FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Supply Chain tracking and management with Distributed Ledger

João Malheiro de Sousa

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Alexandre Valle de Carvalho (FEUP)

Second Supervisor: Tiago Rocha (Associação Fraunhofer Portugal)

July 14, 2021

Supply Chain tracking and management with Distributed Ledger

João Malheiro de Sousa

Mestrado Integrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

Chair: Doctor João Correia Lopes
External Examiner: Doctor José Manuel Matos Moreira
Supervisor: Doctor Alexandre Valle de Carvalho

July 14, 2021

Abstract

Supply Chain is a network that establishes a flow of goods and materials between suppliers, clients, transportation, until the end consumer. The management of this network, which is done in order to maximize customer value, is called Supply Chain Management (SCM). In an agriculture supply chain, for this management to be efficient there are some important aspects that need to be tackled like the traceability of the assets and their transaction costs. Furthermore, the current food traceability options are not linked throughout the chain and that makes it hard to validate the quality of the product at any point of the network.

Distributed Ledger is a decentralized immutable network which can provide a more transparent and safer way to manage a business and the logistics of a supply chain. After being approved, transactions are registered in blocks and sent to every peer. By using this approach, every stakeholder has access to vital data, such as the conditions in which the product is kept, making the whole network more efficient and trustworthy. Currently, the use of Distributed Ledger Technology (DLT) in SCM is still in a pilot phase, but there are multiple cases of success of its implementation in the referred context.

As it stands, it is of major importance that the problem of food traceability and its quality gets tackled so that we can achieve lower transaction time and its costs, and by making the process faster, increase the quality of the products.

Some of the work will be focused on replicating the network capable of handling the transactions in an agricultural supply chain certification phase, in a way that is more traceable, secure and verifiable than the currently used centralized solution. Most of the work will center around the evaluation of a DLT system based on Hyperledger Sawtooth applied to an AgriFood Supply Chain, specifically its product certification phase. In other words, the work will consist in analysing the scalability, the network design and the reliability of the network.

The study shows that such a solution is a good alternative for distributed persistency where all the actors will be able to participate not only by interacting with the system, but also optionally, in the validation process (as validation nodes). The stakeholders can easily track their products and, as a result of that data, they will be able to provide proof of their product's certification, processes and traceability, resulting in the improvement of their methods. Furthermore, the end customer will be able to verify the quality of the products.

Keywords: Supply Chain Management and Tracking, Distributed Ledger Technologies, Food traceability

Resumo

Uma Cadeia de Abastecimento (SC) é uma rede que estabelece um fluxo de mercadorias e materiais entre fornecedores, clientes, transportadoras e todos os elementos relacionados até ao consumidor final. A gestão dessa rede, que é feita com o objetivo de maximizar o valor do consumidor e ganhar vantagens competitivas, é denominada de Supply Chain Management (SCM). Numa cadeia de abastecimento agrícola, para que essa gestão seja eficiente, existem aspectos importantes que precisam de ser abordados, como por exemplo a rastreabilidade dos produtos e seus custos associados quando as mercadorias transitam entre atores da rede logística. Para além disso, as opções atuais de rastreabilidade de alimentos não se extendem ao longo da cadeia e isso dificulta o processo de validar a qualidade do produto em qualquer ponto da rede.

Distributed Ledger é uma rede descentralizada e imutável que pode fornecer uma solução mais transparente e segura de gerir um negócio e a logística de uma cadeia de abastecimento. Depois de aprovadas, as transações são registadas em blocos e enviadas a todos os pares da rede. Utilizando esta abordagem, todos os pares da base de dados distribuída têm acesso a dados vitais, como as condições em que o produto é mantido, tornando toda a rede mais eficiente e confiável. Atualmente, a utilização de Distributed Ledger Technology (DLT) em SCM ainda se encontra numa fase inicial, mas existem múltiplos casos de sucesso da sua implementação no contexto referido.

Atualmente, é de extrema importância que se resolva o problema da rastreabilidade e da qualidade dos alimentos para que se consiga assim tornar o processo mais rápido, a cadeira logística mais eficiente (reduzindo custos) e, assim, aumentar a qualidade dos produtos.

Parte do trabalho tem como foco prototipar e estudar uma rede capaz de lidar com parte das transações de uma cadeia de abastecimento agrícola relativas ao processo de certificação de bens, de uma forma mais eficiente do que a solução atualmente utilizada pela indústria. A maior parte do trabalho concentra-se na avaliação do sistema DLT baseado na tecnologia Hyperledger Sawtooth aplicado à cadeia de abastecimento de alimentos agrícolas. Por outras palavras, o trabalho consistirá em analisar o design da rede e a sua escalabilidade como solução de persistência distribuída.

Com solução similares, todos os atores da rede serão beneficiados. As partes interessadas podem rastrear facilmente os seus produtos e, como resultado dos dados recolhidos, poderão analisar e melhorar os seus métodos. Além disso, o cliente final poderá verificar a qualidade dos produtos que são adquiridos.

Acknowledgements

I want to utilize this opportunity to acknowledge the people that had the biggest impact in not only the conclusion of this work but also in my personal development in the college period of my life.

Firstly, this dissertation could not have been done without the support of my supervisors, Alexandre Valle de Carvalho and Tiago Rocha. They provided guidance and the help I needed to perform this study. Furthermore, they were reliable and always available to mentor me.

Following, my family. They supported me and gave me everything I needed to reach this stage of my life and I am thankful for their love and support.

Lastly, I would like to thank the amazing friends I am fortunate to have. Namely, my dearest friend Inês, Pedro without forgetting the collective friend groups “Road”, “Dança com as Estrelas”, and “Bola de Basket”. There were all, in one way or another, always been there for me and I am extremely grateful for it.

João Malheiro

"We're not doomed. In the great grand scheme of things, we're just tiny specks that will one day be forgotten. So, it doesn't matter what we did in the past, or how we'll be remembered. The only thing that matters is right now, this moment, this one spectacular moment we are sharing."

Bojack Horseman

Contents

List of Figures

List of Tables

Abbreviations

DLT	Distributed Ledger Technology
SCM	Supply Chain Management
SC	Supply Chain ("Cadeia de Abastecimento")
API	Application Programming Interface
TP	Transaction Processor
VM	Virtual Machine
DAG	Directed Acyclic Graph
IoT	Internet of Things

Chapter 1

Introduction

This chapter presents the context of the dissertation and the motivation behind it, the approach, and expected results. It has a total of 5 sections. Section 1.1 aims to explain the context of the project. Section 1.2 is the motivation behind the work. Section 1.3 is the hypothesis around which the dissertation will be developed. Next, section 1.4 is the chosen approach to fix the problem presented in section 1.2 and the expected results of the implementation. Finally, Section 1.5 contains the document's structure.

1.1 Context

Supply Chain is a network that establishes a flow of goods and materials between suppliers, clients, transportation, until the end consumer. The management of a supply chain network aims to maximize customer value to improve brand image and gather a competitive advantage. Supply Chain Management (SCM) is vital for a company because it heavily influences the efficiency of a company's production cycle and, therefore, profit [26][23].

Companies use different solutions to help make the process of managing the supply chain more efficient. Most organizations used to rely on internal solutions to manage transactions and all the data it encompasses, but the globalization of supply chains due to the evolution of developing countries demands for an adaptation into a more global approach [35].

Another possible way to do it is through a Distributed Ledger Technology (DLT). DLT is a decentralized, immutable, and transparent network that can register and store data relevant to SCM. Currently, the use of Distributed Ledger Technology in Supply Chain Management is still in a pilot phase. However, there are multiple cases of success of its implementation in the referred context [22][25]. This work will analyse these case studies in section 2.3.

1.2 Motivation

Supply Chain (SC) networks providing agricultural products can be split into four simple phases as depicted in Figure 1.1. A SC is more efficient if they tackle critical aspects regarding traceability of their assets and transaction costs. These chains, with the advance in technology, are becoming more and more global [32]. However, the current food traceability options are not connected throughout the chain, and that makes it hard to validate the quality of the product at any point of the network [36], which goes directly against the increasing need for companies to establish the provenance of a product [1]. A study from 2017 showed that 69% out of 408 organizations do not have complete visibility into their respective supply chain [1]. This is due to companies only knowing their immediately close partners (supplier and client).

Furthermore, the fragmentation of the network opens the door for food security fraud which can cause harm to the remaining actors [36]. This was the case in 2017 when the United Kingdom's top chicken supplier was discovered to be tampering with product records in order for their customers to buy out-of-date chicken [2]. Worldwide, WHO (World Health Organization) states that each year, roughly one in ten people experience food poisoning and the death count reaches the values of 420000 per year [25].

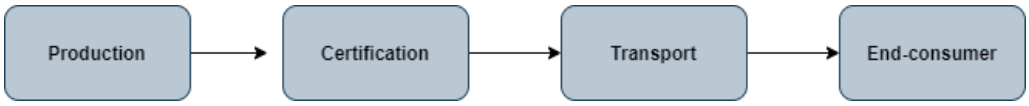

Figure 1.1: Agricultural Supply Chain Phases.

As it stands, it is of significant importance that the problem of food quality and traceability gets addressed to achieve lower transaction costs. This would make the process faster and the foot items more easily traceable, increasing the overall quality of the products.

1.3 Hypothesis

Regarding the current problem existing with Agricultural Supply Chains, it can be summarized in lack of traceability and security of product information and time it takes to exchange information between two participating entities. For that reason, a hypothesis regarding a DLT solution to this problem was developed for this study.

DLT network for Supply Chain Management of agricultural products can provide a trustworthy, secure, traceable, immutable, efficient, and affordable solution to every actor involved in the chain, thus lowering process costs and improving traceability across the logistics supply chain and, therefore, improving food quality.

[1] https://www2.deloitte.com/us/en/insights/industry/retail-distribution/food-labeling-laws-iot-blockchain.html Accessed on: January 2021

[2] TheGuardian - https://www.theguardian.com/business/2017/sep/28/uks-top-supplier-of-supermarket-chicken-fiddles-food-safety-dates Accessed on: January 2021

1.4 Approach and expected results

A Distributed Ledger is a peer-to-peer(P2P) decentralized and synchronized database. This type of network can provide a more transparent and safer way to manage a business and the logistics of a supply chain. After being approved, transactions between actors are registered in blocks and are then available to be read and analyzed. By utilizing this approach (as opposed to centralized databases, for instance), every peer in the distributed database would have access to vital data, such as the conditions in which the product is kept or transporting details, making the whole network more efficient and trustworthy. The stakeholders can easily track their products and, as a result of the collected data, analyze and improve their methods. Furthermore, the end customer will be able to access and verify the quality of the products.

Product certification carries many advantages for companies that adopt such a process: it reinforces customers' trust, improves the organization's brand, and opens up potential new markets for the certificated product [42]. There are multiple rules and stipulations when getting a certification of quality for a product. After inputting product data in the system, a certificate issuing entity would then verify and issue a certificate for the product. This certificate provides the company with an opportunity to include it in transactions of logistics and commercial operations.

After the prototype design and implementation, a study about the solution's network design, scalability, and reliability will be conducted. The underlying reason to proceed with the study is to analyze how a DLT solution of this type can handle a large number of transactions a global SCM would require.

The main objectives of this dissertation are to create, test, and validate a technological ecosystem of product certification in the agricultural goods domain. At the core of this work is a prototype to be developed supported on a DLT that can:

- Identify different actors and their roles in the system (producer, transportation, end-consumer);
- Support representation of product definition;
- Support the access of a product's history and information regarding who changed what information about that said product;

1.5 Document structure

Apart from this introduction, this document contains 5 chapters.

Chapter 2 introduces key concepts and ideas about DLT and its potential use in an AgriFood Supply Chain. It contains details of DLT, such as consensus algorithms and smart contracts. Furthermore, this chapter analyzes some real-world cases of DLT utilized in this study's domain. After, Chapter 3 explains the problem with AgriFood Supply Chains, and it contains implementation details of the developed prototype solution utilized for our experiments. Chapter 4 provides the experimentation phases and environment utilized in this study to run and collect results from

our prototype. Following, Chapter 5 discusses and analyses the experimentation results. Finally, chapter 6 provides conclusions about the study and also future work.

Chapter 2

State of the art

This chapter addresses state of the art regarding Distributed Ledger Technology and its application in an agriculture Supply Chain. Firstly, Section 2.1 discusses Distributed Ledger Technology and all of its components, properties, taxonomies, and implementations of DLT. Following, section 2.2 presents DLT by the domain of application in Supply Chain Management, asset production, goods transportation, and the end-consumer. Finally, section 2.3 contains and analyses implementations of DLT technologies in the agricultural supply chain context.

In the next section, and because of its relevance to this study, Distributed Ledger Technology is presented and discussed.

2.1 Distributed Ledger Technology

Historically, the term "ledger" refers to the storage of relevant information to economic activity. One example is the ownership and the relationships between individual actors or other different entities. In other words, a ledger is a collection of transactions [10]. As the economy became more global, the need for traceability, reliability, and trust increased, which can be achieved with decentralization, and so the concept of Distributed Ledger Technology was created.

Multiple definitions of DLT exist since it is a complex concept subject to much research throughout the most recent years. Benos et al. [9] states that a DLT is a "database architecture which enables the keeping and sharing of records in a distributed and decentralized way, while ensuring its integrity through the use of consensus-based validation protocols and cryptographic signatures". In other words, it is a database spread across multiple peers with the data being replicated securely. The properties inherent to a DLT that define this technology are explained throughout this section to better understand how a solution of this type can help an Agricultural Supply Chain. The difference between the centralized and the distributed approach is depicted in Figure 2.1.

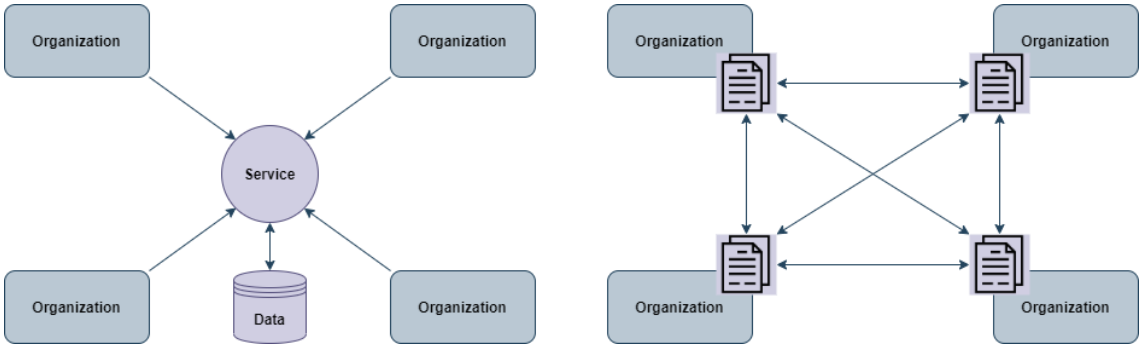

Figure 2.1: Centralized Ledger (left) and Distributed Ledger (right).

There are many distinct DLT platform types, each with its different purpose, characteristics, properties, and structure. This section analyzes each studied DLT platform regarding a set of properties and identifies the differences between these types.

This type of solution can bring significant benefits when it comes to its application in the domain of the agriculture supply chain. The ability to verify both backward and forward traceability of a product in the case of a food-related safety issue [36] is an example where DLT can be advantageous. It is important to note that every use case is different with Distributed Ledger Technology, so one has to study the pros and cons of the technology to better understand if it is applicable. Even when it is adequate, multiple properties, implementations, and system design options must be considered before development.

There are multiple cases where DLT is not the best solution to the problem due to the potential high cost to implement or high energy demand.

DLT approaches come with some notorious downsides. One of which is making system design decisions in this fast-changing technological landscape since by the time implementation is made,

the product use cases and details might not be what is required [7]. Another example is the scalability problem associated with DLT solutions 2.1.2.

2.1.1 Storing Transactions

This section explains the types of data structures that a DLT can have to store its transactions while also containing advantages and disadvantages to the different approaches.

Blockchain

There are multiple types of distributed approaches for building a ledger that will connect the transactions in the network. Firstly, there is the blockchain which presents a linked list of blocks that contain the transactions. This differs from a standard database since, in those databases, data is stored in tables, while in a blockchain implementation the information is aggregated into a block and stored after the previously committed block. Another very different aspect this technology presents is that when utilizing blockchain, the insertion order of the data matters. This is due to many blockchain's use cases being in the financial applications sector and managing money. That is not the case with a legacy centralized database.

Transactions can be split into two main components: body and header. The headers hold a hash of that block's predecessor [8] whereas the data is the information to be stored. The first block of the chain is named genesis and does not hold a predecessor block hash in its header. Besides the hash, the header contains a timestamp and the Merkle Tree root hash. The timestamp is essential to validate the block since it contains the time when the block was created. The Merkle value is a hash of all the transactions inside the block. When it comes to the other components in a block, the block body can be split into two parts: a transaction counter and the transactions.

A blockchain example structure is depicted in Figure 2.2.

