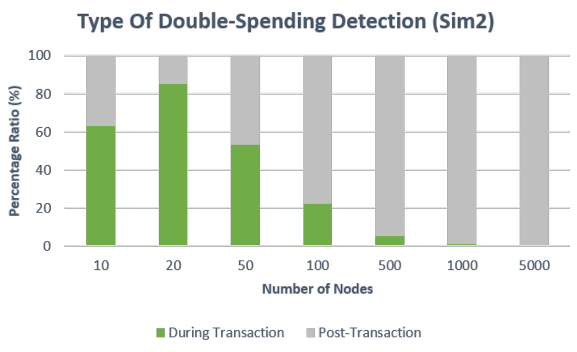
Figure 6.19: Results from number of nodes variations experiments.

In terms of performance, solution 1 is the fastest one, providing the quickest detection times on all experiments. However, solution 2 is ~6000% faster to the current implemented solution but only ~150% slower to the best one (if we consider the scenario with 5000 nodes).

From these figures, it might suggest that solution 1 has exceptional performance in scaling up as it is not varying much, but it is not like this. With this solution, every new node added to the network increases the traffic required to disseminate the whole network, but also the memory usage that every node has to dedicate to store dissemination information form all other agents. More nodes require more data to be transferred and stored.



(a) Results of solution 0.



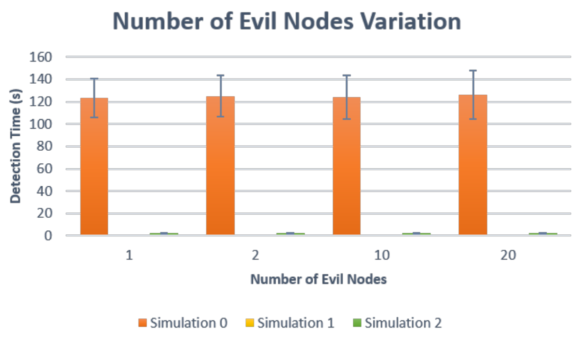
(b) Results of solution 2.

Figure 6.20: Types of double-spending detection for the "variation of number of nodes".

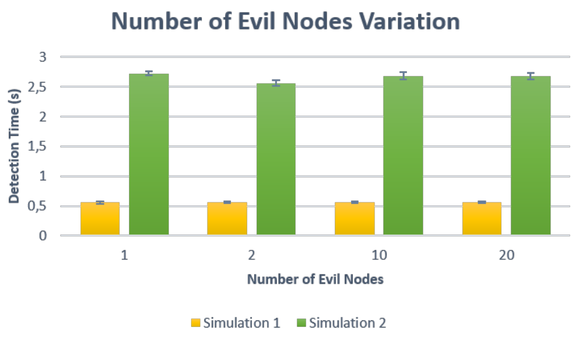
From Figure 6.20 it is possible to see how increasing the number of nodes changes the type of double-spending detection. From an initial during-transaction detection (chain verification) in a network of just 10 nodes, up to a total post-transaction detection realized during dissemination or in a later phase.
The type of detections for solution 1 are always "during detection". However, in very large networks we expect that this behaviour will change.

Number of Evil Nodes Variation

In this evaluation set, all three solutions seem not to have any influence on the number of evil nodes present in the network.
Solution 1 is the quickest one to detect double-spending, followed by solution 2 and the slowest solution 0. This reasoning are clearly visible from the Figure 6.21.



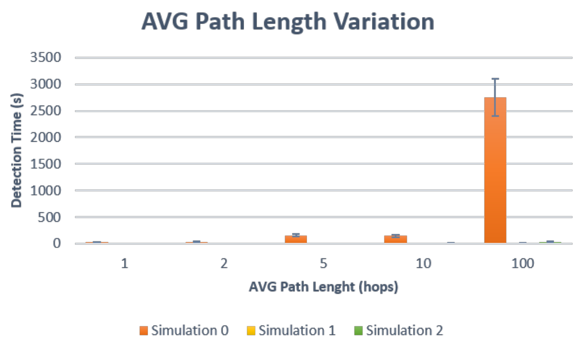
(a) Simulations 0, 1 and 2



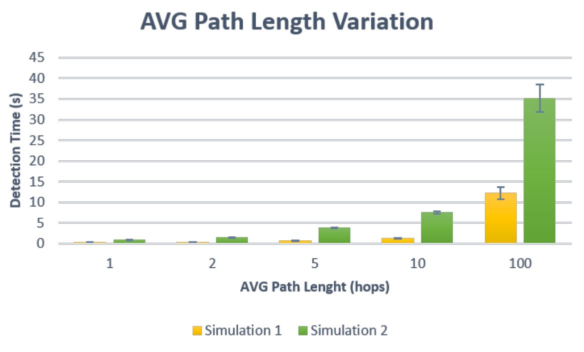
(b) Simulations 1 and 2

Figure 6.21: Results from number of evil nodes variations experiments.

Average Path Length Variation



(a) Simulations 0, 1 and 2

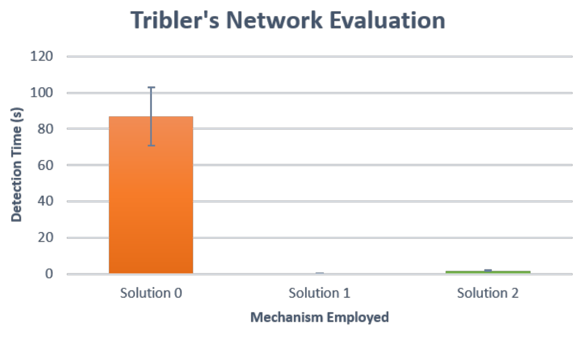


(b) Simulations 1 and 2

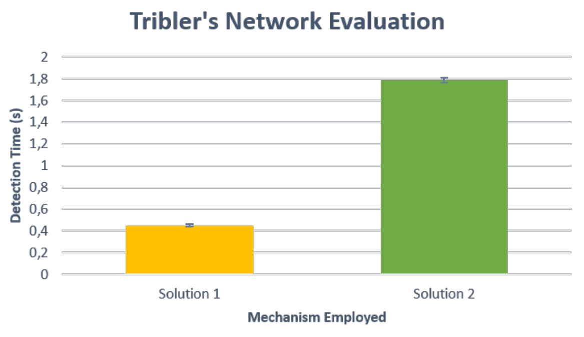
Figure 6.22: Graphs illustrating results from AVG path length variaiton experiments.

