Exploring the Scalability, Throughput and Security Characteristics of the Tangle Distributed Ledger Technology through Simulation Analysis

Nahid Alimohammadi Madenouei

A THESIS SUBMITTED TO
THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE
OF
MASTER OF SCIENCE

GRADUATE PROGRAM IN COMPUTER SCIENCE
YORK UNIVERSITY
TORONTO, ONTARIO

December 2020

Abstract

During the past few years, blockchain technologies have attracted substantial attention from researchers, engineers, and enterprises. These technologies provide decentralized platforms to validate different types of transactions without relying on a central authority. Since the advent of the popular Bitcoin network [2], [3], various types of protocols, among them DAG-based distributed ledgers, have been offered to improve the Bitcoin network's shortcomings in the areas of scalability, throughput, and security. Researchers have also been motivated to study blockchain-based applications across multiple domains with novel design and capabilities [4]. However, when used in such capacities, failures of blockchain networks (BNs) imply catastrophes that extend beyond individuals, organizations, and countries [5]. In light of such mission criticality, the vision of turning BNs into decentralized transaction processing systems that can sustainably and securely compete with the centralized state of the art poses substantial methodological challenges for researchers [5]. Before considered for wide adoption, BN protocols and technologies must be assessed with different analytical and empirical methodologies. In this research, a novel DAG-based distributed ledger, the Tangle, has been studied and several of its features have been investigated through simulation analysis. Contributions of this research are two-fold. First, a Tangle simulator has been designed and implemented based on model-oriented design and development principles. Second, three important characteristics of Tangle networks including scalability, throughput and their security against double-spending attacks have been investigated.

Acknowledgements

The support provided by the program in Dependable Internet of Things Applications (DITA) created by NSERC at York University is gratefully acknowledged. As for myself, I would like to thank my supervisor, Professor Sotirios Liaskos, for his professional direction and help. The support provided by the committee members of this thesis and MAIST lab mates is also gratefully acknowledged.

Table of Contents

List of Tables

List of Figures

List of Symbols

$\boldsymbol{\lambda}$: transaction arrival rate

$\boldsymbol{\alpha}$: randomness factor for weighted random walk

Glossary of Terms

- **Bitcoin Blockchain:** A growing chain of blocks that are linked using cryptography. Each block consists of several transactions up to a size limit.
- **Tangle/DAG:** A directed acyclic graph where each vertex is a transaction, and each edge shows that the transaction at the source approves the transaction at the destination.
- **Node:** A computer which is part of the consensus network. Its role is to issue new transactions and to validate already existing ones.
- **Transaction**: A message that transfers funds or information between two or more nodes.
- **Tip:** A Tangle transaction that has not been approved yet.
- **Transaction Arrival Rate:** Number of arrived transactions per second.
- **MCMC Algorithm:** Markov Chain Monte Carlo, is a technique the Tangle uses to calculate the probability for random walking in the DAG to select two tips.
- **Cumulative Weight:** It is defined as the own weight of a transaction plus the sum of own weights of all transactions that directly or indirectly approve this transaction.
- **Confidence:** Confidence is a measure of a transaction's level of acceptance in a Tangle.
- **High-Load Tangle:** A state of the Tangle in which the flow of transactions in the network is high, and the network has computational delays together with the latency. Thus, there exists great number of the tips in the Tangle.
- **Low-Load Tangle:** A state of the Tangle in which the flow of transactions in the network is low, and the network is fast and does not have computational delays or latency. As a result, there exists just a few tips in the Tangle.
- **Proof of Work:** A piece of data which is difficult (costly, time-consuming) to produce but easy for others to verify and which satisfies certain requirements. Implemented in our context through hashcash [1], in which suffix to a piece of data needs to be found so that its cryptographic hash meets a certain condition.
- **Difficulty:** Difficulty is a value in blockchain networks used to show how hard the Proof of Work is, and it refers to the average number of trials each node needs to generate the Proof of Work.
- **MWM:** Minimum Weight Magnitude which indicates Proof of Work difficulty level in the Tangle network.

- **Transmission rate:** Average propagation rate in bits per second among nodes of the network.
- **Security:** In this research, it means how much the network can resist double-spending attacks.
- **Scalability:** Indicates the number of incoming transactions, nodes and simulated network operation time our simulations can handle within realistic time.
- **Throughput:** It equals the number of confirmed transactions per second in the network. Depends on average confidence of the transaction in the network.
- **Tip Selection Algorithm:** There are three types of tip selection algorithm in the Tangle: completely random selection, unweighted random walk and weighted random walk. In this research, we have implemented and studied the weighted random walk.

Preface

This research has been conducted in the Master of Arts in Information Systems and Technology (MAIST) Lab under supervision of Professor Sotirios Liaskos at York University. In this thesis, my original work includes the design and implementation of the Tangle simulator. I am also responsible for the model validation and all the experiments such as experiment design, environment set up, data collection, processing and visualization.

Chapter One - Introduction

Implementation of Blockchain Networks (BNs) has inspired a large number of novel applications ranging from e-finance, e-healthcare, smart home, Internet of Things, social security and logistics [4]. Thus, researchers have been motivated to study the blockchain-based applications across multiple domains. However, since failures of BNs might result in dire consequences, before considered for wide adoption, BN protocols and technologies must be assessed with different analytical and empirical methodologies to examine their features such as scalability, throughput and security.

Focusing on empirical evaluation, the exact reproduction and validation of open-access BNs in their envisioned scale in a lab environment is impossible [5]. Performing experiments on actual blockchains are difficult since a large number of nodes are necessary. If a private experimental network is constructed, the information on the entire network can be obtained, nevertheless, setting up a large number of nodes is costly, and the power consumption of a large-scale system may have to be taken into account. In addition, experimental conditions and network configuration cannot be easily changed [6]. On the other hand, simulators provide an environment that simplifies the implementation and deployment of protocols. With simulation, it is possible to study a large-scale system with thousands of nodes in a single machine and gather results in a reasonable time [7]. Hence, the use of simulation has been put forth as an alternative [6] – [35]. A BN simulation abstracts aspects of the actual or proposed network to allow cost- and time-effective study of its behavior. However, to have value as research instruments, such simulators need to be widely validated for their accuracy [36].

In this research, we develop a simulator aimed at assessing scalability, throughput, and security of a distributed ledger technology of increasing popularity, the Tangle [37], [38]. The Tangle is a Directed Acyclic Graph (DAG) based distributed ledger designed for devices such as sensors to participate in low-energy networks. Tangle eliminates the need for external validators or miners. It lets the users of the network take up the computation for transaction validations. This means zero transaction fees for small value exchanges. As a result, IoT-based ecosystems can afford to initiate huge volumes of nano- and micro-transactions. They do not have to worry about unpredictable or high fees [39]. As the worldwide number of IoT-connected devices is projected to increase to 43 billion by 2023 [40], Tangle networks might be used as IoT platforms allowing devices to communicate securely, exchange data, services and payments and provide a

decentralized, immutable audit trail for asset offers, requests, and orders [41]. It is claimed that the Tangle network is highly scalable, resilient and robust against attacks and transactions get confirmed fast [42]. Therefore, we are interested in empirically studying different aspects and claims about the Tangle network.

Our objectives are two-fold. Firstly, we are interested in producing a design that is widely reusable by the research community. Having a validated and accurate simulator will help other researchers to engage in the process of broadly exploring various aspects of DAG-based blockchain networks and visualize the results to provide valuable insights for the community. Secondly, we are interested in studying the Tangle network scalability, throughput and its resistance to double-spending attacks by leveraging the Tangle simulator.

1.1 Motivation

In order to use BNs for practical applications, understanding various factors such as system scalability, throughput, and its resistance to security attacks is highly important. Traditional blockchains such as Bitcoin are known to be secure from a double-spending attack point of view, but they have limited throughput. For example, Bitcoin requires about ten minutes for a transaction to be included in a block. It is recommended for merchants to wait for six block confirmations which means one hour to complete the payment process. On the other hand, as a new blockchain protocol, the Tangle claims to have high throughput compared to traditional BNs because of its innovative data structure and efficient consensus design for transaction confirmation. However, the security of this distributed ledger has been largely assessed only theoretically. Despite the Tangle's increasing popularity, it did not undergo substantial arms-length validation and analysis. Hence, we are interested in empirically studying this distributed ledger, and we focus on scalability, throughput, and security, which we believe are critical factors to be assessed before the successful adoption of this technology.

1.2 Research Questions

1.2.1 Discrete Event Simulation Design and Feasibility

We take a discrete event simulation (DES) approach for the Tangle simulation. DES models dynamic systems which evolve in time by the occurrence of events at possibly irregular time intervals. It acts quite similar to the queueing system. Each event occurs at a particular instant in

time and marks a change of state in the system. Between consecutive events, no change in the system is assumed to occur. Thus, the simulation time can directly jump to the occurrence time of the next event [43]. An event queue is utilized as the central event management tool in our simulation. Events are created, inserted in a priority queue, and later executed based on their times. Execution of each event might result in creating other types of events. DES approach best suits our simulation since we can simply simulate different BNs' events such as transaction arrival, transaction validation, and transaction propagation. We also follow a model-driven approach, whereby the abstract model can be extended to simulate other BNs without the burden of implementing them from scratch, enhancing the re-usability of the simulator and comparability of different protocols.

RQ1: Is this strategy feasible and up to how many nodes in the network, given a constant value of the inflow of transactions and realistic computational resources and time?

1.2.2 Validation of the Tangle's Main Claims

The Tangle simulator has been developed based on the algorithms proposed by Popov in the corresponding white paper [38]. The author mentions two main properties of the Tangle. First, in a high-load Tangle network, the number of tips fluctuates around an average value (L0) on the examined time interval, and L0 is linearly related to the transaction arrival rate ($\boldsymbol{\lambda}$). The author claims that assuming the time needed for proof of work is one second, L0 is 2 times $\boldsymbol{\lambda}$. However, another paper from IOTA foundation [25] claims that L0 equals 1.25628 times $\boldsymbol{\lambda}$ for the discrete event simulation of the Tangle network. Second, in a high-load Tangle network, the cumulative weight has two phases of growth. The first phase is called the adaptation period (exponential growth) and the second phase is the linear growth.

RQ2: Does the Tangle simulator verify that for a high-load state of the Tangle network, the number of tips becomes stable over time and that it has a linear relationship with $\boldsymbol{\lambda}$? Does it verify the existence of two phases of growth in the cumulative weight?

Verifying these claims offer validity evidence both for the proposed simulation and for the Tangle itself.

1.2.3 Study of the Tangle's Performance and Security Parameters

It is vital to know about the system capacity in processing generated transactions in the Tangle network. It is also very important to know to what extent the system is secure against double-spending attacks. There are various parameters configurable in the Tangle network such as

$\boldsymbol{\lambda}$ (transaction arrival rate), $\boldsymbol{\alpha}$ (randomness factor for the weighted random walk), the network difficulty or MWM (Minimum Weight Magnitude), the network transmission rate, confidence threshold, and the malicious hash power. These parameters affect both the number of confirmed transactions per second, and the Tangle network's resistance to double-spending attacks.

RQ3: From the network configuration perspective, what parameters affect the Tangle's throughput and how these parameters, quantitatively, affect the Tangle network?

RQ4: From the network configuration perspective, what parameters affect the Tangle's security and how these parameters, quantitatively, affect the Tangle network? Can we find a parameter set to have a secure Tangle network against double-spending attacks?

1.3 Methodology

To address the above research questions, we perform several steps. Figure 1 shows the steps of our methodology.

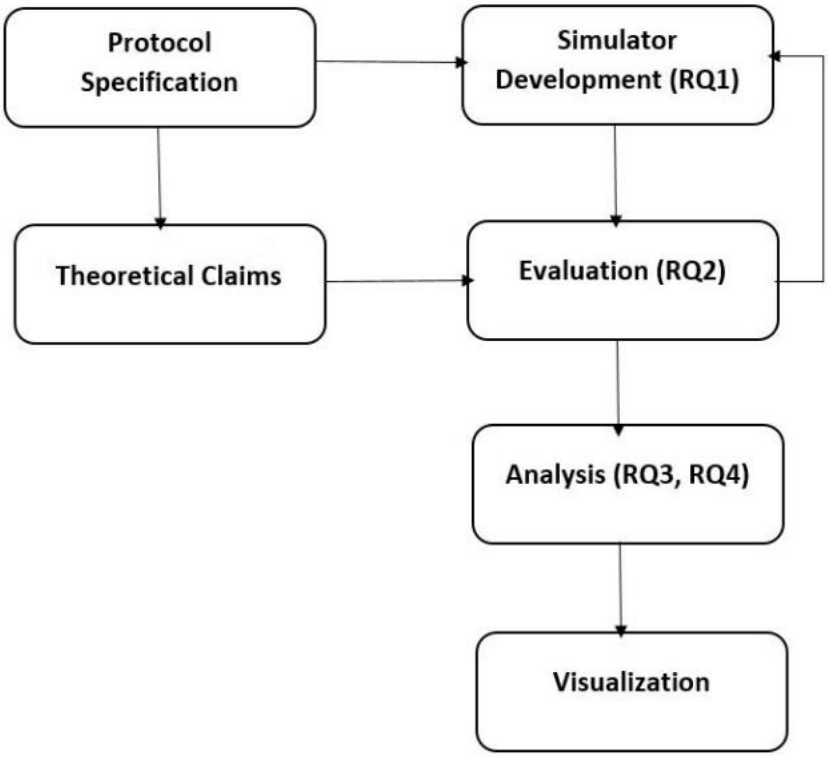


Figure 1: A Scheme of the Methodology

The simulator developed in this research follows the standards of BNs [2], [3], and [38], and processes have been designed according to the specified descriptions. Then, we validate the

simulator against the theoretical claims mentioned in the Tangle white paper [38]. Having a consistent simulation with the theory, we perform analysis on the scalability, throughput, security, and finally visualize the results.

1.4 Contributions

This research has two main contributions. First, we have designed and implemented a Tangle simulator based on model-oriented design and development principles to achieve reusability. This simulator is validated upon the theoretical claims in the Tangle white paper [38] and can help other researchers to engage in the process of broadly exploring various aspects of DAG-based blockchain networks to provide valuable insights for the community. Second, we leverage this simulator to assess three important characteristics of Tangle networks such as scalability, throughput and their security against double-spending attacks. For each of these characteristics, we configure the simulator parameters, run empirical experiments and process and visualize the results which help answering the mentioned research questions in part 1.2.

1.5 Thesis Organization

In this thesis, we first present a detailed overview of the Blockchain networks and the Tangle and then conduct a review on related works on the simulation and analysis of traditional blockchain networks as well as DAG-based blockchain networks in Chapter 2. In the remainder of the thesis, Chapter 3 presents a detailed description of the Tangle simulator model. Chapter 3 also describes the Tangle simulator validation in response to RQ2, and then provides the scalability analysis of the Tangle simulator in response to RQ1. In continuation, Chapter 4 presents the Tangle system throughput and security in response to RQ3 and RQ4. Our experimental results, and main findings are also presented in Chapter 4. Finally, Chapter 5 concludes the thesis and states some potential future directions of the research.

Chapter Two - Background and Related Work

The first part of this chapter states the background of blockchain networks. In the second section, related works both for the traditional blockchains and the Tangle are explained.

2.1 Background

In this section, we provide a detailed description of the Tangle network and we offer a general presentation of blockchain networks focussing on the pioneering Bitcoin network.

2.1.1 Bitcoin Blockchain

Bitcoin blockchain was first introduced by Satoshi Nakamoto in 2008 [2] in order to create the digital cryptocurrency, also called Bitcoin. Bitcoin is still the most popular kind of blockchain both technologically and as a running network. For a more comprehensive understanding of traditional blockchains, in this section we describe how they work and what problems they solve. In the following, some of the main components and properties of the Bitcoin blockchain as well as the transaction life cycle inside the Bitcoin network are explained.

2.1.1.1 Components of the Bitcoin Blockchain

Peer-to-Peer (P2P) Network

Bitcoin network is a peer-to-peer network consisting of many computers/nodes connected together. A network in which participants engage directly over decentralized interactions. Nodes send and receive Bitcoins by broadcasting digitally signed messages to each other in the network.

Blockchain

A blockchain is a public distributed ledger in which transactions assembled in blocks are recorded in a linear chronological order. It is a constantly growing database that has all the mined blocks. Its volume is ever increasing since blocks are mined constantly and are added to the ledger [9].

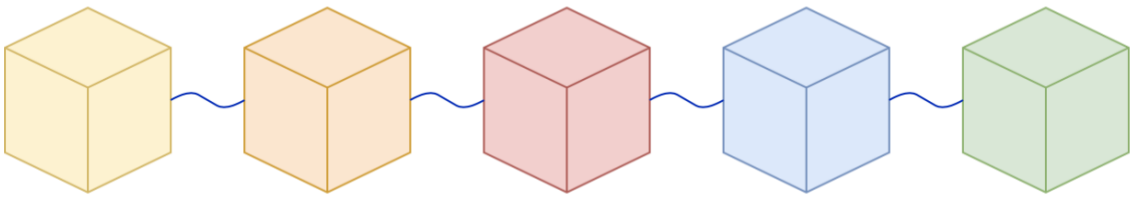


Figure 2: Blockchain as a chain of blocks

Block

As the smallest unit of a blockchain, block is a data structure which assembles transactions. Each block contains some data, hash of the block, hash of the previous block and a nonce value which is adjusted by miners so that the hash of the block has a specific format. Each block's hash value is like its fingerprint that identifies a block and all its content and is always unique.

Node/Miner

Nodes are participants of the peer-to-peer network that keep a local copy of the blockchain to verify that everything is valid and in order. A node in the blockchain network might solve the proof of work and create a new block which in this case is called a miner.

Proof of Work (PoW)

Proof of work is an artifact whose generation requires substantial computational resource investment. In Bitcoin this is hashcash, in which, nodes repeatedly calculate a cryptographic hash (SHA256) of the information to be validated, prefixed by a varying counter – the nonce – aiming at finding a hash output that meets certain criteria – e.g., a certain number of leading zeros [1]. PoW is difficult to compute but easy for other nodes to verify. In the Bitcoin network, it takes about 10 minutes to calculate the required proof of work and add a new block to the chain. This mechanism makes it very difficult for an attacker to tamper with a block because if the attacker tampers with one block, he needs to recalculate the proof of work for all the following blocks, which is very costly. One important factor that involves in proof of work is the network difficulty which is the average number of trials that nodes have to perform to solve the puzzle. The other important factor is a miner's hash rate which is the number of hashes that the node can compute per second.

Consensus

When a new block is created by a miner, it gets propagated to other nodes in the blockchain network. Each node then verifies the block to make sure that it is valid and that it has not been tampered with. If everything is valid, each node adds the new block to its own local copy of the blockchain. Otherwise, the new block is rejected by nodes in the network. Nodes express their acceptance of the block by including its hash in the next block they create.

Transaction

A transaction refers to an exchange between two nodes without the interference of a third intermediary. It can be visualized as a data structure that creates a message initiating a transfer of value or information between nodes taking part in the bitcoin network [9].

Transaction Pool

Transaction pool is the local pool kept within each node containing all the arrived transactions. These transactions are made available to the node for block generation and subsequent mining.

Transaction Fee

The priority of a transaction to be included in a block depends on a number of factors. Transaction fee is one of the most important incentives for a miner to include a particular transaction into the next block [9]. It is a fee that is associated with each transaction. In Bitcoin blockchain, transaction fees depend upon the size of the transaction and not on the transaction amount.

Reward

Miners spend vast amounts of computing power and energy doing proof of work for a financial reward. With every block added to the blockchain comes a block reward, as well as all the fees sent with the transactions that were included in the block.

2.1.1.2 Transaction lifecycle inside the Bitcoin network

Figure 3 explains what happens to a new transaction from its creation to its confirmation in the Bitcoin blockchain network.

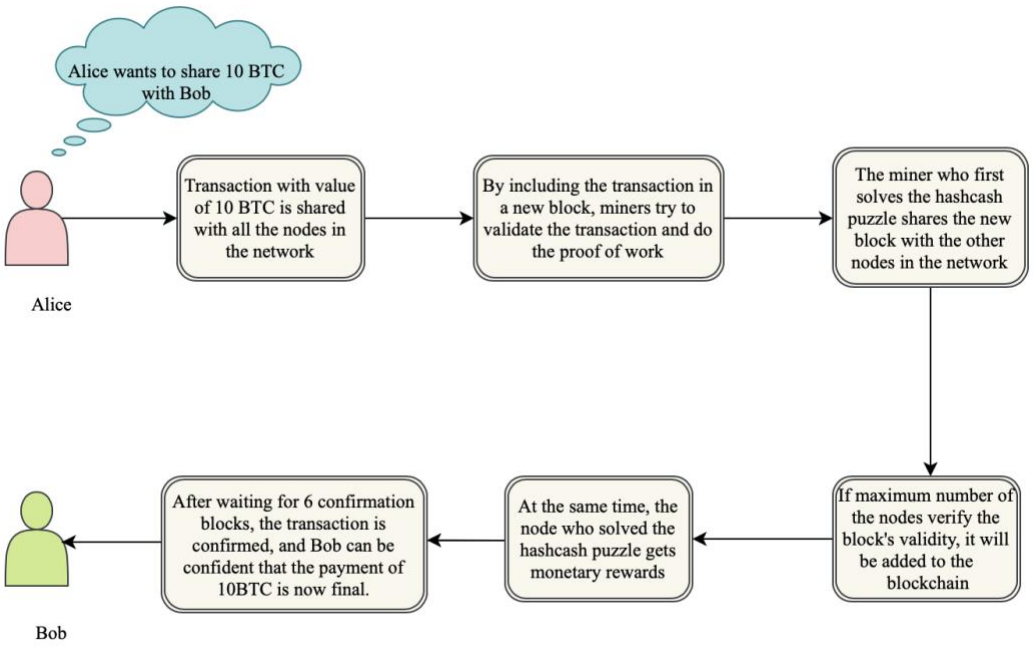


Figure 3: Creation to confirmation of a new transaction in the Bitcoin blockchain network

If the block including the transaction does not get 6 confirmations, the transaction is not confirmed, i.e., transacting parties act as if the payment did not "go through".

2.1.1.3 Properties of the Bitcoin Blockchain

Some of the properties of Bitcoin blockchains that are the reasons for their popularity are mentioned here.

Decentralization

In the Bitcoin network, the entire blockchain is shared among all the computers of the network and no single authority controls. As opposed to a client-server system that is controlled and parameterized by a central authority, in blockchains the peer-to-peer network is controlled by consensus.

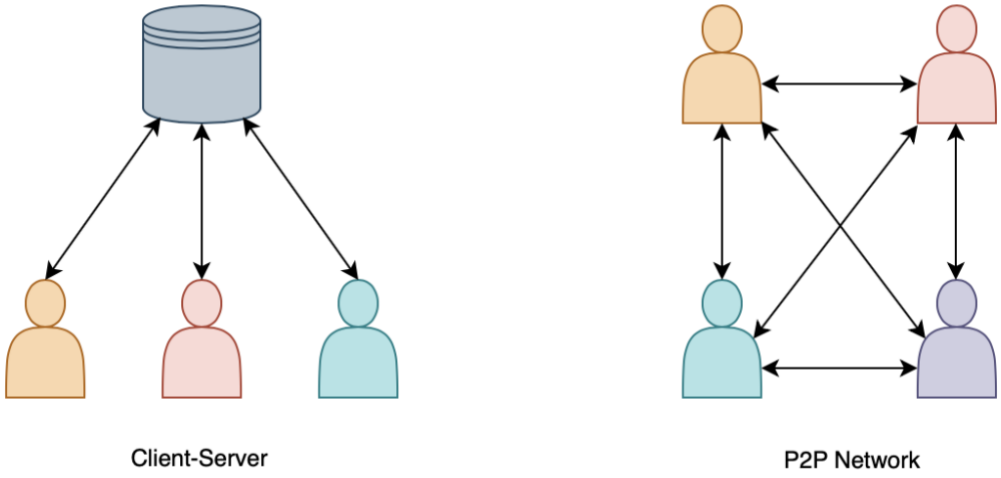


Figure 4: P2P network vs. client-server network

Security

The fact that each block keeps the previous block's hash value makes the Bitcoin blockchain very secure. Once a block is created, its hash gets calculated. Thus, changing something inside the block causes the hash to change and if the fingerprint of a block changes, it is no longer the same block. For example, in Figure 5, block number 3 points to block number 2 and block number 2 points to block number 1.