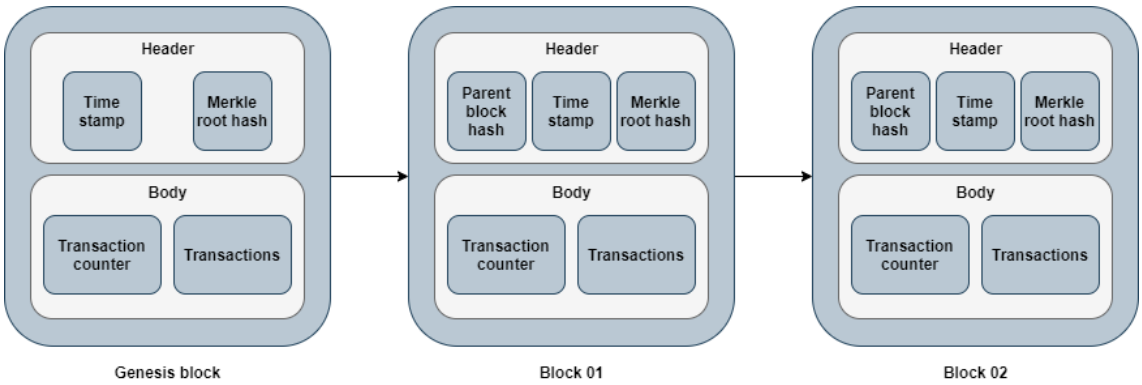

Figure 2.2: Blockchain structure.

As previously mentioned, transactions are stored in the body of a block in the form of a Merkle tree. A Merkle tree is a binary tree of hash values. The root of the tree, stored in the header, is the combination of all of its children's nodes hashes. This detail makes it so that any attempt to

tamper with any child node is noticeable by comparing the root hash with the combination of its branches and leaves, providing the chain with more security. Furthermore, this is a very efficient data structure since the verification of transactions can be done in logarithmic time by computing hashes in the path to the root of the tree [46]. This process is explained in Figure 2.3.

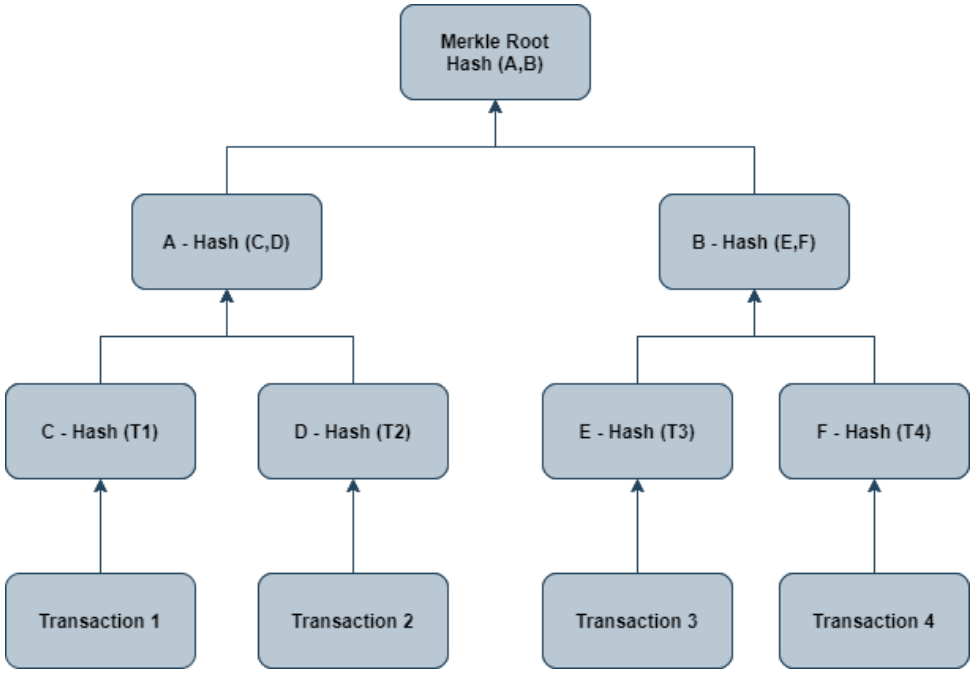

Figure 2.3: Example of a Merkle Tree.

Directed Acyclic Graph

In a Directed Acyclic Graph(DAG), data is stored in nodes instead of blocks where each node only possesses a single transaction. Nodes are linked through edges that represent each transaction validation.

This implementation is built on the belief that chaining the blocks with multiple transactions is a bottleneck for the architecture. Therefore, separating the transactions will battle this problem [20]. One example of a DAG network is Tangle developed by IOTA [37]. One of the main reasons for the creation of Tangle is IOTA's objective of eliminating transaction fees. In this network, a user can only add a transaction after picking and validating two different transactions. This characteristic contributes to an increase in the solution's scalability. Apart from that, there is a need to solve a cryptographic challenge to prove some work to defend the network against spam attacks.

In these systems, as more users connect to the network, the time it takes to validate a transaction decreases [37]. This can be a hurdle in adopting these solutions since the network's performance is directly tied to how large its user base is.

Hashgraph

Alternatively, there is the hashgraph solution. This solution uses a Directed Acyclic Graph, and it differentiates itself from the blockchain approach by not discarding any branches [6]. This technology gives peers in the network two possible actions: submit a transaction or gossip about one [20]. The gossip protocol utilizes gossiping generally found in people's interactions but applied to network nodes. New information spreads exponentially fast.

2.1.2 Properties

In this section, properties inherited by a DLT ledger that are relevant to this dissertation are presented. All of the following properties will help in establishing the hypothesis (Section 1.3).

Immutability

In a Distributed Ledger, the blocks of transactions cannot be altered, so the records are, therefore, permanent. Actors in these chains can only check the data and add more if that is their desire, making the network tamper-proof: for an attacker to alter the chain in any way, it would need to control at least 51% of the network's nodes. In an extensive network like Bitcoin, it is theoretically impossible to achieve such computational power during that span [39]. Although this technology is considered immutable by many, there are still valid security threats that can cause harm to a network of this design. For example, spam attacks and double-spending [31].

Immutability is an advantage of implementing with a DLT, but in some specific cases, it can also be seen as a disadvantage because it is good to, for example, be able to alter something in the chain due to a change in a company's business model or logic [36].

Decentralization

The data in the ledger is spread between all the peers in the network, making a DLT decentralized. The information is replicated and synchronized. Each node possesses a copy of the ledger, and after a consensus is reached to update the ledger, every node will make the changes necessary to update and synchronize with its peers. This is called the global state of the network.

An important concept to have in mind is that a DLT solution is not always fully decentralized. Decentralization is a spectrum.

An increase in decentralization typically comes with its trade-offs. This property decreases the user's dependency on centralized providers. As a result, we get an increase in their data protection against those same providers. However, the trade-off is caused by the inherent transparency of the data we get when utilizing a standard DLT solution. While their data is better protected against third parties, the network's transparency requires some metadata to be more visible to the entire network [17].

Security

These platforms are often responsible for handling large transactions or vital company data that is too important to lose. Many reasons contribute to DLT being considered secure. Firstly, the decision-making is not unilateral, which means that for a block of transactions to be validated and added to the ledger, most nodes have to authorize it. The next reason is the cryptography used. The network's overall security depends on the strength of the cryptography algorithms used for the generation of the authentication keys, encryption of the data itself in the blocks, and the consensus algorithm. Lastly, if the network is private, it ensures that only a select few nodes have permission to visualize the data, which can also increase security by restricting access. Despite all of these contributing factors, there are still some possible exploits and problems when it comes to DLT security, which are explained below:

- Majority attack - this exploit highly depends on the type of consensus algorithm used (subsection 2.1.5). However, it is defined by the malicious actor having control of the consensus in the network. This can lead to faulty transactions and malicious blocks. When a network grows in size, this is less and less probable to happen since it takes more computing power to acquire the control of the network [29];
- Double Spending - this security issue is typically connected with online payments. It occurs when a malicious actor duplicates a digital token and sends it to multiple parties. If digital information can be duplicated, this problem can arise [46];
- DDoS attack - denial of service is when the attacker makes the network service unavailable for its intended users. This is performed by feeding the system with a large number of redundant requests, stressing it with the amount of processing this exercise requires;
- Eclipse attack - this attack isolates the target from the remaining peers in the network by interfering with its connections. Following, the attacker has the power over what the victim can view (control over the entire network view) [28].

Transparency

While having to agree that an increase in transparency in, for example, a public DLT, when compared to a private one, can increase security problems in the ledger, the security features mentioned are capable of securing the network while it is fully transparent. Ghode et al. [18] states that "The challenge is building a Distributed Ledger which can achieve a balance between transparency and confidentiality of data". Records are auditable by a group of participant nodes. For example, in public DLT, every node holds equal rights and can access the ledger and even mine it (update). The records are therefore transparent in that situation. "Poor transparency through multiple supply chain steps can promote or conceal fraud" [36] is an example of what the lack of

transparency can cause in a DLT platform when applied to a Supply Chain Management context.

Scalability

Scalability is one of the main concerns one has to consider when defining the structure and implementation of a solution, mainly because of node data replication. Pearson et al. [36] states that given a DLT complexity, it is likely that blockchain solutions will first emerge in niche, controlled, or high-risk areas of the supply chain, where the technology may have the most significant impact. This is due to its relatively low performance when accounting for an extensive global network. For example, Visa needs more than 4000 transactions per second which are very distant from what the current global-scale DLT implementations offer [47]. Given the issue at hand, this type of system should first be implemented on a small scale. After that, and when the research develops more on the subject, one can think about implementing this concept on a global level.

There are several metrics with a large effect on a DLT solution's scalability:

- Block size - while increasing a block size can theoretically increase the network transaction throughput, it also causes a block propagation delay in the system which can in turn have large performance deficits in the system [47];
- Block time - altering the time it takes for a blockchain network to propose a new block (depending on the consensus used) can alter the network scalability;
- Transaction validation time - another metric with a huge impact on how a solution performs is the time it takes for a transaction to be validated. While this largely depends on the consensus algorithm chosen (see 2.1.5), it presents a considerable scalability problem if it depends on a vast number of nodes approving those transactions [14]. This does not present such a problem to scalability in solutions with designated validating nodes since an increase in network size does not necessarily mean that transactions take more time to process and validate. A possible approach to decrease the transaction time in a system can be Sharding. This method separates the nodes into shards, and it shard processes a different set of transactions. It is a way of parallel executing the transactions [44].

2.1.3 Permission access

DLTs vary regarding the definition of permission access to block transactions or network mining capabilities. Therefore, a DLT can be either public or private.

Public

Public ledgers, also known as permissionless ledgers, are made of computer nodes that can join the network without permission. These nodes are called miners. Permissionless ledgers allow all the peers to mine the network. The state of the network and its blocks are visible and accessible

to everyone in this type of DLT. One of the disadvantages of transparency between every node is, for example, not being able to hide some data that a company might not want to share with everyone [45].

Private

Private ledgers, or permissioned, are the opposite of the public DLT and possess, therefore, restricted access. Only previously authorized entities are allowed to participate in events in the ledger. This ensures the privacy of the data stored in the network. While the privacy of the data stored might be beneficial for some, it could also raise trust issues between actors in the system.

An overview of the trade-off between public and private DLT solutions is presented in Table 2.1.

Public DLT	Private DLT
Any user can participate	Participants are validated and authorized
Anonymous	Participants know each other
Lower throughput	Higher throughput
Higher level of decentralization	Lower level of decentralization
Higher energy consumption	Lower energy consumption

Table 2.1: Overview of public and private DLT solutions.

2.1.4 Smart Contracts

A smart contract is defined as a computerized transaction protocol that executes the terms of a contract [40]. This concept was introduced by Nick Szabo in 1994, who suggested that contract clauses should be translated into code. To simplify, smart contracts are the logic behind all the transactions, either inside a company or between different organizations. One of the main aspects of smart contracts is that it computerizes a process that would have to be done by a person. Therefore, this feature can save money resources in the companies that adopted these types of systems. In other words, in present time distributed systems, a smart contract validates transactions while using programmed logic and changing the ledger's state.

Some platforms allow for the implementation of code, for example, in Java or Go, to increase the complexity of a contract, so it feeds the needs of the platform [16].

Even though smart contracts have evolved in time and are now a pillar of blockchain technology, some security issues have emerged from hacks that exploited flaws in smart contracts logic. An example of this is the 2016 DAO attack on Ethereum smart contracts [4]. The attacker took advantage of the ether being paid out of the contract before the credit value was updated. Therefore, by taking advantage of a callback, he was able to utilize recursion and steal ether. This could be prevented by only allowing the withdrawal after the credit value of the contract was updated,

not allowing the reentrancy attack to happen. A mock example of this attack can be observed in Figure 2.4.

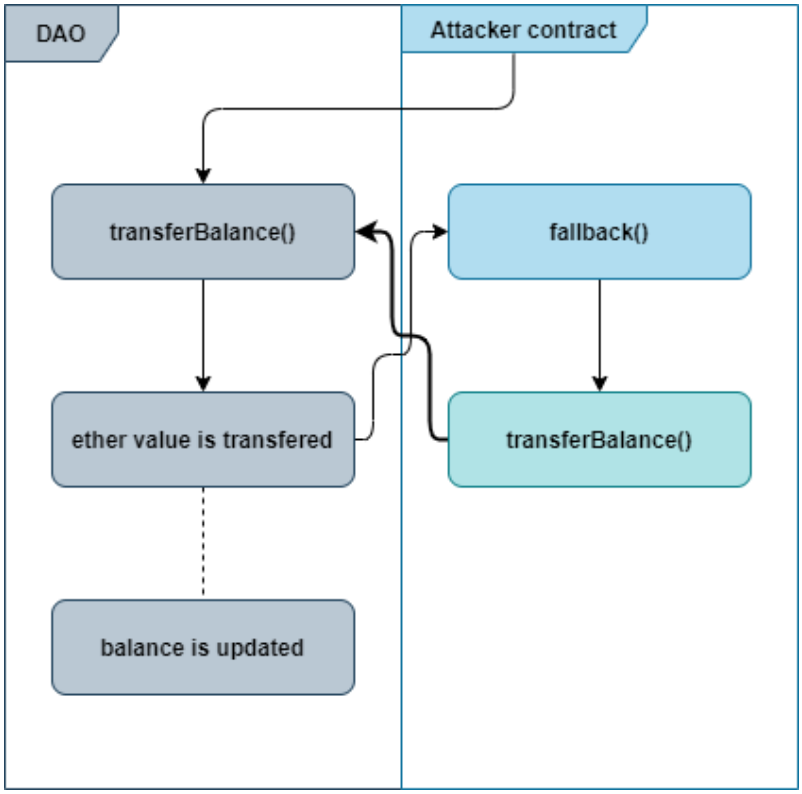

Figure 2.4: 2016 DAO reentrancy attack on an Ethereum smart contract.

2.1.5 Consensus Algorithms

One of the essential components in a DLT is its consensus algorithm. A consensus is achieved when a certain number of peer nodes agree and approve a transaction. This feature helps protect the system against attacks.

For consensus to be reached even in the face of malicious nodes, there are certain aspects a consensus algorithm needs to address and trade-offs to be analyzed. A consensus algorithm has to contain two properties: liveness and persistence. Persistence is what ensures a consistent response from the nodes in the network regarding a transaction, and Liveness expresses that, after some time, the consensus in the system is always reached [46].

There are multiple types of consensus algorithms:

- Proof-of-Work (PoW) - algorithm where each user (miner) has to solve a computationally difficult problem to ensure the validity of new transactions. After solving the problem, the peer receives a token to prove his work. This helps reduce spam attacks on the system. One of the problems PoW faces is that because it is used in a public DLT it is impossible to delete a malicious node from the system. Furthermore, the computational power it takes to

solve the puzzle requires much energy in large networks, and that power is used only to get tokens without presenting a valuable output. As a result of the high investment needed to run this network, centralization of mining power in the long term can occur [41].

- Proof-of-Stake (PoS) - in this algorithm, a peer can validate or mine a new block according to how much buying power (ether) he holds. There is not a need to solve a problem to be able to mine like in Blockchain. This algorithm was created to solve the underlying issue of PoW, which is energy consumption.
- Proof of Burn (PoB) - follows the principle of "burning" the coins/tokens held by the miners that grant them mining rights once they have been used. This algorithm aims to stop the double spending of tokens, one of DLT's most dangerous attacks.
- Proof of Elapsed Time (PoET) - Intel developed this algorithm for their Hyperledger project, and it enables its permissioned networks to elect leaders (who proposes the next block). PoET algorithm generates a random wait time for each node, and each node has to sleep for the duration of that random time. The first peer to wake up from their sleep wins the mining rights of the new block. The setback with this algorithm is the need for fault-proof software to generate the times randomly.
- Practical Byzantine Fault Tolerance (PBFT) - designed to be able to handle byzantine faults. A three-phase process requires a node to receive votes from 2/3 of the other existing nodes on the network. This algorithm helps in preventing spoofing and to detect altered messages sent from malicious nodes [12].

A concept existing in DLT implementations related to consensus is the fork. A fork occurs when, for some reason, nodes do not reach a consensus in the expected time. Normally, this is due to upgraded versions on some nodes and older versions on others [29]. There are two types of forks:

- Hard Fork - older nodes and newer ones cannot coexist on the same chain with their differences in protocol, so a hard fork occurs in which two chains are being built;
- Soft Fork - in this case, older protocol nodes can still interact with the chain as long as they respect the new protocol rules implemented in the newer software version. Both new and old nodes can still work on the same chain and. Eventually, after some amount of time, the old nodes upgrade.