Having nodes that are far away for each other has the effect of slowdown transactions and information dissemination. This behaviour is easily observable from the results of this evaluation set, illustrated in the Figure 6.22.
All three solutions have effects as a consequence of the variation of this parameter. The most affected is the solution 0, of which results displays a variation of more than 131 times across all the experiments.
The quickest detection time is provided by simulation 1, followed by simulation 2.

Tribler's Network Structure



(a) Simulations 0, 1 and 2



(b) Simulations 1 and 2

Figure 6.23: Graphs illustrating results from simulations on the network used by Tribler.

From the results illustrated on Figure 6.23, it is clearly noticeable the large performance margin of the proposed solutions over the current one which detects the attack in average in ~86.68 seconds.
The global dissemination solution is the fastest to detected the double-spending in just ~0.45 sec, but it is closely followed by solution 2 which identifies the attack in ~1.79 sec.
It is worth noticing that the results from simulation 0 have a much higher variance if compared to the other two, hence its detection times vary a lot and are spread far from the mean value.

6.7.5. Results Summary

By examining all the results in this section, we can assert that the current detection mechanism offers inadequate performance in all the experiments, achieving detections in average hundreds or even thousands of times slower than the two proposed solutions. Scalability is also an issue, as visible in the experiments evaluating the effects of the variation of the number of nodes, where this mechanism seems to have an much pronounced increasing trend if compared to the proposed solutions.

Simulation1, which employ global dissemination, is the fastest in all examined the scenarios, being able to detect issues hundreds of times faster if compared to the original solution. Simulation1 produce consistent performances as we observed very small variations among the outcomes between the different experiments of most of the evaluated sets.

Finally, Simulation2 achieved detection times that were slower to the Simulation1, but still much faster than the original detection mechanism, dozens of times quicker in average. However, Simulation2 outperform Simulation1 in terms of bandwidth consumption and data storage as can be seen in Section 6.4, especially if we analyze these parameters in large networks.

7 Conclusion and Future Work

Cryptocurrencies are emerging and becoming more and more popular. Traditional blockchain architecture that employs a single global ledger have hard limitations regards to scalability and performance. Global consensus provides strong security but at the expense of having low transactions throughput.
There exists alternative architecture that solves this problem, and one of them is called Trustchain.

The research objective of this thesis is to design and evaluate a solution that can be implemented inside the Trustchain fabric and can offer superior double-spending detection performance.

To pursue this goal, we initially examined the inner workings of Trustchain and investigated the reasons behind the limitations of its current detection system. Inspired by papers from the area of peers eviction mechanism for P2P networks we designed two solutions.
The first one is based on the total broadcast of every new transaction in the network. The second solution, which is much more cost-efficient, is based on a two-step mechanism, an initial anonymous auditing scheme to verify the partner's ledger and localized dissemination of the transaction information.
To evaluate the solutions, we created a simulator of Trustchain utilizing OMNeT++. On this, we then implemented all three solutions and executed the simulator under different scenarios/circumstances to compare the outcome.
The results showed that the current detection system, that we called Solution0, is the slowest, performing on average up to hundreds of times worse than the proposed solutions.
Solution1 (broadcast dissemination) is the quickest to detect double spending in all experimented scenarios, but Solution2 (anonymizer and localized dissemination) with excellent performance and has also the significant advantage of producing a much smaller load on the network.

In conclusion, our solution (Solution2) is capable to offer better performance than the current double-spending detection system, at some small expense. We require more messages transmitted in the network and the verification auditing extends the time needed to complete a transaction.
Security always has a price to pay, it can be represented as extra payload or as additional computation.

Thus, our work provides new insights regarding the difficulties involved in detecting double-spending in a pair-wise ledger system, and propose and evaluate a solution to address the problem.

7.1. Future Research

The following points suggest some of the future research work:

- **Post Detection Events**
  Once the double-spending is detected it is necessary to proceed with the reporting and correction of the issues generated by it, by addressing all the nodes involved in the attack. This protocol needs to be designed by considering the fact that victims of an attack might not be online when this gets detected,

and that there can be multiple detection of the same attack that can happen sequentially/simultaneously.

- **Offline Nodes**
  We haven't examined how the systems behave with nodes that goes offline or are not working. The performance of the solutions might differ in this scenario, and might revel interesting results. Further experiments can help to address this. Network entities like watchtowers [33] can be an interesting approach to deal with this problem. Watchtowers are a well-researched component that is also being employed for other blockchain problems that require always-on nodes. These entities will stay online 24/7 and relay chain request and dissemination for offline nodes.