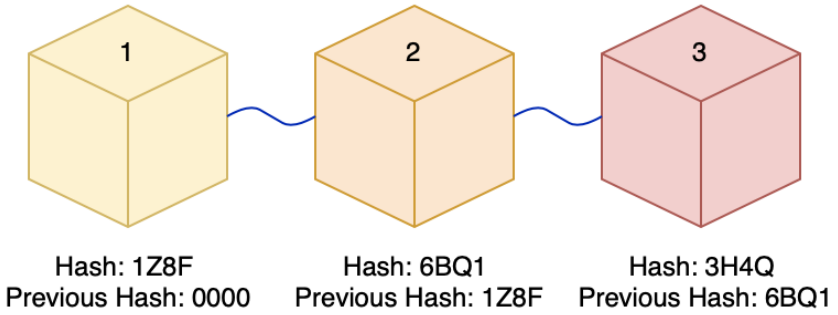


Figure 5: Each block inside the blockchain keeps its own hash and the hash of its previous block

According to Figure 6, if an attacker tampers with the second block, the hash of the second block will change. In turn, that would make block 2 and all the following blocks invalid because they no longer store a valid hash of their previous block.

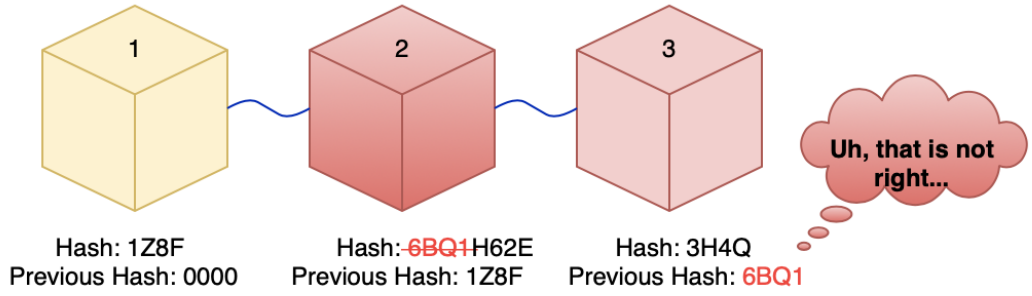


Figure 6: An attacker tampers with block 2

Thus, changing a single block will make all the following blocks invalid. However, using hashes is not enough to avoid tampering. Computers are very fast nowadays and can calculate hundreds of thousands of hashes per second. An attacker can effectively tamper with a block and change all the hash values of the following blocks to make the blockchain valid again. To mitigate this issue, Bitcoin blockchain employs proof of work, as explained in part 2.1.1.1. Thus, security of the Bitcoin blockchain comes from both its creative use of hashing and its proof of work mechanism.

Immutability

Any transaction made in a blockchain cannot be changed or deleted without producing a change in all the subsequent blocks, which is very costly. That is the reason when a transaction is recorded in the blockchain, it is assumed unchangeable.

Provenance

Provenance property of the Bitcoin blockchain refers to the fact that it is possible to track the origin of every transaction inside the blockchain ledger.

Anonymity

Each blockchain network participant has a generated address, not the real user's identity. This keeps users' anonymity, especially in a public blockchain structure. However, there still is publicity meaning that anyone can check the validity of the transactions, which promotes transparency.

2.1.2 The Tangle

In this section, we describe the Tangle as the next evolutionary step of Bitcoin blockchains. According to the Tangle white paper [38], Bitcoin blockchain technology has some drawbacks that

prevent it from being used as a generic blockchain platform across the globe. One such drawback is the concept of the transaction fee for transactions of any value. The importance of micropayments increases in the rapidly developing IoT industry and paying a fee that is larger than the amount of value being transferred seems not logical. However, it is not trivial to simply eliminate fees in the blockchain infrastructure since they serve as an incentive for the miners [38]. Another drawback is the storage such blockchains needed to be saved on each node in the network. For instance, Bitcoin blockchain needs about 250 GB today to get saved on a node [44]. Moreover, Bitcoin blockchains are not quantum-resistant and the elliptic curve signature scheme used by Bitcoin could be completely broken by a quantum computer [45]. A quantum computer can perform the bitcoin proof-of-work using quadratically fewer hashes than are needed classically [45]. Hence, it is easy for an attacker to tamper with a block and recalculate the hash of all the blocks afterwards. More importantly, traditional blockchains still have the throughput issue. They cannot process and confirm high numbers of transactions per second. For example, Bitcoin blockchain can confirm about 7 transactions per second.

Tangle, on the other hand, is an innovative approach that does not incorporate the current blockchain technology. Instead of a chain of blocks, Tangle uses a directed acyclic graph (DAG) as its data structure for storing transactions, allowing transactions to be added in parallel, unlike other blockchain alternatives such as the Bitcoin blockchain. Tangle solves the current problems of the traditional blockchains with some of the important properties it has. First, there is no transaction fee in the Tangle network which makes it proper for the IoT micropayments [38]. Second, in traditional blockchains, nodes need to have a full copy of the chain which becomes larger and larger over time. However, in the Tangle nodes do not need to have a full copy of the DAG to create and verify new transactions. Third, Tangle is quantum resistant since its security comes from its random walk algorithm, not the proof of work [38]. Fourth, there are no limitations on transaction arrival rate in the Tangle network in terms of its throughput. In addition, there is no concept of mining in the sense that miners get monetary rewards and no new coin/token is created since all the tokens are once created in the genesis transaction at the beginning of the Tangle [38] and there is no difference between transaction makers and transaction verifiers in the Tangle network as is in traditional blockchain networks [38].

In the following, we explain some of the components and properties of the Tangle network.

2.1.2.1 Components of the Tangle

Tangle Network

Tangle network is a collection of interconnected nodes that each stores a copy of the DAG. Difficulty and transmission rate are two important properties of the Tangle network. Difficulty refers to the average number of trials for each node to do the PoW and transmission rate is the average propagation rate in bits per second among nodes in the Tangle network.

It is important to observe that the Tangle network is asynchronous. In general, nodes do not necessarily see the same set of transactions in their local version of the DAG. It should also be noted that the Tangle may contain conflicting transactions. The nodes do not have to achieve consensus on which valid transactions have the right to be in the ledger, meaning all of the transactions can be there.

Directed Acyclic Graph (DAG)

DAG is a directed acyclic graph in which transactions issued by nodes in the network constitute the set of the vertices (also called *sites*), while the edge set of the DAG is obtained in the following way: when a node issues a new transaction (i.e. attaches a new transaction to the DAG), the new transaction should approve two unverified transactions in the DAG. These approvals are represented by directed edges, as shown in Figure 7.

In order to issue a transaction, a node does the following:

- The node chooses two other transactions according to the MCMC algorithm, which is described later in this section, and points to them. These two transactions may coincide.
- The node checks if the two transactions are not conflicting and does not approve conflicting transactions. If a node issues a new transaction that approves conflicting transactions, then it risks that other nodes will not approve its new transaction, which will fall into oblivion.
- For a node to issue a valid transaction, the node must solve a cryptographic puzzle (PoW) similar to those in the Bitcoin blockchain [38].

If there is not a directed edge between transaction A and transaction B, but there is a directed path of length at least two from A to B, we say that A indirectly approves B.

Nodes

Nodes are the core of a Tangle network. Each node has a copy of the DAG which hosts all the transactions. Nodes, thus, read from and write to the DAG transaction information. Nodes are responsible for the following:

- Performing the proof of work
- Attaching new transactions to the DAG: When nodes receive a new transaction, they attach it to their local version of the DAG. As a result, at any point in time, all nodes may have different transactions in their local databases. These transactions make up a node's view of the Tangle.
- Deciding which transactions are confirmed: All transactions remain in a pending state until the node establishes the validity of the transaction. One such validity test is when nodes detect double spends and they must decide which transaction should be confirmed and which one should be ignored.
- Selecting tip transactions: All nodes run an algorithm for selecting the so-called tips of the Tangle, which are the vertices of the DAG on which new transactions can be connected.

Transaction

A transaction is a single instruction that can either withdraw cryptocurrencies from a node, deposit them into another node, or have zero-value and contain data, a message, or a signature.

Site

Each vertex in a DAG is called a site. Each site keeps a transaction and its corresponding cumulative weight in the local version of the DAG.

Genesis Transaction

At the beginning of the Tangle, there is a transaction with a balance that contains all of the tokens. The genesis transaction sends these tokens to several other founder addresses. All of the tokens are created in the genesis transaction and no tokens get created afterwards. Therefore, in the Tangle, there is no mining in a sense that miners receive monetary rewards.

Cumulative Weight

Every site in the DAG has a positive value, its own weight, attached to it. To avoid spamming and certain kinds of attacks (see section 4 of the Tangle white paper [38]), it is assumed that sites' own weight is equal to one. Cumulative weight is then defined as the own weight of a particular site plus the sum of own weights of all sites that directly or indirectly point to this site.

Examples of cumulative weight calculation are illustrated in Figure 7, in which the small number in the south-east corner of each box denotes its own weight, and the bold number in the north-west corner of each box denotes the cumulative weight.

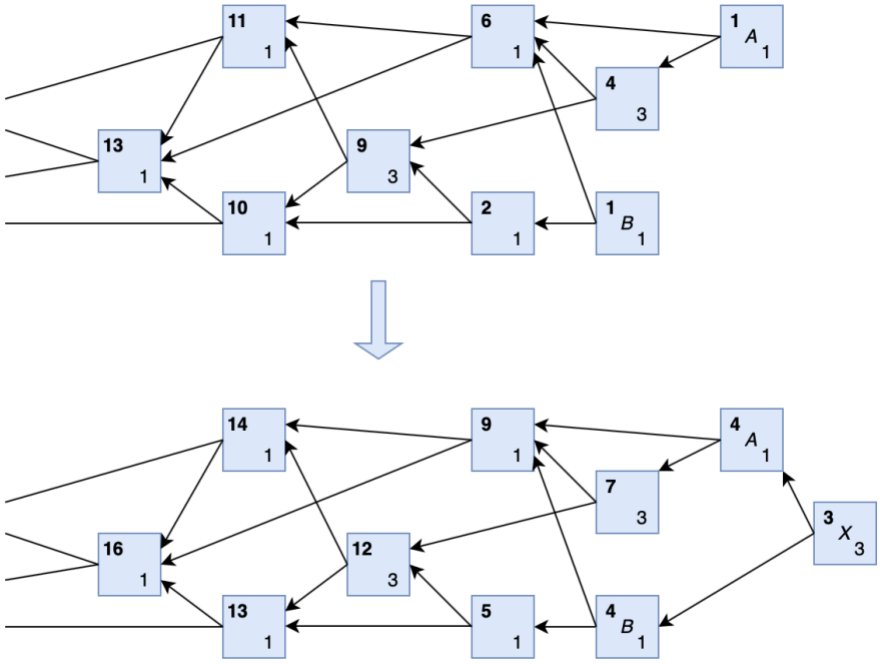


Figure 7: DAG before and after a newly issued transaction (x)

Tips

Unapproved sites in the DAG are called tips which have cumulative weight equal to one. Ideally, we would like to have every site approved. We denote by L(t) the number of tips at time t.

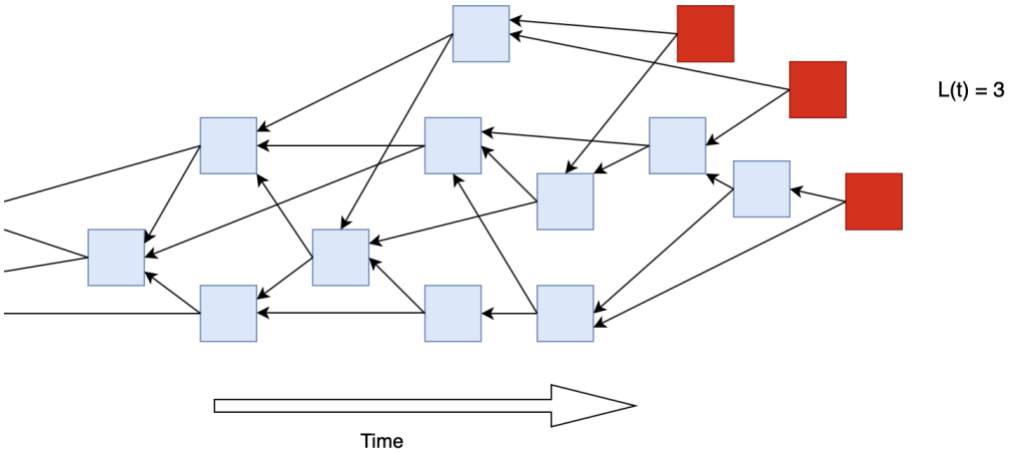


Figure 8: An example of Tangle with the value of L(t) at time t

Lazy Nodes

Lazy nodes are those nodes that do not perform the PoW in order to verify two tips in the Tangle network. A lazy node may always approve a fixed pair of very old transactions, therefore not contributing to the approval of more recent transactions. This way, lazy nodes risk that their transactions will not get approved by other nodes in the network.

2.1.2.2 Properties of the Tangle

Different States of the Tangle Network

There are two states of the Tangle network: low load and high load.

- **Low load:** In this state of the Tangle network, the typical number of tips is small, and frequently becomes one. This may happen when the arrival rate of transactions is so small that it is not probable that several different transactions approve the same tip. Also, if the network latency is very low and nodes compute fast, it is unlikely that many tips would appear. This even holds true in the case when the flow of transactions is reasonably large [38].
- **High load:** In this state of the Tangle network, the typical number of tips is large. This may happen when the flow of transactions is large, and computational delays together with network latency make it likely that several different transactions approve the same tip [38].

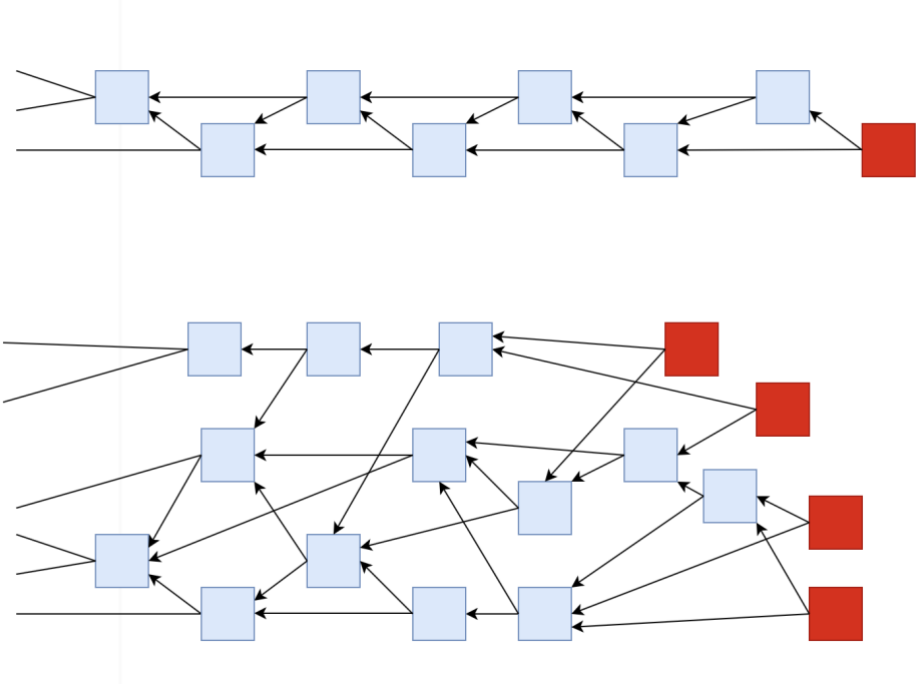


Figure 9: Low-load (top) and high-load (bottom) states of the Tangle network

In Figure 9, blue squares represent verified sites, while red squares represent tips or unverified sites. The situation in the low-load state is relatively simple. Considering $\boldsymbol{\lambda}$ as the transaction arrival rate, the first approval happens on an average timescale of order $1/\boldsymbol{\lambda}$, since one of the first few incoming transactions will approve a given tip.

Stability of the network

In a low-load state of the Tangle network, as mentioned above, the number of the tips is very small and frequently becomes one. For a high-load state of the Tangle network, considering L(t) as the total number of the tips at time t, we expect that L(t) should fluctuate around a constant value, and not escape to infinity. This implies that the limit of $P[L(t) = k]$ as $t \to \infty$ should exist and be positive for all $k \geq 1$. We assume that the number of tips remains roughly stationary in time and is concentrated around a number $L0 > 0$. In the Tangle white paper [38], the author calculates L0 as a function of $\boldsymbol{\lambda}$ and h (PoW time). If a transaction uses the strategy of approving two tips, the typical number of tips is given by $L0 = 2\boldsymbol{\lambda}h$.

Cumulative Weight Growth

In a low-load state of the Tangle network, after a transaction gets approved several times, its cumulative weight will grow with speed $\boldsymbol{\lambda}$ because all new transactions will indirectly reference this transaction. On the other hand, in the high-load state, an old transaction with a large cumulative

weight will experience weight growth with speed $\boldsymbol{\lambda}$ because essentially all new transactions will indirectly reference it. However, when the transaction is first added to the Tangle it may have to wait for some time to be approved. In this time interval, the transaction's cumulative weight behaves in a random fashion.

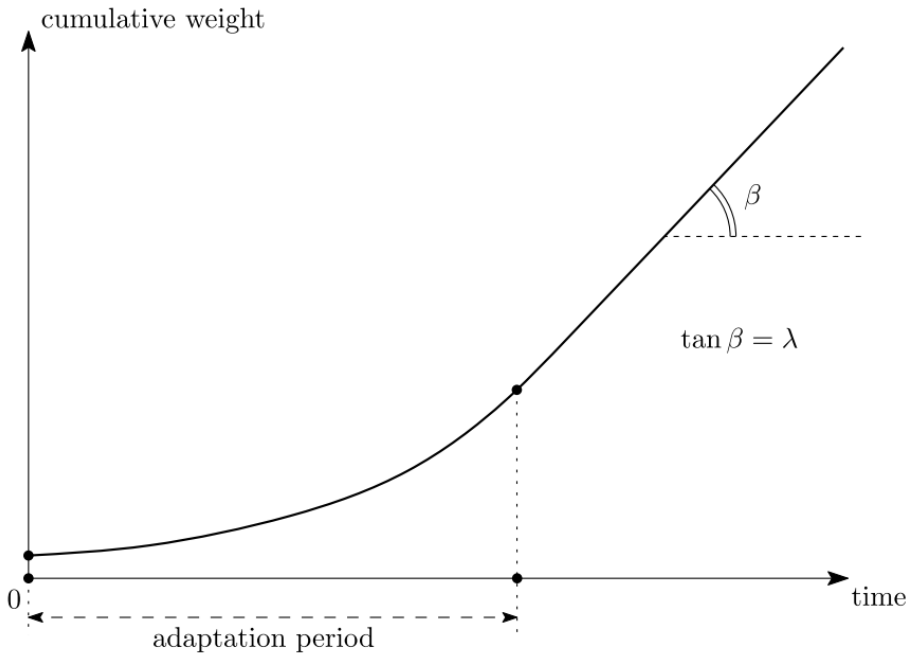


Figure 10: Cumulative weight vs. time for the high-load state of the Tangle network [38]

If we define H(t) as the expected cumulative weight at time t, according to the Tangle white paper [38], H(t) grows exponentially during the adaptation period (see section 3.1(8) of this paper [38]).

$$H(t) \approx 2\exp\left(W\left(\tfrac{1}{2}\right)\frac{t}{h}\right) \approx 2\exp\left(0.352\frac{t}{h}\right)$$

The exponential growth does not mean that the cumulative weight grows very quickly during the adaptation period. Rather, the behavior is as depicted in Figure 10. After the adaptation period, the cumulative weight H(t) grows linearly with speed $\boldsymbol{\lambda}$.

MCMC Algorithm

The MCMC algorithm is described in the following way:

1. Consider all sites on the interval [W, 2W]. W is the cumulative weight and is reasonably large.

2. Independently place N particles (a.k.a. entry points for random walkers) on sites in that interval.
3. Let these particles perform independent discrete-time random walks towards the tips, meaning that a transition from x to y is possible if and only if y approves x ($y \rightarrow x$).
4. The two random walkers that reach the tip set first will sit on the two tips that will be approved.
5. The transition probabilities of the walkers are defined in the following way:

$$P_{xy} = \exp\big(-\alpha(\mathcal{H}_x - \mathcal{H}_y)\big)\Big(\sum_{z:z\rightsquigarrow x} \exp\big(-\alpha(\mathcal{H}_x - \mathcal{H}_z)\big)\Big)^{-1}$$

In this formula, z refers to any site that approves x.

Parasite Chain Attacks

Among possible attack scenarios of the Tangle network, we focus on the parasite chain attack [38] where the attacker tries to outpace the network. In this type of attack, the attacker secretly builds a sub-tangle, including a double-spending transaction, that occasionally references the main Tangle. This specific attack motivates the algorithm which Tangle nodes must use to select tips, the MCMC algorithm. Let us have a closer look at the parasite chain attack. Different steps of this attack are explained in the following:

1. An attacker sends a payment to a merchant and receives the goods after the merchant decides the transaction has a sufficiently large confidence rate, which we see how is calculated later in this chapter.
2. The attacker issues a double-spending transaction, which aims to reverse the transaction that has already been accepted by the merchant for re-spending the same coin.
3. The attacker uses their computing power to issue many transactions that approve the double-spending transaction, but do not approve the original transaction that they sent to the merchant.
4. The attacker hopes that their dishonest sub-tangle outpaces the honest sub-tangle. If this happens, the main Tangle continues growing from the double-spending transaction, and the legitimate branch with the original payment to the merchant is orphaned. Orphaned transactions are not indirectly approved by incoming transactions anymore.

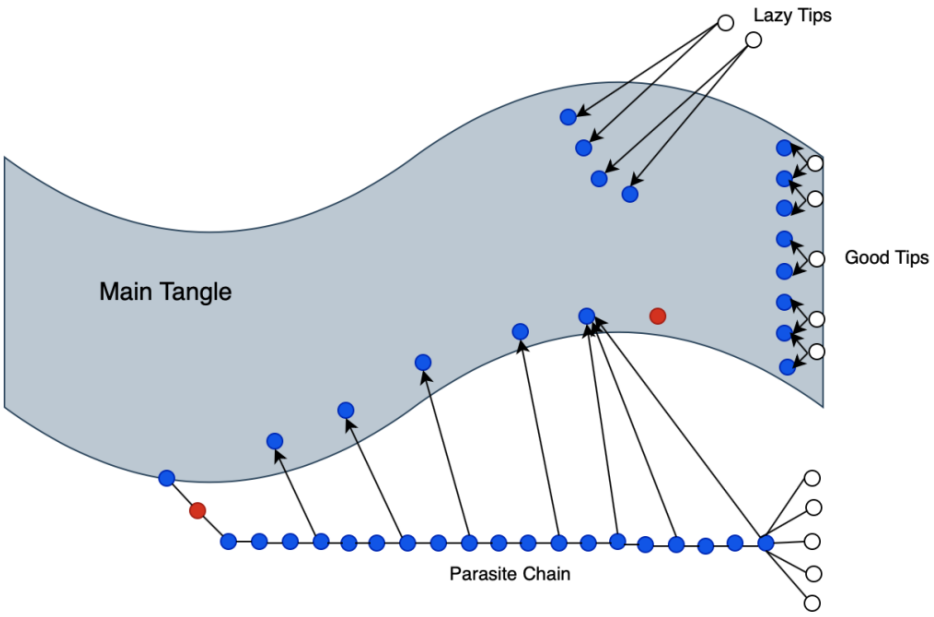


Figure 11: Visual representation of the parasite chain

In Figure 11, the two red circles indicate an attempted double-spending by the attacker.