2.1.6 Oracles

A DLT oracle is any device or entity that connects a DLT with off-chain data (data that is not in the network, for example, an external API to check the weather conditions in which a product is kept). In other words, an oracle is a smart contract that serves real-world data at the request of another smart contract in the chain so that it can execute the agreement [3]. This has to be

implemented in the network with care because it can centralize it [3]. For example, if an API is called to check a product's price a moment later after another peer does the same, the value can be different, invalidating the transaction in that case. A representation of this service is depicted in Figure 2.5.

To battle the issue at hand, one can use a decentralized oracle network like Chainlink [1]. This technology possesses a consensus system close to that of blockchain. By using this platform, we can integrate the DLT with real-world data in a safer way that helps fight malfunctions" [11].

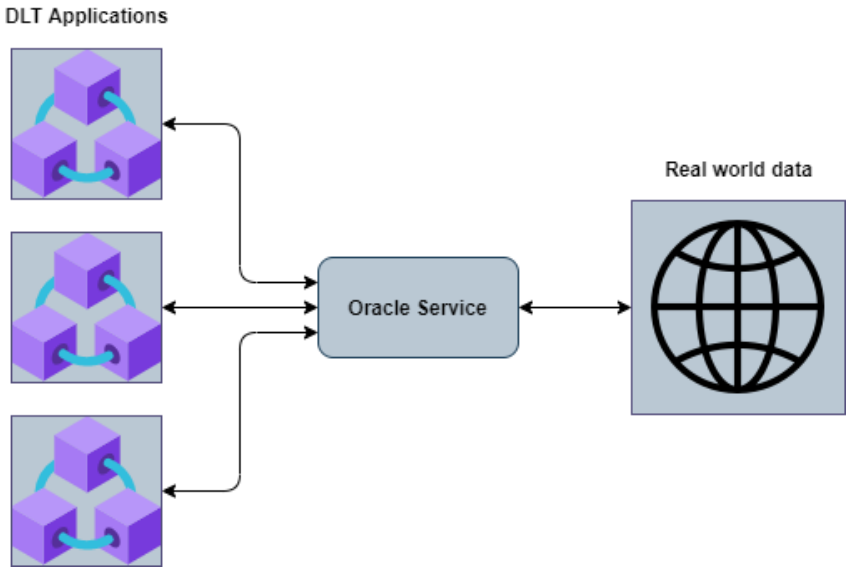

Figure 2.5: Oracle service representation.

2.1.7 Implementations

The following sections contain DLT implementations that are relevant to this study. Furthermore, the differences and similarities between them are analysed to find the best approach when applied to a supply chain in an agricultural domain. This information is all summarized in Table 2.2.

Bitcoin

Bitcoin is the most well-known blockchain implementation in the world. It was invented by Satoshi Nakamoto in 2008 [33]. This is a decentralized, distributed, and public network. Because of its public nature, everyone in the world can join and interact with the network. This technology uses a proof-of-work algorithm where each user (miner) has to solve a computationally difficult problem to ensure the validity of new transactions. This mining, which is, in other words, a sort of race between every participating node in the network, provides users with a monetary reward for the energy consumption that was spent solving the problem. However, it creates a considerable

[1] Chainlink - https://chain.link/ Accessed on: January 2021

world energy problem.

Ethereum

Ethereum is an open-source DLT technology that features smart contracts through its virtual machine(EVM), allowing developers to develop blockchain applications around it. These contracts are immutable and are used as functions. Ether is the cryptocurrency that is the base of the Ethereum network, and it is the currency utilized to pay transactions fees, which go to the network's miners. Each account has a balance of Ether and can transfer ETH to other accounts of its choice.

Hyperledger Fabric

Hyperledger Fabric is a private DLT system. Linux Foundation created it, and it presents a flexible design. Its most differentiable feature is being able to contain multiple ledgers within its network as opposed to other implementations with only one ledger [15]. Another advantage is its versatility. One can plug different consensus algorithms or encryptions which increases the use cases in which the platform can be used.

Hyperledger Sawtooth

Sawtooth is a software framework that can suit a different number of use cases. As Hyperledger Fabric, Sawtooth also allows choosing different consensus algorithms, but, on the other hand, it differentiates from Fabric on the permission access. Sawtooth allows choosing between both permissioned or permissionless network implementation. This implementation is highly modular and it contains multiples SDKs in different languages such as Python, Javascript, Go, etc.

Hyperledger Burrow

This DLT implementation can be seen as a private deployment of Ethereum, which means that only authenticated peers can execute code [15]. This platform utilizes Byzantine fault-tolerant Tendermint [2], which works with authorized validator nodes. One of the disadvantages of this implementation is that the network can fail if more than one-third of the validators are malicious or down [38].

[2] Tendermint - https://tendermint.com/ Accessed on: June 2021

R3 Corda

R3 Corda [3] is a public but permissioned implementation of the blockchain logic built for supporting financial services. Corda presents extreme flexibility when it comes to its consensus. It allows its notary nodes to pick what type of consensus to utilize. Since its consensus relies only on specific nodes validating the transactions, the performance is superior when comparing to a network using, for example, a PoW algorithm. A differentiating factor of this solution is the evolution Corda provides when it comes to smart contracts. In this network, the smart contracts contain not only code but also legal prone (logic to support legal regulations) [43].

Table 2.2: Overview of DLT Implementations utilizing Blockchain.

Implementation	Permission Access	Consensus
Bitcoin	Public	PoW
Ethereum	Public	PoW
Fabric	Private	PBFT
Sawtooth	Public or Private	PBFT, PoET, and Raft
Burrow	Optimized for Public but can be Private	Tendermint
R3 Corda	Private	Pluggable

2.2 DLT for Agricultural Supply Chain Management domain

This section presents the problems in the current supply chain and the benefits of applying DLT in an agricultural supply chain. Supply chains have some inherited problems like traceability of food items, dispute resolution, digitization of the data and DLT possesses the capabilities of presenting a solution to these issues [13]. Making information safe and available to every actor in the chain will improve efficiency(making the network more Farm-to-Fork) and allow for better cooperation between the organizations. By using this technology, the immutability property can flag any adulteration of data by a hostile peer [36].

Despite those advantages, the use of DLT in SCM still presents some disadvantages. Chang et al. [13] state the top challenge for blockchain implementation is the ROI (return on investment), especially in an agricultural context compared to, for example, big technological companies. This is due to the costs compared to the monetary advantages of implementing a DLT in an agricultural SC. Another issue could be standardizing the data. Although there have been efforts deployed to developing standards so that there are no conflicts between networks, this is still a significant problem in this context [34]. Finally, the decentralization nature of the network allows for peers to be scattered around the globe. Because they are not in the same countries, different laws and regulations might apply, complicating the development of the logic in the ledger's smart contracts.

[3] Corda - https://www.r3.com/ Accessed on: June 2021

Jabbari et al. [21] states that one of the biggest problems of DLT for a food supply chain is the fact that most of the technology used to keep track and provenance of the food along the chain (RFID, temperature and humidity sensors, etc.) can be easily replaced or tampered with which defeats the whole purpose of the implementation.

For farmers, the most influential factor in adopting DLT is the cost. Keeping the cost low and the application simple is vital when it comes to these actors. A distributed ledger solution lets farmers have forward traceability, which can improve a farmer's knowledge of the network, improving the actor's ability to plan its harvesting and delivery schedule [27]. Furthermore, a farmer can grade its crops before delivery to decrease product rejections further along in the chain.

Currently, transporting a product between continents takes cooperation between multiple entities and even more paperwork data. The digitization that would be inherited by applying DLT to the logistics part of the supply chain can make this process digital and, therefore, faster.

When it comes to the end-consumer, a DLT solution allows a customer to check the full provenance of a product, including the conditions in which it was kept. In other words, a person would be able to authenticate and verify the quality of a food item. This would increase customer satisfaction while giving an organization an advantage over its competitors [24].

2.3 Implementation in Agricultural Supply Chains

This section presents relevant implementations of the dissertation. Firstly, subsection 2.3.1 is used to analyse the implementation of DLT by Walmart in cooperation with IBM to transport perishable goods(mango, pork). Subsection 2.3.2 discusses Tradelens, by IBM and Maersk, which is a global blockchain developed by the two companies. It also presents Provenance along with its pilot project. Furthermore, OpenSC attempt to improve the fish supply chain in case is analysed and then, finally, Decapolis, which is a Jordan based startup providing knowledge to developing countries farmers, is presented. All of the analysis on the featured case studies attempt to approach why that technological solution was implemented and differentiating factors such as IoT devices used, for example.

2.3.1 Walmart and IBM

Walmart, which is a retail organization that operates grocery stores and hypermarkets, partnered with IBM to implement a blockchain system to improve food item's traceability. The objectives of the company were to increase traceability, transparency and increase food safety. This technology was applied to both Mangoes from the Americas, and pork from China [25].

The platform was developed using Hyperledger Fabric and was centered around allowing actors in the entire supply chain to access information about a product's provenance. In this implementation, there are, both for the mangoes and the pork, sensors and IoT devices integrated with the chain that help the network gather even more information. An application was developed that allows every user to both insert data into the network regarding the conditions in which the product

is kept (temperature, pesticides that were used in its development) and search for the origin of a product in case of, for example, rotten items [5].

Using this DLT, Walmart Vice-President of Food Safety Frank Yiannas stated that it was able to reduce the time it takes to scan a mango's origins from seven days to only 2.2 seconds [25].

IBM then followed this project by creating IBM Food Trust [4] to expand their technology to other organizations and to help increase food security and traceability even further.

2.3.2 Maersk and IBM - Tradelens

Maersk is the biggest shipping organization in the world. The company discovered that for a single product to be shipped from Kenya to Europe took "200 separate communications and interactions among nearly 30 organizations" [13]. Additionally, it is estimated that 15% of transportation costs in the global market are due to time spent handling paperwork [19]. To try and resolve this issue, Maersk partnered up with IBM to develop a blockchain solution that would allow for faster shipping and tracking. Ibrahim Gokcen, CDO at Maersk stated that this project would reduce products cost for the end-consumers but also improve the accessibility to the supply chain for actors from developed and emerging countries [19].

This is a platform where actors can share transaction data transparently and safely. It uses smart contracts to manage inter-organizational business processes. TradeLens can be split into three different components: Network, Platform and Applications/Services. Like the solution in the previous section, IoT devices were used in the implementation. This implementation of IBM and Maersk is a permissioned blockchain where all participants know each other and are previously authorized to participate.

As for the results, Maersk estimates that TradeLens was able to reduce the time of a shipment of materials to the U.S. by 40% [5].

2.3.3 Provenance

Provenance [6], a London-based startup, started in 2013 with a desire to channel the power of blockchain for social good by using it to impact food supply chains. This enterprise provides companies with a way to implement blockchain solutions in their respective domains in order to increase its transparency, in a business or product level. The company's CEO stated that Provenance is driven because people were "very concerned with how this data is presented to consumers, and not just end consumers but consumers along the chain". In other words, the problem presents itself not only in hiding or altering information from the end-consumer, but also between cooperating elements in a product's supply chain.

[4] FoodTrust - https://www.ibm.com/blockchain/solutions/food-trust Accessed on: January 2021
[5] https://newsroom.ibm.com/2018-08-09-Maersk-and-IBM-Introduce-TradeLens-Blockchain-Shipping-Solution Accessed on: February 2021
[6] Provenance - https://www.provenance.org/ Accessed on: May 2021

The company pilot project saw them giving a digital token to every caught fish by some small-scale fisherman organization in Indonesia. They provide the worker with a phone with its pilot application in it and, every time the actor catches a fish, he registers it into the mobile application. After the fish goes through the entire supply chain, the end-consumer could then check, using its digital token, the entire history of the fish from the day it was caught to the transportation details.

Provenance's mission is to involve more actors and small companies from developing countries in the current technology landscape. However, the company's CEO warns that despite blockchain providing trust and transparency, the industry is dominated by financial services and greedy venture capitalists [7] which could stop its evolution in this context.

Since its pilot project, the organization is building on Ethereum and developed some projects worth mentioning. One of which is its cooperation with Princes Tune. Princes is a significant European food conglomerate and is currently utilizing blockchain technology to fuel its "Check je Vis" application. This work allows people to check provenance of the tuna they are buying [2].

2.3.4 OpenSC

Like Provenance, OpenSC [8] is a company proving an increase in food supply chains transparency through the use of blockchain solutions. The company helped Austral Fisheries track its Chilean sea bass products throughout its entire global supply chain. This was done in order to help combat the amount of illegally caught fish in the world.

The difference between this project and Provenance pilot is that OpenSC integrates the blockchain with multiple IoT (Internet of Things) devices to help collect data that enriches the available information in the network. On top of that, OpenSC also utilizes a machine learning algorithm developed by the company to ensure the fish was caught in legal areas. After being caught, each fish is tagged with an RFID and the GPS information of the ship it was caught by which allows Austral to then share that data with its direct consumers. All of the collected data was made available in a dashboard so that Austral could use it to further improve its operation and logistics [9].

2.3.5 Decapolis

Decapolis [10] is a Jordan based company also utilizing blockchain for slightly different reasons. They encountered the problem that small and uneducated producers with their lower quality crops could not compete with the healthier crops the big organizations were producing. They pointed out bad water supply and lousy food safety general knowledge as part of the problem.

In cooperation with the World Food Programme, they launched a pilot private blockchain solution to trace the products from the farm until its last phase. Decapolis also taught the struggling small farmers better agricultural practices such as using pesticides and how to test the water and

[7] https://www.coindesk.com/provenance-channeling-blockchain-social-good Accessed on: June 2021
[8] OpenSC - https://opensc.org/ Accessed on: May 2021
[9] Austral - https://opensc.org/case-studies.html Accessed on: May 2021
[10] Decapolis - https://www.decapolis.io/ Accessed on: June 2021

the soil for detrimental agents that may be in it. To conclude, this platform allows the smaller farm holders to produce better quality products which will increase their income while putting better quality products on the market for all of the area's population [11].

An overview containing all these examples of implementations is presented in Table 2.3.

Table 2.3: Overview of Real World Implementations utilizing Blockchain in Agricultural SC.

Case	Primary Reason for the Project	Technology
Walmart and IBM	Traceability, Transparency, and increase food safety	Fabric
Tradelens	Reduce shipping time and Product Cost	Fabric
Provenance	Food Data Fraud, Companies from Small Countries	Ethereum
OpenSC	Food Transparency	Ethereum
Decapolis	Help small farmers and organizations	Private

2.4 Chapter Summary

In this chapter, we began by analysing the evolution of ledgers and, specifically, Distributed Ledger Technology was presented. We explored DLT definitions, and its most utilized types were explained (blockchain, hashgraph and DAG). Following, we summarized inherent properties to a DLT system, detailing some of them, such as security or scalability. The most important design features of a DLT implementation are its consensus mechanisms and smart contracts, explained in detail. Furthermore, different types of DLT implementations were listed. To provide context, the chapter's second section contains an analysis of DLT solutions for the Agricultural SCM domain, including advantages for all the multiple actors along a supply chain and bottlenecks for its adoption in such domain. To conclude the chapter, real-world examples of DLT solutions to agricultural problems were presented.

[11] https://innovation.wfp.org/project/decapolis Accessed on: June 2021

Chapter 3

Implementation

This chapter presents the problem currently found in Agricultural SC while addressing the implementation design and structure of the DLT solution chose to help perform this study. First, section 3.1 identifies the current problems in current supply chains in section. Following, section 3.2 presents the research questions that this study will try to fully address and provide a complete answer to. Section 3.3 discusses the possible solutions that were investigated in order to provide a solution to the problem. Finally, section 3.4 contains implementation details and the overall structure of the chosen solution.

3.1 Problem definition

Supply Chain Management aims to maximize customer value and gain an advantage when comparing the business to its competitors. This can be done by improving the flow of either raw or finished products in the network.

Currently, in agricultural supply chains, this concept presents some flaws. Food is a very fragile good which can be subject to a multitude of possible malicious acts performed by those who wish to gain something from it. Providing a secure way to store, transfer and visualize data regarding food items is a key to surpass the obstacles that are currently present in Agricultural Supply Chains.

In this context and in order to maximize customer value, one of the main factors one can optimize is the time it takes for actors to execute transactions between each other and along the chain. By lowering this time, we speed up the process, reducing the time it takes for a product to go from the production phase to the end-consumer. This has a direct influence on all the actors involved by delivering better and fresher products. Furthermore, another important detail to consider is the high usage by different users. In a centralized solution, if the system cannot deal with the high usage of transactions at a certain point, the chain can be disrupted. Consequently, agricultural products can lose quality or even go bad.

One of the most essential segments of a complete supply chain is its certification phase. This part of the SC (Supply Chain) was chosen for this study because it usually contains a lot of back-and-forth communication between the producer and the certifying entity. Commonly, this is a process that relies on communication either via e-mail, phone or a centralized solution that does not provide sufficient security and trust levels since those methods can be easily tampered with. For those reasons, studying the performance of a decentralized solution to this problem is even more relevant.

3.1.1 Certification process

Meuwissen et al. [30] states that "certification is the (voluntary) assessment and approval by a (accredited) party on a (accredited) standard". In other words, and the agricultural domain, it can be said that the certification phase is the process where a certifying entity provides an agricultural producer with an official document attesting to the origin, quality, and safety of a specific product.

The first step to a certification process of agricultural goods is when a producer registers agricultural goods on a system with its properties (name, quantity, etc.). Subsequently, a request for the certification of such product is made to a certifying entity.