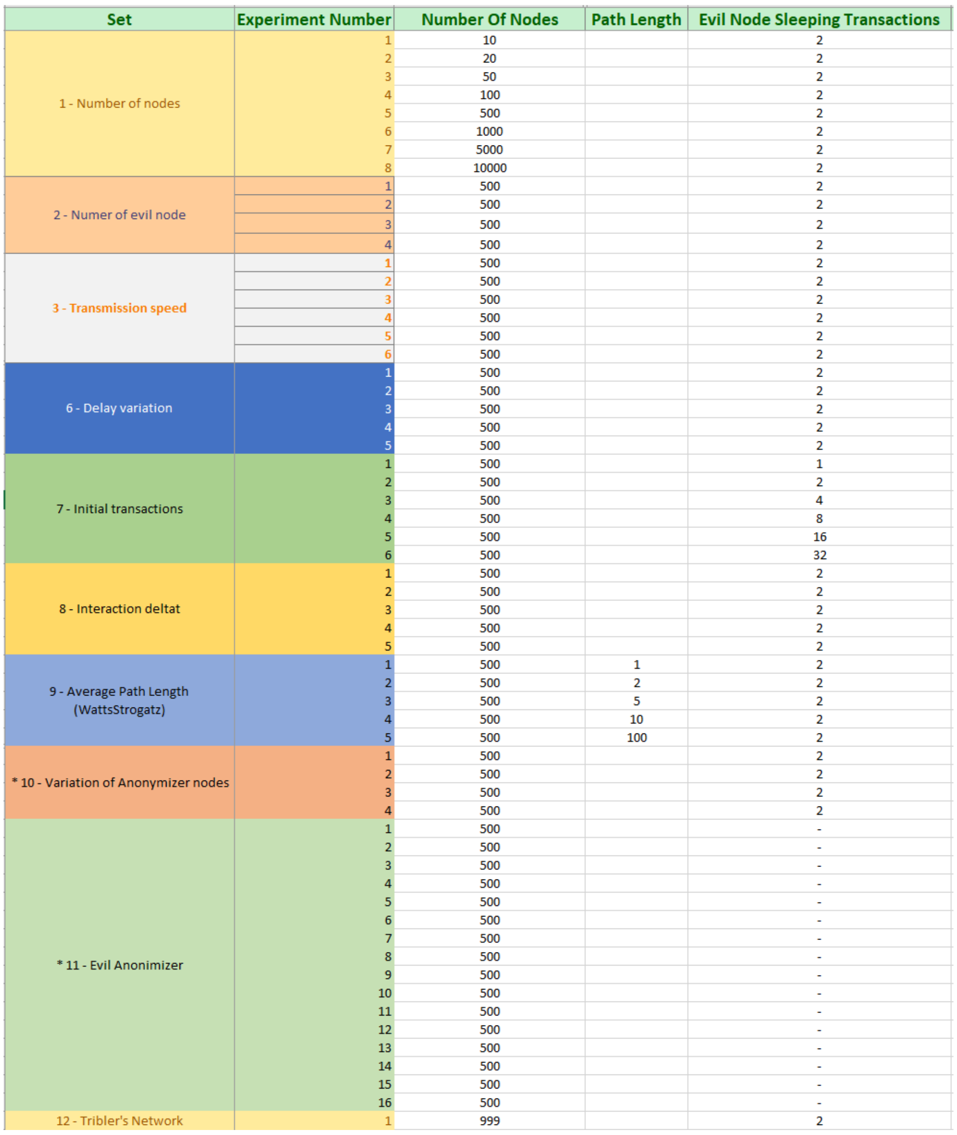
- **Emulation**
  In this work, we investigate the performance of the solutions utilizing simulations on OMNeT++. Trial of the solutions on a Trustchain emulator should be the next step, as this can provide a more realistic behavior, more alike to what we would get from a real implementation.

To conclude the thesis, pair-based architecture like Trustchain can be enabled to detect double-spending in a reasonable amount of time. This thesis provides the design of an effective and efficient detection system, compatible with Trustchain, that is 13000% faster at a cost of a ~7 times larger overhead [1] if compare to the current existing solution.

[1] Calculation based on a network of 5k nodes, the usage of 10 anonymizer nodes and dissemination post-transaction to 50 nodes. Complete details of this assessment can be found in Section 6.7.3 and Section 6.4.