To look at the attack with more detail, assume that a given transaction gained cumulative weight w0 in t0 time units after the moment when it was issued, and that the adaptation period for that transaction is over. In this situation, the transaction's cumulative weight increases linearly with speed $\boldsymbol{\lambda}$. Now, imagine that the attacker wants to double-spend this transaction. To do so, the attacker secretly prepares the double-spending transaction, and starts generating nonsense transactions that approve the double-spending transaction at the time when the original transaction was issued to the merchant. If the attacker's sub-tangle outpaces the legitimate sub-tangle at some moment after the merchant decides to accept the victim transaction, then the double-spending attack would be successful. If that does not happen, then the double-spending transaction would not be approved by others because the victim transaction would acquire more cumulative weight and essentially all new tips would indirectly approve it. The double-spending transaction would be orphaned in this scenario.

To defend against this attack style, we can use the fact that the main Tangle is supposed to have more active hashing power than the attacker. Therefore, the main Tangle is able to produce larger increases in cumulative weight for more transactions than the attacker. The white paper [38] author claims that the tip approval strategy is the most important ingredient for constructing a Tangle-based network and he suggests the idea of using the Markov Chain Monte Carlo (MCMC)

algorithm to select the two tips to reference. To check that the algorithm works as intended, first consider the lazy tips. These tips intentionally approve some old transactions to avoid performing proof of work. Even if the particle chosen in the DAG to start the random walk is on a site approved by a lazy tip, it is not probable that the lazy tip would be selected as the next move by MCMC algorithm because the difference between cumulative weights would be very large and transition probability (Pxy) would be small [38].

Next, consider the attack style where the attacker hiddenly builds a chain containing a transaction that empties their account balance to another account under their control, indicated as the leftmost red circle in Figure 11. Then, the attacker issues a transaction on the main Tangle, represented by the rightmost red circle, and waits for the merchant to accept it. The hidden parasite chain occasionally references the main Tangle. However, the cumulative weight is not very large in the parasite chain. It should be noted that the parasite chain cannot reference the main Tangle after the victim/merchant's transaction since nodes in the Tangle network check for the consistency in history of the Tangle and will not accept the double-spending transaction as a valid transaction. Furthermore, the attacker might try to artificially inflate the number of tips in their parasite chain at the moment of the attack. The attacker's idea is to make the nodes issuing new transactions reference the parasite chain so that the honest branch of the Tangle will be orphaned. The reason why the MCMC algorithm will not select one of the attacker's tips with high probability is that the sites on the parasite chain will have a cumulative weight that is smaller than the sites that they reference on the main Tangle. Therefore, it has low probability that the random walker jumps to the parasite chain unless it begins there, and this event is not very probable either because the main Tangle contains more sites [38].

Confidence Rate

The MCMC algorithm described above is used by nodes to calculate a transaction's confidence rate [38]:

1- Each node runs the random walk 100 times to select 100 tips in the local version of its DAG.

2- Then the node counts how many of the selected tips directly or indirectly approve the given transaction. If this number is less than 50 the transaction is not yet confirmed, otherwise it is assumed confirmed.

In addition, the main rule that the nodes use when they face two conflicting transactions is the following: nodes run the random walk a hundred times and see which of the two transactions

is more likely to be indirectly approved by the selected tip. For example, if a transaction is selected 97 times during 100 runs of the tip selection algorithm, it is confirmed with 97% confidence.

2.2 Related Works

In this section, we review and compare some of the related efforts done by other researchers to simulate blockchain networks including Bitcoin blockchain and the Tangle distributed ledger. Simulation modeling is the process of creating and analyzing a simulator of a system to learn its behavior and predict its performance in the real world. Since distributed ledger technologies (DLT) are usually complicated and computational resource consuming, it is important to utilize simulation modeling to answer performance related questions such as throughput and security [36].

2.2.1 Blockchain Simulation

A number of efforts have been presented for simulating BNs in order to study them from a variety of angles. First, we review the related works on simulating the blockchain networks such as Bitcoin and Ethereum, and then we compare them from various angles in Table 1.

2.2.1.2 Related Works on Blockchain Simulation

Stoykov et al. introduce VIBES, a simulator originally for the study of Bitcoin-like blockchains [6], [7] which is capable of conducting large-scale network simulations, focusing on scalability and speed. Later, the author introduces another configurable event-driven blockchain simulator for large-scale network simulations for gaining empirical insights into the dynamic properties of Ethereum (eVibes [8]).

Goswami [9], on the other hand, uses a custom-made Java-based simulator to study scalability factors of Bitcoin-style blockchains. The research addresses various mechanisms that can be employed to resolve the scalability limitation of Bitcoin-style blockchains through measuring the differences between the simulator and real time scenarios.

To offer a performance modelling approach to analyse BNs, Yasaweerasinghelage et al. [10] uses simulators built for such purpose (Simu-Com). The authors show the feasibility of using architectural performance modeling and simulation tools to predict the system latency arising from blockchain-related factors, such as the configuration of the number of confirmation blocks and inter-block times. The authors, then, discuss how to leverage this simulation model to support architectural decision-making, during the design of blockchain-based systems.

As a different work, Gervais et al. [11] provide a complete Bitcoin simulator written by NS2. Their work aims at introducing a novel quantitative framework to analyze the security and performance implications of various consensus and network parameters of Proof of Work (PoW) blockchains. They specifically developed a method to fight against the double-spending and selfish mining under their framework.

Kreku et al. [12] re-use the ABSOLUT simulator focussing on power/energy consumption and network latency, specifically in Ethereum. The authors present a way to apply ABSOLUT in evaluating blockchain implementations on embedded IoT devices to help in rapid prototyping of different blockchains on IoT platforms and bring savings in projects that need to develop blockchain implementations.

To study the effect of communication delay in Bitcoin blockchain under a selfish-mine strategy, Gobel et al. [13] develop two blockchain simulators, one in C++ and one in Java, based on the DESMO-J simulation framework [14]. They use discrete-event simulation to study the behaviour of a network of Bitcoin miners, which includes selfish-mine action under a network with communication delay between miners. They found out that if dishonest miners exist, the performance of both honest and dishonest miners will become worse.

Liaskos et al., on the other hand, have employed simple custom-made R-based simulations of PoW competitions to study the role of governance parameters in network sustainability [15], [16]. In the first paper [15], they attempt a simple model for assisting comprehension of and qualitative reasoning about the sustainability of public PoW BNs. The proposed model focuses on the parameters that affect the decision of an individual node to participate in transaction validation. This model aims at understanding how various configurable and observed network parameters interact with each other and how such understanding can help reconfigure the network's protocol to support sustainability in response to external or internal disruptions. Later on [16], they proposed a simple model for quantitatively reasoning about the parameters that affect and are affected by the consensus process of public open-access BNs and attempt a first-cut formulation of such a model as an adaptive system. Then, they demonstrate some of the challenges in designing real-world controllers including limited observability, complex and heterogeneous local decision-making methods and adaptation consensus issues.

A relatively different approach is followed by Neudecker et al. [17] who work by abstracting the Bitcoin reference implementation to evaluate the feasibility and cost of partitioning attacks on the Bitcoin network at full scale of 6,000 nodes. Applying the simulation model, they study the feasibility of an attack aiming to partition the network. They demonstrate the trade off in the costs of the attack with respect to the amount of network resources that must be available to the attacker and the time required to perform the attack.

Wang et al. [18] leverage SimPy 3.0.10 (open source [19]) under python 3.6.0 to demonstrate how to simulate a simple blockchain. They identify key requirements for the simulation software for implementing PoW protocols and define a number of performance metrics to quantify the Quality of Blockchain (QoB). The authors show that it is possible and practical to use a simulation approach to study Blockchain networks with different network sizes and protocols.

Two more recent efforts are BlockSim [20] and SimBlock [21]. The former, an open-source Python project [22], focusses on Bitcoin PoW modelling and simulation. Blocksim is helpful to understand the details in the block generation process and PoW and it provides abstractions that make it easy for designers and analysts to explore design trade-offs and configuration questions related to performance, reliability, security or other properties of interest. However, the authors left the defined test cases validation and verification for future work. The latter, a Java open-source project [23], is also focussing on Bitcoin-like blockchains also considering latency-defining network/region characteristics and their effect on the consensus process. The authors investigate the influence of nodes' behavior on blockchains by conducting some experiments which clarify the influence of neighbor node selection algorithms and relay networks on the block propagation time.

2.2.1.2 A Comparison among Related Works on Blockchain Simulation

Table 1: A comparison among simulations of the blockchain networks

Authors	Language	Blockchain	Method	Validated	Open Source	year
Lyubomir Stoykovet al.	Scala	Bitcoin	discrete-event simulation	-	✓	2017

Aditya Deshpande et al.	Scala	Ethereum	discrete-event simulation	-	✓	2018
Sneha Goswami	Java	Bitcoin	-	-	-	2017
Rajitha Yasaweerasinghelage et al.	Python	Bitcoin	Palladio workbench as a modeling tool	✓	-	2017
Arthur Gervais et al.	-	Bitcoin and Ethereum	Quantitative model, NS2	✓	-	2017
Kreku et al.	-	Ethereum	ABSOLUTE, an existing simulation tool	✓	-	2017
Gobel et al.	Java, C++	Bitcoin	discrete-event simulation	✓	✓	2013
Liaskos et al.	R	Bitcoin	discrete-event simulation	-	-	2019
Neudecker et al.	C++	Bitcoin	discrete-event simulation	✓	-	2015
Wang et al.	Python	Bitcoin	-	-	-	2018

Alharby et al.	Python	Bitcoin	discrete-event simulation	✕	✓	2019
Aoki et al.	Java	Bitcoin	discrete-event simulation	✓	✓	2019

In Table 1, '-' means that the required information could not be found. Symbol '✓' means 'yes' and symbol '✕' means 'no'.

2.2.2 Tangle Simulation

The efforts mentioned in section 2.2.1 primarily focus on original block-based blockchain (BC) architectures that literally involve a blockchain, a chain of blocks, as the main data structure. Recently a different data structure for storing validated transactions has been proposed that is based on Directed Acyclic Graphs (DAGs). In DAGs, transactions are validated individually as vertices of a shared graph. The most prominent DAG is the Tangle [38], the base data structure and consensus protocol behind IOTA cryptocurrency. Compared to BC systems, DAG-based distributed ledger (DL) systems are usually more complicated in terms of data structures and consensus achieving, and so more difficult to understand [41]. Thus, since the introduction of the Tangle, several efforts have been made to simulate this DL technology. Many modeling studies on DAG-based DL have been performed from different performance perspectives, providing good resources for learning about and designing DAG systems. First, we review the related works on simulating the Tangle networks and then we compare them from various angles in Table 2. We also review some of the other related works studying other aspects of the Tangle.

2.2.2.1 Related Works on the Tangle Simulation

To help understand the Tangle, two IOTA Foundation papers [24] and [25] build a discrete model and a continuous time model for IOTA Tangle, respectively. The former gives a first glance of IOTA Tangle by introducing a discrete model and discusses the relationship between cumulative weights of transactions and tip numbers over discrete time steps. They find that in the high-load Tangle networks, the cumulative weight contains two phases of growth, namely the exponential and the linear. The simulation results also reveal that the numbers of tips L(t) as a function of time remains stable under random tip selection strategy. In contrast, for Markov Chain Monte Carlo (MCMC) tip selection strategy, it is stable only for small $\boldsymbol{\alpha}$ (≤ 0.001) in examined time intervals.

The later provides a continuous time simulation model to validate the analytical prediction about the number of tips L(t)=(k/k−1)*$\boldsymbol{\lambda}$h, which is initially proposed by Popov [38]. The authors also explore the cumulative transaction weights and find that there is a non-negligible probability of transactions being left behind for larger values of $\boldsymbol{\alpha}$. For simulator design, they generalize the tip selection number to be k rather than 2 and choose 3k + 4 particles to ensure k distinct tips selected from random walks. Each walk starting position is chosen randomly (with uniform distribution) from transactions issued between 100$\boldsymbol{\lambda}$ and 200$\boldsymbol{\lambda}$ transactions before. According to the simulation results, they empirically conclude that any starting position placed further than 10$\boldsymbol{\lambda}$ to 20$\boldsymbol{\lambda}$ provides the same growth of the number of tips, and it would not influence the tips number. These empirical results could provide directions for further study to improve the simulation efficiency. However, these works do not directly examine or analyze any specific system performance metrics.

M. N. Nguyen [26], attempts to replicate IOTA's Tangle simulations from the paper [25]. The author proposes a solution to create the simulation properties as described in the paper [25]. However, he limits the number of transactions only to a few thousand and the transaction issuing nodes have limited configuration options.

On the other hand, Zander et al. [27] provide DAGsim, an open-source python project [28], which is a simulation of DAG-based distributed ledger where each vertex can be either a transaction or a block. By modeling both honest and semi-honest actors, the simulations show that the agents with lower latency and a higher connection degree have a higher probability of having their transactions accepted in the network.

Fan [29] adopts DAGsim and builds into it the Coordinator confirmation consensus to develop a performance simulation model. The paper leverages the extended simulator to simulate IOTA for conducting study on the performance and re-attachment waiting time on a small-scale network.

Likewise, Lathif et al. [30] propose CIDDS, a configurable open-source [31] DAG-based DLT simulation framework, inspired by VIBES [6], to enable large-scale simulations with thousands-nodes level. Beside the DAG, the system provides simulation outcomes such as the number of unapproved tips and the number of approved transactions with approval confidence. However, the paper does not provide the simulation validation and as future work the authors plan

to extend the simulator with attack simulation functionalities where the effect of malicious nodes can be investigated.

As another work, Sanders et al. develop a simulator to study number of the tips, cumulative weight, first transaction approval time and computational overhead of the Tangle as a function of Tangle size based on various tip selection algorithms, focused on the Uniform Random Tip Selection (URTS) which selects tips with uniform probability from the set of available tips, and the Unbiased Random Walk (URW) which uses a random walk beginning at the genesis [32].

In order to illustrate and visualize the Tangle, Gal [33] develops an IOTA visualization simulator. Through this simulator, one could intuitively observe different Tangles generated under various tip selection algorithms such as uniformed random, weighted and unweighted random walk. In the weighted random walk, the simulator could show how the model parameters such as arrival rate $\boldsymbol{\lambda}$ and the built-in randomness factor $\boldsymbol{\alpha}$ change the Tangle shape and the transaction confirmation ratio. However, the total transaction number in this simulator is limited to 500 and performance metrics are not quantified.

Moreover, Bottone et al. [34] present and develop an extendable multi-agent simulator, an open-source simulation environment, in which the authors employed NetLogo [35] to provide a 3D visualization of the Tangle.

To improve the tip selection algorithm on the Tangle, Ferraro et al. [46] develop a new tip selection algorithm that ensures that all honest transactions get eventually approved, while at the same time preserves the basic properties of the MCMC tip selection algorithm.

Chafjiri et al. [47], on the other hand, presented a new random walk algorithm having the benefits of both unweighted random walk algorithm and weighted MCMC algorithm at the same time. The authors claim that an unweighted random walk algorithm can approve transactions nearly proportional to the time of their arrivals while a weighted algorithm can better defend against lazy and malicious transactions.

Bramas [48] theoretically studies the security aspect of the Tangle. He claims and mathematically proves that in order for the Tangle to be secure, the hashing power of honest nodes should be constantly used to generate sites and that it is strictly greater than the hashing power of the adversary.

Cullen et al. [49] analyze the efficacy of IOTA's core MCMC algorithm using a matrix model for the parasite chain attack and present an extension that utilises the first order time

derivative of the cumulative weight in calculating the jumping probabilities which improves the ledger's resistance to these types of attacks.

As can be seen, none of these related works provide scalability analysis of the Tangle network in terms of network ability to simulate different numbers of nodes, given realistic computing resources. Moreover, none of the above papers simulated the weighted MCMC random walk to investigate transaction throughput. And none of them simulate malicious actors in the Tangle network to empirically explore the resistance of the network to double-spending attacks. In this research, we analyze scalability and throughput of the Tangle network in different parameter settings. In addition, we simulate an attacker in the Tangle network in order to assess various parameters influencing the network security against double-spending attacks.

2.2.2.2 A Comparison among Related Works on the Tangle Simulation

Table 2: A comparison among simulations of the Tangle networks

Authors	Language	Blockchain	Method	Validated	Open Source	year
B. Ku´smierz	-	Tangle	discrete-event simulation	✓	-	2017
B. Ku´smierz et al.	-	Tangle	continuous-time simulation	✓	-	2018
M. N. Nguyen	Python	Tangle	continuous-time simulation	-	✓	2018
Zander et al.	Python	Tangle	continuous-time simulation	-	✓	2019
Fan	Python	Tangle	continuous-time simulation	✓	✓	2019
Lathif et al.	Python	Tangle	-	-	✓	2018
Sanders et al.	-	Tangle	continuous-time simulation	-	-	2020
Gal	JavaScript	Tangle	-	✓	✓	2018

Bottone et al.	Java	Tangle	NetLogo as a well-known, widely used, cross-platform modelling and simulation environment	-	✓	2018

In Table 2, '-' means that the required information could not be found. Symbol '✓' means 'yes' and symbol '✕' means 'no'.

Chapter Three – Simulation Implementation and Validation

In this chapter, we explain different components of the Tangle simulator and the way we implemented them. In addition, we provide the details of how the simulator gets validated.

3.1 Simulation Implementation

To design and implement the Tangle simulator, we follow the model-oriented design and development principles whereby an abstract conceptualization of the simulator is first devised. The Tangle simulator is based on an instantiation of this abstract model [5]. The abstract conceptualization (the UML class diagram) can be seen in Figure 12. In the following, we explain the different components of our Tangle simulator, how they work and how they relate to each other.

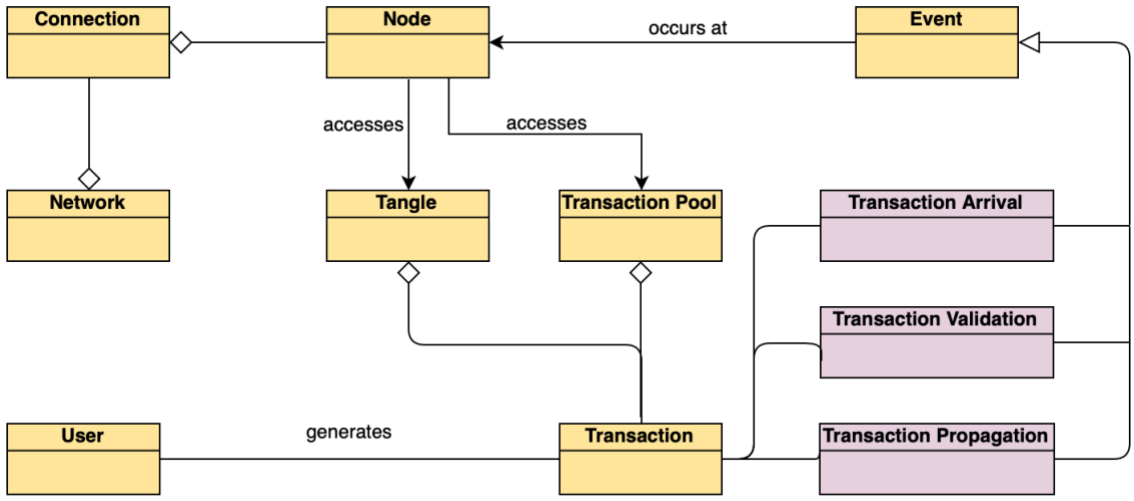


Figure 12: A Meta-Model for the Tangle Simulation

3.1.1 A Network of Nodes

The Tangle network consists of connections between two or more nodes. Each node has a Transaction Pool, which contains the Transactions arrived at the node as well as a local version of the Tangle or DAG consisting of the transactions. Nodes use their Transaction Pool to validate incoming Transactions in order of their arrival time. Each node has a specific value of hash power which shows how many trials per second the node can perform to solve the PoW in order to validate each incoming transaction.

We also model the network transmission rate between any two nodes in the Tangle network. We define a network $N \equiv (V, E, f)$ where V is the set of nodes, $E \subseteq V \times V$ is the set of edges, and $f: V \times V \rightarrow R$ is a function that determines the transmission rate value. We enforce $(v1, v2) \in E \Leftrightarrow (v2, v1) \in E$, i.e., N is undirected, $f(v1, v2) \geq 0$ (non-negative weights), $f(v1, v1) = 0$ (zero self-distance) [27]. Here, for simplicity, we assume that all the nodes in the network are connected ($E = V \times V$). Network transmission rates between the nodes are stored in a matrix. Values in this matrix are obtained by f which is a random number generator with normal distribution.

3.1.2 The DAG Data Structure

The DAG is a directed acyclic graph (V, E) a.k.a. the Tangle with the following properties [27]:

- The root of the DAG is called genesis (G), such that $degout(v) \geq 1$ for all $v \in V \backslash \{G\}$, and $degout(G) = 0$. degout(v) function returns the number of outgoing edges from vertex v.
- Any $v \in V$ such that v != G references G.
- At time $t \geq 0$ the state of the DAG ledger is $L(t) \equiv (V(t), E(t))$, where V(t) is the set of vertices, and E(t) is the multi-set of directed edges in the DAG. Furthermore, the following dynamics hold:
    - The initial state of the DAG at $t = 0$ is defined by $V(0) = \{G\}$, $E(0) = /0$.
    - Over time, the DAG grows: if $0 < t1 < t2$ then $V(t1) \subseteq V(t2)$ and $E(t1) \subseteq E(t2)$.
    - Transactions arrive with a rate $\boldsymbol{\lambda} > 0$ according to a Poisson process and they become vertices of the DAG.
    - A node attaches a vertex v (i.e. a transaction) to the Tangle. Later, the consensus protocol votes on confirming the vertex.

We remind that there is no concept of the block in the Tangle. Thus, every single transaction is directly attached to the DAG data structure as a vertex in this graph, see Figure 13. [29].

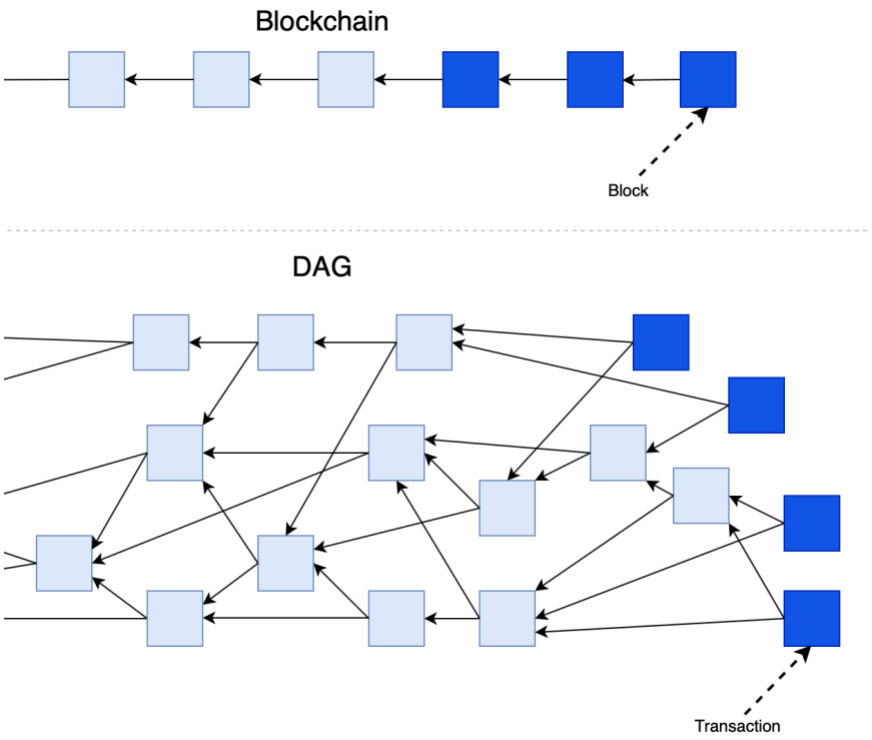


Figure 13: Blockchain vs. DAG

3.1.3 Transaction flow during the simulation

Each Transaction in the Tangle simulator has a specific life cycle from creation to confirmation/ rejection. There are six steps in the Transaction flow.