Three scenarios were found that describe the most standard practices in which a certification process can occur in an actual situation within an agricultural domain. The most basic one consists of having the responsible certifying agent validate the product's properties and then issue the certificate that will later be used to prove the product's quality and integrity. This action is shown in Figure 3.1.

The second scenario requires the certifying entity to perform an inspection on the producer's facilities, and product [1] [2]. Furthermore, they will submit a report explaining that the producing operation complies with certifying industry standards.

The third scenario involves the certification report indicating some failures in the operation, which will involve an extra step of communication from the producing company telling the certifying organization that the changes to fulfill the requirements were made. Test scenarios 2 and 3 are represented in Figure 3.2 in green and purple, respectively.

The way these three different processes were replicated in the blockchain is detailed in section 3.4.8.

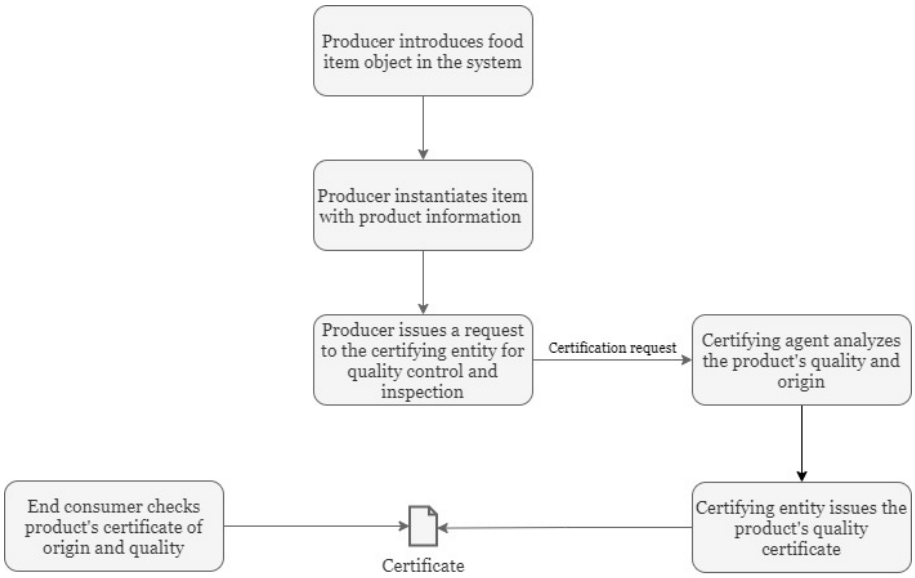

Figure 3.1: Test scenario 1.

3.2 Research Questions

We aim to tackle the problem found in Agricultural Supply Chains with a Distributed Ledger approach that can help solve many of the issues encountered. This dissertation strives to find if a DLT solution is well fitted and capable of efficiently dealing with such a supply chain. This is achieved by testing the network with different system designs and block characteristics. On top of that, it raises the hypothesis question:

DLT network for Supply Chain Management of agricultural products provides a trustworthy, secure, traceable, immutable, and affordable solution to every actor involved in the chain while lowering transaction costs and improving food quality.

For a decentralized solution to be adequate in the AgriFood Supply Chain domain, it needs to help fix the problems inherent with utilizing a legacy centralized ledger and present an efficient

[1] https://www.usda.gov/media/blog/2012/10/10/organic-101-five-steps-organic-certification Accessed on: March 2021

[2] https://www.qcsinfo.org/food-safety-gaps/ Accessed on: March 2021

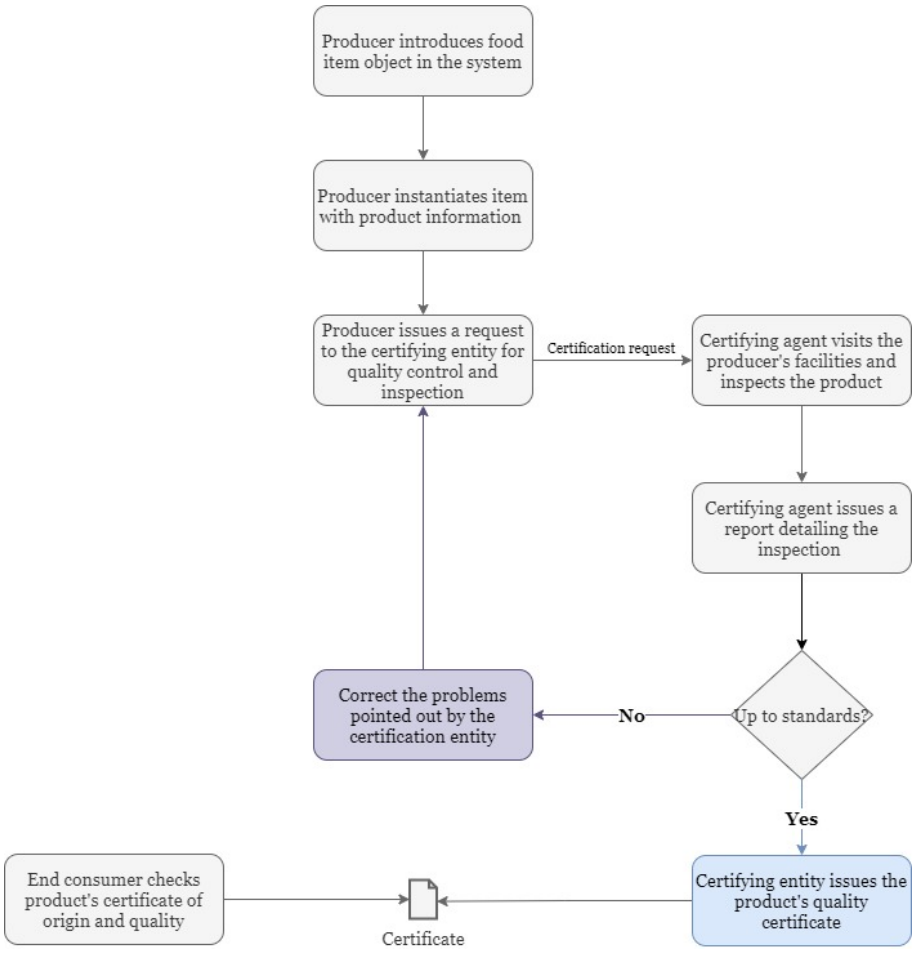

Figure 3.2: Test scenario 2 and 3.

way to operate it. There is a couple of important system design choices and attributes that can help study a solution's performance: the number of the nodes in the network and batching transactions together.

Additionally, we find that answering these research questions can provide extra information when concluding if the solution we propose is scalable and reliable:

- RQ01 - How does block size bottleneck the network?
- RQ02 - Does putting transactions into batches (sets) of transactions help the network performance?
- RQ03 - What effects does the network size (number of nodes) have on the system performance?

3.3 Hyperledger Sawtooth

Several blockchain implementation options were analysed before a decision on the underlying technology was performed. Ethereum, for example, is the most used blockchain technology in

the world. However, its transaction costs (ether) are significant and not the best approach in the context of agricultural goods.

Hyperledger Fabric was another analysed technology. Although it has some significant advantages like, for example, being able to build multiple layers of chains (multiple supply chains in the same network), the fact that it only supports the implementation of permissioned networks was a deterring factor.

Hyperledger Grid [3] is a promising project that originated from the Sawtooth supply chain dedicated project but was not chosen for this dissertation because it is still in the incubation phase.

Hyperledger Sawtooth was the underlying blockchain implementation chosen to perform this study in the SCM domain, specifically in the certification phase. This technology provides extreme flexibility not only by having different methods to achieve consensus (see Section 2.1.5) but also because of its modular design. Furthermore, this software allows for parallel execution of transactions that can consequently increase its transaction rate capability.

In order to test the blockchain implementation and answer this study's research questions, two different experiments will be run using different Sawtooth configurations and network sizes. Following the experimentation, metrics will be extracted out of these tests, which will help validate the hypothesis. Further detailed explanation on Chapter 4.

3.4 SmartAgriChain Technical Implementation

This section explains the specific implementation of the Sawtooth underlying blockchain technology. This prototype is designed for the agricultural supply chain domain and is utilized to perform a study in order to validate this work's hypothesis.

3.4.1 Structure

SmartAgriChain is divided in 5 components as detailed in Figure 3.3:

- Frontend Client: frontend built using ReactJS [4] that serves to interact with the blockchain by submitting requests to the Rest API via proxy (Server);
- Rest API: used to connect the client to the nodes;
- Server: built using Express [5]. It is utilized as a proxy to serve requests from the browser to the Rest API;
- Node: contains the main logic of the service in its Transaction Processors (TP), particularly the SAC Transaction Processor. The TP is responsible for executing and handling the payload of a transaction. More details can be found at section 3.4.5;

[3] Grid - https://www.hyperledger.org/use/grid Accessed on: May 2021
[4] React - https://reactjs.org/ Accessed on: June 2021
[5] Express - https://expressjs.com/ Accessed on: June 2021

- Kotlin client: extra application used to feed requests to the network to get performance metrics analysed later in chapter 5.

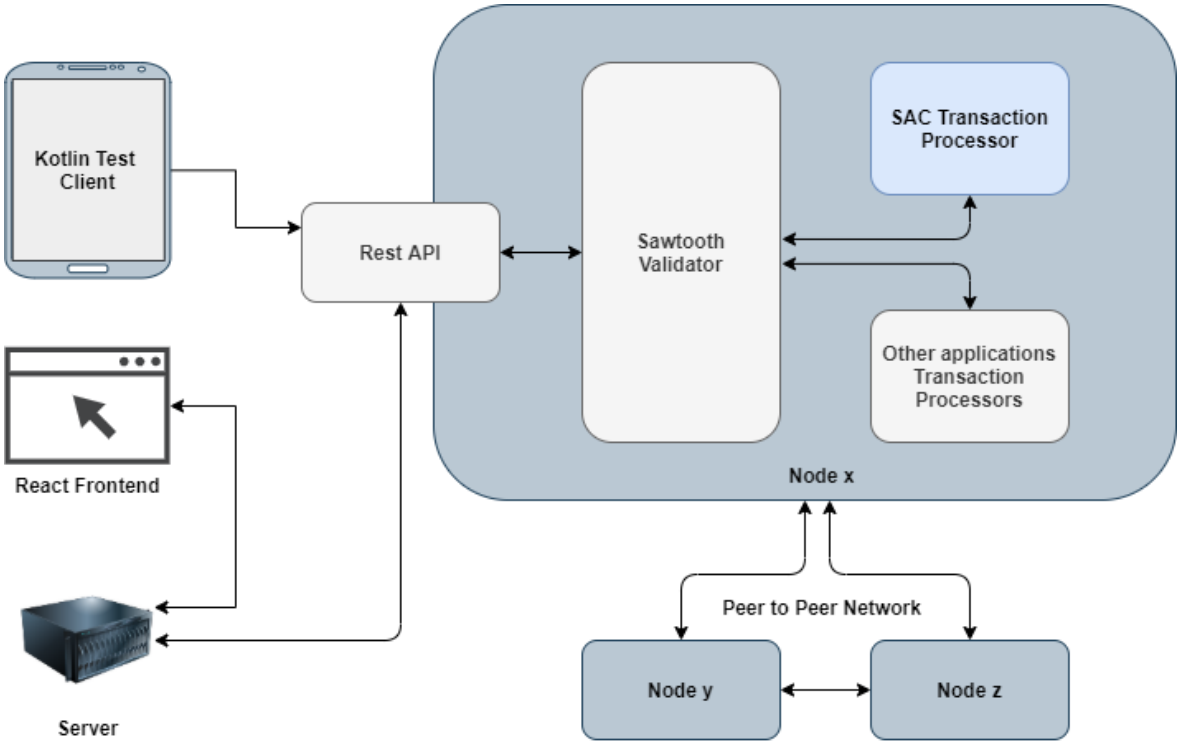

Figure 3.3: SmartAgriChain structure specification.

3.4.2 Sawtooth Validator

The Sawtooth Validator is one of the main components in the Sawtooth network design. It has a multitude of responsibilities that are critical to the system. First, the validator oversees all the connections between the nodes (this is what allows the nodes to establish communication). Furthermore, it must communicate with the other nodes to reach a consensus and maintain the global state. The global state is another name for the set of chained blocks every node contains. By maintaining consensus, every node contains the same state which is of significant importance to maintain the network's immutability. Finally, this component receives transaction requests and sends them to the correct transaction processor for further validation (section 3.4.5).

There are two ways Sawtooth can reject transactions. One of which is when the validator is full of transactions to process and its queue has also reached full capacity. The other is called back pressure test and it helps prevent Denial of Service attacks. This attacks are executed in order to stop a system from being accessible to its users. Following the system being overwhelmed it will come back to its normal state of processing transactions.

3.4.3 Sawtooth REST API

The Sawtooth REST API is an interface used by a client to communicate with the validator component mentioned in section 3.4.2. This communication is done through HTTP requests and allows to submit transactions or query blocks (the API treats the validator as a black box). Furthermore, the developed React Client has to establish the communication to the Rest API by using a server as a proxy because the Sawtooth component does not support CORS (cross-origin resource sharing).

Another Sawtooth feature is the batching of transactions. Blockchain technology allows for the transactions to be put and committed together. It is important to note that they are submitted according to their order inside the batch, so if they contain dependencies, it is essential to be careful when setting the batch request. We will test the effect batching has on the system's performance in the experiments of this study.

3.4.4 Collections

In order to prove the hypothesis and to study the performance of a DLT implementation in a certification process context, a series of data collections were defined to store in the system and be used in the experiments.

This implementation of the Sawtooth blockchain holds four different collections, explained below, and its fields are specified in Table 3.1:

- Organization: collective entity with associated details. Each organization object can have multiple associated agents via its address (details explained in section 3.4.6);
- Agent: a single entity that can be part of an Organization and perform actions in the chain. An example is, for instance, creating or altering either a product or a certification process.
- Product: goods or a set of goods (batch). One of the most important aspects of a product is its ownership. A product can have an owner (Agent). It also allows for a vast range of types of products (simple or created using multiple simple components, etc.) since it contains optional fields that allow customization;
- Certification: current state and some core data of a certification process and the steps that make such process.

3.4.5 Transaction Processor

Like Ethereum (section 2.1.7) has the concept of smart contracts, Hyperledger Sawtooth contains its Transaction processor (TP) to validate the business logic in the request's data. In other words, this is the component responsible for the execution of a transaction in the network.

Sawtooth possesses multiple built-in transaction processors like the settings TP or the identity TP which are used to add or change in-chain network policies. This is a way of, for example, altering consensus settings while keeping the network operational.

Table 3.1: Collections in the Hyperledger Sawtooth blockchain.

Agent	Organization
docType: String name: String active: boolean roles: List<Role> publicKey: ByteArray	docType: String name: String address: String ID: String
Product	**Certification**
docType: String name: String custom: Map<String, Any> onwerPKey: ByteArray producerPKey: ByteArray productionDate: Long expirationDate: Long quantity: Int components: List<String> description: String	docType: String ID: String productID: String certificationType: String timestamp: Long currentStep: Int numberOfSteps: Int assigneePKey: ByteArray certificationPKey: ByteArray producerPKey: ByteArray certificationStatus: Enum

The custom transaction processor for this network was developed using Javascript, and it is in charge of handling the request's payload.

To communicate and submit transactions, a user must create its identity with a pair composed of a public and a private key. He utilizes this authentication mechanism in all of its interactions with the system.

The data handling is made of three distinct phases. The transaction processor first starts by deserializing the data encoded in Protocol Buffer (Protobuff). Afterwards, the payload is processed depending on the collection it serves (product, agent, organization, certification) and its respective fields. For example, a specific collection field might have a size limit. The final phase consists of serializing the data so that the transaction can go through and be inserted in the chain, altering the ledger state. This process is shown on Figure 3.4.

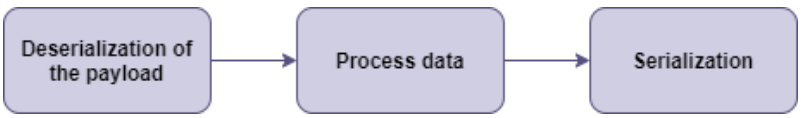

Figure 3.4: Transaction Processor phases.

3.4.6 Addressing

Addressing is one of the main implementing details one must look to when deploying a Sawtooth Network. With this technology, queries are always based on the addresses in the transaction.

Sawtooth stores its data in a Merkle Tree (see section 2.1.1). In this network design, each leaf node can be accessed through a 35 bytes address (70 hex characters).

Every sawtooth address begins with six-digit prefix used to let the validator know to which transaction processor it should send the data (namespace prefix). Sawtooth leaves the rest of the address to be thought out and designed by the developers. One of the main advantages of addressing with a transaction family prefix is that since Sawtooth completely separates its application level (transaction processor) from the network core, it can utilize the same network for different applications other than SmartAgriChain. This is possible due to each application having its own prefix.

Following the prefix, we have a two-character indicator representing one of the four collections (either Agent, Organization, Product, or Certification). The remainder of the address, which is 62 characters long, is interpreted based on the collection identifier used in the previous space.

For example, the agent collection has two equal-sized parts in its custom 62 characters long section. The first contains the organization ID, and the second contains the agent ID. All of these fields are generated utilizing cryptography hash algorithms (in this case, "sha()" function). To query all the agents belonging to a specific organization, we only have to use a partial address without the agent section. An example of an Agent collection address is shown in Table 3.2.

