Use of IOTA 1.0 for Micropayments in Transport and IoT

Pedro Miguel Fernandes de Bastos Sécio Gomes
pedro.miguel.gomes@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

Setembro 2020

Abstract

Less than 12 years after their inception, cryptographic coins have emerged from obscurity to excite businesses and consumers, as well as central banks and other authorities. They attract because they promise to replace trust in long-standing institutions with trust in a new, fully decentralised system based on blockchain and distributed ledger technology (DLTs). This technology has considerable processing costs, bearable for high-value financial transactions, but unsuitable to support numerous low-value transactions. New variants of the algorithms initially used to allow the implementation of micro-payments in Machine-to-Machine (M2M) transactions associated with the Internet of Things (IoT) have been proposed, as in the case of IOTA, a digital currency that promises to be suitable to support payments associated with IoT, which avoids the weight associated with implementing the blockchain while maintaining distributed validation of transactions. The aim was to assess the suitability of the IOTA for the implementation of micro-payments in the area of IoT applied to public transport. As a basis, we intended to take advantage of the immutability of the issued tokens, allowing the registration of transactions in an asynchronous way, keeping the validation and accounting of these transactions in a safe and distributed manner. In the proposed solution, the various models for implementing IOTA in its application to public transport have been studied and deepened due to their limitations and advantages. The results of the work have demonstrated a solution with adequate processing capacity, but with many limitations for implementation in a real case scenario, in its current state. No paragraph breaks.

Keywords: Blockchain, Distributed Ledger Technology, Integrated ticketing, Internet of Things, IOTA, Micropayments

1. Introduction

Blockchain, IOTA and other forms of Distributed Ledger Technologies (DLT) have been identified as one of the most exciting sets of technologies since the appearance of the Internet. The DLT is based on a shared, encrypted database to store, protect and validate electronic transactions without the need for a central validation system. In this way, it presents itself as a decentralised, open and public" ledger system", similar to a database, whose validation is done, according to a certain consensus protocol, by its users. The great leap in popularity of the DLT took place in 2007 with the creation of Bitcoin Blockchain. However, when looking at its applicability to the Internet of Things (IoT), issues such as scalability, offline accessibility, transaction fees and quantum security were not resolved. Alternatively, the IOTA Foundation developed and published the tangle, which it claimed resolved these issues inherent in the Blockchain, and which wanted to equip the IOTA with an appropriate distributed ledger. A scalable ledger was essential to be able to handle the huge number of txs sent by millions of devices, including micro-txs and zerovalue txs, as messages or sensor data, in other to reduce the high transaction costs for numerous IoT scenarios. The Tangle, while presenting very similar characteristics to Blockchain, would allegedly be able to solve all the above-mentioned problems. This growing popularity of DLT has triggered a wave of innovations, experiments, analysis and research, which has sparked, among others, the financial sector, which has been faced with the emergence of a vast number of explorations made using this technology for payments and settlements. These experiences with DLT demonstrate their potential for the next generation of payment systems, improving the integration and reconciliation of settlement accounts and their ledgers, involving electronic and mobile payment programmes that enable integrated payments, in real time, flatter structures, continuous operations and global reach. and end-to-end payment and settlement transfers. In addition to monetary transactions, experts have already begun to perceive and study the usefulness of this technology in changing the mobility sector. European Commissioner for Transport, Violeta Bulc. recognised that it could help in the challenge of integrating shared mobility services. Applications in this area allow actors to establish direct relations between themselves according to a commonly agreed set of rules and a high degree of trust, without having to go through a central authority. By combining a common language and syntax for the "mobility internet" and new means of accounting transactions, these applications can help redefine the way people and organisations access, pay for, use and/or manage mobility services through a wide network of unrelated and competing transport service providers and platforms. Thus, the main objective of this pioneering research work was to investigate the applicability of IOTA in the mobility sector by conducting a series of benchmarks on the technology and developing a Proof of Concept to prove the feasibility of the designed solution. To this end, it was essential to understand how well IOTA integrates with existing different operation models, to develop a seamless proof-of-concept demonstrator for the practical validation of results and to explore the resources needed for its practical implementation. As a result of this research and experience on the use of DLT for micropayments in a transportation ticketing system, has provided information on its potential benefits, risks, limitations, and implementation challenges, considering that, despite its limitations and lack of maturity, this technology presents great potential for implementation and likely long-term applications and benefits for the development of the micropayment system in IoT, in future versions.