- **Transaction creation**: Transaction arrives at a randomly selected node in the network. For each incoming Transaction, a Site is created. Each Site includes the corresponding Transaction and its cumulative weight. Sites, then, constitute the vertex set of a DAG.
- **Tip selection**: Each new Site in DAG should approve two other unverified Sites (tips) in the node's local version of the DAG.
- **Proof of work**: After selecting two tips, the node tries to find a proper nonce value and compute a hash value that has a specific format.
- **Transaction validation**: After selecting the two tips and computing the PoW, Transaction is considered validated and it is ready to get propagated to other nodes in the network.
- **Transaction propagation**: After validating the Transaction, the node uses a gossip protocol to propagate the Transaction to all other nodes in the network.

- **Transaction confirmation/ rejection**: After receiving the Transaction, each node creates a new Site including the propagated Transaction and attaches the Site to the DAG. Each node, then, calculates the confirmation rate of the Transaction (according to the consensus protocol in section 3.1.5) to see if the Transaction is confirmed or not.

3.1.4 Events

The Tangle simulator is based on the discrete-event simulation (DES). DES models the operation of a system as a discrete sequence of events in time. Each event occurs at a particular instant in time and marks a change of state in the system. Between consecutive events, no change in the system is assumed to occur. Thus, the simulation time can directly jump to the occurrence time of the next event [43]. Nodes in the Tangle network witness a variety of events including New Transaction Arrival, Transaction Validation, and Transaction Propagation.

Design of the Tangle simulator relates to the queueing theory in a sense that transaction arrivals are modeled based on Poisson distribution which means that transaction inter-arrival times follow an exponential distribution and $\boldsymbol{\lambda}$ is the rate of the new transaction arrival in the Poisson process. Then, the service time for each transaction which equals the validation time plus propagation time of the transaction is modeled based on the normal distribution in the Tangle simulator. However, a complete queue-theoretical analysis of the Tangle simulator design is a topic for future investigations.

Events in the Tangle simulator form a chronologically ordered queue and get processed one at a time, the result of such processing might be the scheduling of more events in the simulated future. We benefit from priority queue in Java which is a special type of queue wherein all the elements are ordered as per their time of creation. Thus, when an element from the priority queue is popped, the least element according to the specified ordering is removed first. Each node responds to the New Transaction Arrival event by creating a new Transaction Validation event. In modeling the Tangle, a Transaction Validation event represents a successful solution of the hash cash "puzzle" [5]. Upon occurrence, the node propagates the Transaction through the network through scheduling Transaction Propagation events for other nodes in the network.

3.1.5 Transaction Confirmation

In the Tangle, confirmation confidence comes in the form of a confirmation degree rather than binary confirmed or not [29]. Thus, confirmation consensus relies on a threshold to define

confirmation. In the Tangle simulator, we assume that this threshold is 50%. Then, for a specific transaction X, the calculation of its confirmation confidence in a node is determined using the steps mentioned in section 2.1.2.2 (Confidence Rate).

3.1.6 MCMC Tip Selection Algorithm

In the Tangle network, the number of tips increases [29] as new transactions keep adding to the ledger. It becomes crucial how to wisely select two tips from all the tips so that the whole Tangle can grow in a stable way i.e. the number of tips fluctuates around a constant value. Let us first review how DAG develops with the arrival of transactions.

Entry Point: There must be a starting position to start random walking. It is easy to think of taking the genesis vertex in the DAG as the entry point. However, this will bring obvious efficiency decrease while traversing a long path when the graph grows larger. In the Tangle simulator, an entry point is a random position from the first half part of the Tangle for each random walk.

Cumulative Weight: When a new transaction comes to the DAG, all previous transactions connected to it in the subgraph get updated by adding the new transaction's own weight to their cumulative weight. For simplicity, we have assumed that the own weight of all transactions in the Tangle network is equal to 1.

MCMC Random Walk: Having the entry point and current Tangle state of cumulative weights for all sites, a random walker walks through the DAG, from an entry point to reach a tip, so as to get the tip in a probability determined by the cumulative weights and a randomness parameter, called $\boldsymbol{\alpha}$.

3.1.7 Design and Implementation Details

Each Simulation is executed as follows. Firstly, we initialize the Nodes and Transactions instances. Then, we create New Transaction Arrival events when each Transaction arrives at a randomly selected Node and put all these events inside a priority queue. The simulator iterates over all New Transaction Arrival events in the queue. This is, in fact, the main simulation loop, i.e. the arrival of each transaction is one event. Arrival times of incoming transactions are sampled from the exponential distribution according to the rate of incoming transactions $\boldsymbol{\lambda}$.

Let h be the average time a device needs to perform calculations that are required to issue a transaction (mainly PoW time). Together with 'h', the network latencies are crucial in deciding

which part of the network is visible for a node at a certain point in time during the simulation. We designed the network somehow that agents have different computational powers, so h might be different for various nodes and transactions. Each node in every iteration of the main simulation loop deals with the currently visible part of the DAG, i.e. it is bounded by the network latencies and PoW time.

Here is the pseudocode of the main loop in the Tangle simulator. The queue inside the while loop is a queue of events. Events are pulled out from the queue and executed one by one.

```
Function run(){
  Event e;
  while (!queue.isEmpty()){ // while queue is not empty, continue the loop
    e = queue.poll(); // remove the last Event from the queue and put it in e
    Globals.currTime = e.getTime(); // change the system global time to the event time
    e.happen(); // execute the logic behind the Event e
  }
  Log(); //log anything needed to output files
}
```

3.2 Simulation Validation

The validity of the developed Tangle simulator is investigated at two levels (as per RQ2):

- **Implementation-Level:** We follow established software engineering practices to ensure that errors and omissions do not happen at implementation level. Specific measures include code reviews with the rest of the Enterprise Systems Group (ESG) and interdependent testing processes. Figure 14 shows a part of the code review documentation we utilized to track and solve bugs/changes during and after the simulator implementation.
- **Design-Level:** The Tangle is compared with the theoretical claims mentioned in [24], [25], [38] via experimental data. We assess if the number of tips has a linear relationship with

transaction arrival rate in a high-load state of the Tangle network. We also evaluate if cumulative weight in the simulator shows two phases of growth (exponential and linear).

3.2.1 Simulation Implementation Validation

Three steps have been done to validate the simulator for code quality and correctness of the algorithms.

1. **Code Review:** We organized regular semi-formal meetings [50] to review the Tangle simulator code. Before each code review meeting, the deliverable code was circulated to the review audience so that they can familiarize themselves with the deliverable, and then a meeting with the reviewers was scheduled. During the meeting, each section of the code was explained: what each section is aiming to achieve. Code reviews were both focused on coding standards and finding defects in the simulator code. Then, errors, decisions, and any action items were recorded on an excel document (Figure 14), and a meeting outcome was agreed. Agreed items were addressed afterwards and if the meeting outcome was that a second review should be held, then another meeting was scheduled. Our meetings included four reviewers, all from the MAIST lab and we had five meetings in total to review main parts of the Tangle simulator such as implementation of different events (New Transaction Arrival, Transaction Validation and Transaction Propagation), the MCMC tip selection algorithm, transactions' confidence rate calculation, the malicious node creation process, parasite chain creation and revealing, and R scripts used to plot diagrams based on log files.
2. **Qualitative Analysis:** We read the papers, websites and forums [24], [25], [38], [51]-[54] to understand and analyze all the algorithms behind the Tangle, also aided, when applicable, by visualizations.
3. **Quantitative Analysis:** We plot the results of the experiments, which show the same patterns expected according to the theoretical claims [38].

Date	Type	Class	Method	Priority	Review Notes	Status
2020-04-23	Reusability	Simulation.java	addNodes(int num)	Medium	keep the malicious node in the NodeSet	Completed
2020-04-23	Reusability	Simulation.java	addNodes(int num)	Medium	keep the legitimate and malicious txs in the TrasactionSet	Completed
2020-04-23	Reusability	NodeSet.java	LegitimateTxId	Medium	middle tx in the txset	Completed
2020-04-23	Reusability	Simulation.java	run()	Medium	all the codes belonging to calcConfidenceRate is part of the malNode job and go	Completed
2020-04-23	Reusability	Simulation.java	run()	Medium	create 3D matrix in Event.java	Completed
2020-04-27	Reusability	Node.java	calcConfidenceRate	Medium	should go to the data structure-use signatures in the blockchain.java	Completed
2020-04-27	Reusability	Transaction.java		Medium	clean state from transaction.java	Completed
2020-04-27	Reusability	Transaction.java	node_site_list	Medium	site refers to node, site referes to tx, tx keeps a siteList	Completed
2020-04-27	Reusability	Transaction.java		Medium	Is the tx in the tangle compatible with tx in bitcoin? the same interface?	Completed
2020-04-27	Reusability	Node.java	isTxApprovedByTheTip	Medium	isSiteApprovedByTheTip - check everywhere for this	Completed
2020-04-27	Reusability	Node.java	calcConfidenceRate	Medium	change 100 in the calcConfRate to a parameter	Completed
2020-04-27	Reusability	Node.java	calcConfidenceRate	Medium	calcConfidenceRate and isSiteApprovedByTheTip should be tested	Completed
2020-04-27	Reusability	Transaction.java		Medium	change the name of getTipList - it is vague	Completed
2020-04-27	Reusability	Transaction.java	TransactionValidation.java	Medium	create a txPool and change timing of txValidation event - a function: validateTx	Completed
2020-04-27	Reusability	Transaction.java	TransactionValidation.java	Medium	change the validationEvent for mal node (add the propagationEvents in a list and	Completed
2020-04-27	Reusability	Transaction.java		Medium	mal node does not validate honest txs except the one it wants to dbl spend	Completed
2020-04-30	Reusability	NewTransactionArrival	happen()	Medium	put the code inside the node	Completed
2020-09-09	Reusability	Tangle.java	isSiteApprovedByTheTip	Medium	change flag to found and alreadyUpdated to alreadyVisited	Completed
2020-09-09	Reusability	Tangle.java	isSiteApprovedByTheTip	Medium	make it more readable by defining another function on top while removing the las	Completed
2020-09-09	Reusability	TransactionSet.java		Medium	change legitimateTx to victimTx	Completed
2020-09-09	Reusability	Node.java		Medium	in attacking mode, if the attacker receives honest txs, it acts like fake txs with the	Completed

Figure 14: Code review documentation

3.2.2 Simulation Design Validation

This part explains different behaviours we expect from our Tangle simulator. We present some results obtained from the experiments on the Tangle simulator and then we validate the simulator by comparing these results with the theoretical claims mentioned in the Tangle white paper [38] and two other papers [24], [25] from IOTA Foundation on simulation of the Tangle. We confirm the behaviour of the number of tips as a function of time and behaviour of the cumulative weight growth versus time both for the high-load and low-load states of the Tangle network. In fact, according to [24], [25], [38] for our simulator to be valid it has to satisfy the following theoretically established properties.

1. **Hypothesis 1:** The number of tips fluctuates around a constant value for small values of $\boldsymbol{\alpha}$ and all values of $\boldsymbol{\lambda}$ during the simulation time in high-load Tangle networks. However, there are theoretical arguments [24], [25] that for larger values of $\boldsymbol{\alpha}$, the number of tips actually diverges and will become infinite in the limit of large time.
2. **Hypothesis 2:** The average number of tips in a high-load Tangle network has a linear relationship with the transaction arrival rate ($\boldsymbol{\lambda}$) assuming that PoW time (h) is one second.
3. **Hypothesis 3:** There are just a few tips in a low-load Tangle network for all values of $\boldsymbol{\alpha}$ and $\boldsymbol{\lambda}$ during the simulation time.
4. **Hypothesis 4:** Cumulative weight of transactions reveals two distinct growth phases in a high-load Tangle network for all values of $\boldsymbol{\alpha}$ and $\boldsymbol{\lambda}$ during the simulation time. The first phase is called adaptation period which has an exponential growth (according to

the white paper [38], section 3.1(8)) and the second phase has a linear growth with slope $\boldsymbol{\lambda}$w, where w is the mean weight of a generic transaction.

5. **Hypothesis 5:** Cumulative weight of transactions shows one phase of growth, linear growth with slope $\boldsymbol{\lambda}$w where w is the mean weight of a generic transaction, in a low-load Tangle network for all values of $\boldsymbol{\alpha}$ and $\boldsymbol{\lambda}$ during the simulation time.

To collect simulation data, we run more than 30 simulations with $\boldsymbol{\lambda}$ values of 10, 20, 30, 40, 50, 60, 70, 80, 90 and 100, and $\boldsymbol{\alpha}$ values of 0, 0.0001, 0.001, 0.01, 0.1, 1 and 10. All the experiments are based on 30-minute running of the Tangle network except for showing the linear relationship between the average number of the tips and $\boldsymbol{\lambda}$ where experiments are based on 10-minute running of the Tangle network because the number of tips gets stable soon after around 10 seconds of running the Tangle network simulation.

Moreover, in each experiment, we consider 10 nodes and up to 60,000 transactions. For the high-load state of the Tangle, transmission rate between each two nodes is normally distributed at an average of 10,000 bits/sec and standard deviation of 1,000 bits/sec. Likewise, for the low-load state of the Tangle, transmission rate between each two nodes is normally distributed at an average of 1,000,000,000 bits/sec and standard deviation of 1,000 bits/sec. Values of the network difficulty and nodes' hash power is not important because we have assumed that PoW time is one second. It is also important to notice that all values of the number of tips and cumulative weight in each simulation time is the average value of the number of tips and cumulative weight of all ten nodes of the Tangle network.

3.2.2.1 Stability of the system (the number of tips)

In this section, we show that our Tangle simulator confirms the first three hypotheses mentioned above. According to the Tangle white paper [38], L(t) is the number of tips in the Tangle at time t which is expected to remain stable in time. More precisely, it is expected that L(t) be positive recurrent, which implies that the limit of $P[L(t) = k]$ as $t \to \infty$ should exist and be positive for all $k \geq 1$. We expect that L(t) fluctuates around a constant value, and not escape to infinity since if L(t) escapes to infinity, many transactions would be left unapproved perpetually.

To analyse the stability properties of L(t), we consider the same assumptions mentioned in the white paper [38].

1. The first assumption is that transactions are issued by a number of roughly independent nodes, so the process of incoming transactions can be modelled by a Poisson point process

that is the transaction inter-arrival time follows an exponential distribution. $\boldsymbol{\lambda}$ is the rate of the new transaction arrival in the Poisson process.

2. Second, for simplicity we assume that $\boldsymbol{\lambda}$ remains constant during the simulation time.
3. The third assumption is that all devices have the same computing power so that h = 1.
4. Fourth, we assume that the own weight of each transaction is equal to 1.
5. The last assumption is that all nodes behave the same in the Tangle network. To issue a transaction, a node chooses two tips with MCMC tip selection algorithm and approves them. However, while this implies that the average number of direct approvers is equal to two, the actual number of approvers can vary significantly between transactions in the Tangle [53].

3.2.2.1.1 Tip counts in the high-load state of the Tangle network (hypothesis 1)

In this part, we examine the number of tips in a high-load Tangle network for various values of $\boldsymbol{\alpha}$ and we observe that hypothesis 1 is valid in our Tangle simulator. Average and standard deviation of the number of tips for different values of $\boldsymbol{\alpha}$ in 30-minute running of the high-load Tangle network simulations with $\boldsymbol{\lambda}$ = 10 and 18,000 transactions can be seen in Table 3.

Table 3: Average and standard deviation of the number of tips for various values of $\boldsymbol{\alpha}$

α	**Average number of tips**	**Standard deviation of the number of tips**
0	11.33908	2.442012
0.0001	11.44702	2.276785
0.001	11.42208	2.310337
0.01	11.80169	2.253086
0.1	16.64326	4.503251
1	58.76637	29.73114
10	97.06706	51.55213

On one hand, for smaller values of $\boldsymbol{\alpha}$ such as $\boldsymbol{\alpha}$ = 0, $\boldsymbol{\alpha}$ = 0.0001, $\boldsymbol{\alpha}$ = 0.001 and $\boldsymbol{\alpha}$ = 0.01, L(t) fluctuates around a constant value and remains stable over the examined time interval. We see

the behaviour of L(t) as a function of time and we observe that although the number of tips varies against time, it fluctuates around an average L0. Figure 15, 16 as two examples show that L(t) remains stable in the simulation time.

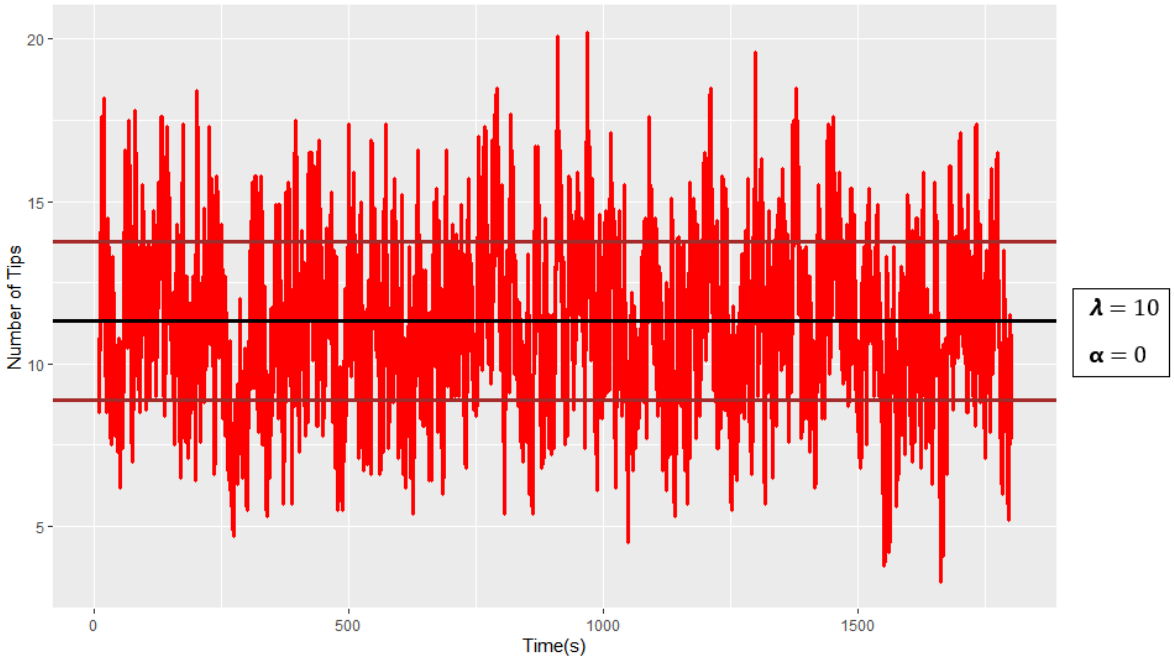


Figure 15: Values of L(t) in a high-load Tangle network as a function of time with $\alpha = 0$ *and* $\lambda = 10$

In Figure 15, we see that L(t) fluctuates around an average value (black line) and most of the points are no further from average than standard deviation (brown lines). Here, according to the Tangle literature [54] the number of tips is recurrent.

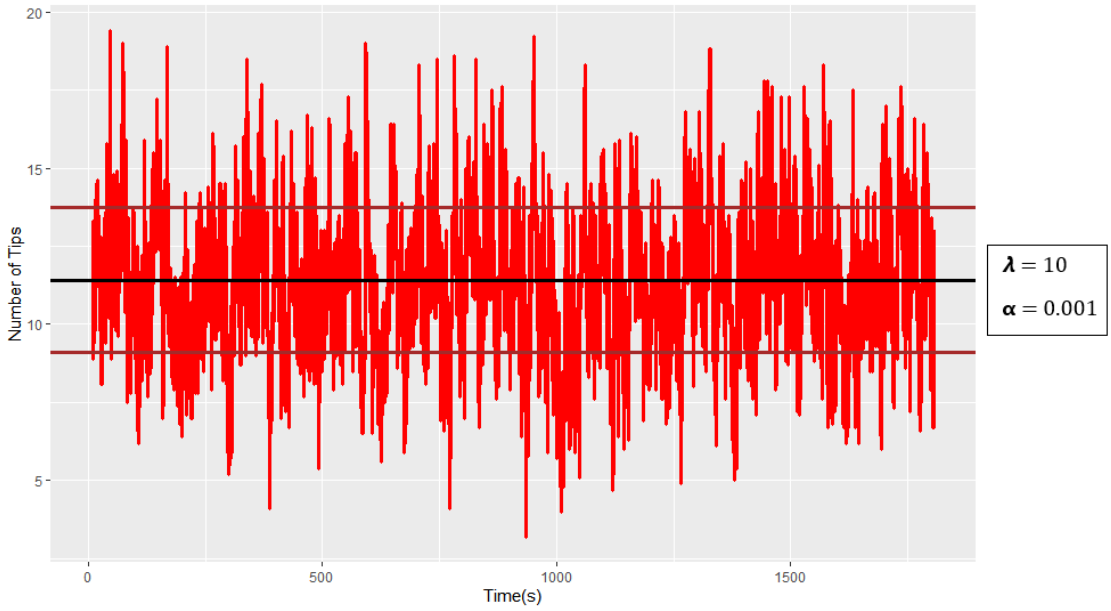


Figure 16: Values of L(t) in a high-load Tangle network as a function of time with $\alpha = 0.001$ *and* $\lambda = 10$

In Figure 16, we observe that L(t) is recurrent [54] and fluctuates around an average value (black line) and most of the points are no further from average than standard deviation (brown lines).

On the other hand, larger values of α increases the probability to select heavier tips so that new coming transactions will be attached to the heaviest path eventually [29], and other transactions attached to other parts of the Tangle remain as tips. As can be seen in Figure 17, the same pattern happens in our simulation. When α gets larger values such as $\alpha = 0.1$, $\alpha = 1$ and $\alpha = 10$, the average number of the tips increases over the simulation time.

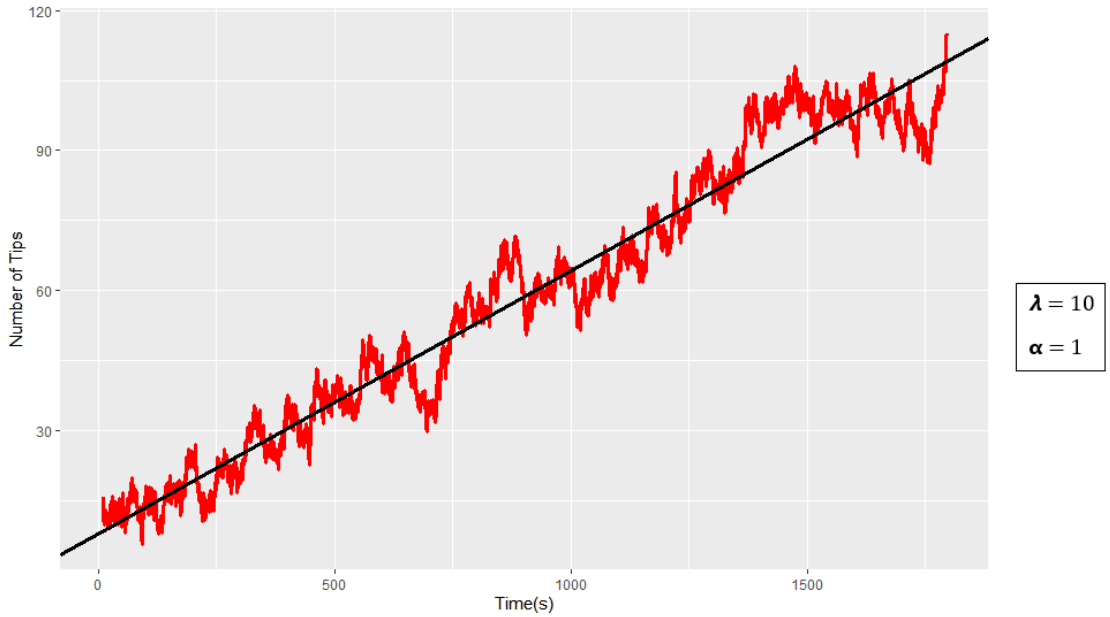


Figure 17: Values of L(t) in a high-load Tangle network as a function of time with $\alpha = 1$ *and* $\lambda = 10$

Figure 17 reveals a linear trend indicated by a fitted linear function (black line). Here, according to the Tangle literature [54] the number of tips is transient.