General Addressing sections are depicted in Figure 3.5.

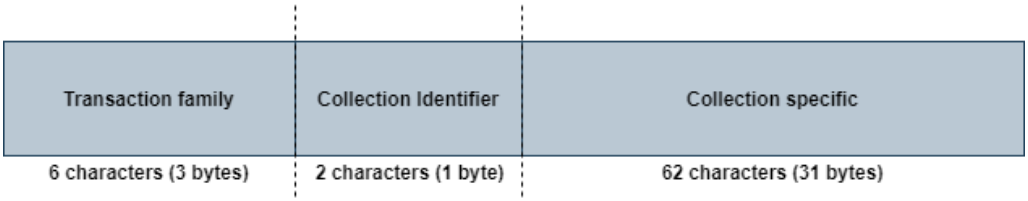

Figure 3.5: Addressing in Sawtooth.

Table 3.2: Example of Agent collection addressing.

Transaction Family	Collection ID	Organization ID	Agent ID
4c0aba	ed	sha (org,31)	sha (agent,31)

3.4.7 Consensus

Hyperledger Sawtooth gives the developer the possibility of choosing between four different types of dynamic consensus algorithms. These are the PBFT, PoET (SGX or CFT), Raft, and Devmode (the first three are explained below). All of these algorithms have its advantages and disadvantages. Devmode was firstly used in order to develop the network and then PoET was chosen in the deployment phase. The reason for these choices is detailed below.

Sawtooth devmode consensus is equipped to be used in development to test a transaction processor with one single node. Regarding Proof of Elapsed Time, this is a byzantine fault-tolerant

(BFT) algorithm where every participating node waits for a randomly distributed time value assigned by a trusted timer. This time is free of malicious intervention because it is obtained via Intel Software Guard Extensions (SGX), a TEE. A TEE (Trusted Execution Environment) is software that runs a particular set of instructors in a protected environment in memory. After, the first node that finishes its waiting period can mine the upcoming block. In other words, the node that wins that election is the elected leader. After that, the block is easily verified by the remainder of its peers, and the process repeats itself [6]. This process is depicted in Figure 3.6.

One of the most notable advantages of this algorithm is the efficiency created in allowing the processor of the miner to sleep instead of solving a computationally heavy problem, as it happens with blockchains utilizing PoW. In that time frame, the processor can then focus on other tasks or requests. Furthermore, Proof of Elapsed Time does not rely on any kind of cryptocurrency to function which largely weights in a Supply Chain Management implementation since some transactions might not involve monetary affairs.

Sawtooth contains a simulator of this algorithm that does not require specific hardware. It is called PoET CFT since instead of using the SGX, it is simulated in the Sawtooth system. This algorithm is Crash Fault Tolerant (CFT) because a trusted agent does not give the time for each node. Despite that security difference, we chose to implement this system for testing with this algorithm since it does not need specific hardware for its execution.

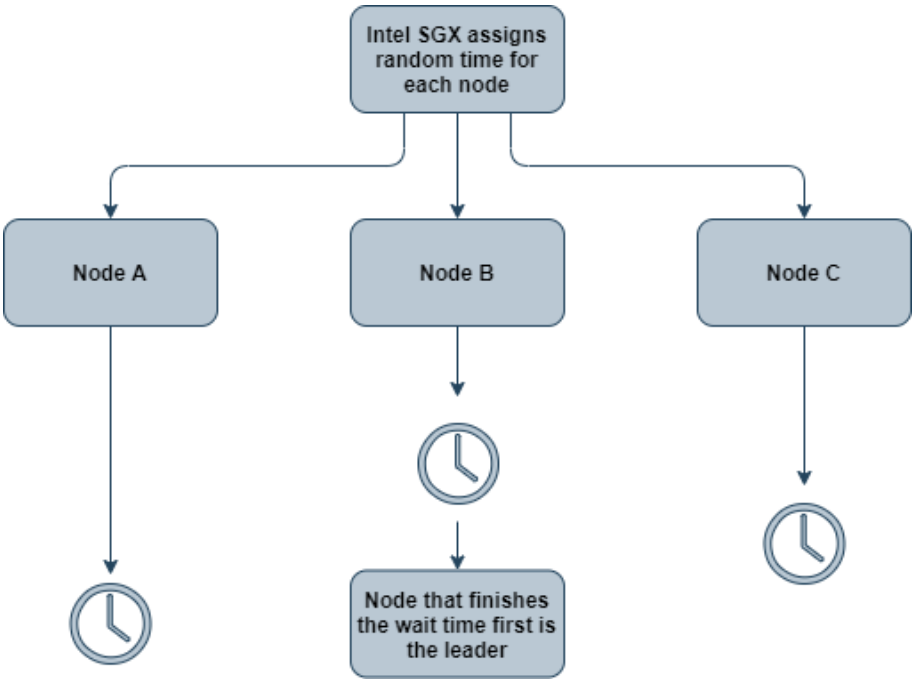

Figure 3.6: Proof of Elapsed Time.

[6] Sawtooth Docs - https://sawtooth.hyperledger.org/docs/core/releases/latest/ Accessed on: June 2021

3.4.8 Certification process in Sawtooth

To study and analyse this study's hypothesis, certification processes must be committed through transactions into the blockchain.

The three scenarios detailed in section 3.1.1 are replicated in this dissertation by using the Certification collection with a various number of steps. This can be either 1, 2, or 3 total steps.

For that purpose, a test class for the whole process of certification was developed, which allows simulating the insertion of a certification process and its completion. This class was used to mock the entire certification scenario and prove this network's purpose of allowing producers and certifying entities to create or alter a certification process.

We will assume the network is empty and does not contain any actor or organization registered to explain the whole process.

The process of certification in the blockchain is depicted in Figure 3.7, and it can be divided into three main phases:

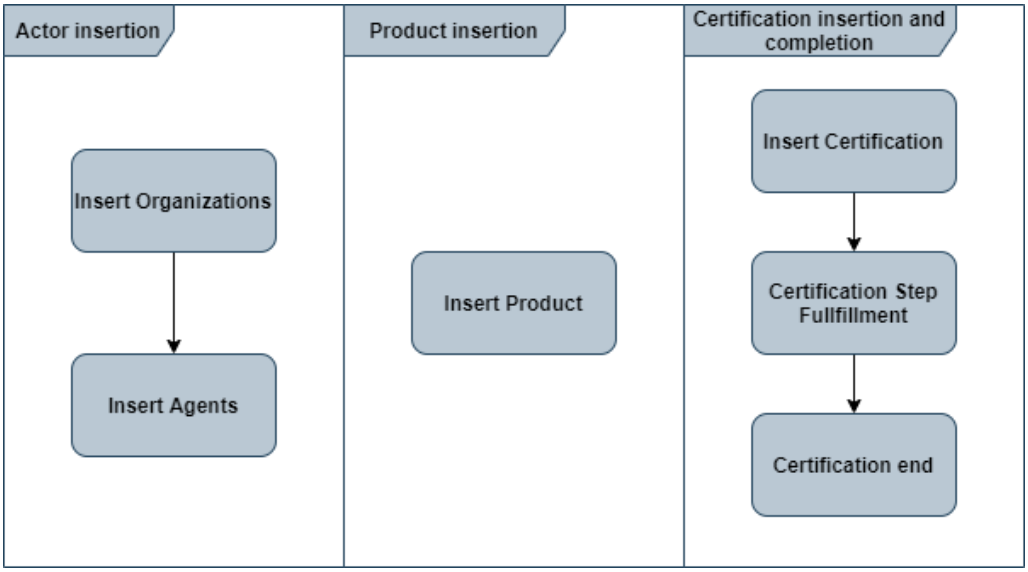

Figure 3.7: Certification process phases.

- Actors insertion phase - phase in which the organizations and the actors responsible for the certification are created in the system; Firstly, two organizations are inserted using a name and an address. Following this, we need to set up the actors that will later interact in the certification process. Two different agents are inserted in the network through two different transactions with their names, roles, public key. Another important implementation detail is that we can associate the inserted agents with their organizations via the address (see section 3.4.6. This segment is the one that initiates the whole process of certification. By inserting the actors in the network, we can then verify ownership of the products or certifications and apply logic accordingly if it is required;

- Product insertion phase - in this phase, the product subject to the future certification will be sent to the network. In this transactions payload, we can provide the following data: product's name, owner's public key, quantity, and product origin information like producer identification through its public key, producing data, expiration date. Depending on if the product is a complex item with multiple products used in its production, we can also provide a list of components used to make the item. It is important to note that product details can be modified. The blockchain will always keep a history of the product data and know who changed it.

- Certification insertion and completion phase - this is the final phase of this scenario. Begins with the sending of a transaction to the network containing a certification payload. In this payload, we can have the product, the producer and the organization identifiers, certification status (accepted, pending, rejected) and certification steps. A certification step is a phase in which the producer has to submit new information or a file to attest to the product's quality and provenance. After the insertion, we have the back-and-forth transactions between the producer and the certifying agent to fulfill the requirements of the step. After the steps are covered, the certifying agent alters the certification status to accepted and can provide a document link with the field certificationDocument;

3.4.9 Unit Testing

To test this network, unit tests in Kotlin were developed for each collection class presented in section 3.4.4. This is done to validate the insertion of transactions containing each one of the four collections. These test classes are built to send a request to deliver one transaction with a specific payload. Following that, it queries the blockchain to check if the previously requested transaction was successful and if all its information is correctly stored in the blockchain network.

3.5 Chapter Summary

In this chapter, we began by presenting the problem in current agricultural supply chains with a specification of the certification phase to better understand this study's scope. After, we discussed the technology options that could be a fit as a solution to the problem. Following, an explanation of Hyperledger Sawtooth components and system structure is presented, including information about its Rest-API, Validator node, transaction processor, addressing of transactions in the network, and the consensus algorithm specification. We ended this chapter with a description of how a certification process is replicated in the blockchain solution.

Chapter 4

Experimentation

This chapter contains the details of the experimentation performed on the Hyperledger Sawtooth implementation prototype detailed in chapter 3. First, section 4.1 contains the experimentation phases. Following, section 4.2 presents the technology stack that was used to perform the multiple experiments on the system. Furthermore, section 4.3 provides system details of the machines where the tests were run. Section 4.4 explains the multiple metrics utilized to compare experiments and, finally, section 4.5 contains specific details about the tests.

4.1 Experimentation phases

To perform tests, metrics from the Sawtooth network are to be collected and analysed. For that purpose, the experimentation is composed of two separate experiments. Furthermore, the client is used to feed the requests to Sawtooth's Rest API.

We can divide each experiment into four distinct phases that are performed in this order:

- Load Simulation - feed Certification Processes transactions with mock payload to the system with multiple input rates (4.5);
- Metrics Extraction - extract the metrics (4.4) from the transaction stress tests mentioned in the previous phase;

- Results discussion - analyse the collected results with a focus on exposing the information that is more relevant to answer the research questions and to validate the hypothesis (chapter 3.2). Following, a discussion on the data to answer the questions (chapter 5);
- Draw conclusions - conclusions are taken from the experimentation used to answer the research questions. In this phase of the dissertation, future work and some difficulties faced in the present study are also presented (6);

The experiment method is depicted in Figure 4.1.

These experiments aim to stress the network to study its behaviour and conclude if it has a big enough transaction throughput to hold a real-life agricultural supply chain and all of its actors, organizations, and products. Stress and load tests will be performed to find the limits in terms of transactions per second, scalability, response time, and system behavior with more participating nodes.

Figure 4.1: Scientific method of experimentation.

4.2 Test Technology stack

In order to achieve the goal of studying the distributed network and Hyperledger Sawtooth implementation, we first need to have access to the relevant metrics that will allow us to measure the system's performance. System performance is a term that, in this case, can be related to hardware performance (CPU and RAM) or network response time and transaction execution rate. With a stress test in which we will try to bring the system closer to its limits, it is important to understand its limits. These limits can be split into two different categories:

- Network design limit - meaning that a network setting or a certain way of implementing the system causes a bottleneck (changing the number of transactions per batch).
- Computational limit - meaning the hardware itself is being utilized at its maximum potential and can not perform any better with the current design.

To extract the desired metrics to prove the hypothesis, a test technology stack composed of Telegraf [1], InfluxDB [2] and Grafana [3] was required. These software components are individually explained below:

[1] Telegraf - https://www.influxdata.com/time-series-platform/telegraf/ Accessed on: June 2021
[2] InfluxDB - https://www.influxdata.com/products/influxdb Accessed on: June 2021
[3] Grafana - https://grafana.com/ Accessed on: June 2021

- Telegraf - open-source server agent used to collect, aggregate, and send system-related metrics like CPU and RAM usage to a database. This component is fundamental to analyse the computational limits;

- InfluxDB - this component is a time-series database (TSDB) developed using Go by InfluxData [4]. It is utilized as storage for all the collected data. Furthermore, it is responsible for storing the hardware system metrics sent by Telegraf and the Sawtooth internal metrics sent by the Validator (section 3.4.2). The latter are important to analyse the network design limit (transaction processing data);

- Grafana - software used to display the metrics collected by the database. Here we can filter and query the database for specific data.

These components are running in the genesis node (node responsible for initializing the network and posting the genesis block to the network). Ideally, InfluxDB and Grafana would be outside this machine because they are extra programs running on the same machine. Regardless of that need, Telegraf must be on the genesis machine to collect system information.

This scheme is depicted in Figure 4.2.

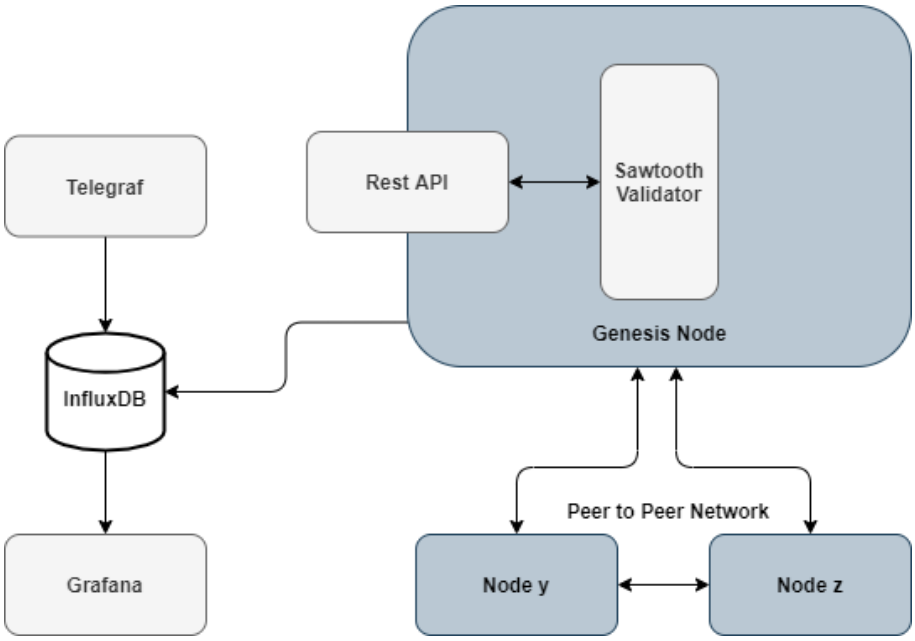

Figure 4.2: Test stack to measure the desired metrics.

[4] Influxdata - https://www.influxdata.com/ Accessed on: June 2021

4.3 Environment

All the experiments detailed in this study are run in Virtual Machines containing instances of Ubuntu 18.04. These virtual machines are running in the same network. Furthermore, during the testing, transaction latency is dependent on network delays.

It is essential to understand that the VM running the genesis node requires extra computational power compared to the other machines. This particular machine needs that advantage since it contains services that contribute to the extraction of performance data from the network (Telegraf, Grafana, and InfluxDB running).

Each non-genesis node is composed of a Fraunhofer AICOS m1 large instance with the specs being as follows: 0.4 CPU (allocation of 40% of 1 core of Intel Xeon (R) X5675 @3.07Ghz CPU); 1 GB of RAM, and 16GB of disk space. On the other hand, the machine containing the genesis node has 2GB of RAM, one core of the same CPU, and 16GB of disk space. An overview of these specifications is presented in Table 4.1.

Table 4.1: System specifications.

Machine type	RAM	CPU	Disk Space
Genesis	2GB	1	16GB
Non-Genesis	1GB	0.4	16GB

This experimentation structure is depicted on Figure 4.3.

Having analysed the hardware specifications, we need to look at Software versions used to perform this study.

An overview of the software versions utilized in this dissertation is detailed in Table 4.2. The only service not running on its current latest released version is the database which is InfluxDB, version 1.7. This occurs due to the lack of compatibility found when trying to run Influx 2.x with Sawtooth.

Table 4.2: Software versions.

Sawtooth	Telegraf	Grafana	InfluxDB
1.2.6	1.18	7.5.7	1.7

All the applications and services are run using docker services (dockerized) except Telegraf, a Ubuntu system service.