2. Background and Related Work

2.1. Distributed Ledger Technologies

Distributed Ledger Technology (DLT) is defined by Prableen Bajpai as "... the technological infrastructure and protocol that allow simultaneous access, validation and record updating in an immutable manner across a network spread across multiple entities or locations". Its characteristics of decentralisation, immutability, and scalability have given it the potential to manage and register secure micropayments, constituting a possibility to overcome problems in IOT systems. There are fundamentally two main types of DLT, blockchain based on blocks or like a directed acyclic graph (DAG) where there are no blocks (for example, IOTA Tangle). Blockchain is a distributed ledger for storing and sharing data across all nodes in a network. This ledger is considered tamper-proof, making it exceptionally suitable for cryptocurrencies, for example, Bitcoin, which, however, has posed problems of scalability, as it is slow and expensive to perform transactions in the Bitcoin blockchain, due to the single chain of blocks being linear, and the blocks cannot be created simultaneously.

2.2. Distributed Ledger Technology in Cryptocurrencies

The blockchain is a distributed ledger that is updated in groups of transactions called blocks. Blocks are then chained sequentially via the use of cryptographic hashes to form the blockchain, que have seven principal characteristics: (1) Distributed databases and ledgers, (2) Irreversibility of records, (3) Transparent identity management with pseudonymity, (4) Robust validation and consensus, (5) Peer-to-peer transmission, (6) Computational logic. Blockchain-based permissionless cryptocurrencies have two groups of participants: “miners” who act as bookkeepers and “users” who want to transact in the cryptocurrency. At face value, the idea underlying these cryptocurrencies is simple: the ledger is updated by a miner, and the update is subsequently stored by all users and miners. Underlying this setup, the key feature of these cryptocurrencies is the implementation of a set of rules (the protocol) that aim to align the incentives of all participants to create a reliable decentralised payment technology. An oversight committee roposes that any transaction could be authenticated, and any transmitted piece of information maintained by an emergent process of consensus among a globally distributed network of peers that follow a precise, incorruptible method to check any change in the system. The cryptographic identity of each new block in a blockchain must be validated before it can be included in the latest iteration of the ledger that is propagated to, and recognized by, all nodes. The recording of the blockchain database at any given time is permanent, chronologically ordered, and available to all others on the network. This immutability is at the heart of the “trustfulness” of the blockchain. What makes this technology so appealing and game changing is the absence of third parties, such as payment processors, during the exchanges. This means that for a transaction to be made, it has to be validated by the community (peer-to-peer).

3. IOTA

3.1. Overview

The Tangle runs an asynchronous protocol in a peer-to-peer network to facilitate transaction processing on an immutable, distributed, decentralized ledger secured by cryptographic measures. The Tangle is comprised of transactions, or sites, and edges, which connect sites and form a DAG. The network consists of nodes and each node stores its version of the Tangle. An edge indicates that one site directly approves another. A path symbolizes indirect approval. When a new transaction is issued, it approves two other transactions, which are previous transactions that have never been approved by 16 other transactions. The protocol validates if two approved transactions conflict by examining the Tangle history, and if it discovers a conflict between them, it will not approve those transactions, otherwise a node will store the transaction in its ledger and broadcast it to its neighbors. Each transaction's weight determines its importance in the Tangle. The Tangle defines a transaction's cumulative weight as the sum of the weights of other nodes that directly or indirectly approve the transaction, including itself. Therefore, the number of nodes that directly or indirectly approve the node represents the cumulative weight. The transactions that are chosen to be approved, are done so using a random walk tip selection algorithm. Currently, the algorithm that biases the random walk is the Markov Chain Monte Carlo (MCMC). In a Markov chain each step does not depend on the previous one, but follows from a rule that is decided in advance, taking into consideration the cumulative weight of the transactions. One of the main concerns of blockchain security is the malicious node issue. Without the use of a tip selection algorithm, like MCMC random walk, nodes can become lazy and allow lazy tips, which are tip transactions that point to older transactions, to be confirmed. Confirming old transaction is unwanted since it increases the branching factor of the graph and thus, it increases the number of tips. Furthermore, lazy nodes do not help the network to grow since no new unapproved transactions get confirmed and can easily allow the occurrence of double-spending attacks on the network, such as the parasite-chain. Despite this, this type of attacks can still occur when using the tip selection algorithm if an attacker can generate enough cumulative weight on his transactions to surpass the cumulative weight of the Tangle. To avoid this, the current IOTA system's stability relies also on a particular type of node, called the coordinator node. The coordinator issues signed zerovalued transactions at a certain rate, which are called milestones, and confirms every transaction in the path of the selected transactions to approve to the latest milestone (at least). The above mentioned tip selection algorithm starts the random walk from a milestone at a certain depth. That said, the nodes validate transactions upon its receipt and during the tip selection. The coordinator confirms transactions by issuing milestones. A transaction is only considered for confirmation if the node has its transaction history path until a milestone and if the node is synchronized with the network i.e. possesses a ledger equal to the other nodes and the latest milestone index.