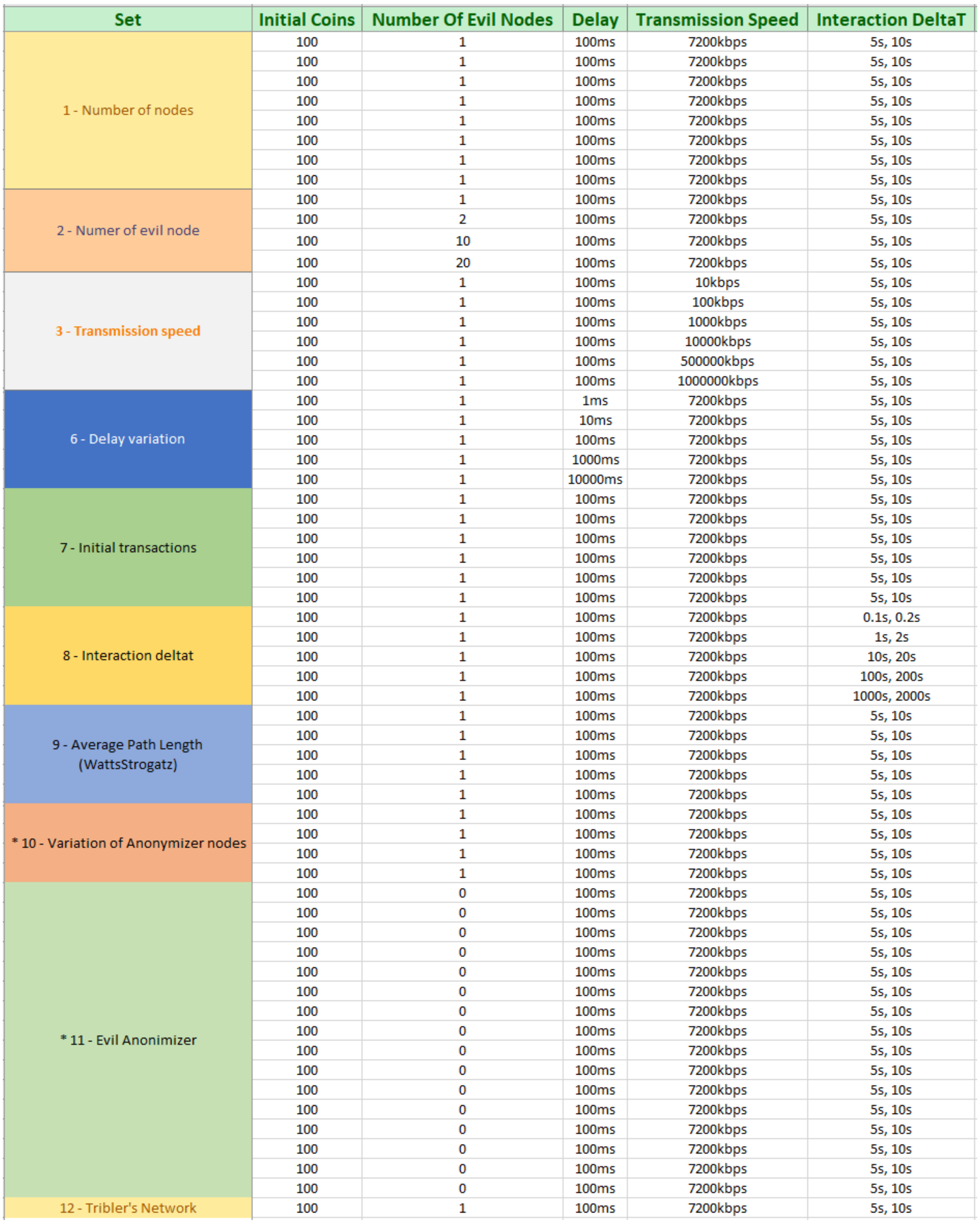
A

Evaluations Configurations



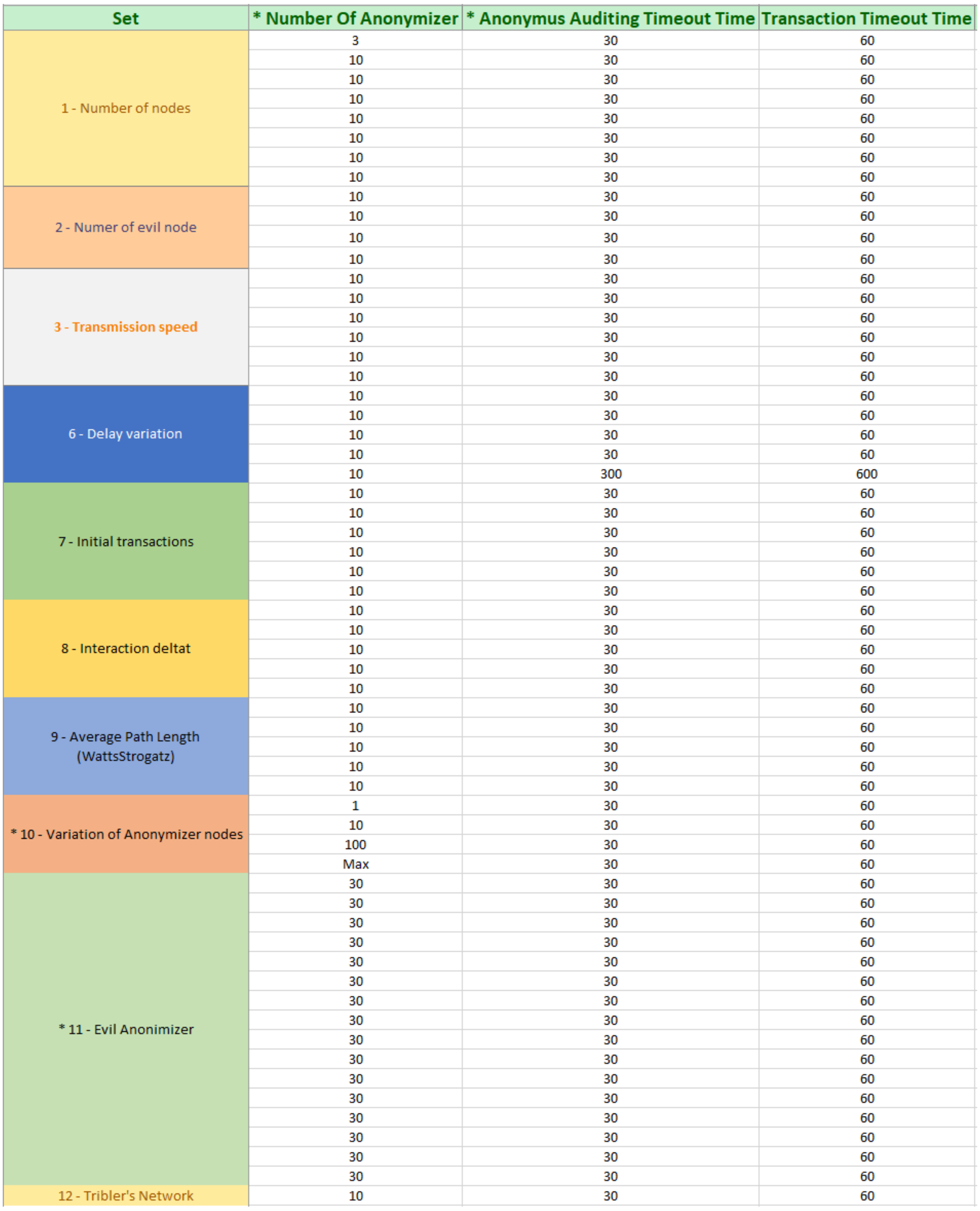
Set	Experiment Number	Number Of Nodes	Path Length	Evil Node Sleeping Transactions
1 - Number of nodes	1	10		2
	2	20		2
	3	50		2
	4	100		2
	5	500		2
	6	1000		2
	7	5000		2
	8	10000		2
2 - Numer of evil node	1	500		2
	2	500		2
	3	500		2
	4	500		2
3 - Transmission speed	1	500		2
	2	500		2
	3	500		2
	4	500		2
	5	500		2
	6	500		2
6 - Delay variation	1	500		2
	2	500		2
	3	500		2
	4	500		2
	5	500		2
7 - Initial transactions	1	500		1
	2	500		2
	3	500		4
	4	500		8
	5	500		16
	6	500		32
8 - Interaction deltat	1	500		2
	2	500		2
	3	500		2
	4	500		2
	5	500		2
9 - Average Path Length (WattsStrogatz)	1	500	1	2
	2	500	2	2
	3	500	5	2
	4	500	10	2
	5	500	100	2
* 10 - Variation of Anonymizer nodes	1	500		2
	2	500		2
	3	500		2
	4	500		2
* 11 - Evil Anonimizer	1	500		-
	2	500		-
	3	500		-
	4	500		-
	5	500		-
	6	500		-
	7	500		-
	8	500		-
	9	500		-
	10	500		-
	11	500		-
	12	500		-
	13	500		-
	14	500		-
	15	500		-
	16	500		-
12 - Tribler's Network	1	999		2