4.4 Metrics

For experimentation purposes, a set of metrics was selected. These metrics become available to the user in a Grafana dashboard. Metrics can be split into two distinct categories. The first group

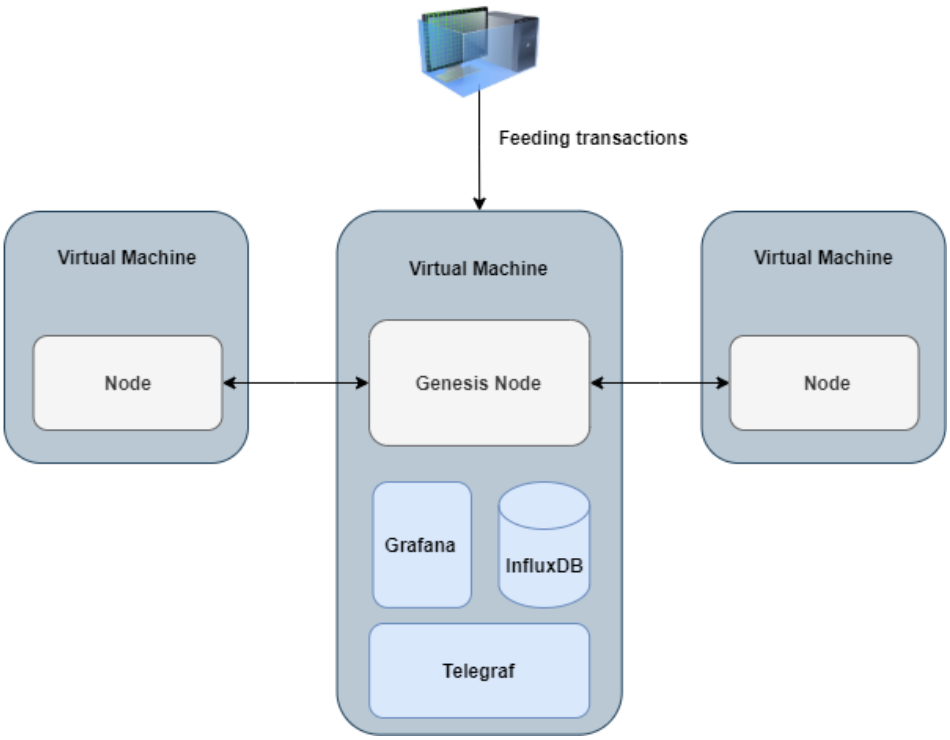

Figure 4.3: Experimentation Virtual Machine and node structure.

consists of Sawtooth metrics from its Validator:

- Committed Transactions - total number of transactions that were successfully validated and committed in the network. This metric is useful to see the percentage of transactions that are commited and the ones that stay pending or are rejected;
- Transaction Execution rate - number of transactions committed to the network considering a certain time period (transactions per second). An important metric to compare the performances in the system with different input rates;
- Transaction Processing duration (99th percentile) - value below which 99% of the transactions committed in the network take to be processed. For instance, for a value of 0,1 seconds, this metric means that 99% of the transactions were processed in that time frame. Percentile metrics have the advantage of being more resilient to outliers when compared to averages, etc. This data will more accurately tell how much time a transaction takes to be processed by the TP. We utilized this metric to calculate the standard deviation of the transaction processing time, which will be a good indicator of how stable the processing duration is over a while;
- Pending Batches - number of transaction batches in the queue pending further processing and validation. It presents a good indicator of the existing cluster of transactions flooding the system. When this value reaches large numbers, the system starts to reject transactions;

- Batches rejected (back-pressure tests) - number of batches of transactions that were rejected by Sawtooth due to the queue being full (explanation on section 3.4.2). An important metric to know when the system triggers its security measures of rejecting batches to deny Denial of Service attacks;

The second group of metrics is composed of System hardware related metrics:

- RAM used - total Random Access Memory used by the system during the time it was being monitored;
- CPU usage (user) - shows CPU used by userspace processes;

Both of these system hardware metrics are used to know how the network is affecting system performance and, more importantly, know when the hardware components are in total capacity and, for that reason, are holding the blockchain performance back.

4.5 Validation

In order to validate this study's hypothesis presented in 1.3, we will study the performance of the blockchain network with transactions sent to the system at a fixed rate. In order to do so, the experiment runs a thread that is executed every x amount of milliseconds and for 150 seconds. The period of 150 seconds was chosen because we believe it is long enough so that the data shows an accurate depiction of the network's performance.

To study the scalability of the blockchain solution implemented regarding its nodes, we need to understand how network design and the number of transactions being sent affect the overall system performance.

Two different test network designs experiment, each with a different number of nodes in the network (3 and 5). This increase in the number of nodes in the network is helpful to study the scalability of the network. In other words, how an increase in the number of network nodes affects the system's performance.

Experiment 1 (configuration 1)

In this network design and configuration, we have a system containing three nodes and 100 batches per block. These tests were run with a 50, 100, and 200 milliseconds input rate to be analysed and discussed. This is a valuable test to address Research Question 01 ("How does block size bottleneck the network?"). More details about block size and how this experiment allows to answer this question are presented in section 5.1.

Experiment 1 (configuration 2)

The network structure is the same as in the previous configuration. The difference is how the transactions are sent. In this experiment, the transactions are sent in couples (batched). These tests were run with a input rate of 200, 100 and 50 milliseconds to measure the

impact batching the transactions together (two transactions per batch instead of one) has on the system which will allow addressing Research Question 02 (“Does putting transactions into batches (sets) of transactions help the network performance?”).

Experiment 2 (configuration 1)

This experiment is run with a network size of 5 nodes to check whether the increase in node size bears any changes in the system data collected after the experiment is run. This is a very important experiment to address if the network can expand in a global Supply Chain context and will specifically address Research Question 03 (“What effects does the network size (number of nodes) have on the system performance?”).

The specific input rates and the overview of the different types of experiments run are explained in Table 4.3.

Table 4.3: Experiments overview.

Experiment 1 - 3 node network		E2 - 5 node network
1 transactions per batch	2 transactions per batch	1 transactions per batch
50 ms - 20 t/s	50 ms - 40 t/s	100 ms - 10 t/s
100 ms - 10 t/s	100 ms - 20 t/s	200 ms - 5 t/s
200 ms - 5 t/s	200 ms - 10 t/s	-

4.6 Chapter Summary

In this chapter, we explained the experimentation process of this study. It starts by explaining the experimentation phases (load simulation, metrics extraction, results discussion, and conclusions). We then present the technology stack set up to collect metrics from the Sawtooth solution (InfluxDB, Telegraf, and Grafana). After, we provide details of the system design (Virtual Machines specifications and Software versions). System hardware and Sawtooth internal metrics that will help answer the research questions and validate this disseration’s hypothesis are then discussed. Lastly, the experiments used to validate this study are shown and explained.

Chapter 5

Results and Discussion

This chapter contains the results of the experimentation conducted on the Sawtooth Supply Chain solution and testing environment. Section 5.2 presents not only statistical data of the fixed rate testing with a three node network but also an analysis of the results. Section 5.3 serves the same purpose but for a network design of five nodes. This will all be done regarding this study's domain and its hypothesis.

To summarize what was previously stated in section 4.5, configuration 1 and 2 will try to answer how does block size and batching transactions together affect the network (RQ01 and RQ02). Comparing the two experiments with different network designs (amount of nodes) will answer how network size impacts system performance (RQ03).

5.1 Sawtooth details

In this section, some details about Sawtooth settings that significantly impact this study are analysed.

Firstly, each block of data is published into the system every 10 seconds (average time). After, we have that each block only holds a maximum of one hundred (100) batches. It is important to note that both of these settings were constant throughout the two different experiments.

The mentioned one hundred maximum batches per block value is the default in Sawtooth, and tuning it into higher values can create some problems in the network. Increasing block size causes the network to increase the time it takes to publish a new block and communicate that larger block to all the peers in the system, which can cause the network to fork since the global state is not maintained so easily.

With those settings in mind, we can summarize that, theoretically, every 10 seconds, the blockchain can only publish a maximum of 100 batches. If every batch only holds a single transaction, this will cap the network at a maximum of ten transactions per second (10/s). What happens if we exceed that input throughput value will be studied in the first experiment since one of the test rates is 20 transactions submitted per second (section 5.2).

5.2 Experiment 1 - Insertion with fixed rate in 3 node network

In this section, the results of the experiments conducted with a fixed input rate are going to be presented. The network utilized for this experimentation is built using three nodes. For this purpose, experiments using different input throughput were deployed. The different chosen values were 50, 100, and 200 milliseconds. These equal 20, 10, and 5 transactions per second. The first (20 transactions per second) surpasses the theoretical limit of 10 transactions per second inherent from the Sawtooth settings chosen. In this experiment, we will study what happens in that situation where the input rate is larger than the publishing capacity of the system.

Sawtooth allows to group transactions in batches. Regarding that, and because this work is in the supply chain product certification domain, we believe that each transaction should be sent to the network either alone or coupled with another transaction (batch of two) to better depict a real-world scenario. We will study how a variation in this configuration can alter the performance on an overwhelmed system between configuration 1 (section 5.2.1) and configuration 2 (section 5.2.2).

5.2.1 Configuration 1

This subsection analyzes and discusses the experiments conducted within a three node network design with one transaction per batch. This is probably the most utilized use case in this study's context of Agricultural Supply Chain since an actor may not want to submit a large number of transactions at the same time.

It is essential to understand each input rate and how many transactions per second are being submitted into the system. Firstly, the 200 milliseconds test has an average submission rate of 5 transactions per second (tps) and, therefore, totals 750 committed batches (in this configuration, each batch contains one transaction). Following, the remaining two tests of 100 milliseconds and 50 milliseconds submit 10 tps and 20 tps, respectively, which totals to 1500 and 3000 total transactions in the 150 seconds in which the test occurs.

Total number of committed transactions

As a starting point, the total number of committed transactions is depicted in Figure 5.1. In that plot, we can observe that the faster throughput rate doesn't equal to more total transactions committed. In 150 seconds, the 100 milliseconds rate is the one presenting the largest throughput of transactions committed.

It is also imperative to note that during the 150 seconds, the slower input committed the best percentage of the total transactions sent. At a 5 tps rate, and in 150 seconds, it committed 720, which equals to 96% of the total 750 that were submitted. This value is not 100% since the network only committed the remainder of 30 transactions after the 150 seconds studied period. Following, the input rates of 100 milliseconds and 50 milliseconds presented a percentage of 76% and 31%, respectively, on their 1500 and 3000 total inserted transactions.

It is important to note that the transactions that were not committed in the 150 seconds span were not lost. Due to the higher value of input throughput they were either rejected or are in a pending state and will be processed when the system has the needed processing power. When developing an application utilizing this underlying technology, a client should have a workflow that allows for transactions to be re-sent after they are rejected.

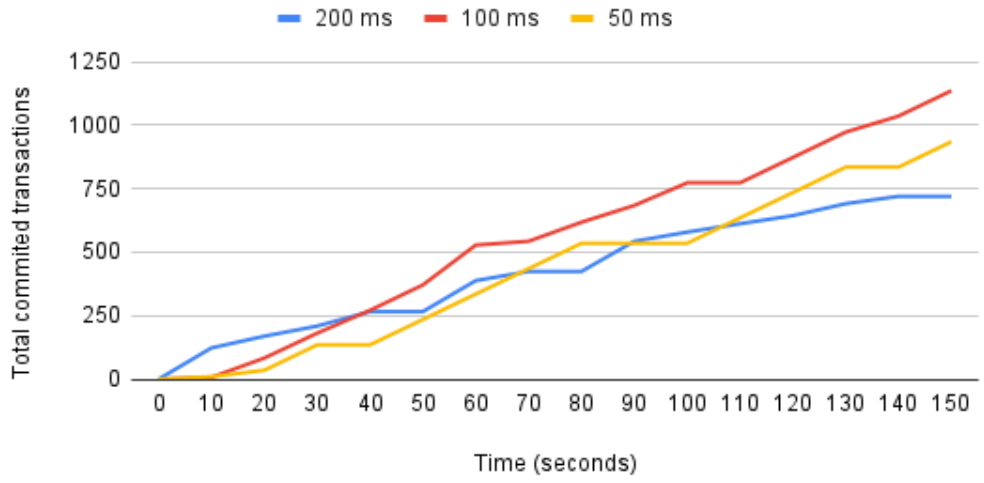

Figure 5.1: Total number of committed transactions in a 3 node network with 1 transaction per batch.

Transaction execution rate

The transaction execution rate is an excellent metric to study the speed at which transactions were committed into the blockchain. As we can observe in Table 5.1, the faster input rate (50 ms) is only able to commit 6,32 batches per second to the network, while the 100 ms test presented a value of 7,57. It is essential to note that the 200 milliseconds test presented an execution rate (4,8

batches committed per second) very close to its input rate. In other words, this means the system was able to process the transactions that were being sent efficiently.

The maximum execution rate occurs in the 100 milliseconds test, which is the best output value out of the three rate configurations. It presents the best execution rate results since the block size is being limited to a theoretical maximum capacity of 10 tps (explained on section 5.1).

Standard deviation is used to express how much the values of a metric are different from the average. This is a good metric to study the instability of the performance in a system. Higher values of standard deviation mean that the values differ more when comparing to the average. In this case, when a system has a more significant throughput, it presents a higher standard deviation which translates to its performance being more unstable. The 50 milliseconds test presented a larger standard deviation value (4,81) compared to the other two values (4,16 for 100 ms and 4,17 for 200 ms). In other words, and regarding execution rate, we can conclude that the system is more unstable when we increase the input throughput, which is expectable in this scenario.

Table 5.1: Transaction execution rate statistical data in a 3 node network with 1 transaction per batch.

Fixed input rate (ms and tps)	Average Ex. rate (tps)	Maximum Ex. rate	Standard deviation
50 ms (20 tps)	6,23	10,0	4,81
100 ms (10 tps)	7,57	15,7	4,16
200 ms (5 tps)	4,8	12,3	4,17

Pending and Rejected Batches

The results in table 5.2 show an expected value of pending batches, with the 50 milliseconds test being the highest average pending batches with 236,75. This is a very high value compared to its peers. It is explained by the input throughput (20 t/s) being higher than the network theoretical maximum processing value (10 t/s) with the defined setting. Some factors with influence on these metrics are the possible network delays and bottlenecks. This can cause events that should ideally be received 50 ms apart from each other to be received in bulks depending on network conditions.

Furthermore, we can also note that no tests triggered Sawtooth back-pressure test mechanism since no transactions were rejected due to the security mechanism (explained on 3.4.2).

Table 5.2: Pending and Rejected batches in a 3 node network with 1 transaction per batch.

Fixed input (ms and tps)	Avg. Pending batches	Max. Pending batches	Total rejected b.
50 ms (20 tps)	236,75	517	0
100 ms (10 tps)	42,69	139	0
200 ms (5 tps)	5,34	41	0

Transaction Processing Duration (99th percentile)

Following, we detail the results achieved regarding the Transaction Processing Duration (99th percentile). This metric most accurately depicts how long it takes to process a single transaction at a specific time (for 99% of the transactions processed). As we can observe in Figure 5.2, at the end of the 150 second experiment time, all the tests followed different paths until they started to converge into the same value of around 0,3. This action is explainable by the faster input rate (50 milliseconds) leaving a more significant amount of batches in a pending state when comparing to the other tests. As a cause of leaving batches in the pending queue, the system transaction processing duration value decreases and converges into the values presented by the other two experiments.

Furthermore, these results are backed up by the data in Table 5.3. We can observe that the 50 milliseconds input rate test presents a higher average transaction processing duration and the maximum value registered at 0,46 seconds. The instability mentioned in the transaction rate subsection is also visible in this table, with a higher value of standard deviation the 50 ms test presents when compared to the two tests that were within the system settings limit.

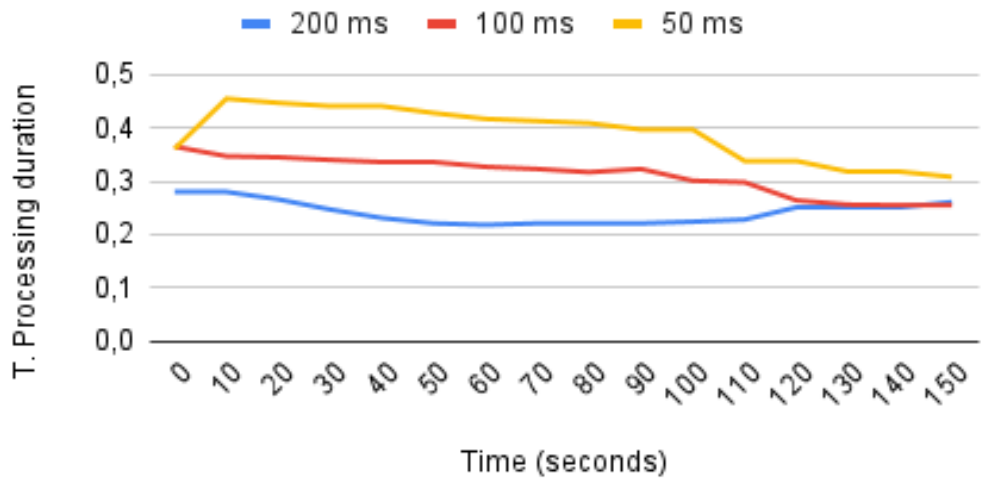

Figure 5.2: Transaction processing duration 99th percentile in a 3 node network with 1 transaction per batch.

Table 5.3: Transaction Processing Duration (99th Percentile) in a 3 node network with 1 transaction per batch.