3.2. Data Models

IOTA uses a balanced ternary numeral system composed of trits and trytes. This system is used all through the protocol, from seed and address generation to transaction generation, content and validation. The current implementation of the Tangle uses 81-character seeds, which can be seen as the access-key, necessary to perform operations on the Tangle. An address is like an account that belongs to a seed and that has a 0 or greater balance of tokens. Addresses are derived from the seed and can be seen as the public half of a public/private key pair, except both keys are generated using the Kerl [21] hash function, which is based on SHA-3 [20]. The corresponding private keys are used to sign transactions to prove their ownership and as such, they should not be reused. A transaction is a single transfer instruction that can either withdraw IOTA tokens from an address and deposit them into another address or have zero-value. Transactions considered by the Tangle protocol can be in two states: validated and unconfirmed (called tips) or confirmed. A bundle is a group of one or multiple related transactions, that rely on each other's validity. It acts as a transactions container to transfer data or tokens where transactions reference each other through their hashes and, when sent to the Tangle, the bundle must reference other two transactions. It is always an atomic operation, i.e., either all transactions are successful or none.

3.3. Network

In IOTA there are no access controls for participants to join the network so that anyone can run a node to read from and write into the public ledger. Nodes are the core of the Tangle and connected to others (neighbors) they form the IOTA network. They are responsible for the following key functions: (1) Keeping a record of the addresses with a balance greater than 0, (2) Validating transactions when performing the tip selection, (3) Attaching valid transactions to the Tangle upon its receipt and (4) Broadcasting transactions to neighbors. The process of validating a transaction is to make sure that their histories in the ledger do not conflict and that counterfeit transactions are never confirmed. Transactions are only confirmed by the coordinator. The coordinator is an application whose purpose is protect the Tangle from double-spending attacks. Therefore, nodes use the signed transactions issued by the coordinator, called milestones, to reach a consensus on which transactions are confirmed. When a valid milestone references an existing transaction in the Tangle, nodes mark the state of that existing transaction and its entire history as confirmed and the tokens are transferred.

3.4. Tip Selection

The selection of each tip transaction is done by using a random walk biased with the Markov Chain Monte Carlo (MCMC) algorithm. A walker starts from a milestone at a given depth and transverses a path, validating every transaction, until it reaches a tip. At each transaction in the path, it uses a transition function depending of the the transaction's cumulative weight. The higher the cumulative weight, the higher the probability of that transaction being confirmed later.

3.5. Proof-of-work (PoW)

In the Tangle, as a spam prevention measure such as Hashcash [16], each transaction must include a PoW result to be valid. A proof of work (PoW) is a piece of data that is calculated using trial and error to meet certain requirements. This PoW can be difficult to do, depending on the chosen difficulty level in the network (Minimum Weight Magnitude).

3.6. Transaction workflow

The process flow to complete a transaction is described below: (1) Generate bundle hash: Kerl hash function with sponge constructor absorbs all transactions objects necessary for validation and squeezes the hash, (2) Sign Input Transactions: The signature is generated using the bundle hash and a private key., (3) Tip selection: MCMC is used to randomly select two unvalidated transactions., (4) Proof-of-work: For each transaction included in the bundle, the PoW is computed., (5) Broadcast and Validation: Each transactions is broadcasted. Transaction validation and confirmation is then required to transfer the funds.

3.7. Security Analysis

There are multiple ways to attempt a double-spending attack in IOTA. A few of them are described in the white paper [22]. These are the large weight / outpace attack [22] [19], the splitting attack [22] and the parasite chain attack [22] [17]. The following does not attempt to be a comprehensive list. There are other possible attacks, e.g., replay attacks [18], 34% attack [14], Sybil Attack [22].

4. Solution

The main goal of this solution is to create an immutable ticketing system as a proof of concept using IOTA, and allow further benchmarks. This system will allow users to execute different use cases on different transport models, such as the traditional model (e.g., metro) where there is a ticket validation component at the entrance of the service provider gateway and the ticket is handled to this gateway, the taxi model where the client can travel and pay based on distance to travel and finally the wallet consumption model (e.g., eletrical scooter) where a check-in and check-out are requested to start and stop the journey respectively.