Figure A.1: 1st group of configurations parameters. *THESE SETTINGS ARE ONLY APPLICABLE TO **SIMULATION2**



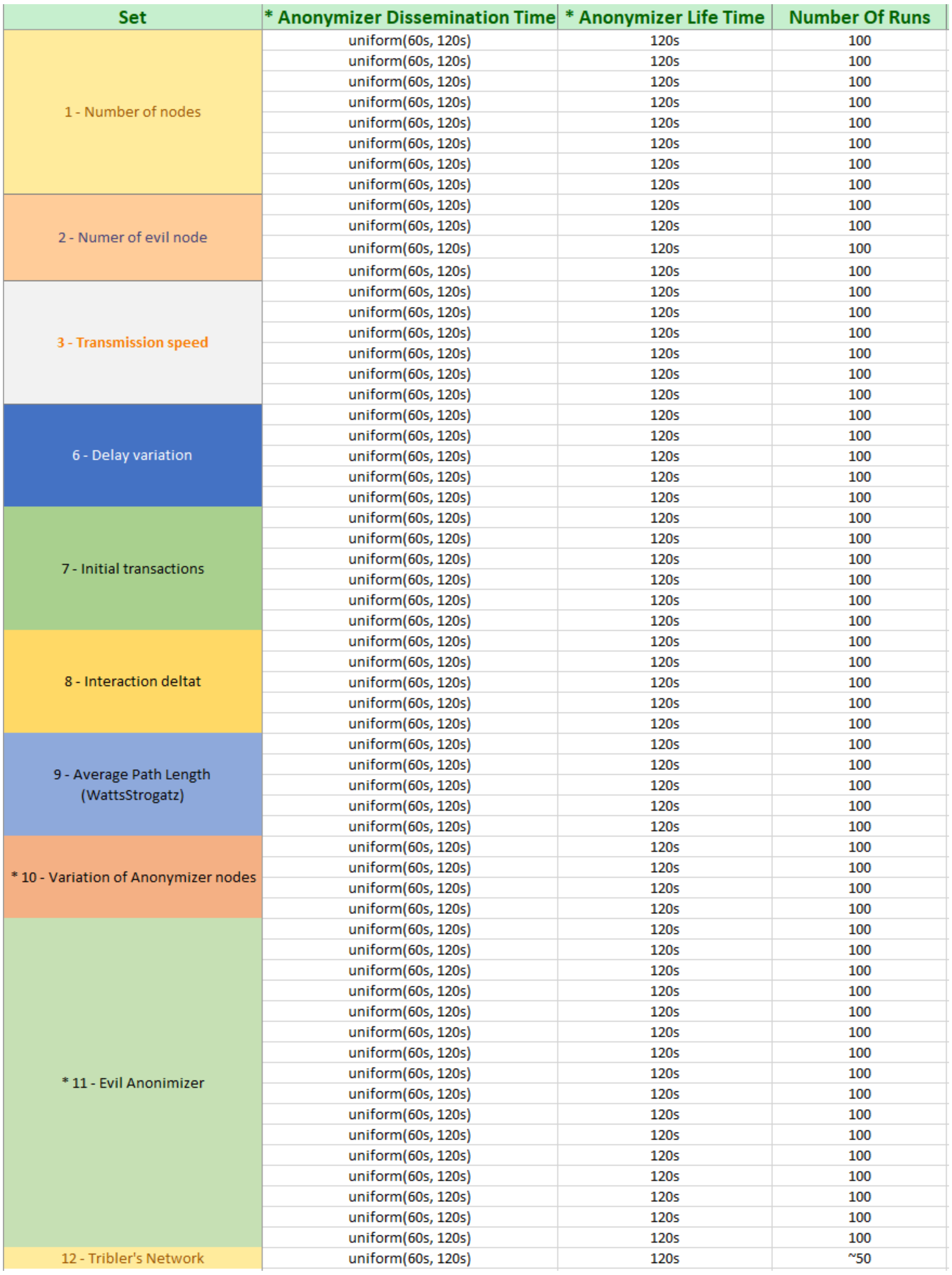
Set	Initial Coins	Number Of Evil Nodes	Delay	Transmission Speed	Interaction DeltaT
1 - Number of nodes	100	1	100ms	7200kbps	5s, 10s
	100	1	100ms	7200kbps	5s, 10s
	100	1	100ms	7200kbps	5s, 10s
	100	1	100ms	7200kbps	5s, 10s
	100	1	100ms	7200kbps	5s, 10s
	100	1	100ms	7200kbps	5s, 10s
	100	1	100ms	7200kbps	5s, 10s
	100	1	100ms	7200kbps	5s, 10s
2 - Numer of evil node	100	1	100ms	7200kbps	5s, 10s
	100	2	100ms	7200kbps	5s, 10s
	100	10	100ms	7200kbps	5s, 10s
	100	20	100ms	7200kbps	5s, 10s
3 - Transmission speed	100	1	100ms	10kbps	5s, 10s
	100	1	100ms	100kbps	5s, 10s
	100	1	100ms	1000kbps	5s, 10s
	100	1	100ms	10000kbps	5s, 10s
	100	1	100ms	500000kbps	5s, 10s
	100	1	100ms	1000000kbps	5s, 10s
6 - Delay variation	100	1	1ms	7200kbps	5s, 10s
	100	1	10ms	7200kbps	5s, 10s
	100	1	100ms	7200kbps	5s, 10s
	100	1	1000ms	7200kbps	5s, 10s
	100	1	10000ms	7200kbps	5s, 10s
7 - Initial transactions	100	1	100ms	7200kbps	5s, 10s
	100	1	100ms	7200kbps	5s, 10s
	100	1	100ms	7200kbps	5s, 10s
	100	1	100ms	7200kbps	5s, 10s
	100	1	100ms	7200kbps	5s, 10s
	100	1	100ms	7200kbps	5s, 10s
8 - Interaction deltat	100	1	100ms	7200kbps	0.1s, 0.2s
	100	1	100ms	7200kbps	1s, 2s
	100	1	100ms	7200kbps	10s, 20s
	100	1	100ms	7200kbps	100s, 200s
	100	1	100ms	7200kbps	1000s, 2000s
9 - Average Path Length (WattsStrogatz)	100	1	100ms	7200kbps	5s, 10s
	100	1	100ms	7200kbps	5s, 10s
	100	1	100ms	7200kbps	5s, 10s
	100	1	100ms	7200kbps	5s, 10s
	100	1	100ms	7200kbps	5s, 10s
* 10 - Variation of Anonymizer nodes	100	1	100ms	7200kbps	5s, 10s
	100	1	100ms	7200kbps	5s, 10s
	100	1	100ms	7200kbps	5s, 10s
	100	1	100ms	7200kbps	5s, 10s
* 11 - Evil Anonimizer	100	0	100ms	7200kbps	5s, 10s
	100	0	100ms	7200kbps	5s, 10s
	100	0	100ms	7200kbps	5s, 10s
	100	0	100ms	7200kbps	5s, 10s
	100	0	100ms	7200kbps	5s, 10s
	100	0	100ms	7200kbps	5s, 10s
	100	0	100ms	7200kbps	5s, 10s
	100	0	100ms	7200kbps	5s, 10s
	100	0	100ms	7200kbps	5s, 10s
	100	0	100ms	7200kbps	5s, 10s
	100	0	100ms	7200kbps	5s, 10s
	100	0	100ms	7200kbps	5s, 10s
	100	0	100ms	7200kbps	5s, 10s
	100	0	100ms	7200kbps	5s, 10s
	100	0	100ms	7200kbps	5s, 10s
	100	0	100ms	7200kbps	5s, 10s
12 - Tribler's Network	100	1	100ms	7200kbps	5s, 10s