Fixed input rate	Avg. T. Proc. Duration (s)	Max. T. Proc. Duration	Standard Deviation
50 ms	0,39	0,46	0,048
100 ms	0,31	0,37	0,034
200 ms	0,24	0,28	0,022

Another aspect we have to account for while experimenting is the system limiting factors. We

have to try to understand better what can be prohibiting the network from operating at its fullest potential. One of these reasons could be the machines in which the experiments are being run. The machines not having enough computational power to run the network efficiently can be a factor that limits system performance. We are trying to analyse this while looking at system hardware data like CPU and RAM usage.

RAM usage

As we can see in Figure 5.3, the RAM usage value does not change very much when comparing the three tests, and it converges to a value of around 55%. This indicates that the RAM in the system is not presenting a bottleneck to the network's performance.

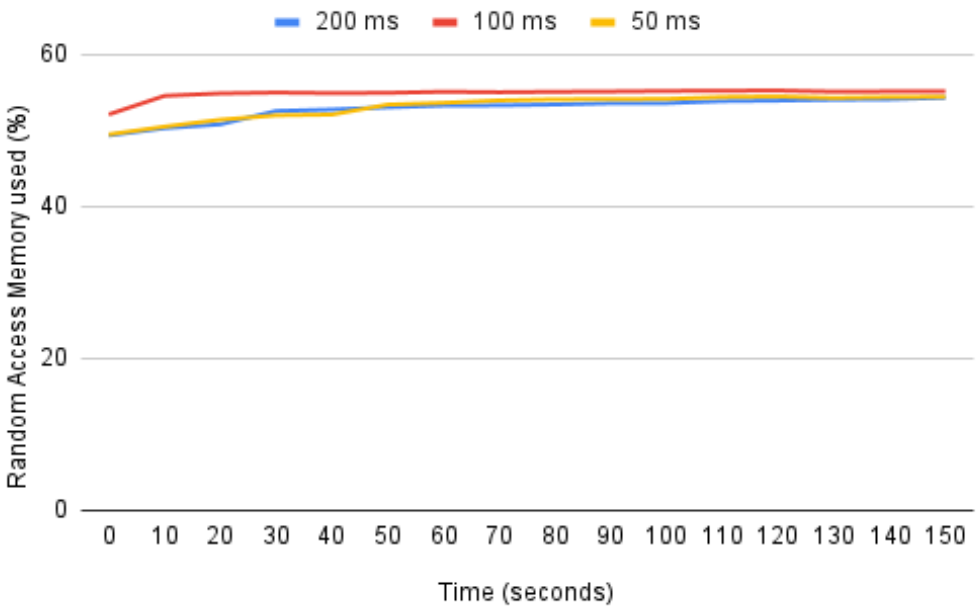

Figure 5.3: RAM usage in a 3 node network with 1 transaction per batch.

CPU usage

As for table 5.4, it presents a distinct behaviour when comparing to the RAM system metric. While the higher input rate tests presented an average CPU usage at around 42%, the 200 milliseconds experiment showed only a 24% usage. Since the average values are very distant from the maximum system capacity, we can conclude that it is not presenting a bottleneck to the blockchain solution performance. Nonetheless, it is essential to note the difference in usage when comparing the two higher throughputs (50 and 100 ms) that presented a value around 42% to the lower one (200 ms) with 24%. We can also note a significant difference in standard deviation indicating again that, in the 50 milliseconds test, the whole network was much more unstable.

Table 5.4: CPU usage in a 3 node network with 1 transaction per batch.

Fixed input rate	Average CPU usage (%)	Maximum CPU usage (%)	Standard Deviation
50 ms	42,78	89,2	25,57
100 ms	42,19	65,6	17,14
200 ms	24,43	56,5	13,87

Discussion

To summarize, we obtained the theoretically expected results in most of the metrics. First, we have the Sawtooth transaction-related metrics. We noted that the faster input rate of 50 milliseconds did not have the highest transaction execution rate out of the three tests since the system was over the theoretical limit of 10 tps and was working over its limit. For that reason, in the 50 milliseconds scenario, the transaction processing duration and the average pending batches in the queue were also much higher than the other two scenarios. Regarding the system hardware metrics, while the RAM usage was very similar in all of the tests, the CPU usage was much higher in the 50 and 100 ms experiments, presenting a value of around 42% when comparing to the 200 ms test with 24%.

We can conclude that, based on the low values the system hardware metrics present, in this experiment, the hardware is not a bottleneck to the system throughput.

Furthermore, we believe the current bottleneck is the number of transactions being published in each block that is not letting the system process transactions faster while writing them in the blockchain. There are two different ways to increase this number. Either increase the number of batches per block (currently at 100) or group more than one transaction per batch. However, increasing the number of batches per block (increasing block size) can negatively affect transactions processing time and even cause network forks (explained in section 2.1.5) since the nodes take more time to communicate and transmit blocks, delaying the update of the global state. Another solution that can be applied is batching the transactions together. That would keep the network stable while also achieving a higher transaction rate. With that in mind, increasing the number of transactions per batch to two should allow blocks to hold much more transactions while keeping the block size at 100 batches. In 5.2.2 we detail how the system performed with two transactions per batch.

5.2.2 Configuration 2

This subsection is used to explain and analyse the experiments conducted within a three node network design with every existing batch containing a total of two transactions instead of one as in Configuration 1.

To summarize, this experiment will double the number of transactions being sent to the network per time unit while leaving the number of batches being sent unaltered. Firstly, the 50 milliseconds rate will submit 20 batches per second to the network (40 tps). After, we have the

100 milliseconds test, which will send 10 batches per second (20 tps). Finally, the 200 ms input throughput sends 5 batches per second (10 tps).

In this experiment, we expect a transaction rate increase for all the input rates and a significant increase in system hardware usage (CPU and RAM). To compare the values obtained in Configuration 1 and 2, we will present the values from the second experiment and compare them to the first set of tests in each subsection analysis.

Total number of committed transactions

Starting the analysis of this configuration, we have the total number of committed transactions, depicted in Figure 5.4. There is not a surprise in these results since the faster throughput (50 milliseconds) ends up, after the 150 seconds time frame, committing more transactions than the other two.

It is important to note that the total transactions sent to the system double while utilizing this configuration. The 50 milliseconds test submits 6000 transactions, while the 100 ms test sends 3000 transactions and, finally, the 200 ms input rate is responsible for sending 1500 total transactions in the 150 seconds time frame. Furthermore, the batches being published per second are the same as in Configuration 1, so the 50 ms test presents 20 batches per second which is still higher than the 10 batches per second the network can publish.

However, it is very interesting that the 50 milliseconds experiment, because it is still over the limit in terms of batches per block, is not committing close to what it should (total of around 2500 out of the 6000). The other two rates show much more efficient rates, with the 100 ms test committing 2100 out of 3000 and the 200 ms presenting a value of 1200 out of 1500 transactions.

To summarize, the two lowest input rates commit almost all of their transactions in due time, while the 50 ms input rate is only halfway to its total value.

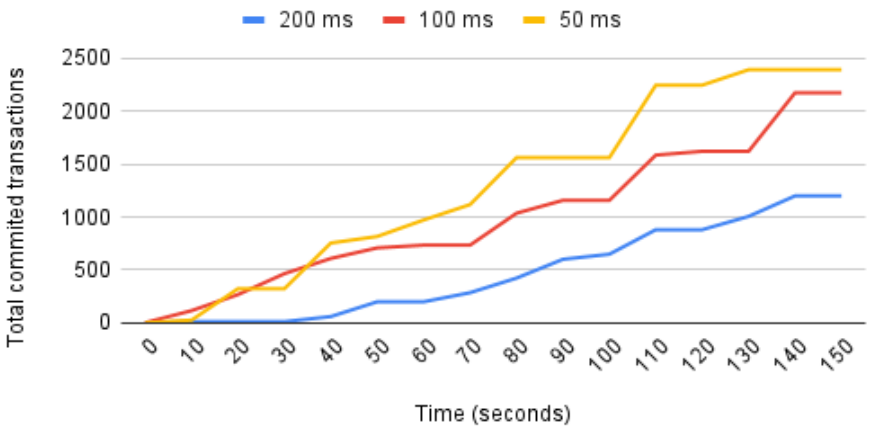

Figure 5.4: Total number of committed transactions in a 3 node network with 2 transactions per batch.

Transaction execution rate

As we can confirm based on the data presented in Table 5.5, a faster input rate translates into a faster transaction execution rate.

The maximum execution rate is 68,6 transactions per second in the 50 milliseconds test. It is higher than the 55,4 batches/second recorded in the 100 milliseconds input rate, which is what one would expect. The 50 milliseconds input rate is sending 2 transactions every 50 ms, which equals a total of 40 transactions per second, while the 100 milliseconds only delivers 20 a second. Therefore, we can also note that the 50 ms test presents a 44% efficiency rate when comparing the input with output throughput, which is lower than the 77% the 100 milliseconds experiment shows.

The standard deviation metric demonstrates the same it did in the first configuration. When we increase the input rate, the stability of the Sawtooth blockchain decreases, creating a more significant variation of execution rate values.

Table 5.5: Transaction execution rate data in a 3 node network with 2 transactions per batch.

Fixed input rate (ms and tps)	Avg. Exec. rate (tps)	Max. Exec. rate	Standard Deviation
50 ms (40 tps)	17,81	68,6	21,83
100 ms (20 tps)	15,38	55,4	16,46
200 ms (10 tps)	8,21	23,2	7,95

Pending and Rejected Batches

The results in table 5.6 show an expected outcome since this metric refers to the number of batches in a pending or rejected state and not transactions. The number of submitted batches in this configuration and Configuration 1 is the same in each test. Furthermore, there is no sizeable difference in the data between the two configurations, and a conclusion cannot be drawn from it.

Table 5.6: Pending and Rejected batches in a 3 node network with 2 transactions per batch.

Fixed input rate	Avg. Pending batches	Max. Pending batches	Total rejected batches
50 ms	196,75	549	0
100 ms	34,25	190	0
200 ms	7,38	30	0

Transaction Processing Duration (99th percentile)

Looking at the data presented in Table 5.7, and comparing to the first configuration's Transaction Processing Duration values, we can conclude that there is not a sizeable difference in the results. In other words, the time it takes to process one single transaction is similar utilizing the two different configurations, which is very positive since we are submitting a significantly larger amount of transactions.

It is important to note that Sawtooth has a feature of parallel scheduling of transactions, mentioned in section 3.3, which allows the system to process multiple transactions at the same time. Without this feature, and if the system took 0,36 seconds to deal with each transaction (value from the 50 milliseconds test) serially, the network would not be able to process the input rates we are experimenting with and would present a much slower execution rate.

Table 5.7: Transaction Processing Duration (99th Percentile) in a 3 node network with 2 transactions per batch.

Fixed input rate (ms and tps)	Avg. T. Proc. Duration (s)	Max. T. Proc. Dur.	Std. deviation
50 ms (40 tps)	0,36	0,38	0,015
100 ms (20 tps)	0,33	0,36	0,013
200 ms (10 tps)	0,29	0,34	0,053

RAM usage

When it comes to Random Access Memory usage, the results show the fastest throughput presented the most usage in RAM, as expected since it is sending more requests to the validator. It is important to note that there is an increase of RAM usage from around 55% in configuration 1 to around 80% in this set of tests which is a significant difference (see Figure 5.5).

CPU usage

This metric presents the most differentiating factor for the results found in testing the two configurations. According to the data in Table 5.8 we can see a sizeable difference between the values of CPU usage utilizing this configuration and the first one. This is due to the CPU being utilized to process double the amount of transactions it was processing in the previous configuration.

Table 5.8: CPU usage in a 3 node network with 2 transactions per batch.

Fixed input rate	Average CPU usage (%)	Maximum CPU usage (%)	Standard Deviation
50 ms	73,99	93,1	21,03
100 ms	84,51	95,5	13,09
200 ms	35,31	66,1	18,76

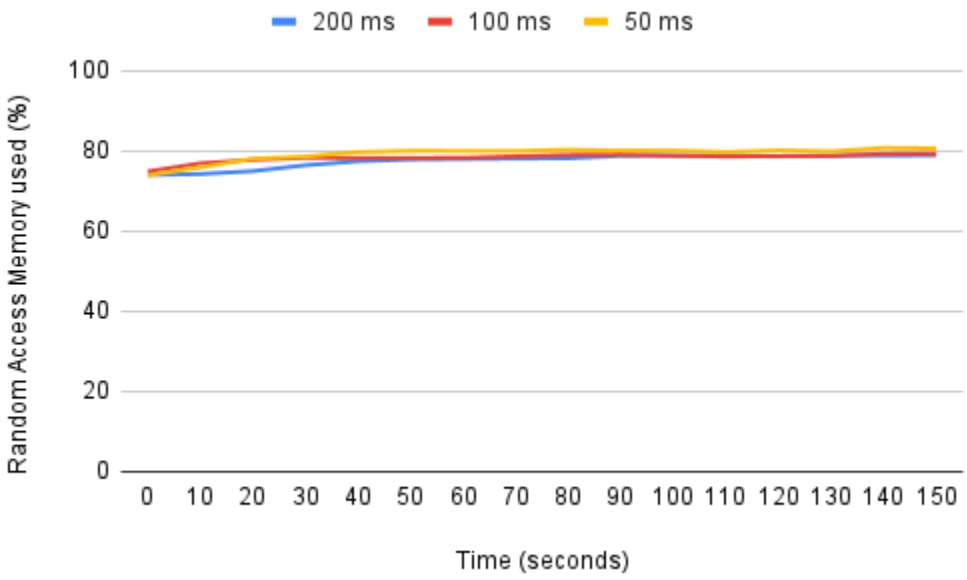

Figure 5.5: RAM usage in the system with 3 nodes and 2 transactions per batch.

Discussion

Using this configuration, we compared the results with the first configuration and noticed that the system hardware metrics are being significantly more used. However, they are not fully stressed yet. Therefore, we can conclude that increasing the number of batched transactions would probably stress the hardware even more and closer to its maximum capacity.

Furthermore, based on the data collected, we believe that this increase in the number of transactions per batch is highly efficient and favorable to the network's performance. For that reason, this increase (which is an application design feature) may be one valid way of dealing with blockchain inherent scalability problem since it allows the network to process more transactions while keeping its system metrics at a standard usage rate.

The scalability problem is going to be looked at in more detail in the second experiment (section 5.3) since the network design contains five nodes instead of three.

5.3 Experiment 2 - Insertion with fixed rate in 5 node network

Having compared how a change in the batch size can alter the network performance, now we will analyse how a change in the network size alters its output and system performance.

With an increase in the number of nodes in a blockchain network, there comes more communication between each node, negatively affecting the system. We will study this effect when comparing Configuration 1 of Experiment 1 with Experiment 2. Since this test aims to study the scalability issue of expanding the network size, there is no need to extract and analyse the input

rate of 50 milliseconds which helped studied the impact of block size in previous configurations. Therefore, this analysis contains two different input rate tests, 100 and 200 milliseconds.

Regarding input rates in the test runs, the 100 milliseconds presents 10 transactions per second and the 200 milliseconds has a rate of 5 transactions per second, the same values of Configuration 1 in Experiment 1.

Total number of committed transactions

Beginning the analysis of this configuration, we have the total number of committed transactions, depicted in Figure 5.6. The values are very close to what we saw on Configuration 1 (720 and 1136 with 200 ms and 100 ms, respectively). When it comes to total committed transactions increasing the network size does not present a scalability concern.

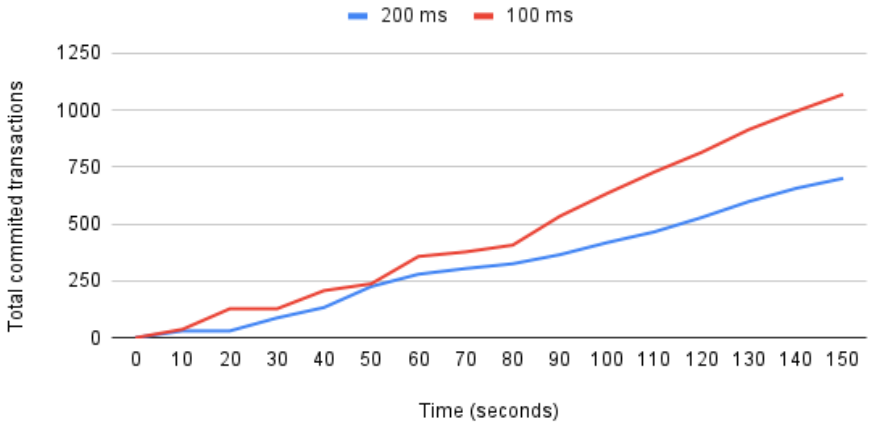

Figure 5.6: Total number of committed transactions on a 5 node network.

Transaction execution rate

Moving on to the execution rate, as we can confirm based on the data available in Table 5.9, the first experiment contains slightly higher values which indicates it was processing transactions at a faster rate, even if it is just a tiny increase.

When it comes to standard deviation, the difference between the two experiments we are comparing is slight. However, it gives the edge to Experiment 2 since its standard deviation values are lower.

To summarize this subsection, even if the transaction execution rate is slower in this experiment, the network does not oscillate its performance as much as in experiment 1.

Table 5.9: Transaction execution rate statistical data in a 5 node network.

Fixed input rate (ms and tps)	Avg Exec. rate (tps)	Max. Exec. rate	Standard deviation
100 ms (10 tps)	7,11	12,7	3,71
200 ms (5 tps)	4,5	9,2	2,23

Pending and Rejected Batches

Following, in table 5.10, we have an expected outcome. The results are similar to the values obtained utilizing the first configuration, so the number of pending batches does not change when the network increases its size from three to five nodes.