4.1. Overview

The solution is divided into three parts: Frontend (client and service provider applications), Wallet Manager API, and IOTA, as shown in Figure 1. A client represents the end-user of the ticketing system, while a service provider represents a transport operator.

The design of the IOTA network for this system had to take into account some assumptions and some limitations, discussed in section 5, and is mainly comprised with three components: the coordinator, which runs Compass [3] and IRI software, the nodes, which run IRI, and the proxies, which run Caddy [2]. All this machines were equipped with a Ubuntu Server 18.04 with 4GB of RAM and a 64 bit processor. A proxy is solely dedicated to performing proof of work and proxying the other client's requests to the node. Considering the functioning and importance of the node, seen previously in section 2, and compared to the proxy, a proxy would apparently be more suited for proof of work computations and handling client's requests (which was later shown in section 5). As such, the design of the network consists in a coordinator, a node per service provider and a proxy per service provider transport gateway/terminal, which could be increased as further needed. These were setup in a virtualized environment (VMware [13]). After configuring each component, the first node on the network was launched from a snapshot containing the genesis transaction of the network, that is, the starting state of the network. In the initial state of this solution's network, all available IOTA tokens are associated to a wallet manager's address. Therefore, it is necessary to create a seed and generate an address for the entity that manages this solution beforehand. All the IOTA tokens in circulation on the network will be distributed by this seed. The Figure 2 demonstrates the network component's software and communication channels.

The wallet manager serves the client and service provider applications through a REST API and it was developed using the Spring Boot [10] for Java. This component acts an intermediary between the users and IOTA network, responsible for keeping client seeds, service provider seeds, addresses and prices in a SQLite [11] database, and performing operations such as balance updates, token transfers and ticket generation. Its main operations are described in Table 1 and in Table 2. It also uses the IOTA IRI API which is the node's API to interact with the network and it was developed using Jota (Official IOTA client library for Java) [5]. The core operations performed when sending a transaction are prepareTransfers, validateTransferAddresses, attachToTangle, storeTransactions and broadcastTransactions, as described in Table 3

The frontend consists in a single multi-platform application developed using the Ionic framework [4] whose intent is to provide an interface to use the

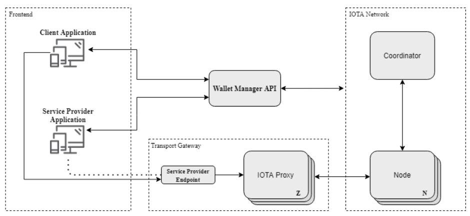

Figure 1: Solution's High-Level Architecture

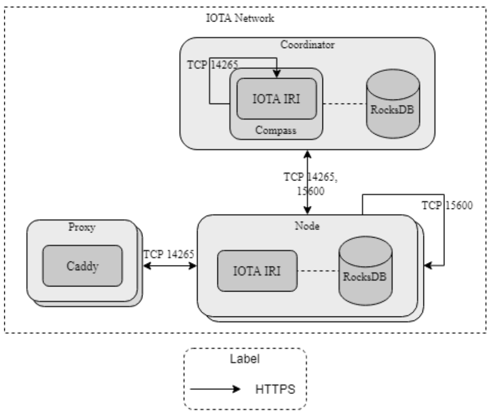

Figure 2: IOTA Network Detailed Architecture

Wallet Manager API (in most cases) and it has different functionalities for both the client and service provider. The client application should be used by the clients of the system (i.e., a person who wants to buy and/or use tickets to ride a form of public transport). Its main functionalities consist in: (1) Balance and transaction history update, (2) Token acquisition and (3) Different ticket generation alternatives. The service provider application should be used by the service providers of the system (i.e., Taxi, Metro, Bus). Its main functionalities consist in: (1) Balance and transaction history update, (2) Address update and (3) Price update. The service provider endpoint was also implemented in the service provider application in order to nimbly simulate a transport gateway/terminal. Service provider's endpoint is able to: (1) Receive tickets and (2) Send transactions.

4.2. Models

As a trust model, we assume the client as the only possible threat to the system. Therefore, client's application can act maliciously: clients can try to get access to other client's accounts, send fake tickets or reuse tickets (double-spending). Following STRIDE, Tampering and Repudiation threats are found in a scenario where a client receives a ticket from the wallet manager and is able to change it before broadcasting it to the service provider endpoint. To solve them, it is necessary to guarantee authenticity, integrity and non-repudiation of the ticket.