Figure A.2: 2nd group of configurations parameters.*These settings are only applicable to **Simulation2**



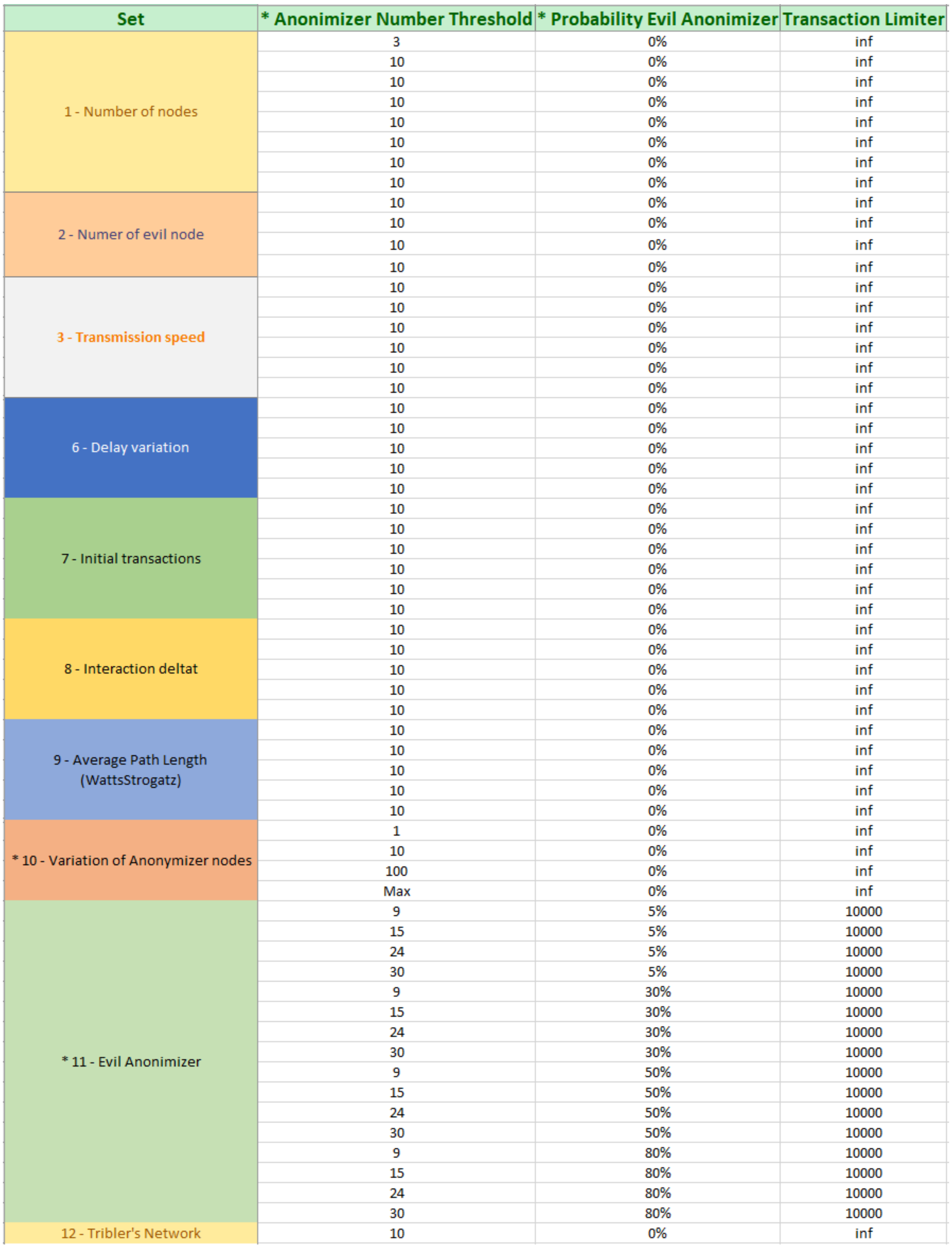
Set	* Number Of Anonymizer	* Anonymus Auditing Timeout Time	Transaction Timeout Time
1 - Number of nodes	3	30	60
	10	30	60
	10	30	60
	10	30	60
	10	30	60
	10	30	60
	10	30	60
	10	30	60
2 - Numer of evil node	10	30	60
	10	30	60
	10	30	60
	10	30	60
3 - Transmission speed	10	30	60
	10	30	60
	10	30	60
	10	30	60
	10	30	60
	10	30	60
6 - Delay variation	10	30	60
	10	30	60
	10	30	60
	10	30	60
	10	300	600
7 - Initial transactions	10	30	60
	10	30	60
	10	30	60
	10	30	60
	10	30	60
	10	30	60
8 - Interaction deltat	10	30	60
	10	30	60
	10	30	60
	10	30	60
	10	30	60
9 - Average Path Length (WattsStrogatz)	10	30	60
	10	30	60
	10	30	60
	10	30	60
	10	30	60
* 10 - Variation of Anonymizer nodes	1	30	60
	10	30	60
	100	30	60
	Max	30	60
* 11 - Evil Anonimizer	30	30	60
	30	30	60
	30	30	60
	30	30	60
	30	30	60
	30	30	60
	30	30	60
	30	30	60
	30	30	60
	30	30	60
	30	30	60
	30	30	60
	30	30	60
	30	30	60
	30	30	60
	30	30	60
12 - Tribler's Network	10	30	60

Figure A.3: 3rd group of configurations parameters.*THESE SETTINGS ARE ONLY APPLICABLE TO **SIMULATION2**



Set	* Anonymizer Dissemination Time	* Anonymizer Life Time	Number Of Runs
1 - Number of nodes	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
2 - Numer of evil node	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
3 - Transmission speed	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
6 - Delay variation	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
7 - Initial transactions	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
8 - Interaction deltat	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
9 - Average Path Length (WattsStrogatz)	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
* 10 - Variation of Anonymizer nodes	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
* 11 - Evil Anonimizer	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
	uniform(60s, 120s)	120s	100
12 - Tribler's Network	uniform(60s, 120s)	120s	~50

Figure A.4: 4th group of configurations parameters.*THESE SETTINGS ARE ONLY APPLICABLE TO **SIMULATION2**



Set	* Anonimizer Number Threshold	* Probability Evil Anonimizer	Transaction Limiter
1 - Number of nodes	3	0%	inf
	10	0%	inf
	10	0%	inf
	10	0%	inf
	10	0%	inf
	10	0%	inf
	10	0%	inf
	10	0%	inf
2 - Numer of evil node	10	0%	inf
	10	0%	inf
	10	0%	inf
	10	0%	inf
3 - Transmission speed	10	0%	inf
	10	0%	inf
	10	0%	inf
	10	0%	inf
	10	0%	inf
	10	0%	inf
6 - Delay variation	10	0%	inf
	10	0%	inf
	10	0%	inf
	10	0%	inf
	10	0%	inf
7 - Initial transactions	10	0%	inf
	10	0%	inf
	10	0%	inf
	10	0%	inf
	10	0%	inf
	10	0%	inf
8 - Interaction deltat	10	0%	inf
	10	0%	inf
	10	0%	inf
	10	0%	inf
	10	0%	inf
9 - Average Path Length (WattsStrogatz)	10	0%	inf
	10	0%	inf
	10	0%	inf
	10	0%	inf
	10	0%	inf
* 10 - Variation of Anonymizer nodes	1	0%	inf
	10	0%	inf
	100	0%	inf
	Max	0%	inf
* 11 - Evil Anonimizer	9	5%	10000
	15	5%	10000
	24	5%	10000
	30	5%	10000
	9	30%	10000
	15	30%	10000
	24	30%	10000
	30	30%	10000
	9	50%	10000
	15	50%	10000
	24	50%	10000
	30	50%	10000
	9	80%	10000
	15	80%	10000
	24	80%	10000
	30	80%	10000
12 - Tribler's Network	10	0%	inf

Figure A.5: 5th group of configurations parameters.*THESE SETTINGS ARE ONLY APPLICABLE TO **SIMULATION2**

Bibliography

[1] Bitcoin block propagation delay. https://dsn.tm.kit.edu/bitcoin/,.