The behaviour of the number of tips with $\alpha = 0$ is the most desired one since number of the tips varies over time, but it stays bounded, not growing on average. The other values of alpha have a less ideal behaviour. The number of tips grows on average over time, for example for $\alpha = 0.001$ this is only visible after a very long time of running the simulation [54].

However, despite the fact that it is desirable to have a recurrent Tangle with $\alpha = 0$, because we do not want unapproved transactions, positive value of α plays a role of security against lazy tips and parasite sub-tangles [29]. $\alpha > 0$ keeps recent transactions from approving old ones and contributes to the good behaviour in the Tangle network. Thus, it is better for the Tangle to employ a very small positive value of alpha such as $\alpha = 0.001$, making the growth of the number of tips quite slow, so the few transactions that get left behind just need to reattach to the Tangle [54]. Thus, in our simulations we set the α to 0.001 to have a stable Tangle network during the examined time and analyse the impact of other parameters on the Tangle network.

3.2.2.1.2 Linear relationship between the number of tips and λ in the high-load state of the Tangle network (hypothesis 2)

Our simulations show that in the high-load Tangle network, there is a linear relationship between the average number of tips (L0) and λ for α equal to 0.001, considering h equal to one second. All experiments are for 10-minute running of the Tangle simulator.

Table 4: Average and standard deviation of the number of tips for various values of λ

λ	Number of Transactions	Average number of tips	Standard deviation of the number of tips
10	6,000	11.48979	2.205524
20	12,000	23.80224	3.52489
30	18,000	34.53555	3.751544
40	24,000	44.37097	4.543612
50	30,000	56.41663	5.170574
60	36,000	68.38212	6.397714
70	42,000	78.61165	6.219493
80	48,000	89.11171	6.685825
90	54,000	100.9682	7.168516
100	60,000	112.7592	6.744268

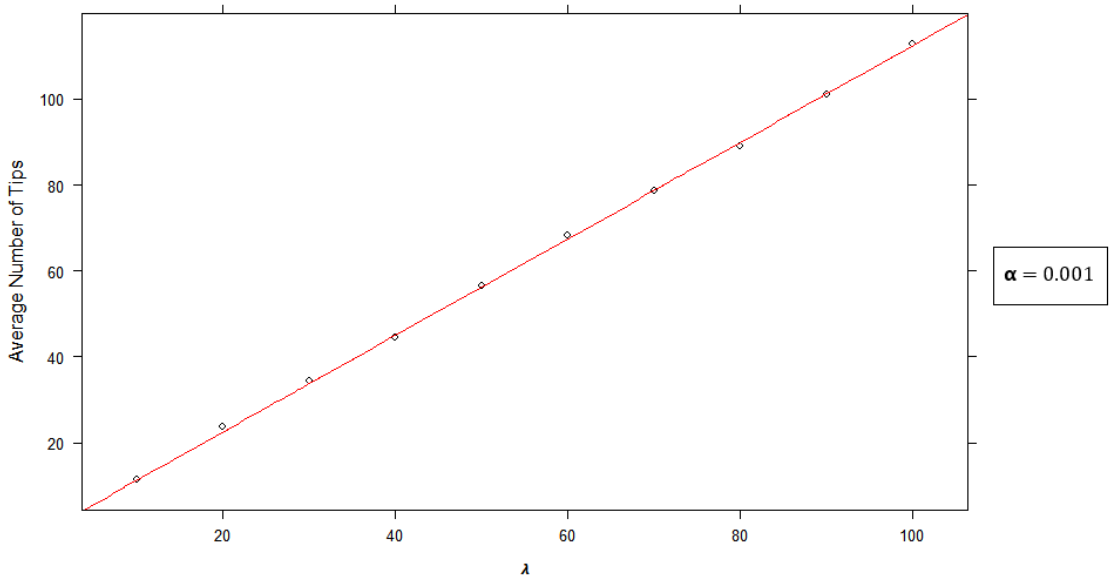


Figure 18: Average number of tips vs. λ

In Figure 18, each black point shows the value of the average number of tips against $\boldsymbol{\lambda}$. The red line is the best fitted linear function ($F(\boldsymbol{\lambda}) = A * \boldsymbol{\lambda}$) that fits the data points. We observe that A equals to 1.125206 ± 0.003614.

3.2.2.1.3 Tip counts in the low-load state of the Tangle network (hypothesis 3)

In a low-load state of the Tangle network, even with setting various values for network parameters such as $\boldsymbol{\lambda}$ and $\boldsymbol{\alpha}$, we observe that the average number of tips remains just a few over the simulation time. Table 5 and 6 show the frequency of the average number of tips during the simulation time for different parameter settings.

Table 5: The average number of tips for a low-load Tangle network simulation with $\boldsymbol{\alpha} = 0.001$ and $\boldsymbol{\lambda} = 50$

Average Number of Tips	Frequency
1	323,556
1.1	176
1.2	72
1.3	64
1.4	70
1.5	73
1.6	65
1.7	64
1.8	64
1.9	64
2	66

Table 6: The average number of tips for a low-load Tangle network simulation with $\boldsymbol{\alpha} = 1$ and $\boldsymbol{\lambda} = 10$

Average Number of Tips	Frequency
1	196,523
1.1	34
1.2	14
1.3	12
1.4	12
1.5	12

1.6	14
1.7	12
1.8	12
1.9	12
2	12

3.2.2.2 Growth of the Cumulative Weight

In the case of cumulative weight growth, performed simulations confirm findings of the white paper [38] and [24], [25] both for the high-load and low-load Tangle networks. In this section we show that hypotheses 4 and 5 are valid in our Tangle simulator.

3.2.2.2.1 Cumulative weight growth in the high-load state of the Tangle network (hypothesis 4)

In the high-load Tangle network, we observe two phases of growth for transactions' cumulative weight: exponential growth (adaptation period) and linear growth. As can be seen in Figure 19, we examine the cumulative weight growth of the 500th transaction as a sample over the simulation time and we observe two phases of growth according to hypothesis 4. Figure 20 shows that behaviour similar to what is shown in Figure 19 is typical for other transactions (such as 1000th, 1500th, 2000th and 2500th transactions) in the high-load Tangle network.

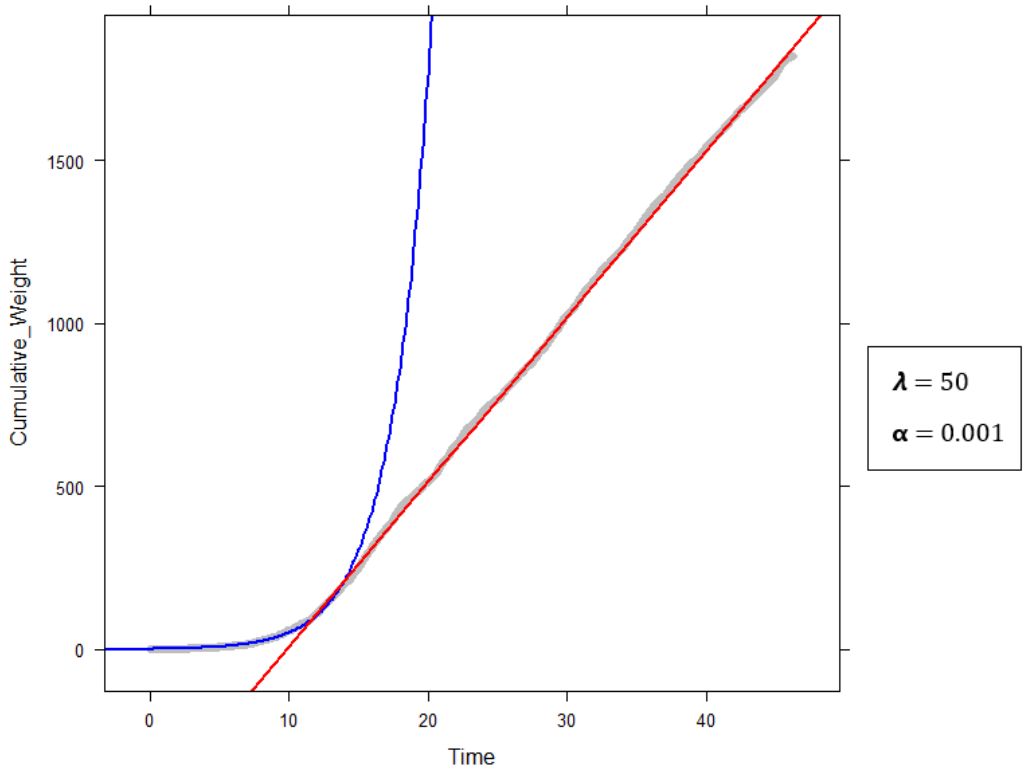


Figure 19: Cumulative weight of the 500th transaction in a high-load Tangle network with $\alpha = 0.001$ *and* $\lambda = 50$

Figure 19 shows the cumulative weight of the 500th transaction during the simulation time (grey line). Blue and red lines are the best fitted exponential and linear functions respectively. Linear trend of the cumulative weight continues for the length of the simulation. F(t) = A exp(Bt) is the exponential function and g(t) = Ct + D is the linear function. F(t) has been fitted on the interval [0, 13.7] and g(t) has been fitted on (13.7, 55]. Values of parameters: A=1.5400821 ± 0.0024782, B=0.3521739 ± 0.0001301, C=50.677873 ± 0.002853, D=-498.175285 ± 0.088332. We see that B is similar to the predictions in the white paper [38] (see section 3.1(8)) and also C is close to the value of λ.

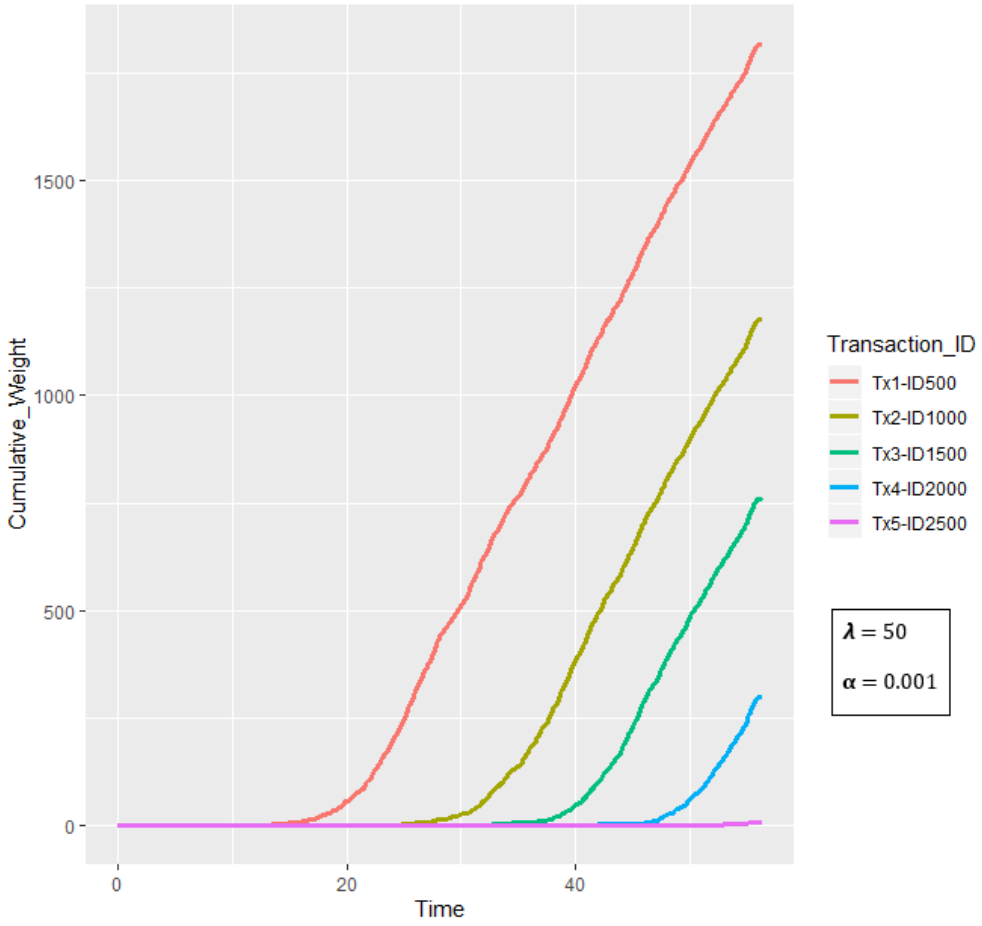


Figure 20: Cumulative weight growth of 500th, 1000th, 1500th, 2000th and 2500th transactions in a high-load Tangle network with $\alpha = 0.001$ and $\lambda = 50$

3.2.2.2.2 Cumulative weight growth in the low-load state of the Tangle network (hypothesis 5)

Results of simulations in a low-load Tangle network confirm hypothesis 5. As can be seen in Figure 21, we examine the cumulative weight growth of the 500th transaction as a sample over the simulation time and we observe only one phase of growth. Figure 21 shows that there is one phase of growth: linear growth with slope λw and since we assume that w=1, this slope is equal to λ.

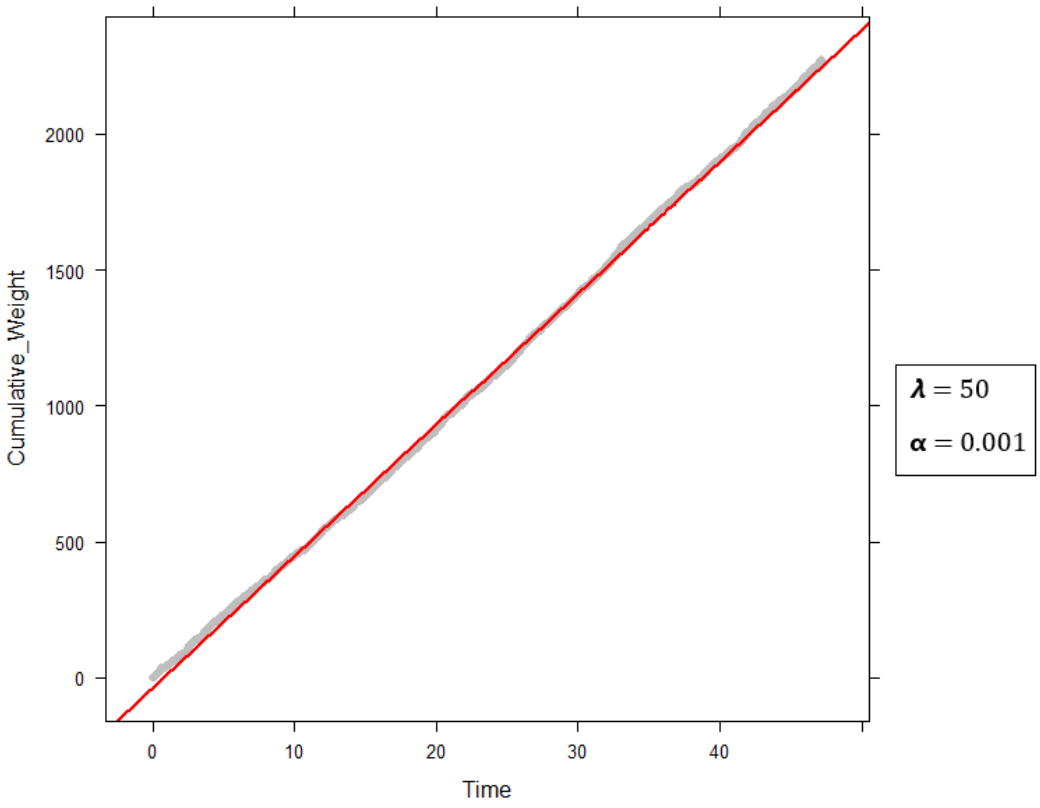


Figure 21: Cumulative weight of the 500th transaction in a low-load Tangle network with $\alpha = 0.001$ and $\lambda = 50$

Figure 21 shows the cumulative weight of the 500th transaction during the simulation time (grey line). Red line is the best fitted linear function. F(t) = At + B is the linear function. Values of parameters: A= 48.439776 ± 0.002422, B= -37.068975 ± 0.067179. We see that A is close to the value of λ.

Figure 22 shows that behaviour similar to what is shown in Figure 21 is typical for other transactions (such as 1000th, 1500th, 2000th and 2500th transactions) in the low-load Tangle network.

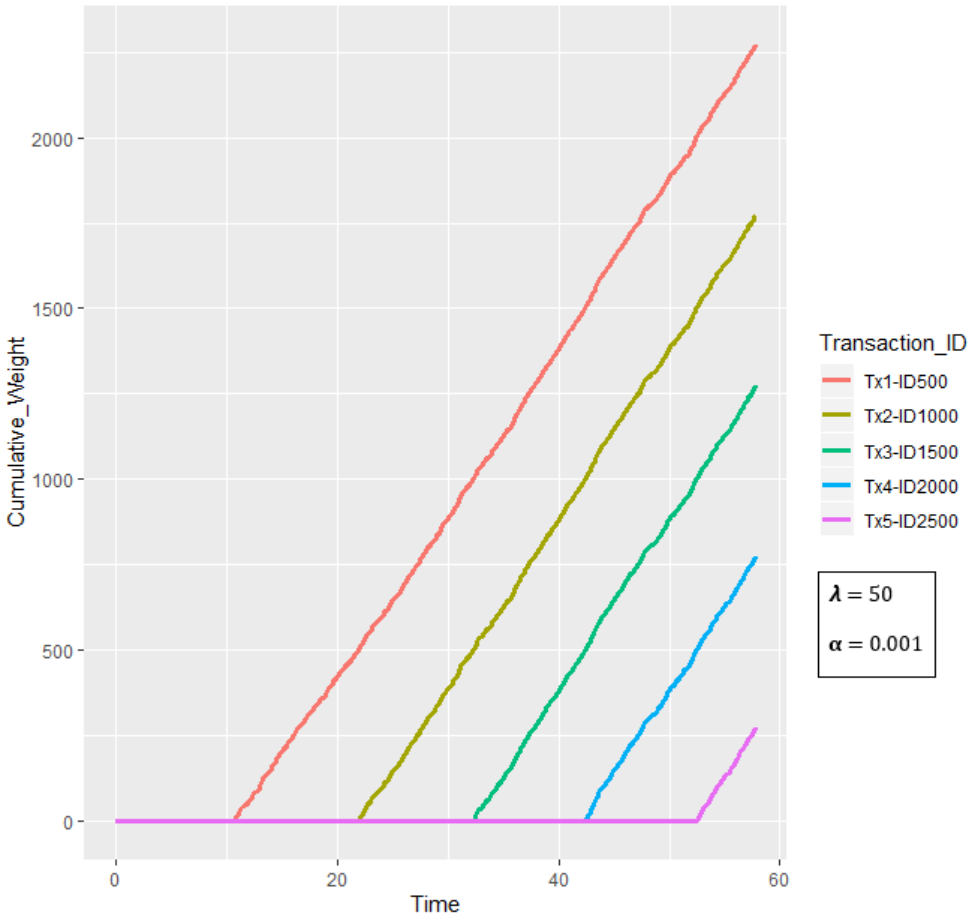


Figure 22: Cumulative weight growth of 500th, 1000th, 1500th, 2000th and 2500th transactions in a low-load Tangle network with $\alpha = 0.001$ *and* $\lambda = 50$

By confirming all the five hypotheses mentioned in this section, the presented data from our Tangle simulator confirm the theoretical claims in the Tangle white paper [38] and [24], [25] papers from IOTA Foundation. Using our Tangle simulation, we are able to examine the number of tips, L(t), as a function of time. We examined values of L(t) when tip selection is guided by MCMC random walk in high-load Tangle networks. For α=0, L(t) remains stable in examined time interval. Even though analytical evidence indicates that for positive values of α, $L(t) \rightarrow \infty$ when $t \rightarrow \infty$ (in the absence of additional efforts by the users to re-attach these transactions). For smaller values of α the rate of the divergence is insignificant, but for larger values of α growth rate of L(t) is significant and follows linear fashion. In response to RQ2, we also confirmed findings of the white paper [38] in the matter of a linear relationship between the average number of tips (L0) and λ in high-load Tangle networks. For the low-load Tangle network, we showed

that the average number of tips remains a few over the simulation time. Moreover, similar to the claims in the Tangle white paper [38] analysis of the cumulative weight growth over time in high-load Tangle networks in our simulator revealed two phases of growth (exponential growth and linear growth) while analysis of the cumulative weight growth in low-load Tangle networks revealed just one phase of growth (linear growth).

3.3 Simulator Scalability

Having a validated Tangle simulator, it is vital to investigate up to how many nodes can be simulated in the Tangle network and how much real time (in seconds) it takes to simulate each number of nodes in the network. All the parameters of the Tangle simulator can be set in a single Java file. Thus, to test the Tangle simulator scalability, we set all the simulator parameters to constant values and only change the "number of nodes" to see its effect on the real time of the simulation. We run several experiments to answer the following question:

1- How many nodes can be simulated in the Tangle network given realistic time and realistic computational resources?

In all the experiments, $\boldsymbol{\lambda}$ equals 1, number of transactions equals 1,800 (30-minute simulation time), $\boldsymbol{\alpha}$ equals 0.001, MWM equals 100, transmission rate between each two nodes is normally distributed at an average of 5,000 bits/sec and standard deviation of 1,000 bits/sec, and hash power of each node is normally distributed at an average hash power of 12e9 trials/sec and standard deviation of 1e9 trials/sec. We execute 12 experiments with various values of the number of nodes such as 10, 100, 500, 1,000, 1,500, 2,000, 2,500, 3,000, 3,500, 4,000, 4,500, and 5,000.

It is also important to consider the computational resources to run these experiments. All the experiments are run on a MacBook Pro with Processor 2.3 GHz Quad-Core Intel Core i7 and Memory 16 GB 3733 MHz LPDDR4.

In response to RQ1, Table 7 shows how much real time it takes for the simulator to run each of these experiments with the specific number of nodes.

Table 7: Number of nodes and the corresponding real time of the simulation

Number of the Nodes	Real Time (second)
10	20.016
100	208.128

500	1,102.013
1,000	2,128.576
1,500	3,047.033
2,000	4,394.714
2,500	5,909.547
3,000	6,935.245
3,500	7,644.145
4,000	9,642.248
4,500	11,344.002
5,000	14,647.480

Figure 23 shows the relationship between the number of nodes and the corresponding real time of the simulation in seconds. The result shows that with greater running time of the simulation, our simulator is able to simulate larger networks with thousands of nodes. For example, it approximately takes 4 hours to run a simulation of a 30-minute Tangle network with 5,000 nodes. As we discuss in the conclusion section, there are ways by which this performance can be improved through the use of multithreading and parallelism.