Operation	**Action**
Create User	Registers user login information in database
Login	Login client
Check Balance	Get user's balance, transaction history and addresses with tokens
Charge Wallet	Transfers tokens to a client address
Transport Price	Get requested service provider's transport price
Taxi Price	Get requested service provider's transport price based on distance
Ticket	Generates a ticket for the selected service provider transportation option
Check-in ticket	Generates a check-in ticket for the selected service provider transport
Check-out ticket	Generates a check-out ticket for the check-in ticket previously generated

Table 1: Client description of operations

Operation	**Action**
Create Service Provider	Registers service provider login information in database
Login	Login service provider
Check Balance	Get providers's balance, transaction history and addresses with tokens
Updated Price	Change service provider transport price
Update Address	Generate a new receiver address for the service provider

Table 2: Service Provider description of operations

Operation	**Action**
attachToTangle	Does proof of work for the given transaction
broadcastTransactions	Sends transaction trytes to a node
checkConsistency	Checks the consistency of transactions
findTransactions	Finds transactions that contain the given values in their fields
generateNewAddresses	Checks if a set of transactions is confirmed
getAccountData	Returns addresses balance, transaction history and associated tokens
getBalances	Calculates the confirmed and unconfirmed balance of addresses
getInclusionStates	Checks if a set of transactions is confirmed
getInputs	Gets the input addresses of a seed
getTransactionsToApprove	Gets two consistent tip transaction hashes to use as tips
getTransfers	Finds all the bundles for all the addresses based on the seed
getTrytes	Gets a transaction's contents in trytes
prepareTransfers	Prepares transfer by generating bundle, finding and signing inputs
sendTransfer	Runs prepareTransfers, attachToTangle and then storeAndBroadcast
storeAndBroadcast	Runs storeTransactions and broadcastTransactions
storeTransactions	Stores transactions in a node's view of the Tangle
validateTransfersAddresses	Validates the supplied transactions for correct input/output and key reuse
wereAddressSpentFrom	Check if a list of addresses was ever spent from

Table 3: Jota library used operations

4.3. Overlayer

The requirements presented in the previous section are fulfilled using digital signatures. In order to do so, the wallet manager needs to generate a public / private key pair for each new registered service provider and the public key must be known by all service provider endpoints. After that, the user can then request a ticket for that service provider. The ticket is generated and the aggregation of the ticket with the ticket time of expiration is signed. Both the signature and the ticket are sent to the client. The client then sends this data to the service provider endpoint which will verify the signature and if the ticket did not expire yet. If everything is validated, the transactions is sent to the node and the transaction ID contained in the ticket is broadcasted to all other endpoints in the network. The expiration date together with this broadcast, provide an early attempt to stop double-spending.

4.4. Use Cases

To validate the architecture of this system, four specific use cases were designed to exemplify the main functionalities of the project's features:

1. Check Balance. This operation allows a client or service provider to check its account balance and transaction history. In this use case the client starts by requesting an update of his account status to the wallet manager, which

will request that data to a node in the IOTA network. The node responds with the contabilistic balance (obtained from tip transactions - unconfirmed), the confirmed balance (obtained from confirmed transactions), the history of transactions from/to the client and its addresses which contain balance.

2. Charge Wallet. This operation transfers balance to the client. In this use case the client starts by requesting a debit to his account to the wallet manager, which will generate a transaction from his address to the client's address. When this process has completed, the wallet manager will confirm the debit to the client. The necessary steps to generate an IOTA transaction are represented in order to provide a better understanding on further chapters.

3. Use Transport. This operation allows a client to use a transportation method, transfering value from his account to the service provider. In this use case the client starts by requesting the price of the chosen transport. Then, he requests a ticket to the wallet manager which will generate an unfinalized bundle with more parameters, called a ticket, and send it to the client. The client can then use this ticket by sending it to the service provider endpoint. The latter will validate the parameters added by the wallet manager and allow or deny the use of the transport. At the same time, or when available, the proxy will compute the proof of work and broadcast the bundle to a node in the IOTA network, at the service provider endpoint's request.

4. Use Transport (Wallet Consumption). This operation allows a client to use a transportation method by doing a check-in and to stop using by doing a check-out. In this use case the client starts by requesting a check-in ticket to the wallet manager which will generate an unfinalized bundle with more parameters and a transactional value of 0, called a check-in ticket, and send it to the client. The client can then use this ticket by sending it to the service provider endpoint. The latter will validate the parameters added by the wallet manager and allow or deny the use of the transport. At the same time, or when available, the proxy will compute the proof of work and broadcast the bundle to a node in the IOTA network, at the service provider endpoint's request. After some time, the client does a check-out, requesting it to the service provider endpoint. The endpoint then calculates the cost of the trip and request a check-out transaction to the wallet manager which will generate an unfinalized bundle with more parameters and a transactional value of the difference between the time of check-out and time of check-in times the service provider rate, called a check-out ticket, and send it to the service provider endpoint. Then the proxy will compute the proof of work and broadcast the bundle to a node in the IOTA network, at the service provider endpoint's request. In the end, the service provider endpoint informs the client of the paid price of the trip.