[2] Bitcoin system's graph of historical transaction size. https://tradeblock.com/bitcoin/historical/1h-f-tsize_per_avg-01101,.

[3] Bitcoin system's graph of historical block creation time. https://data.bitcoinity.org/bitcoin/block_time/5y?f=m10&t=l,.

[4] Das-4 (the distributed asci supercomputer 4). https://www.cs.vu.nl/das4/.

[5] Lightning network has 1% success rate with transactions larger than $200, controversial research says. https://diar.co/volume-2-issue-25/.

[6] International ping statistics. https://wondernetwork.com/pings,.

[7] Ip latency statistics. https://enterprise.verizon.com/terms/latency/,.

[8] Global network latency monitor. https://www.dotcom-tools.com/internet-backbone-latency.aspx,.

[9] Annual report analysis. https://www.akamai.com/us/en/about/news/press/2017-press/akamai-release\s-first-quarter-2017-state-of-the-internet-connectivity-report.jsp,.

[10] Average internet speeds by country. https://www.fastmetrics.com/internet-connection-speed-by-country.php#top-10-worldwide,.

[11] Trustchain application inside tribler. https://tribler.readthedocs.io/en/next/trustchain.html.

[12] Carl Hauser Wes Irish John Larson Scott Shenker Howard Sturgis Dan Swinehart Alan Demers, Dan Greene and Doug Terry. Epidemic algorithms for replicated database maintenance., 1987.

[13] Eduardo Alchieri Alysson Neves Bessani. A guided tour on the theory and practice of state machine replication.

[14] Adam. Back. Hashcash-a denial of service counter-measure. In *Hashcash-a denial of service counter-measure.*, 2002.

[15] Vitalik Buterin et al. Qna video: Omisego holiday special ama. omisego.network, 2018.

[16] Bin Chang and Shantanu Dutta. Internet banking and online trading. In *Internet Banking and Online Trading*. University of Ontario Institute of Technology, Canada, 2012.

[17] Eyal Ittay Croman, Kyle. On scaling decentralized blockchains. NUS Computing, 2006.

[18] Chaum David. Blind signatures for untraceable payments. In *Blind signatures for untraceable payments*. Department of Computer Science, University of California, Santa Barbara, CA, 1982.

[19] Paul Erdős and Alfréd Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1): 17–60, 1960.

[20] Ittay Eyal and EminGünSirer. Majority is not enough: Bitcoin mining is vulnerable. In *Majority is not enough: Bitcoin mining is vulnerable.*, pages 436–454, 2014.

[21] D. Germanus, S. Roos, T. Strufe, and N. Suri. Mitigating eclipse attacks in peer-to-peer networks. In *2014 IEEE Conference on Communications and Network Security*, pages 400–408, Oct 2014. doi: 10.1109/CNS.2014.6997509.

[22] Steve Hanke. Hyperinflation – a kaleidoscope of uses and abuses. `https://www.forbes.com/sites/stevehanke/2019/01/30/hyperinflation-a-kaleidoscope-of-uses-and-abuses/#23f34c7be198`, Jan 2019.

[23] H. Ismail, S. Roos, and N. Suri. A composite malicious peer eviction mechanism for super-p2p systems. In *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 456–464, Aug 2018. doi: 10.1109/TrustCom/BigDataSE.2018.00072.

[24] R. Koch, M. Golling, and G. D. Rodosek. How anonymous is the tor network? a long-term black-box investigation. *Computer*, 49(3):42–49, Mar 2016. doi: 10.1109/MC.2016.73.

[25] Hubert W Lilliefors. On the kolmogorov-smirnov test for normality with mean and variance unknown. *Journal of the American statistical Association*, 62(318):399–402, 1967.

[26] Dick HJ Epema Michel Meulpolder, Johan A Pouwelse and Henk J Sips. Bartercast: A practical approach to prevent lazy freeriding in p2p networks., 2009.

[27] Azizah Abdul Manaf Mohammed A. Saleh. Optimal specifications for a protective framework against http-based dos and ddos attacks, Aug 2014.

[28] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. In *Bitcoin: A Peer-to-Peer Electronic Cash System*.

[29] Fanny Chevalier Jennifer Mankoff Anind Dey Nikola Banovic, Tofi Buzali. Modeling and understanding human routine behavior. *ACM CHI Conference on Human Factors in Computing Systems 2016*, pages 248–260, 2016. doi: 10.1145/2858036.2858557.

[30] Martijn de Vos Pim Otte and Johan Pouwelse. Trustchain: A sybil-resistant scalable blockchain. `https://www.sciencedirect.com/science/article/pii/S0167739X17318988/`, Sep 2017.

[31] Julie Pitta. Requiem for a bright idea. `https://www.forbes.com/forbes/1999/1101/6411390a.html#4252c3cb715f`, Nov 1999.

[32] Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments. `https://lightning.network/lightning-network-paper.pdf`, 2016.

[33] Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments. In *The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments*, January 14, 2016.

[34] Serguei Popov. The tangle. *cit. on*, page 131, Oct 2017.

[35] Fergal Reid and Martin Harrigan. An analysis of anonymity in the bitcoin system. In *An analysis of anonymity in the bitcoin system.*, 2013.

[36] Stephen T Hedetniemi Sandra M Hedetniemi and Arthur L Liestman. A survey of gossiping and broadcasting in communication networks., 1988.

[37] Atul Singh et al. Eclipse attacks on overlay networks: Threats and defenses. In *In IEEE INFOCOM*. Citeseer, 2006.

[38] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393(6684): 440, 1998.

[39] Paul R. Wilson. Locality of reference, patterns in program behavior, memory management, and memory hierarchies. In *Locality of Reference, Patterns in Program Behavior, Memory Management, and Memory Hierarchies*, 2007.