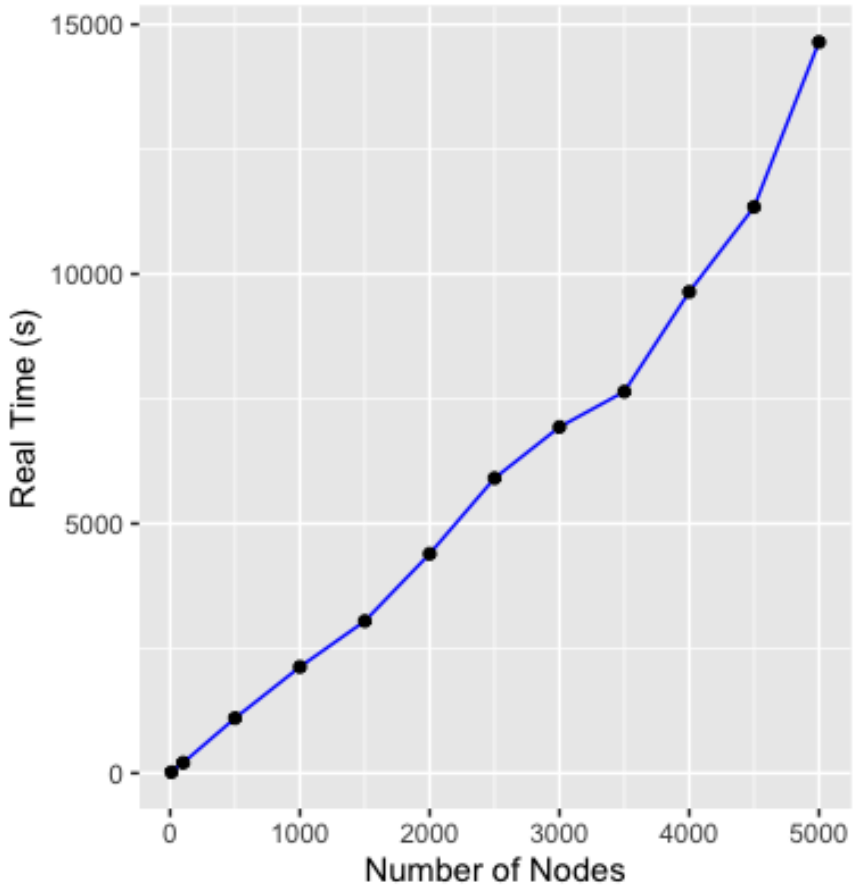


Figure 23: Real Time of the Simulation vs. Number of the Nodes

Chapter Four – Results and Experiments

This chapter provides a detailed explanation of assessing transaction throughput and security against parasite chain attacks in the Tangle Networks.

4.1 Throughput of the Tangle Network

In this section, we evaluate the throughput of the Tangle network. Throughput is a system-level performance metric that defines how many jobs can be processed per second [29]. In the Tangle simulator, the throughput is defined as the number of confirmed transactions per second, which represents the transaction processing power of the entire network. To assess the Tangle network's throughput, we run the simulator with a set of parameters to answer the following questions:

1- Which parameters influence the throughput of the Tangle network?

2- Given the parameters what is the average number of confirmed transactions per second?

Here, the output of each experiment is a 3D matrix in which the first dimension is nodes of the Tangle network, the second dimension is transactions, the third dimension is the simulation time, and the value of each cell is the transaction confirmation rate. Then, we visualize the average number of confirmed transactions per second versus different values of transaction arrival rate ($\boldsymbol{\lambda}$), randomness factor in random walk ($\boldsymbol{\alpha}$), network difficulty, and network transmission rate.

4.1.1 Review of Transaction Confirmation

In this section, we provide a detailed description of the transaction confirmation process in the Tangle network since it is used in transaction throughput calculation. As previously stated, Tangle is an ever-expanding Directed Acyclic Graph (DAG) as time moves forward. In practice, all new transactions are generated within a PoW time and attached to the DAG with an arrival rate $\boldsymbol{\lambda}$. The DAG is expanding in a way that the later arriving transactions need to approve two tips. Here, approval is a relationship of direct reference. For example, in Figure 24, node A approves B, which means that A is directly pointing to B [29].

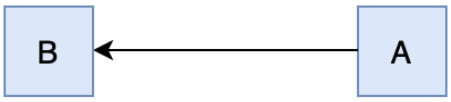


Figure 24: Approval Example

These two tips are chosen by following the MCMC tip selection algorithm (see section 3.1.6). As we have already seen, the MCMC algorithm includes three steps:

1. Defining entry points which are random positions from the first half part of the DAG.

2. Updating cumulative weight of transactions when a new transaction arrives at the DAG.

3. Parallel random walks through the Tangle starting from entry points to reach tips, so as to get the two first tips. For example, the selected tips in Figure 25 are v9 and v8, with the random walk paths (v1, v3, v6, v7, v9) and (v1, v2, v4, v5, v8), shown by the red dashed lines.

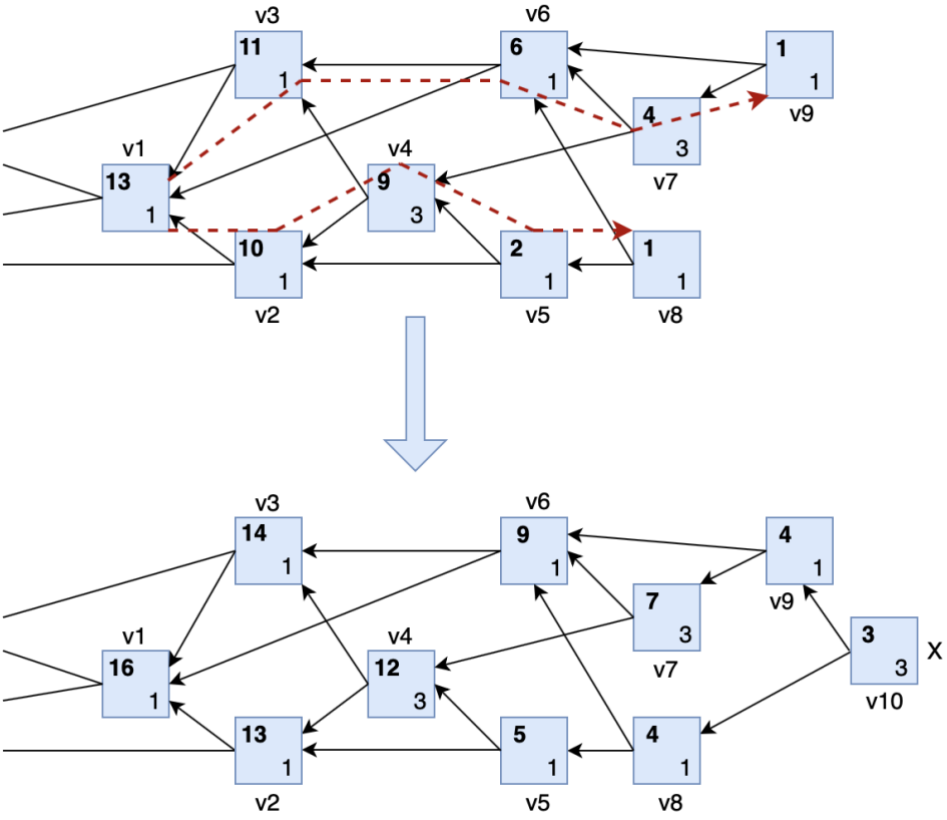


Figure 25: Subgraph with Cumulative Weight Update before and after a Newly Issued Transaction (X)

Here, v1 is the entry point. The small number in the lower-right corner of each box denotes "own weight", and the bold number in the upper-left corner denotes the "cumulative weight". According to the implementations of MCMC random walk, the probability to walk towards a specific approver is as the following formula [38]:

$$P_{xy} = \exp\big(-\alpha(\mathcal{H}_x - \mathcal{H}_y)\big)\Big(\sum_{z:z\rightsquigarrow x} \exp\big(-\alpha(\mathcal{H}_x - \mathcal{H}_z)\big)\Big)^{-1}$$

where Pxy is the probability to walk from vertex x to y, Hy is the cumulative weight of vertex y, and z → x means that z directly approves x. Therefore, in the weighted random walk, a transaction with higher cumulative weight has a higher probability to be selected and approved. In other words, the probability of walking from x to y increases exponentially with the cumulative weight of y, multiplied by **α**. Here, **α** is a randomness factor, which means that an **α** = 0 factor makes the walk completely random, and with an **α** = ∞, the transaction with the highest cumulative weight is chosen as the next step. The sum in the denominator is a normalization factor, which sets the total transition probabilities sum to one. Confirmation rate of each transaction is then determined using the following steps:

- Each node runs the MCMC random walks to select 100 starting points at tips. Several or all of the starting points can be the same tip.
- The node calculates how many starting points confirm the transaction, that is, can directly or indirectly reach the transaction, moving backwards along the chain of confirmations.
- If this number is less than 50, the transaction is not yet confirmed. If it is more equal than 50, the transaction is considered confirmed.

4.1.2 Experiment Setup

We leverage the Tangle simulator to generate transactions in the network, then collect their confirmation data to understand the relationship between the transaction throughput of the network and configured parameters in the experiments. To collect the data, we consider 10 nodes. We choose this size of network for experimental efficiency, given that larger networks require considerably more time as we saw in Section 3.3. Nevertheless, assuming that end-to-end connection latencies in the network do not decrease as we add more nodes, the confirmation level distributions should remain unaffected by network (i.e. sample) size. Likewise, the realistic scenario in which a larger network implies lower end-to-end latency can be simulated by lowering the end-to-end latencies while keeping the network size constant – assuming also proportional adaptation of λ – and, thus, the simulation time tractable. Nevertheless, given our computational limitations we are not able to empirically test this assumption. Regardless, small blockchain

networks can still be conceived in practical applications, e.g. a set of devices with a building, or a permissioned B2B transaction settling application.

All the experiments are run for a ten-minute simulation time in the Tangle simulator. To explore the influence of the transaction arrival rate $\boldsymbol{\lambda}$, randomness factor $\boldsymbol{\alpha}$, network difficulty MWM, and network transmission rate on the transaction throughput, we run several groups of simulations. In each experiment group, we fix all the parameters and change only one parameter to observe its impact on the transaction throughput.

We assume a network of 10 nodes (e.g. a network of 10 Internet-of-Things devices). we set the transaction arrival rate ($\boldsymbol{\lambda}$) to 5 (3000 transactions in 10-minute running of the simulation) to have a reasonable running time for each simulation. We also set the randomness factor ($\boldsymbol{\alpha}$) to 0.001 (according to Section 3.2.2.1.1) in order to keep the Tangle network stable, with constant number of tips during the simulation. To configure the network, transmission rate between each two nodes is normally distributed at an average of 5,000 bits/sec and standard deviation of 1,000 bits/sec. Moreover, transaction size is arbitrarily selected and is normally distributed at an average of 5,000 bytes and standard deviation of 500 bytes. Network difficulty is arbitrarily set to 1e5 and hash power of each node is arbitrarily chosen and is normally distributed at an average hash power of 12e9 trials/sec and standard deviation of 1e9 trials/sec.

During each simulation, the proposed transaction confirmation rate calculation is used to extract the transaction confirmation data and later to conduct a statistical analysis of the data. To calculate the average number of confirmed transactions per second, we log the data required for throughput analysis 30 times during the simulation time (simulation time divided in 30 intervals). At each log time, we select 30 random transactions among all the arrived transactions to the Tangle network. Arrived transactions include both the transactions attached to the local DAG of nodes and transactions in nodes' pool waiting for validation. At each time, we calculate the average confidence rate of these 30 random transactions and count the number of those transactions with the confidence rate greater than 50 percent. Then, by generalization, at each log time the total number of confirmed transactions among all the arrived transactions to the network is calculated. Reported throughput for each experiment in the following plots are, then, the average number of confirmed transactions.

Table 8 to 11 includes the details of the results including the average throughput and standard deviation (SD) of the throughput in each experiment setting. Average throughput equals the

average number of transactions with confidence rate greater than 50% per second. In addition, Figures 26 to 29 show the influence of each parameter on the transaction throughput.

4.1.3 Results

*4.1.3.1 Impact of Transaction Arrival Rate (**λ**) on Throughput*

In this section, we investigate the transaction throughput as a function of the transaction arrival rate (**λ**). To this purpose, we execute a group of experiments with **α** equals 0.001 and MWM equals 1e5. Transmission rate between each two nodes is normally distributed at an average of 5,000 bits/sec and standard deviation of 1,000 bits/sec. **λ** has various values of 1, 5, 10, 15, 20, 25, 30, 35 and 40.

*Table 8: Average Throughput and Standard Deviation (SD) of the Throughput for Various Values of **λ***

λ	**Number of Transactions**	**Average Throughput**	**SD of the Throughput**
1	600	0.7071785	0.2021667
5	3,000	3.178864	1.338227
10	6,000	5.865795	2.648756
15	9,000	8.53248	3.925367
20	12,000	10.64008	5.079908
25	15,000	13.51908	5.863012
30	18,000	15.96856	7.993904
35	21,000	17.55628	9.217246
40	24,000	20.93969	9.880581

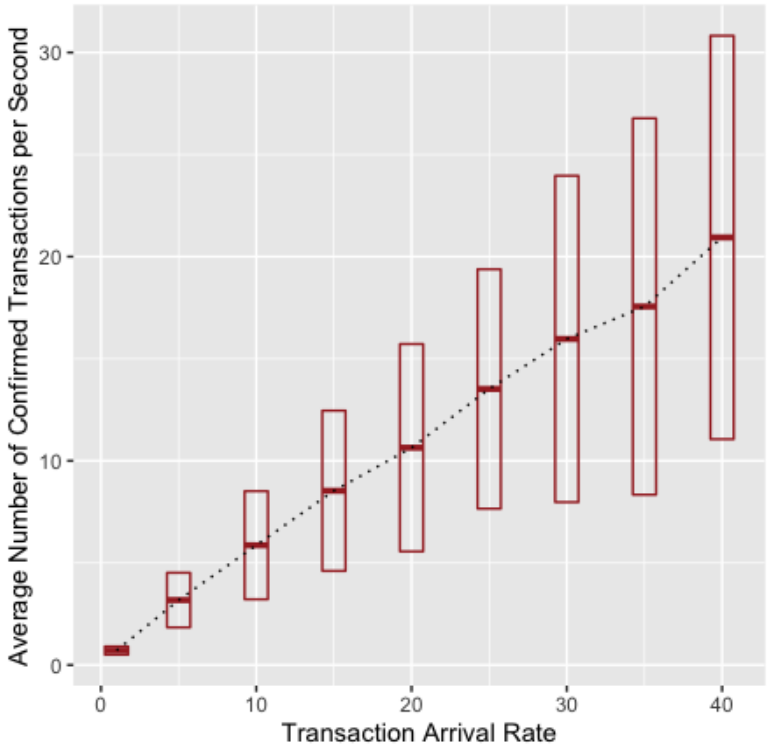


Figure 26: Number of Confirmed Transactions per Second vs. Transaction Arrival Rate

Each vertical bar in Figure 26 shows mean +/- SD of the number of confirmed transactions per second. As shown in this Figure, the plot provides an almost linear relationship between the average number of confirmed transactions per second and the transaction arrival rate. We observe that greater values of transaction arrival rate result in higher values of the number of confirmed transactions per second. The exact values of the numbers are provided in Table 8.

4.1.3.2 Impact of Randomness Factor (α) on Throughput

In this section, we investigate the transaction throughput as a function of the randomness factor in random walk (α). We run another group of simulations with $\lambda = 5$, number of transactions = 3,000, MWM = 1e5, transmission rate between each two nodes is normally distributed at an average transmission rate of 5,000 bits/sec and standard deviation of 1,000 bits/sec, and varying values of α equal to 0, 0.001, 0.01, 0.1, 1, and 10.

Table 9: Average Throughput and SD of the Throughput for Various Values of α

α	Average Throughput	SD of the Throughput
0	3.177717	1.079453
0.001	3.178864	1.338227
0.01	3.291223	1.330975
0.1	3.241293	1.240233
1	3.624654	1.327565
10	3.557073	1.411345

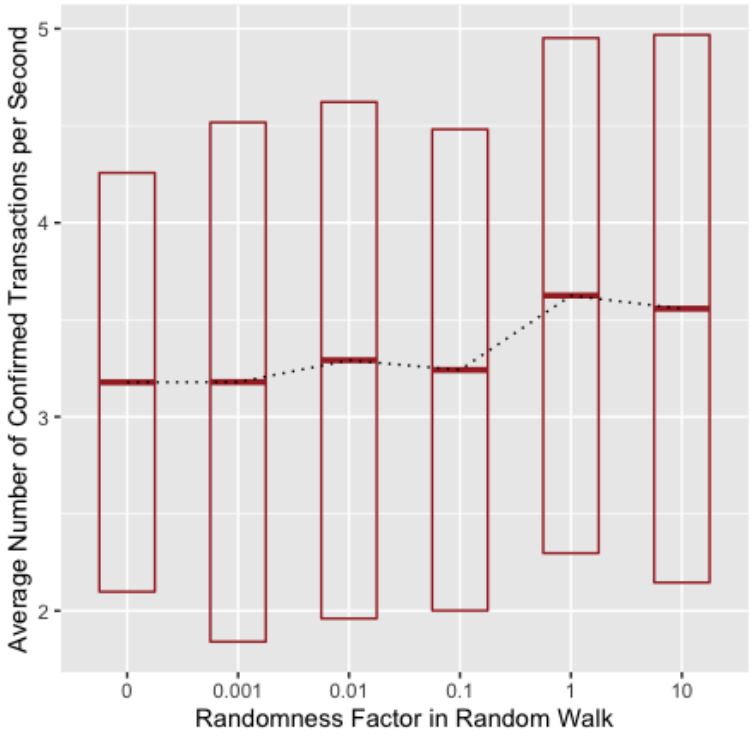


Figure 27: Number of Confirmed Transactions per Second vs. Randomness Factor in Random Walk

In Figure 27, each vertical bar shows mean +/- SD of the number of confirmed transactions per second. According to Figure 27 and Table 9, we do not observe any specific relationship between α and the number of confirmed transactions per second for a constant value of λ in the 10-minute simulation time for each experiment. However, there is a possibility that for longer simulation times the average number of confirmed transactions decreases while α increases. There exist some theoretical arguments [24], [25] that larger values of α increases the probability to select heavier tips in the random walk so that new coming transactions will be attached to the

heaviest path of the Tangle eventually [29] and as a result, those transaction in the heavy sub-tangle get confirmed by the MCMC random walk and other transactions attached to other parts of the Tangle remain unconfirmed as tips.

4.1.3.3 Impact of Network Difficulty (MWM) on Throughput

To investigate the impact of network difficulty (MWM) on the transaction throughput, a group of simulations is run with $\boldsymbol{\lambda}$ = 5, number of transactions = 3,000, $\boldsymbol{\alpha}$ = 0.001, transmission rate between each two nodes is normally distributed at an average of 5,000 bits/sec and standard deviation of 1,000 bits/sec, and MWM values of 1e1 to 1e15 and 1e20.

Table 10: Average Throughput and SD of the Throughput for Various Values of Network Difficulty

Network Difficulty	Average Throughput	SD of the Throughput
1e1	3.163705	1.24673
1e2	3.244671	1.146522
1e3	3.339393	1.26073
1e4	3.27187	0.9759385
1e5	3.178864	1.338227
1e6	3.181419	1.348995
1e7	3.22961	1.190247
1e8	3.184152	1.289947
1e9	3.028559	1.195984
1e10	3.290772	1.299867
1e11	1.042299	0.3842319
1e12	0.1300998	0.07160918
1e13	0.0113468	0.003314126
1e14	0.001271697	0.000373298
1e15	0.0001225991	5.997222e-05
1e20	1.143567e-09	2.716162e-10

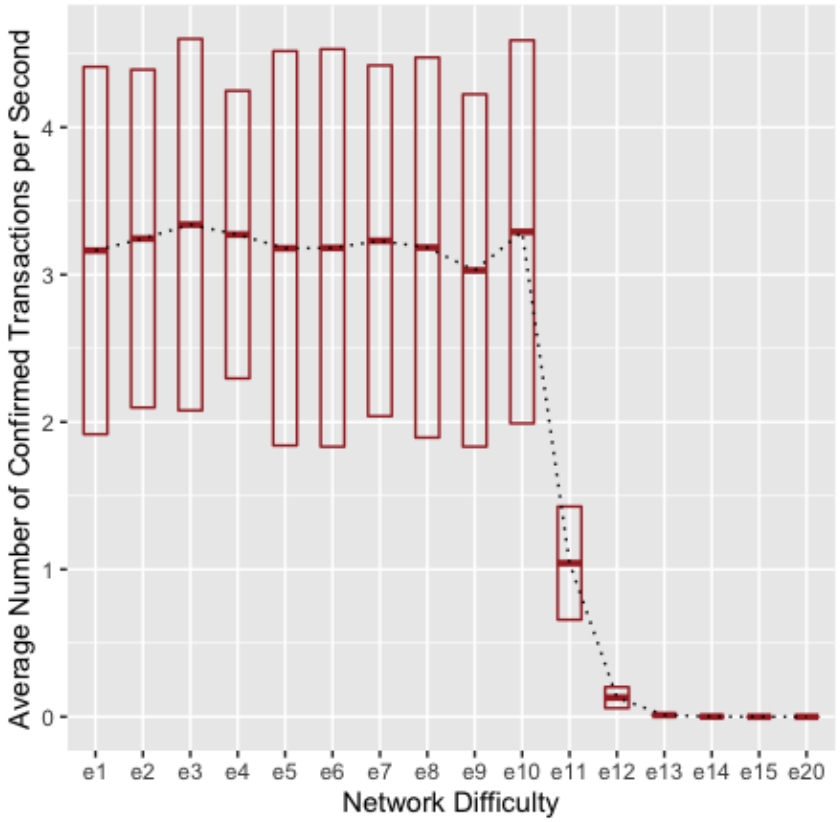


Figure 28: Number of Confirmed Transactions per Second vs. Network Difficulty

Each vertical bar in Figure 28 shows mean +/- SD of the number of confirmed transactions per second. When network difficulty is less equal than e10, the number of confirmed transactions per second is almost the same. However, we observe that greater values of difficulty, more than e10, result in much less values of the number of confirmed transactions per second.

When difficulty is less equal than e10, validation times of incoming transactions are small. Our experiments in Table 10 show that for these values of difficulty, validation of a new transaction happens right after its arrival to the network. The reason is that the transaction arrival rate is 5 transactions per second, which means on average each new transaction arrives every 200 milliseconds to the network and when difficulty is less equal than e10, average validation time is still less than 200 milliseconds. Thus, for these experiments (when difficulty <= e10), arrival, validation and propagation of new transactions has the same pattern among nodes in the network. Therefore, at each log time, most of the 30 random transactions are attached to local DAGs instead of waiting in nodes' pools for validation. As a result, most of these 30 transactions have a confidence rate greater than zero, and among them are several transactions with confidence rates

greater than 50 percent. That is the reason, we observe almost the same result for experiments with the difficulty less equal than e10.

However, when difficulty is more than e10, because of higher transaction validation times, we observe that even before the first log time most of the transactions have arrived at the network, but they are in nodes' pools waiting for validation. Therefore, at each log time only few of the 30 random transactions are actually attached to local DAGs and have confidence rates greater than zero. We observe that for difficulty greater than e10, if transactions are attached to local DAGs, they have confidence rate equals 1 since because of high value of the difficulty there only exist one or two tips in DAGs at a time. In addition, for higher values of difficulty, validation time is greater, and it even takes much more time for transactions to get validated and attach to local DAGs. That is the reason that for higher values of difficulty such as e11 to e20, the total number of confirmed transactions decreases over the examined simulation time.

4.1.3.4 Impact of Network Transmission rate on Throughput

Finally, to check the influence of network transmission rate on the throughput, we run a group of simulations with $\boldsymbol{\lambda}$ = 5, number of transactions = 3,000, $\boldsymbol{\alpha}$ = 0.001, MWM = 1e5, and network transmission rate between each two nodes is normally distributed at standard deviation of 1,000 bits/sec and average transmission rate with various values of 1,000, 2,000, 3,000, 4,000, 5,000, 10,000, 100,000, 1,000,000, 10,000,000 and 100,000,000 bits/sec.