Table 5.10: Pending and Rejected Batches in a 5 node network.

Fixed input rate	Average Pending Batches	Maximum Pending Batches	Rejected Batches
100 ms	38,44	145	0
200 ms	8,1	37	0

Transaction Processing Duration (99th percentile)

With regards to the data presented in Table 5.11, and comparing to Configuration 1, we can point out an increase in Transaction Processing Duration values which backs up the data analysed in the previous section. In other words, in this second experiment, transactions take more time to be executed.

Table 5.11: Transaction Processing Duration (99th Percentile) in a 5 node network.

Fixed input rate (ms and tps)	Avg. T. Proc. Duration (s)	Max. T. Proc. Duration	Std. Dev.
100 ms (10 tps)	0,33	0,38	0,047
200 ms (5 tps)	0,29	0,35	0,029

RAM usage

Analysing Figure 5.7, we can note that both of the test values start at 60 and increase slightly into the 65% range. This is about 10% higher than the number registered in configuration 1.

CPU usage

Regarding CPU, with regards to table 5.12, we can see the values slightly increase from 34% to 38% in the test with an input rate of 100 milliseconds and from 24% to 26% in the 200 ms run.

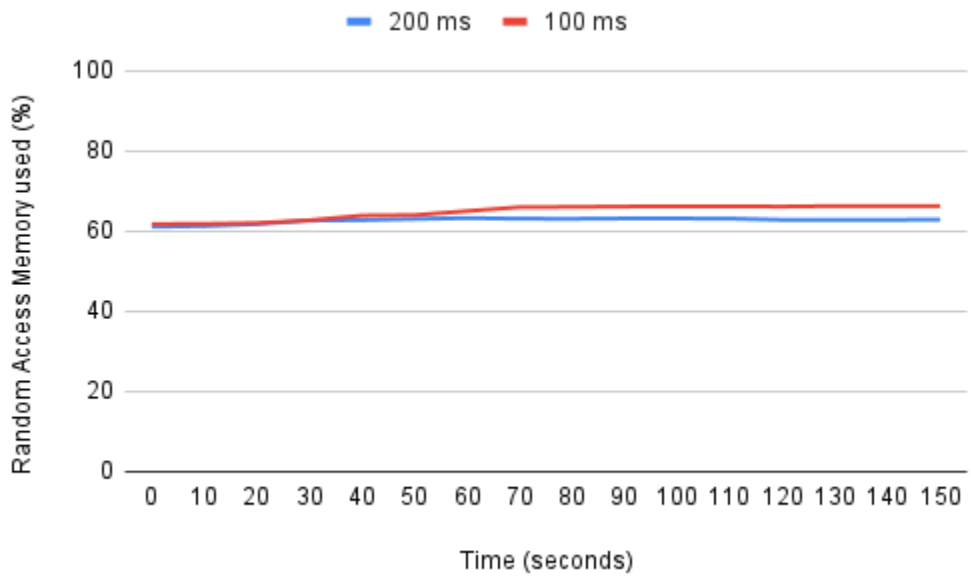

Figure 5.7: RAM usage in the system in a 5 node network.

In other words, CPU usage increases slightly when we inserted two more nodes in the network design.

Table 5.12: CPU usage in a 5 node network.

Fixed input rate	Average CPU usage (%)	Maximum CPU usage (%)	Standard Deviation
100 ms	37,90	88,8	24,95
200 ms	26,46	54,8	13,41

Discussion

By utilizing this configuration and comparing it to the first configuration of experiment 1, we can conclude about the system scalability options.

Addressing the comparison between experiences 1 and 2, a slight decrease in the execution rate and an increase in system hardware usage metrics can be justified by the significantly more considerable amount of communication a two node increase in the network requires. It is essential to note that network latency, utilized system hardware, and variance can affect the results. We cannot conclude that another increase in network size from five to seven nodes or a different scenario would have the same effect. To summarize, the solution could handle the extra nodes added to its design with only a small increase in usage and a slight decrease in transaction rate. These experiment input rates present much higher transactions per second values when compared to what a real-world AgriFoods Supply Chain would require. Therefore, we can conclude that this solution is scalable in that scenario.

5.4 Chapter Summary

In this chapter, we presented the results of the experiments that were conducted for this study. It starts by comparing the results obtained when experimenting with committing batches with one transaction and two transactions (with three nodes in the network). We provide the results for experiment 2 (five nodes), compare the data with the first experiment, and conclude the system. Furthermore, in this section, we try to address the research questions of this study with the discussion regarding the experiments conducted on the Sawtooth prototype.

Chapter 6

Conclusions

The need for decentralized solutions in agricultural supply chains has been increasing due to the globalization of the industry. The objective of this work was to test and validate if a DLT, when applied to an agricultural Supply Chain domain, can become a solution to lower transaction costs while, by providing the inherent advantages of blockchain, allows to improve the traceability, trust, and decentralization of product data, as compared to other legacy (centralized) solutions.

6.1 Main Challenges

Regarding technical challenges, blockchain implementations hold many different settings and structure details one must study and understand to make the best solution. This is due to the large number of technological advancements this field of study has been subject to in the last few years, which holds advantages and disadvantages. The main downside is that the technological landscape is constantly changing, making it hard for developers to make system decisions that can remain relevant through the mentioned evolution. Another adequate downside is that there is no clear implementation leader at this point in time, causing fragmentation and several small and not fully matured/maintained projects.

6.2 Conclusions

This study starts with relevant topics such as DLT, concepts, properties, and different components being analysed. Next, some of the most used implementation technologies were analysed. One of which is this study's underlying blockchain technology, Hyperledger Sawtooth. After, there is an analysis of DLT solutions in Agricultural Supply Chains which contains both advantages and disadvantages of its utilization. Case studies regarding DLT in AgriFood SC are then presented along with their implementation details and primary reasons for implementation. Following, we explain the details of the developed Sawtooth prototype, focusing on the consensus algorithm used

(POET CFT). This study focused heavily on testing the system and its performance to answer the Research Questions and validate the hypothesis.

Regarding the research questions, we can conclude that they were addressed in this study. While research questions 2 and 3 were fully addressed, question 1 was only partially answered. Research Question 01 involved analysing how the network's block size affected the system. Despite discussing that topic, more in-depth work could be done surrounding that aspect of the solution.

After, RQ02 centered around the batching of transactions. By comparing two different configurations of a network containing three nodes, we concluded that increasing the number of transactions in a batch can slightly increase the system workload but has a significant positive impact on the execution rate of the transactions. This is an excellent feature for a complete application with the same use cases. It would make the network more efficient by decreasing the number of batches holding only one transaction and increasing multiple transaction batches.

Finally, RQ03 addressed the scalability problem found in most blockchain solutions. Increasing network size (number of nodes) causes the nodes to increase the amount of network communication required to stabilize the system and reach a consensus. This factor can have a significant effect on system performance. In the solution, while it was proven that an increase in the number of nodes does negatively alter the production, that effect was slight. The network was being stressed with a rate scenario that does not represent the normal workload a supply chain in an agricultural domain would sustain. Therefore, we can conclude that building and deploying a scalable solution regarding a DLT for an agricultural solution is achievable.

Lastly, the hypothesis was validated since the developed network is secure, immutable, affordable, a good candidate to provide traceability of agricultural items, and increase visibility across the entire supply chain network until the end consumer. By utilizing this solution, every actor involved in the chain would process information faster and secure access to immutable information that would directly improve food quality.

While doing this study, I learned a lot about all the small details about doing a dissertation, which I could not imagine before this experience started.

6.3 Future work

Future work should address the in-depth study of how a network reacts, practically, to a block size increase. In Sawtooth's case, that would be achieved by increasing the number of maximum batches allowed per published block. It would be interesting to, while altering the block size, also variate the block publish time. This would allow studying how those blockchain key settings alter the throughput rate a system of this caliber can withstand and fully address research question number 1.

Another item that can be addressed in the future is to get specific blockchain metrics instead of system hardware metrics. For example, having RAM utilized only by the solution and not the

entire system would allow studying, with more detail, its performance while eliminating outliers in the collected data.

Furthermore, a study could be performed regarding the time it takes to complete a whole certification process compared to a centralized legacy system. In this work, we utilized fixed input rate testing to study performance. However, a direct comparison between centralized and decentralized when regarding a complete certification process could result in an excellent study to help further validate the use of DLT in Agricultural Supply Chains.

Future work could also address studying the system with real hardware (raspberry pis, for example) in different locations of the globe in order to better simulate a global Supply Chain.

Finally, in this study we could not run the solution with a greater number of nodes, for example, one hundred and this would be a great study. The large increase in network size would provide extra insight and data regarding the scalability of the solution which would further help validate a DLT solution in an AgriFood Supply Chain domain.

References

[1] Blockchain Supply Chain Transformation E-Book | Microsoft Azure. https://azure.microsoft.com/en-us/resources/how-blockchain-will-transform-modern-supply-chain/. Accessed: January 2021.

[2] Princes uses blockchain to create end-to-end transparency in tuna chain - Supply Chain Movement. https://www.supplychainmovement.com/princes-uses-blockchain-to-create-end-to-end-transparency-in-tuna-chain/. Accessed: June 2021.

[3] Hamda Al-Breiki, Muhammad Habib Ur Rehman, Khaled Salah, and Davor Svetinovic. Trustworthy blockchain oracles: Review, comparison, and open research challenges. *IEEE Access*, 8:85675–85685, 2020.

[4] Nicola Atzei, Massimo Bartoletti, and Tiziana Cimoli. A survey of attacks on ethereum smart contracts (sok). In Matteo Maffei and Mark Ryan, editors, *Principles of Security and Trust*, pages 164–186, Berlin, Heidelberg, 2017. Springer Berlin Heidelberg.

[5] Mohamed Awwad, Sohit Reddy, Varun Kazhana Airpulli, Madhubala Santosh Zambre, Aniket Marathe, and Prasham Jain. Blockchain technology for efficient management of supply chain. 09 2018.

[6] Leemon Baird, Mance Harmon, and Paul Madsen. Hedera: A Public Hashgraph Network & Governing Council. Technical report, 2018.

[7] Mark C. Ballandies, Marcus M. Dapp, and Evangelos Pournaras. Decrypting distributed ledger design - Taxonomy, classification and blockchain community evaluation. *arXiv*, 2018.

[8] Federico Matteo Bencic and Ivana Podnar Zarko. Distributed ledger technology: Blockchain compared to directed acyclic graph. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, July 2018.

[9] Evangelos Benos, Rod Garratt, and Pedro Gurrola Perez. The economics of distributed ledger technology for securities settlement. *Ledger*, 4, 11 2019.

[10] Chris Berg, Sinclair Davidson, and Jason Potts. Ledgers. *SSRN Electronic Journal*, 2018.

[11] Giulio Caldarelli. Understanding the blockchain oracle problem: A call for action. *Information*, 11(11):509, October 2020.

[12] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *Third Symposium on Operating Systems Design and Implementation (OSDI)*, New Orleans, Louisiana, February 1999. USENIX Association, Co-sponsored by IEEE TCOS and ACM SIGOPS.

[13] Yanling Chang, Eleftherios Iakovou, and Weidong Shi. Blockchain in global supply chains and cross border trade: a critical synthesis of the state-of-the-art, challenges and opportunities. *International Journal of Production Research*, 58(7):2082–2099, August 2019.

[14] Anamika Chauhan, Om Prakash Malviya, Madhav Verma, and Tejinder Singh Mor. Blockchain and scalability. In *2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pages 122–128, 2018.

[15] Mohammad Chowdhury, Md. Sadek Ferdous, Kamanashis Biswas, Niaz Chowdhury, A. S. M. Kayes, Mamoun Alazab, and Paul Watters. A comparative analysis of distributed ledger technology platforms. *IEEE Access*, 7:167930–167943, 11 2019.

[16] K Christidis and M Devetsikiotis. Blockchains and Smart Contracts for the Internet of Things," in IEEE Access, vol. 4, no. , pp. 2292-2303, 2016. *IEEE Access*, 4:2292–2303, 2016.

[17] Primavera De Filippi. The interplay between decentralization and privacy: the case of blockchain technologies. 09 2016.

[18] Dnyaneshwar Ghode, Vinod Yadav, Rakesh Jain, and Gunjan Soni. Adoption of blockchain in supply chain: an analysis of influencing factors. *Journal of Enterprise Information Management*, 33(3):437–456, March 2020.

[19] Tom Groenfeld. IBM And Maersk Apply Blockchain To Container Shipping. https://www.forbes.com/sites/tomgroenfeldt/2017/03/05/ibm-and-maersk-apply-blockchain-to-container-shipping/?sh=1666c3743f05. Accessed: January 2021.

[20] Nabil El Ioini and Claus Pahl. A review of distributed ledger technologies. In *Lecture Notes in Computer Science*, pages 277–288. Springer International Publishing, 2018.

[21] Arman Jabbari and Philip Kaminsky. Blockchain and Supply Chain Management. Technical report, 2018.

[22] Binu Jacob, Joseph Porter, Rajesh Khemraj, David Lasecki, and Edmund Miller. Blockchain Technology in the Shipping Industry: What can we learn from early adopters? Technical report, 2019.

[23] Assey Mbang Janvier-James. A new introduction to supply chains and supply chain management: Definitions and theories perspective. *International Business Research*, 5(1), December 2011.

[24] Alen Jugović, Juraj Bukša, Alex Dragoslavić, and David Sopta. The possibilities of applying blockchain technology in shipping. *Pomorstvo*, 33(2):274–279, December 2019.

[25] Reshma Kamath. Food traceability on blockchain: Walmart's pork and mango pilots with IBM. *The Journal of the British Blockchain Association*, 1(1):1–12, July 2018.

[26] Laura Kopczak and M.E. Johnson. The supply-chain management effect. *MIT Sloan Management Review*, 44:27–34, 03 2003.

[27] Malni Kumarathunga. Improving farmers' participation in agri supply chains with blockchain and smart contracts. In *2020 Seventh International Conference on Software Defined Systems (SDS)*. IEEE, April 2020.

[28] Xiaoqi Li, Peng Jiang, T. Chen, X. Luo, and Qiao yan Wen. A survey on the security of blockchain systems. *Future Gener. Comput. Syst.*, 107:841–853, 2020.

[29] I.-C Lin and T.-C Liao. A survey of blockchain security issues and challenges. *International Journal of Network Security*, 19:653–659, 09 2017.

[30] Miranda P.M. Meuwissen, Annet G.J. Velthuis, Henk Hogeveen, and Ruud B.M. Huirne. Traceability And Certification In Meat Supply Chains. *Journal of Agribusiness*, 21(2):1–15, 2003.

[31] Joanna Moubarak, Eric Filiol, and Chamoun Maroun. On blockchain security and relevant attacks. pages 1–6, 04 2018.

[32] Gopal Naik and D.N. Suresh. Challenges of creating sustainable agri-retail supply chains. *IIMB Management Review*, 30(3):270–282, September 2018.

[33] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. Technical report.

[34] World Trade Organization. *World Trade Report 2018*. 2018.

[35] Kirk A. Patterson, Curtis M. Grimm, and Thomas M. Corsi. Adopting new technologies for supply chain management. *Transportation Research Part E: Logistics and Transportation Review*, 39(2):95–121, March 2003.

[36] Simon Pearson, David May, Georgios Leontidis, Mark Swainson, Steve Brewer, Luc Bidaut, Jeremy G. Frey, Gerard Parr, Roger Maull, and Andrea Zisman. Are distributed ledger technologies the panacea for food traceability? *Global Food Security*, 20:145–149, March 2019.

[37] S. Popov. The tangle. 2015.

[38] C. Saraf and S. Sabadra. Blockchain platforms: A compendium. In *2018 IEEE International Conference on Innovative Research and Development (ICIRD)*, pages 1–6, 2018.

[39] G. Thiraviya Suyambu, M. Anand, and M. Janakirani. Blockchain – a most disruptive technology on the spotlight of world engineering education paradigm. *Procedia Computer Science*, 172:152–158, 2020.

[40] Nick Szabo. Smart contracts. *Unpublished manuscript*, 1994.

[41] Paolo Tasca and Claudio Tessone. A taxonomy of blockchain technologies: Principles of identification and classification. *Ledger*, 4, 02 2019.

[42] Ann Terlaak and Andrew A. King. The effect of certification with the ISO 9000 quality management standard: A signaling approach. *Journal of Economic Behavior & Organization*, 60(4):579–602, August 2006.

[43] Martin Valenta and P. Sandner. Comparison of ethereum, hyperledger fabric and corda. 2017.

[44] Junfeng Xie, F. Richard Yu, Tao Huang, Renchao Xie, Jiang Liu, and Yunjie Liu. A survey on the scalability of blockchain systems. *IEEE Network*, 33(5):166–173, 2019.

[45] Victor Zakhary, Mohammad Javad Amiri, Sujaya Maiyya, Divyakant Agrawal, and Amr ElAbbadi. Towards global asset management in blockchain systems. *arXiv*, 2019.

[46] Rui Zhang, Rui Xue, and Ling Liu. Security and privacy on blockchain. *ACM Computing Surveys*, 52:1–34, 07 2019.

[47] Qiheng Zhou, Huawei Huang, Zibin Zheng, and Jing Bian. Solutions to scalability of blockchain: A survey. *IEEE Access*, 8:16440–16455, 2020.