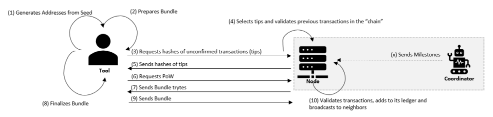
(1) Generates Addresses from Seed
(2) Prepares Bundle
(4) Selects tips and validates previous transactions in the "chain"
(3) Requests hashes of unconfirmed transactions (tips)
(x) Sends Milestones
(5) Sends hashes of tips
Tool
(6) Requests PoW
Node
Coordinator
(7) Sends Bundle trytes
(9) Sends Bundle
(8) Finalizes Bundle
(10) Validates transactions, adds to its ledger and broadcasts to neighbors

Figure 3: Generating and sending a transaction to a node

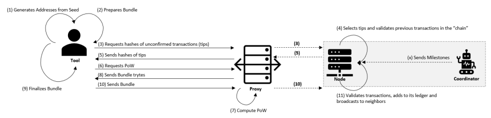
(1) Generates Addresses from Seed
(2) Prepares Bundle
(4) Selects tips and validates previous transactions in the "chain"
(3) Requests hashes of unconfirmed transactions (tips)
(3)
(5) Sends hashes of tips
(5)
(x) Sends Milestones
Tool
(6) Requests PoW
(8) Sends Bundle trytes
Node
Coordinator
(10) Sends Bundle
(10)
(9) Finalizes Bundle
Proxy
(11) Validates transactions, adds to its ledger and broadcasts to neighbors
(7) Compute PoW

Figure 4: Generating and sending a transaction to a proxy

5. Results

The benchmark consists in generating and sending multiple transactions at the same time to a node or/and a proxy. The process of generating and sending a single transaction is represented in Figure 3 for a node and in Figure 4 for a proxy.

IOTA nodes and proxies will be evaluated on two different scenarios via the following metrics: (1) Tip selection (getTips) and proof-of-work (attachToTangle) computation times, (2) Transactions completed per second (Throughput), (3) Time to generate/complete transaction (Latency), (4) Time to confirm transactions (Confirm Latency), (5) CPU and Memory usage. On both scenarios there were also some network variants such as: (1) Type of machine (node or proxy), (2) Number of virtual CPU's/logical cores in the machine (ranging from 1, 2 or 4), (3) Proof-of-work effort (Minimum weight magnitude of 3, 6, 9, 11 or 13) and (4) Different workloads (Number of sent transactions). To achieve this, a self-made benchmark tool was created using Python as the coding language and Pyota [8] as the official Python client library for IOTA since it was the only library that supported asynchronous requests to the nodes. Python, however, does not provide the functionality of "true" threads due to GIL. GIL (Python Global Interpreter Lock) [9] is a mutex (or a lock) that allows only one thread to hold the control of the Python interpreter. This means that only one thread can be in a state of execution at any point in time. Despite this, since most of the tasks were network I/O bound achieving concurrency with asynchronous IO was enough for the task. It was implemented using the asyncio library [1]. Then, after verifying that the tool was not using all the available computing power of the machine that ran the benchmarking tool, the multiprocessing library [6] was used in order to take full advantage of every logical core. On the other hand, the IOTA node was configured to be the publisher of ZeroMQ [15] events, regarding transactions that the IRI node has recently appended to the ledger and transactions that had recently been confirmed, and the tool acted as subscriber. The computational resources of the hosts were logged using the top [12] tool in Linux at a rate of one measure per second. All this data was then compiled to a database and analyzed using Power BI [7]. The environment in which the benchmark was performed consists in two physical machines connected in a LAN. One ran a node and a proxy in a virtualized environment with 4GB of RAM dedicated to each and the number of logical processors of the virtual machines varied between 1, 2 and 4. This physical machine has the following hardware specifications: an AMD Ryzen 7 3700x 8-Core 3.6Ghz with 16 logical processors and 32GB of RAM. The other machine ran the developed benchmark tool once per core variation, that is three runs for the node and three runs for the proxy, making a total of 6 runs. Its hardware specifications are: an Intel I5-8300H 4-Core 2.3Ghz with 8 logical processors and 16GB of RAM. The next sections will describe the gathered results obtained from this tool in different scenarios.