Table 11: Average Throughput and SD of the Throughput for Various Values of the Average Network Transmission rate

Average Transmission rate	Average Throughput	SD of the Throughput
1,000	1.06256	0.4607079
2,000	1.284784	0.6920575
3,000	1.840945	1.003016
4,000	2.555467	1.316067
5,000	3.178864	1.338227
10,0 00	3.82141	1.207057
100,000	5.00837	0.1170402
1,000,000	5.181619	0.139428
10,000,000	5.201643	0.147959
100,000,000	5.004215	0.06326653

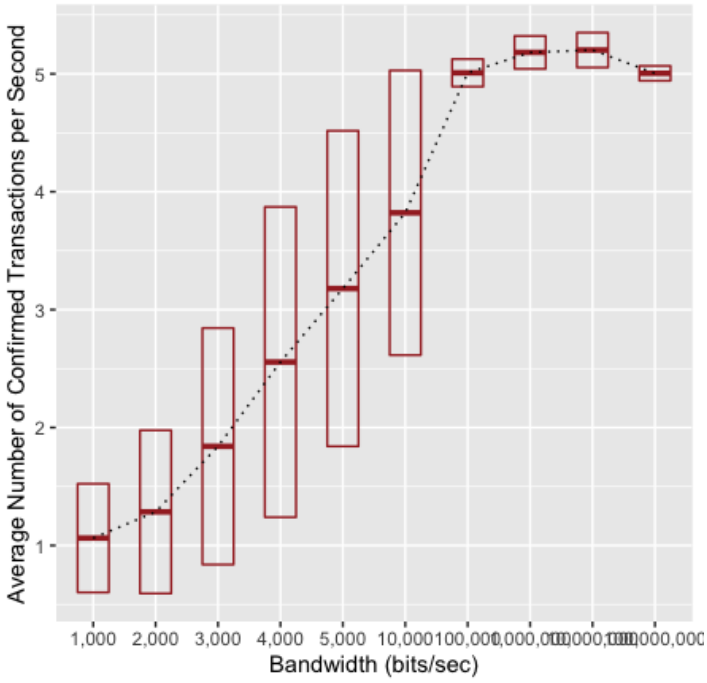


Figure 29: Number of Confirmed Transactions per Second vs. Network Transmission rate

Each vertical bar in Figure 29 shows mean +/- SD of the number of confirmed transactions per second. In Table 11 and Figure 29, higher values of network transmission rate result in higher values of the number of confirmed transactions per second. The higher values of transmission rate make the Tangle network latency low and as a result the typical number of tips is small [38]. Smaller number of the tips in the Tangle means that more transactions get confirmed by the transaction confirmation process mentioned in 4.2.1. Moreover, after a threshold value of transmission rate, here 100,000 bits/sec, we do not observe significant changes in the throughput. The higher values of transmission rate make the Tangle network latency very low and as a result the typical number of tips frequently becomes one or two [38], which means most of the transactions get directly or indirectly confirmed by new incoming transactions. Thus, the number of confirmed transactions per second for the same value of $\boldsymbol{\lambda}$ and higher values of the network transmission rate (transmission rate >= 100,000 bits/sec) is almost the same in the examined setting of the simulations. It is important to mention that with $\boldsymbol{\lambda} = 5$, the maximum number of confirmed transactions per second in the simulation can be 5. In Figure 29, there is a little error which shows that the number of confirmed transactions can be more than 5 for higher values of the transmission rate. We attribute this error to the coarse sampling used for calculating throughput (simulation time divided in 30 intervals) in order to allow for experimental efficiency.

4.1.4 Summary of Findings on Transaction Throughput of the Tangle Networks

In this section, we investigated the transaction throughput of the Tangle network in various parameter settings. We observed that throughput has an almost linear relationship with transaction arrival rate in the examined environment. Moreover, the randomness factor in MCMC tip selection algorithm does not influence the throughput in the examined setting. However, based on theoretical claims in the papers [24], [25], [29], it is possible that higher values of randomness factor result in lower throughput. We also observe that higher values of network difficulty result in greater transaction validation time and as a result decreases the throughput. Finally, higher values of network transmission rate result in higher throughput, but if network transmission rate passes a threshold value, throughput remains the same for the same transaction arrival rate.

4.2 Security of the Tangle

In this section, we assess the security of the Tangle network. We study a specific kind of attack, parasite chain attack, with the aim of answering the following questions:

1. What parameters affect the Tangle's security in the presence of parasite chain attacks and how?
2. Can we find a parameter set corresponding to the malicious hash power in the Tangle network to create a secure network against parasite chain attacks?

4.2.1 Parasite Chain Attack Scenario

There are different attack scenarios against the Tangle network such as large weight attacks, splitting attacks, and parasite chain attacks [38]. We are particularly interested in parasite chain attacks since it acts in a way that is similar to double-spending attacks in traditional bitcoin-style blockchain networks.

The visualization of the parasite chain attack can be seen in Figure 30. This attack includes several steps as explained in below [38].

1. An attacker sends a transaction including a payment to a merchant and receives the goods after the merchant decides the transaction has a sufficiently large confidence rate.
2. The attacker issues a double-spending transaction, which is a transaction that spends the same money twice.

3. The attacker simultaneously uses their computing power to issue many transactions, a.k.a. fake transactions, that approve the double-spending transaction, but do not approve the original transaction that they sent to the merchant.
4. The attacker hopes that their dishonest sub-tangle outpaces the honest sub-tangle. If this happens, the main Tangle continues growing from the double-spending transaction, and the legitimate branch with the original payment to the merchant is orphaned. Orphaned transactions are not approved by incoming transactions anymore. We call the transaction including the original payment to the merchant the victim transaction.

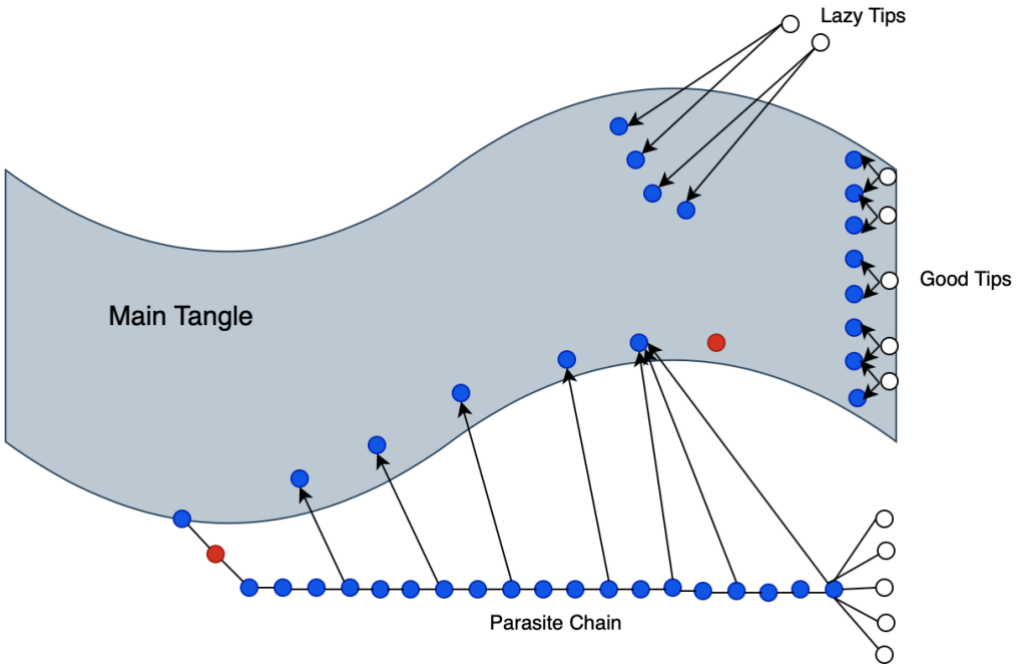


Figure 30: Visual representation of the parasite chain attack

4.2.2 Defend Against Parasite Chain Attack

According to the Tangle white paper [38], to defend against parasite chain attacks, the honest nodes of the Tangle network must be assumed to have more active hashing power than the attacker. Therefore, the main Tangle is able to produce larger increases in cumulative weight for more transactions than the attacker, and the sites on the parasite chain will have a cumulative weight that is smaller than the cumulative weight of the sites on the main Tangle. This results in less confidence rate for the double-spending transaction than the victim transaction in the Tangle network which leads to the attack failure. The Tangle white paper [38] also claims that the security of the Tangle comes from its weighted tip selection algorithm, the MCMC algorithm (see section

3.1.6). Assuming greater hashing power for honest nodes in the Tangle network, it has a low probability that the random walker jumps to the parasite chain unless it begins there, and this event is not very probable either because the main Tangle contains more sites.

4.2.3 Parasite Chain Attack Simulation

We have simulated the parasite chain attack in the Tangle simulator in order to study this attack empirically. To do so, we have implemented several required steps.

1. Based on the parasite chain attack scenario explained above, new parameters are defined in the Tangle simulator as follows: the malicious hash power percentage, attack duration, and confidence rate threshold. Malicious hash power percentage refers to the attacker's computing power percentage of the total hash power in the Tangle network. Malicious hash power is used to create and validate fake transactions by the attacker to support their parasite chain. Attack duration is defined by the number of the fake transactions created and validated by the attacker to append to the parasite chain to increase the double-spending transaction's cumulative weight. Finally, the confidence rate threshold is used in revealing the parasite chain which is explained in step number 5.
2. An attacker is defined as a malicious node in the Tangle network. In the Tangle network setup, we select one random node as the malicious node and set its hash power to the malicious hash power percentage of the total network's hash power.
3. A victim transaction is a transaction that the attacker aims to double spend. The victim transaction is selected as the 30th honest transaction in the Tangle network.
4. During the simulation, when the victim transaction arrives at the malicious node, the attacker hiddenly starts creating the parasite chain. To do so, he first creates a double-spending transaction which approves the same transactions that the victim transaction approves. Then, he creates fake transactions to increase the cumulative weight of the double-spending transaction. Since each new transaction has to verify two tips in the Tangle, each fake transaction verifies the previous fake transaction as one tip and another tip which does not directly or indirectly refer to the victim transaction in the main Tangle.
5. In each simulation state, the victim transaction's confidence rate is calculated. When it passes the confidence rate threshold, the malicious node reveals the parasite chain to other nodes of the Tangle network. The parasite chain includes the double-spending transaction,

fake transactions and all the honest transactions that arrived at the malicious node after starting the attack.

6. After revealing the parasite chain to other nodes of the network, each new-coming honest transaction to an honest node checks for consistency of the Tangle history. It means that incoming transactions check to not verify two tips that are conflicting. Conflicting tips are those tips that one of them directly or indirectly verifies the victim transaction while the other one directly or indirectly verifies the double-spending transaction.
7. We log the victim transaction's confidence rate and the double-spending transaction's confidence rate in each simulation state to visualize them after the simulation is done.

If at some point the double-spending transaction's confidence rate gets greater than the victim transaction's confidence rate, the Tangle network is not secure anymore since the Tangle might get extended from the malicious sub-tangle and the victim transaction might get orphaned.

4.2.4 Experiment Setup

We leverage the Tangle simulator to generate victim and double-spending transactions in the network, and then collect their confirmation data to study the relationship between the network security and configured parameters in the simulation. To collect the data, we consider 10 nodes in the Tangle simulator. In all the experiments we assume that the average hash power of the network is 12e9 trials/sec with a standard deviation of 1e9 trials/sec. We set the transaction arrival rate $\boldsymbol{\lambda}$ to 1, randomness factor $\boldsymbol{\alpha}$ to 0.001, number of the honest transactions to 2500, number of the fake transactions to 8,000, and transmission rate between each two nodes is normally distributed at an average of 5,000 bits/sec and standard deviation of 1,000 bits/sec.

We observe that with the same set of parameters, various values of the network difficulty result in different states of the Tangle network. Low difficulty values compared to the incoming transaction arrival rate $\boldsymbol{\lambda}$, such as 1e4 and 1e8 for $\boldsymbol{\lambda}$ = 1 create a high-load state of the Tangle network: lower values of the difficulty, transaction validation time is lower, and thus several transactions might get validated and propagated to the network at the same time, resulting in higher number of tips. On the other hand, higher values of the difficulty such as 1e12, 1e16, and 1e20, results in low-load states of the Tangle since they create greater validation times. Thus, one transaction gets validated and propagated to the network and then the next transaction gets validated and propagated to the network. That is why at each simulation state we only observe one or two tips in the Tangle.

The experiments we report below cover both low-load and high-load states of the Tangle network. To create a low-load Tangle we set the difficulty to e16 and to have a high-load Tangle the difficulty is set to e8. Moreover, to explore the influence of the malicious hash power and confidence threshold, we run several simulations with 10 different values of the malicious hash power from 10% to 90% with step 10%, and 10 different values of the confidence threshold from 10% to 90% with the step of 10%.

All the experiments are run for the 2500 simulation seconds (~42 simulation minutes). To measure network security, we compare both the victim and the double-spending transactions' average confidence rate among all the 10 nodes during the simulation times. The proposed confirmation rate calculation (see section 4.1.1) is used to extract the transaction confirmation data.

4.2.5 Results

4.2.5.1 High Difficulty Value in the Tangle Network

According to the above parameter setting, in a Tangle network with high value of the difficulty, 1e16, it takes a lot of time for the attacker to validate the transactions. However, the greater the malicious hash power, the greater number of transactions can be created and validated during the simulation time. We observe that after starting the attack while the attacker tries to create and validate the double-spending transaction and as many fake transactions as he can, incoming honest transactions check for the consistency of the Tangle history and if they find two conflicting tips, they choose the one with the greater confidence rate. For lower values of the malicious hash power, independent of what the confidence threshold is, incoming honest transactions select the tip pointing to the victim transaction. For higher values of the malicious hash power, independent of what the confidence threshold is, incoming honest transactions either select the tip pointing to the victim transaction or select the tip pointing to the double-spending transaction. However, in none of the cases the attacker actually manages to win.

Let us look at Figure 31 for an example of this. The simulation scenario includes 2500 honest transactions with transaction arrival rate $\boldsymbol{\lambda} = 1$ and 8000 fake transactions. After the attacker receives the victim transaction, he starts creating a hidden parasite chain. After the victim transaction's confidence rate passes the confidence threshold, the attacker propagates the chain to the network. In Figure 31, when the red line's slope (the attacker's double-spending transaction

confidence rate) starts increasing before the time 1e+08, the double-spending transaction and fake transactions are arriving at other honest nodes in the network, which is why the double-spending transaction's confidence rate is increasing at that point. However, after the time all the nodes receive the transactions on the parasite chain, the victim transaction's confidence rate and the double-spending transaction's confidence rate are both at around 50%. Thus, incoming honest transactions that check for consistency of the Tangle history either select the victim transaction or the double-spending transaction.

In general, our experiments show that for low values of the malicious hash power, the attacker cannot win, and for higher values of the malicious hash power, the attacker only can challenge the honest nodes in the Tangle network but again does not win. In addition, we believe that if the attacker cannot win with this parameter set, then he cannot win with other values of the malicious hash power or confidence threshold. Because either the malicious hash power is less than 90 percent or the confidence threshold is more than 10 percent and according to what we explained earlier in this part, in both cases it is much harder for the attacker to win. In Table 12, the results can be seen for lower values of malicious power ration and higher values of confidence rate, which, as predicted, the attacker cannot win.

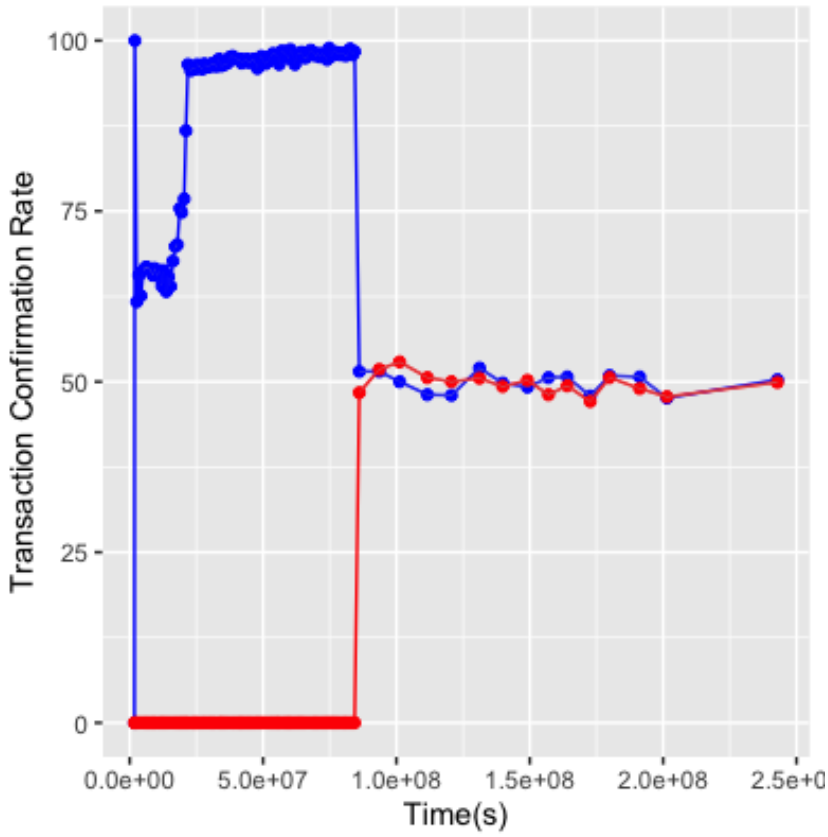
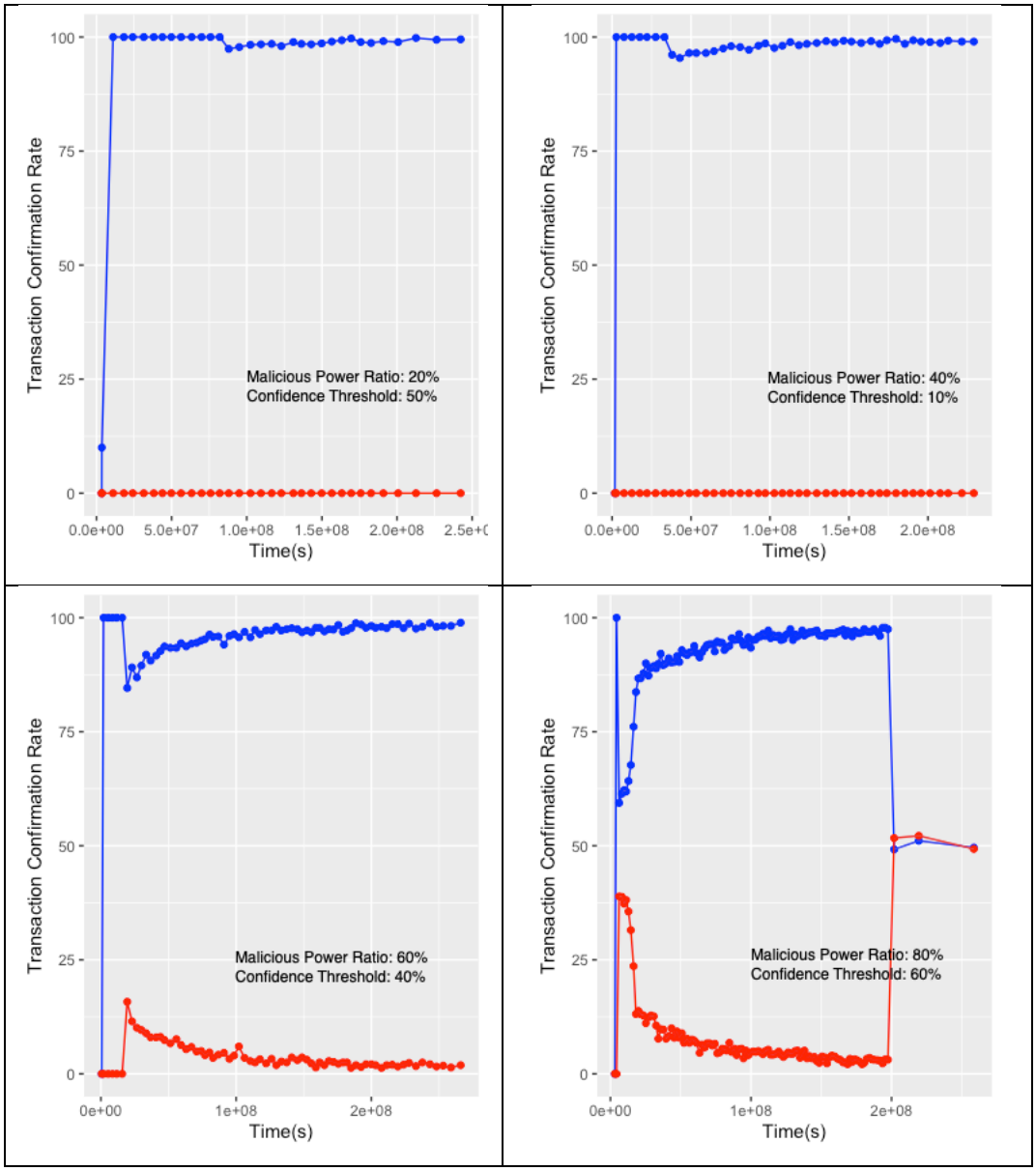


Figure 31: The double-spending transaction's confidence rate and the victim transaction's confidence rate versus the simulation time. Malicious hash power is 90% of total hash power and the confidence threshold is 10%.

In Figure 31 the blue line shows the victim transaction's confidence rate while the red line shows the double-spending transaction's confidence rate.

Table 12: The double-spending transaction's confidence rate and the victim transaction's confidence rate versus the simulation time in Tangle networks with a high difficulty value, 1e16.



The blue line in each plot shows the victim transaction's confidence rate while the red line shows the double-spending transaction's confidence rate.

4.2.5.2 Low Difficulty Value in the Tangle Network

In a Tangle network with low values of the difficulty, we observe that independent of what the malicious hash power is, the attacker eventually wins. When the confidence threshold is lower than 50 percent, the victim transaction's confidence rate gets to this confidence threshold in a

shorter simulation time and thus, the attacker propagates the parasite chain sooner in the Tangle network. In case the confidence threshold is greater than 50 percent, it takes more time for the victim transaction's confidence rate to reach this threshold and the attacker propagates the parasite chain later. However, in both cases the attacker propagates the transactions of the parasite chain at the moment the victim transaction's confidence rate reaches this threshold.

This phenomenon is due to the low difficulty. In low difficulty scenarios, honest transaction rate by which they supply validated transactions to the network is bounded by $\boldsymbol{\lambda}$. That is not the case for the malicious node who generates fake transactions at the rate of their choosing. This way, with low value of difficulty, 1e8, the double-spending transaction and most of the fake transactions get validated and propagated to other nodes in the network in a short amount of simulation time. Later, honest incoming transactions arriving at nodes in the network check for the consistency of the Tangle history. If they see two conflicting transactions, they choose the one with the higher confidence rate. Since there are many fake transactions verifying the double-spending transaction, the double-spending transaction's confidence rate gets higher than the victim transaction's confidence rate. Thus, incoming transactions will choose the double-spending transaction over the victim one. As a result, most of the tips in the Tangle refer to the double-spending transaction. That is why the double-spending transaction gets a higher confidence rate by random walks rather than the victim transaction during the simulation time. In this situation, the victim transaction gets orphaned after a while and the Tangle continues from the parasite sub-tangle.

To perform the experiments, at first, we consider the case of the attack when the malicious hash power is only 10 percent of the total hash power in the Tangle network and the confidence threshold is 90 percent. We believe if the attacker can win with this parameter set, then he can win with other values of the malicious hash power or confidence threshold. Because either the malicious hash power is more than 10 percent or the confidence threshold is less than 90 percent and according to what we explained earlier, in both cases the attacker can win easier.

The behavior of the network in this experimental setting can be seen in Figure 32. The malicious chain gets revealed to the network at simulation time equal to 90.8 seconds. There are 2500 honest transactions with transaction arrival time equal to 1 and there are 8000 fake transactions in the Tangle network. As can be seen, the double-spending transaction and all the fake transactions are validated and propagated to the network soon in the simulation time. In Table

13, the results can be seen for higher values of malicious power ratio and lower values of confidence rate, which, as predicted, follow the same pattern.

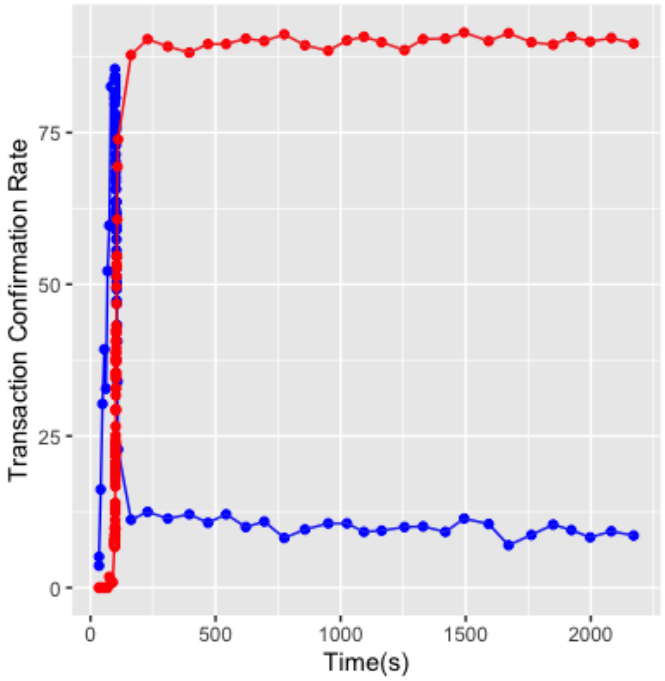
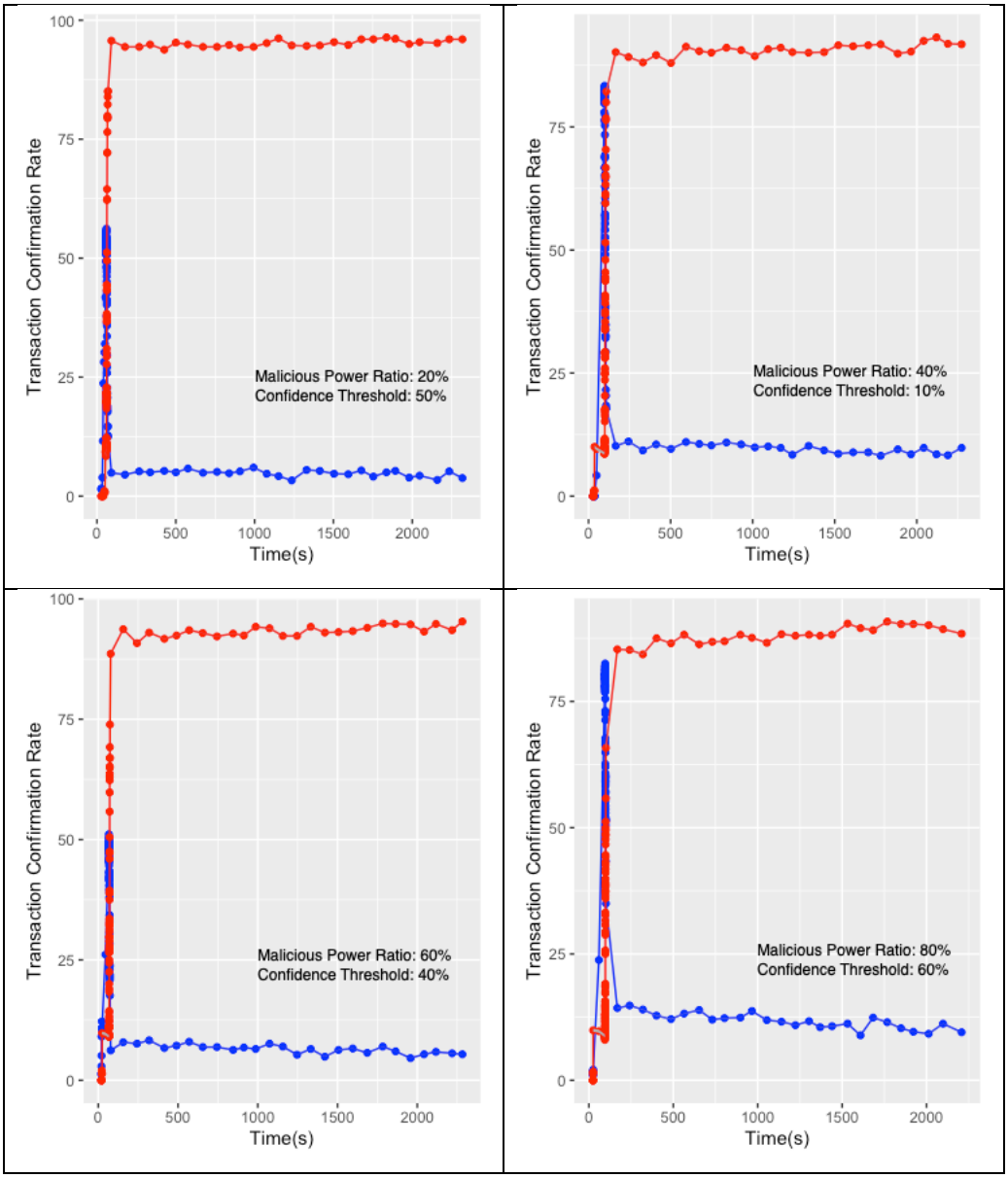


Figure 32: The double-spending transaction's confidence rate and the victim transaction's confidence rate versus the simulation time. Malicious hash power is 10% of total hash power and the confidence threshold is 90%.

In Figure 32 the blue line shows the victim transaction's confidence rate while the red line shows the double-spending transaction's confidence rate.

Table 13: The double-spending transaction's confidence rate and the victim transaction's confidence rate versus the simulation time in Tangle networks with a low difficulty value, 1e8.

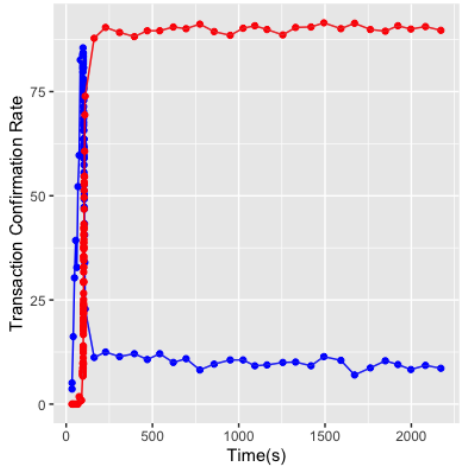
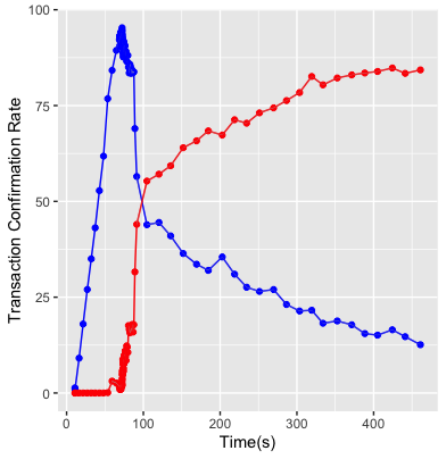


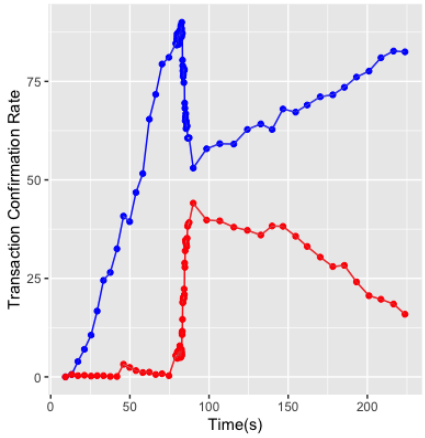
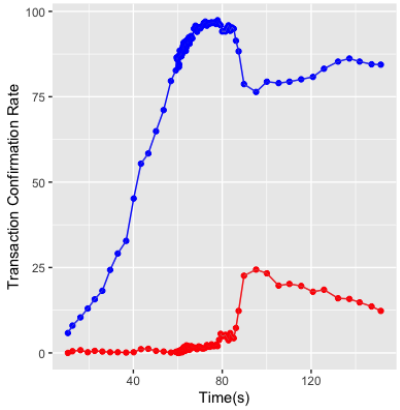
The blue line in each plot shows the victim transaction's confidence rate while the red line shows the double-spending transaction's confidence rate.

According to the experiments above and in response to RQ4, the network difficulty plays an important role in security of the Tangle network. When the difficulty value is high, the malicious hash power affects the experiment results. In experiments with difficulty = e16, the malicious hash power has to be more than 80 percent for the attacker to challenge the honest nodes. However, when the difficulty value is low, e8, malicious hash power is not much important. Instead,

according to Table 14, the transaction arrival rate ($\boldsymbol{\lambda}$) plays a role since for the same parameter setting and for higher values of $\boldsymbol{\lambda}$, honest nodes in the network get busier to validate new incoming honest transactions and it would be more difficult for the attacker to compete with them during the simulation time. Results shown in Table 14 are for different values of $\boldsymbol{\lambda}$, difficulty = 1e8, number of honest transactions = 2500, number of fake transactions = 8000, malicious hash power = 10%, and confidence threshold = 90%.

Table 14: The double-spending transaction's confidence rate and the victim transaction's confidence rate versus the simulation time for various values of $\boldsymbol{\lambda}$ in Tangle networks with a low difficulty value, 1e8.

$\boldsymbol{\lambda}$	**Plot**
1	
5	

10	
15	

4.2.6 Summary of Findings on the Tangle Network Security

In the Tangle network with high values of difficulty, e.g., e16, we observe that for low values of the malicious hash power, the attacker does not win, and for higher values of the malicious hash power (> 80%), the attacker only can challenge the honest nodes in the Tangle network, but again does not win in the examined environment. As a result, it seems that one way to have a secure Tangle network is increasing the network difficulty value. However, as stated previously in section 4.2.2.3, higher values of the difficulty also result in less values of the transaction throughput. Thus, it is vital to find an optimum value of the network difficulty while keeping the transaction throughput high enough.

In a Tangle network with low values of the difficulty, e.g., e8, we observe that independent of what the malicious hash power is, the attacker eventually wins. This phenomenon happens because in low difficulty scenarios, honest transaction rate is bounded by $\boldsymbol{\lambda}$, which is not the case

for the malicious node who can generates fake transactions at an arbitrary rate. This way, the double-spending transaction and most of the fake transactions get validated and propagated to other nodes in the network in a short amount of simulation time, resulting in a higher confidence rate for the malicious transaction. In networks with low value of the difficulty, however, increasing the value of $\boldsymbol{\lambda}$ increases the network resistance to double-spending attacks since honest nodes in the network validate more incoming honest transactions and it gets more difficult for the attacker to compete with them.

Chapter Five - Concluding Remarks

In this thesis, we studied and investigated different features of the Tangle network including scalability, throughput and security. We started the research by first reviewing the literature around the simulation of different blockchain networks aimed to study and evaluate their features. Then, we designed and implemented a Tangle simulator to study the Tangle networks by experimental simulation. We leveraged the simulator to answer the research questions on the Tangle. Through the simulation, we empirically explored and analyzed the influences of various parameters including transaction arrival rate $\boldsymbol{\lambda}$, randomness factor $\boldsymbol{\alpha}$ in weighted random walk, network difficulty, and network transmission rate on the Tangle transaction throughput. Moreover, we assessed the resistance of the Tangle network against double-spending attacks. Through the simulation, we empirically explored and analyzed the influences of various parameters including malicious hash power, confidence threshold, attack duration, network difficulty, and transaction arrival rate $\boldsymbol{\lambda}$ on the Tangle network security.

Our contribution is twofold. First, we designed and implemented a Tangle simulator based on model-oriented design and development principles, which takes a 2D matrix of nodes and transactions, creates a Tangle network according to the algorithms mentioned in the Tangle white paper [38] and outputs a 3D matrix in which the first dimension is nodes, the second dimension is transactions, the third dimension is simulation times, and value of each cell is transactions' confidence rates. In addition, we validated the Tangle simulator against theoretical claims mentioned in the white paper [38]. Second, we leveraged this simulator to investigate three important features of Tangle networks such as scalability, throughput and their security against double-spending attacks. This simulator can be extended for further research on other features of the Tangle network to provide insight for the academic community.

- As for future work, there are several opportunities for follow-up work:
- Improving the scalability and performance of the Tangle simulator through multithreading and parallelism. The most time-consuming computations of the Tangle simulator are first, a recursive function that updates the cumulative weight of all the sites in the DAG whenever a new site is attached to the DAG and second, the MCMC random walk which starts from an entry point and randomly walk towards a tip. Both of these algorithms can be optimized by concurrent execution of more than one thread.
- Study and simulation of other types of blockchain networks using the simulator.

- Evaluation of other BNs features such as latency and privacy.
- Defining common metrics to compare different BNs in terms of scalability, throughput, security, privacy, etc.
- Study and investigation of how other consensus algorithms such as FPC-BI [55] affect the Tangle's security.
- Implementation of transaction reattachment, which is the process of issuing the same original transaction to a new position in the Tangle, to increase the confirmation probability.
- Designing a control-theory model to adjust Tangle network parameters and make it adaptive. This model can be used to adjust an optimum value of the network difficulty to increase the network security against double-spending attacks while keeping the transaction throughput high enough.

References:

[1] S. Liaskos, B. Wang, N. Alimohammadi, “Power Games: Modelling and Simulating Proof-of-Work Network Competitions,” pp. 1–5, 2019, unpublished.

[2] S. Nakamoto,” Bitcoin: A Peer-to-Peer Electronic Cash System,” www.bitcoin.org, Tech. Rep., 2008. [Online]. Available: https://git.dhimmel.com/bitcoin-whitepaper/

[3] J. Garay, A. Kiayias, and N. Leonardos, “The Bitcoin Backbone Protocol: Analysis and Applications,” Advances in Cryptology - EUROCRYPT, 2015, pp. 281–310.

[4] F. Casino, T. K. Dasaklis, and C. Patsakis, “A systematic literature review of blockchain-based applications: Current status, classification and open issues,” Telemat. Informatics, vol. 36, pp. 55–81, 2019.

[5] S. Liaskos, T. Anand and N. Alimohammadi, "Architecting blockchain network simulators: a model-driven perspective," IEEE International Conference on Blockchain and Cryptocurrency (ICBC), Toronto, ON, Canada, 2020, pp. 1-3, doi: 10.1109/ICBC48266.2020.9169413.

[6] L. Stoykov, K. Zhang, and H.-A. Jacobsen, “VIBES: Fast blockchain simulations for large-scale peer-to-peer networks: Demo,” Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference (Middleware’17): Posters and Demos. Las Vegas, Nevada: ACM, 2017, pp. 19–20. [Online]. Available: http://doi.acm.org/10.1145/3155016.3155020

[7] F. Sch¨ussler, P. Nasirifard, and H.-A. Jacobsen, “Attack and vulnerability simulation framework for Bitcoin-like blockchain technologies,” Proceedings of the 19th International Middleware Conference (Posters) (Middleware’18). Rennes, France: ACM, 2018, pp. 5–6. [Online]. Available: http://doi.acm.org/10.1145/3284014.3284017

[8] A. Deshpande, P. Nasirifard, and H.-A. Jacobsen, “eVIBES: Configurable and interactive ethereum blockchain simulation framework,” Proceedings of the 19th International Middleware Conference (Posters) (Middleware’18). Rennes, France: ACM, 2018, pp. 11–12. [Online]. Available: http://doi.acm.org/10.1145/3284014.3284020

[9] S. Goswami, “Scalability analysis of blockchains through blockchain simulation,” Master’s thesis, Department of Computer Science, University of Nevada, Las Vegas, 2017.

[10] R. Yasaweerasinghelage, M. Staples, and I.Weber, “Predicting latency of blockchain-based systems using architectural modelling and simulation,” IEEE International Conference on Software Architecture (ICSA 2017), Gothenburg, Sweden, 2017, pp. 253–256.

[11] Gervais, A., et al., “On the security and performance of proof of work blockchains,” Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016, pp. 3–16.

[12] J. Kreku, V. Vallivaara, K. Halunen, and J. Suomalainen, “Evaluating the efficiency of blockchains in IoT with simulations,” Proceedings of the 2nd International Conference on Internet of Things, Big Data and Security (IoTBDS), Porto, Portugal, 2017, pp. 216–223.

[13] Göbel, J., et al., “Bitcoin blockchain dynamics: the selfish-mine strategy in the presence of propagation delay,” Perform. Eval. 04, pp. 23–41, 2016.

[14] Göbel Johannes, Joschko Philip, Koors Arne, Page Bernd, “The discrete event simulation framework DESMO-J: Review, comparision to other frameworks and latest development”, Proceedings in 27th European Conference on Modelling and Simulation (ECMS), 2013.

[15] S. Liaskos, B. Wang, and N. Alimohammadi, “Blockchain networks as adaptive systems,” Proceedings of the 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS@ICSE 2019), Montreal, QC, Canada, 2019, pp. 139–145. [Online]. Available: https://doi.org/10.1109/SEAMS.2019.00025

[16] S. Liaskos and B. Wang, “Towards a model for comprehending and reasoning about PoW-based blockchain network sustainability,” Proceedings of the 33rd Annual ACM Symposium on Applied Computing, (SAC 2018), Pau, France, 2018, pp. 383–387. [Online]. Available: https://doi.org/10.1145/3167132.3167175

[17] T. Neudecker, P. Andelfinger, and H. Hartenstein, “A simulation model for analysis of attacks on the Bitcoin peer-to-peer network,” Proceedings of the 2015 IFIP/IEEE International

Symposium on Integrated Network Management (IM), Ottawa, Canada, May 2015, pp. 1327–1332.

[18] B. Wang, S. Chen, L. Yao, B. Liu, X. Xu, and L. Zhu, "A Simulation Approach for Studying Behavior and Quality of Blockchain Networks," Proceedings of the International Conference on Blockchain (ICBC 2018), Seattle, USA, 2018, pp. 18–31.

[19] "Blockchain Simulator", https://github.com/concept-inversion/blockchain-simulator, accessed: 2019-12-13.

[20] M. Alharby and A. van Moorsel, "Blocksim: A simulation framework for blockchain systems," SIGMETRICS Performance Evaluation Review, vol. 46, no. 3, Jan. 2019, pp. 135–138. [Online]. Available: http://doi.acm.org/10.1145/3308897.3308956

[21] Y. Aoki, K. Otsuki, T. Kaneko, R. Banno, and K. Shudo, "Simblock: A blockchain network simulator," IEEE Conference on Computer Communications Workshops (INFOCOM 2019 Workshops), Paris, France, April 2019, pp. 325–329.

[22] "BlockSim GitHub Page," https://github.com/blockbirdLabs/blocksim, accessed: 2019-12-13.

[23] "SimBlock GitHub Page," https://dsg-titech.github.io/simblock/, accessed: 2019- 12-13.

[24] B. Kusmierz, "The first glance at the simulation of the Tangle: discrete model," IOTA Found. WhitePaper, 2017, pp. 1–10.

[25] K. Bartosz and S. Philip, "IN PA Extracting Tangle Properties in Continuous Time," IOTA Found., 2018.

[26] M. N. Nguyen, "Tangle Simulation," https://github.com/minh-nghia/TangleSimulator, accessed on 2020-06-22.

[27] M. Zander, T. Waite, and D. Harz, "DAGsim: Simulation of DAG-based distributed ledger protocols," Perform. Eval. Rev., vol. 46, no. 3, 2019, pp. 118–121. [Online]. Available: https://doi.org/10.1145/3308897.3308951

[28] GitHub, “IOTA TANGLE SIMULATION,” https://github.com/IC3RE/DAGsim, accessed on 2020-06-22.
[29] C. Fan, “Performance Analysis and Design of an IoT-Friendly DAG-based Distributed Ledger System,”, Master’s thesis, Department of Computer Science, University of Alberta, Edmonton, Alberta, 2019.
[30] M. R. A. Lathif, P. Nasirifard, and H. A. Jacobsen, “Demo abstract: Cidds: A configurable and distributed dag-based distributed ledger simulation framework,” ACM/IFIP/USENIX Middlew. Conf., 2018, pp. 7–8.
[31] GitHub, “CIDDS: A Configurable and Distributed DAG-based Distributed Ledger Simulation Framework,” https://github.com/i13-msrg/cidds, accessed on 2020-06-22.
[32] W. Sanders, A. Penzkofer, A. Capossele, and A. Gal, “Properties of the Tangle for Uniform Random and Random Walk Tip Selection. (arXiv:2001.07734v1 [cs.DC]),” IEEE International Conference on Blockchain, 2020.
[33] A. Gal, “The Tangle: An Illustrated Introduction,” https://blog.iota.org/the-tangle-an-illustrated-introduction-4d5eae6fe8d4. accessed on 2020-06-24.
[34] M. Bottone, F. Raimondi, G. Primiero, “Multiagent-based simulations of block-free distributed ledgers,” Proceedings - 32nd IEEE International Conference on Advanced Information Networking and Applications Workshops, WAINA, 2018.
[35] Northwestern, NetLogo. [Online]. Available: https://ccl.northwestern.edu/netlogo, 2020
[36] F. Casino, T. K. Dasaklis, and C. Patsakis, “A systematic literature review of blockchain-based applications: Current status, classification and open issues,” Telemat. Informatics, vol. 36, no. November 2018, pp. 55–81, 2019.
[37] S. Popov, “The Tangle (v 0.6),” IOTA Foundation, 2016. [Online]. Available:
[38] S. Popov, “The Tangle (v 1.4.3),” IOTA Foundation, 2018. [Online]. Available: https://assets.ctfassets.net/r1dr6vzfxhev/2t4uxvsIqk0EUau6g2sw0g/45eae33637ca92f85dd9f4a3a218e1ec/iota1_4_3.pdf
[39] DIGNOMINICA, “Can IOTA’S blockless blockchain become the IoT standard?”, https://diginomica.com/can-iotas-blockless-blockchain-become-iot-standard, accessed: 2020-08-05.

[40] MCKINSEY, “Growing opportunities in the Internet of Things”, https://www.mckinsey.com/industries/private-equity-and-principal-investors/our-insights/growing-opportunities-in-the-internet-of-things, accessed: 2020-08-05.

[41] IOTA, “Powering Industry 4.0”, https://www.iota.org/solutions/industrial-iot, accessed: 2020-08-05.

[42] IOTA, “What Is IOTA?”, https://www.iota.org/get-started/what-is-iota, accessed: 2020-08-05.

[43] Carnegie Mellon University, An Introduction to Discrete-Event Simulation, https://www.cs.cmu.edu/~music/cmsip/readings/intro-discrete-event-sim.html, accessed: 2019-09-01.

[44] STATISTA, “Size of the Bitcoin blockchain from 2010 to 2020, by quarter”, https://www.statista.com/statistics/647523/worldwide-bitcoin-blockchain-size/, 2020

[45] D. Aggarwal, G. K. Brennen, T. Lee, M. Santha and M. Tomamichel, "Quantum attacks on bitcoin and how to protect against them", arXiv:1710.10377, 2017, [online] Available: https://arxiv.org/abs/1710.10377, 2008, pp. 1–21.

[46] P. Ferraro, C. King, and R. Shorten, “IOTA-based Directed Acyclic Graphs without Orphans,” ArXiv:1901.07302, 2018, pp. 1–11.

[47] F. S. Chafjiri and M. Mehdi Esnaashari Esfahani, “An Adaptive Random Walk Algorithm for Selecting Tips in the Tangle,” 5th Int. Conf. Web Res. ICWR, 2019, pp. 161–166.

[48] Q. Bramas, “The Stability and the Security of the Tangle,” IOTA Found., 2018.

[49] A. Cullen, P. Ferraro, C. King, and R. Shorten, “Distributed Ledger Technology for IoT: Parasite Chain Attacks,” ArXiv:1904.00996, 2019, pp. 1–9.

[50] G. O’Regan, “Concise Guide to Software Testing”. Springer International Publishing, 2019, pp. 83-85. [Online]. Available: https://doi.org/10.1007/978-3-030-28494-7.

[51] IOTA, “Meet the tangle,” https://www.iota.org/research/meet-the-tangle, accessed: 2020-01-20.

[52] IOTA, “Developer Documentation”, https://docs.iota.org, accessed: 2020-01-20.

[53] IOTA, “The structure of the Tangle - Number of approvers”, https://blog.iota.org/the-structure-of-the-tangle-number-of-approvers, accessed: 2020-01-25.

[54] IOTA, “The many faces of the Tangle”, https://blog.iota.org/the-many-faces-of-the-tangle, accessed: 2020-01-25.

[55] S. Popov and W. J. Buchanan, “FPC-BI: Fast Probabilistic Consensus within Byzantine Infrastructures,” ArXiv:1905.10895, 2019, pp. 1–26.