5.1. Performance Tests

In Scenario I, all previously described network variants were used and the following metrics were retireved:

1. Proof-of-work (attachToTangle) and Tip selection (getTransactionsToApprove) computation times.
2. Transactions completed per second (Throughput).
3. Time to generate/complete transaction (Latency).
4. Time to confirm transactions (Confirm Latency).
5. CPU and Memory usage.

The Scenario II consist is one node and one proxies, both with 4 logical cores each, only a proof-of-work effort (Minimum weight magnitude) of 9 is studied and the network has an even higher workload (Higher number of sent transactions) when compared to Scenario I. This time, the tool sent a total of 1200, 1600, 1840, 2000, 2400, 3200 or 4000 transactions in total and the Throughput was limited by the number of transactions requested per second by the tool. The objective of this scenario was to extend the previous benchmarks presented.

5.2. Functional Tests

There were two types of operations to measure: (1) the Wallet Manager API operations and (2) the service provider endpoint operations. The first was tested using JUnit and filters over HTTP(s) requests that provided the execution times for each endpoint. The tests ran 50 times to get an average result and a more accurate representation. The error was gathered from the minimum and maximum execution times in the test. The second was tested automatically in the application with a simple loop function. It also gathered the results of 50 executions and averaged them, and the error was also measured in the same way. Therefore, since some of the created use cases use wallet manager API and service provider endpoint's operations, the results were then summed to create an approximation of the execution times in a real life scenario with only one client. This results obtained seemed to fit to the gathered results for the IOTA network on a low number of total sent transactions.

6. Discussion

This sections justifies some of the design decisions made for this work by analysing three scenarios. We will consider, in the course of this section, as possible double-spending the scenario where a client issues a fake ticket and is able to use a transport, even though the ticket contained in the transaction later becomes detected as a double-spend in the Tangle

1. Proof-of-work is computed at the transport gateways: In this scenario, the minimum weight magnitude can be a lower value than the standard IOTA network because the transactions can be generated by the gates instead of a client, which are less trusted – this results in a more centralized architecture which requires more computational resources. This architecture is not as centralized as the dedicated nodes alternative but it is not so feasible since the transportation gates are usually low-performance devices, resulting in a very high delay to compute proof-of-work, thus making this scenario unfeasible.
2. Proof-of-work is computed at the client's smartphone: In this scenario, the minimum weight magnitude needs to be similar to the standard IOTA network because the transac-

tions are generated by the client and it is used to prevent spamming of the network – this results in a more decentralized architecture making it possible to save network resources. However, because smartphones do not have the same computing power as a dedicated machine and the minimum weight magnitude needs to be high, this results in a high delay to compute proof-of-work. The processing power of the device would also change the experience a lot from client to client and affect its battery. What also makes this scenario unfeasible is the possibility of double-spending. In this scenario the client would need to have in his possession all the necessary information to compute and generate a transaction and due to the time it could take to approve a transaction, the IOTA network would only detect a double-spend much time after the user had delivered the transaction.

3. Dedicated nodes or proxies to compute proof-of-work: In this scenario, the minimum weight magnitude can be a lower value than the standard IOTA network because the transactions can be generated by a dedicated trusted server instead of a client – this results in a more centralized and secure architecture which requires more computational resources but also makes it possible to achieve better proof-of-work performance since the hosts can be high performance machines. It would also allow to implement a solution for the double-spending problem, by making sure that tickets were authentic and not allowing repeated tickets to enter the system. It was, as such, the most fit scenario to implement in a public transport ticketing system.

The previously obtained results were conducted to evaluate the chosen scenario and to provide better insights when designing the solution. The main takeaways were:

1. Proof-of-work computation times highly depend on the chosen minimum weight magnitude
2. Tip selection computation times highly depend on number of transactions received by the host in a period of time, stabilizing after a certain threshold
3. Not enough computational resources on the proof-of-work computing host will cause a great delay in this operation
4. Node's seem to constantly grow its memory usage with time as more are sent transactions at a fixed throughput
5. A proxy is more reliable and better performing on high demanding environments (e.g., higher minimum weight magnitudes and/or higher number of requests)
6. Coordinator nodes constant high computational resources (CPU and/or Network IO) usage will cause a great delay in transaction confirmation speeds or even failure/crash

Gathering all this data and studying it for the chosen scenario, we could theoretically apply it to a real-life scenario. Using a proxy connected to a node, both with 4 logical processors (2 CPU's) and 4GB of RAM, to receive the requests from transport gateway/terminal at the same fixed rate as the previously executed benchmarks on a host with the equivalent specifications, a Throughput of 6 transactions completed per second would be accomplished. This means that a single proxy connected to a node would be able to send around 500.000 transactions to the Tangle per day. If we were to double its computational power to 8 logical cores (4 CPU's), we could also double its Throughput to 12 transactions completed per second and it would be possible to send around 1 million transactions to the Tangle. All this, while most likely not having more than 90% CPU usage at a given point in time. These results make the primary solution previously presented in Chapter 4 viable in terms of performance/cost relationship.

7. Conclusions

Given the defined requirement for processing micropayments in little time without much computational effort, IOTA promised to be the most interesting DLT to study. It was, at the beginning of this project, a relatively unknown and unstudied DLT. Despite all the advantages of using a DLT, such as IOTA, to implement a ticketing system in the mobility sector, there are still some limitations in this technology that were not addressed by its creators in the course of this work. The most important limitations of IOTA in this context were: (1) The high demanding computational effort required when computing the proof-of-work, which makes it difficult to integrate IOTA into low-performance or batterydependent devices, (2) The coordinator as a single point of failure, makes the confirmation of transactions reliant on a single node. If this node fails, the whole network is vulnerable to double-spending attacks and (3) The lack of performance and scalability benchmarks of its components and operations, makes it difficult to design a network able to handle thousands of requests. Some limitations found in the IOTA had to be addressed and to do so, some benchmarks were created and a solution was designed to mitigate some of the limitations in light of the requirements present in a ticketing system for the mobility sector. It was achieved by: (1) Reducing the proof-of-work effort, (2) Designing an overlayer which not only transformed IOTA into a permissioned network, preventing a client from travelling a number of times before its detected in the IOTA network as a double-spend, but also increased the resiliency of the coordinator and the scalability of the network and (3) Developing a series of benchmarks on IOTA's main components, the node and the proxy, in order to design a well-scaled network to support the developed solution and future works. . With these problems addressed and understood, it was then possible to develop a solution as proof-of-concept of a ticketing system for multiple transport modalities and different usability schemes using IOTA. However, this design of the solution brought some other limitations which were not addressed in this work. In conclusion, despite having addressed some of the issues and having designed a workable proof-of-concept, we found IOTA 1.0 to be an going evolution not yet mature enough to be implemented in a large scale real life scenario. For future work it is suggested: 1- To further improve the references implemented in the IOTA components - distribute the client that is performing the requests so that more requests could be sent per second by not being limited to the computational resources of one machine; increase the number of nodes in the IOTA network to study its scalability; Benchmarks could be also executed on lowperformance devices, fit for IoT, on lower proof-of-work efforts. 2- To further improve the solution, the wallet manager component needed to be benchmarked and audited in order to transform it into a secure and highly available component, since without it would be impossible to use the network. An other component could also be added to the solution to provide better auditing capabilities. As a proof-of-concept, we could distribute the designed components into different networks and implemented the service provider endpoint in a low-performance device (e.g., Raspberry Pi), which would allow a more seamless and real-life experience. Since in January 2020 the IOTA Foundation proposed a number of improvements to what they called Coordicide [24], which were not published in time to be included in this research work, and which would take the IOTA to a place closer to the goals set out here, we advise that this project be repeated at the launch of version 2. 0 of the IOTA, as we believe that the conditions are created for IOTA to be the closest technology to the DLT that was initially promised to process micropayments in a short time, without much computational effort on IoT devices.

References

[1] asyncio - python library.

[2] Caddy.

[3] Compass.

[4] Ionic framework.

[5] Jota.

[6] multiprocessing - python library.

[7] Power bi.

[8] Pyota.

[9] Python global interpreter lock.

[10] Spring boot.

[11] Sqlite.

[12] top - linux command.

[13] Vmware.

[14] Xy attack vector - iotas version of the 34% attack.

[15] Zeromq.

[16] A. Back. Hashcash - a denial of service counter-measure. 09 2002.

[17] A. Cullen, P. Ferraro, C. King, and R. Shorten. Distributed ledger technology for iot: Parasite chain attacks. 03 2019.

[18] G. De Roode, I. Ullah, and P. J. M. Havinga. How to break iota heart by replaying? In *2018 IEEE Globecom Workshops (GC Wkshps)*, pages 1–7, 2018.

[19] L. de Vries. Iota vulnerability: Large weight attack performed in a network, January 2019.

[20] M. P. G. V. A. G. Bertoni, J. Daemen and R. V. Keer. Keccak implementation overview. 2012.

[21] IOTA. Kerl.

[22] S. Popov. The tangle. 2018.