Certitex: a Textile Certified Supply Chain
(Versão Final Após Defesa)

Miguel Alexandre Torrão Alves Brandão

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática
(2º ciclo de estudos)

Orientador: Prof. Luís Filipe Barbosa de Almeida Alexandre
Co-orientador: João Alexandre Aguiar Amaral Santos

Novembro de 2020

Agradecimentos

This work was carried out at SociaLab, in Universidade da Beira Interior's informatics department, partially integrated with the SociaLab's activies and Zirak an italian informatic technologies company under supervision of Professor Luís Filipe Barbosa de Almeida Alexandre, João Alexandre Aguiar Amaral Santos and Fabrizio Turco to whom I deeply thank all the help and supervision.

Resumo

O aparecimento das tecnologias *blockchain* e o seu crescimento e desenvolvimento, têm levado à exploração de aplicações da tecnologia em novas áreas. Inicialmente, e relativamente ao tema desta tese, esta tecnologia foi explorada com o objetivo de prover cadeias de fornecimento alimentícias de rastreabilidade e transparência para o consumidor. Estudos atuais têm provado que a tecnologia apresenta propriedades poderosas para promover a rastreabilidade e não repúdio de informação. Infelizmente todas as soluções baseadas em *blockchain* encontradas na fase de pesquisa são soluções desenvolvidas por entidades privadas não havendo qualquer divulgação de informação relativa ao seu desenvolvimento, e também na maioria esmagadora dos casos sobre o seu funcionamento. Isto levou a que esta dissertação fosse maioritariamente um trabalho de investigação da tecnologia base, e desenvolvimento de raiz de uma solução funcional.

Os problemas das soluções tradicionais, prendem-se com o uso de estratégias de registo de informação não estandardizadas e facilidade de repúdio de informação isto porque cada entidade por norma age independentemente das outras e apenas comunica com aquelas que lhe estão diretamente ligadas. Adicionalmente, é comum verificar que produtos no final da sua cadeia de produção estão danificados e de ser praticamente impossível localizar onde na cadeia os danos ocorreram.

A ideia de adaptar a tecnologia *blockchain* como uma solução para a rastreabilidade de produtos na cadeia de fornecimento apresenta alguns pontos preocupantes, pois as *blockchains* são geralmente associadas a sistemas distribuídos e públicos para manter uma dada criptomoeda. Apesar de este ser o propósito inicial para a sua criação têm vindo a surgir outras tecnologias *blockchain* orientadas para o armazenamento e processamento de dados num modelo *business to business*. Estas *blockchains* possuem medidas de controlo de acesso, e são, portanto chamadas de privadas permitindo apenas acesso por parte de um grupo seleto de entidades. Adicionalmente, o armazenamento de informação numa *blockchain* é também muitas vezes associado a custos elevados, e quando nos referimos a *blockchains* públicas como a Ethereum isto é uma realidade, mas pelo uso de soluções privadas podemos colmatar este custo. É também uma preocupação os custos computacionais associados a *blockchains* de criptomoeadas como a Bitcoin e Etherum. Novamente é possível contornar esta limitação pelo uso de soluções privativas onde podemos usar algoritmos mais leves, pois o ambiente em que o sistema se vai inserir, não carece de tantos cuidados. Através do uso de *blockchain* para certificar a origem e percurso de produtos numa cadeia de fornecimento é também interessante explorar os dados recolhidos no processo e como estes podem ser utilizados para tornar a própria cadeia de fornecimentos mais eficiente.

O objetivo desta dissertação é estudar como a tecnologia *blockchain* pode ser conjugada com uma cadeia de fornecimento para oferecer rastreabilidade de produtos e recolha de informação. Para alcançar este objetivo foi desenvolvido um protótipo de uma aplicação baseada em *blockchain* para recolha de dados numa cadeia de fornecimento, bem como um protótipo de uma aplicação para a visualização e interação remota com os dados e tam-

bém um protótipo de um módulo de *Machine Learning* capaz de fazer uso da informação recolhida pela *blockchain*.

Palavras-chave

Blockchain, Armazenamento de Dados, Machine Learning, Análise de séries temporais, Quorum

Resumo alargado

O aparecimento das tecnologias *blockchain* e o seu crescimento e desenvolvimento, têm levado à exploração de aplicações da tecnologia em novas áreas além da sua original, criptomoeadas. Áreas como a gestão e rastreabilidade de produtos em cadeiras de fornecimento. Inicialmente, e relativamente ao tema desta tese, esta tecnologia foi explorada com o objetivo de prover cadeias de fornecimento alimentícias rastreabilidade e transparência para o consumidor. Atualmente estão a ser estudadas e desenvolvidas soluções aplicadas a uma maior variedade de cadeias de fornecimento. Estudos atuais têm provado que a tecnologia apresenta propriedades poderosas para promover a rastreabilidade e não repúdio de informação. A atual estrutura destas cadeias, várias entidades diferentes localizadas em diferentes espaços físicos, é propensa à aplicação de soluções de *blockchain* pois ajusta-se também à arquitetura da tecnologia em si. Tudo isto leva a um forte interesse pela aplicação da tecnologia de *blockchain* a cadeias de fornecimento. Infelizmente todas as soluções baseadas em *blockchain* encontradas na fase de pesquisa são soluções desenvolvidas por entidades privadas não havendo qualquer divulgação de informação relativa ao seu desenvolvimento, e também na maioria esmagadora dos casos sobre o seu desenvolvimento. Isto levou a que esta dissertação fosse maioritariamente um trabalho de investigação da tecnologia base, e desenvolvimento de raiz de uma solução funcional.
Os problemas das soluções tradicionais prendem-se principalmente com o uso de estratégias de registo de informação não estandardizadas e facilidade de repúdio de informação, isto porque cada entidade por norma age independentemente das outras e apenas comunica com aquelas que lhe estão diretamente ligadas. As demandas atuais dos consumidores pelo conhecimento da origem dos produtos levam à exploração de novas soluções para colmatar este problema. Adicionalmente, é comum verificar que produtos no final da sua cadeia de produção estarem danificados e de ser praticamente impossível localizar onde na cadeia os danos ocorreram.
A ideia de adaptar a tecnologia *blockchain* como uma solução para a rastreabilidade de produtos na cadeia de fornecimento apresenta alguns pontos preocupantes, pois as *blockchains* são geralmente associadas a sistemas distribuídos e públicos para manter uma dada criptomoeda. Apesar de este ser o propósito inicial para a sua criação tem vindo a surgir outras tecnologias *blockchain* orientadas para o armazenamento de dados num modelo *business to business*. Estas *blockchains* possuem medidas de controlo de acesso, e são, portanto chamadas de privadas permitindo apenas acesso por parte de um grupo seleto de entidades. Adicionalmente, o armazenamento de informação numa *blockchain* é também muitas vezes associado a custos elevados, e quando nos referimos a *blockchains* públicas como a Ethereum isto é uma realidade mas pelo uso de soluções privadas podemos colmatar este custo. É também uma preocupação os custos computacionais associados a *blockchains* de criptomoeadas como a Bitcoin e Etherum que são baseadas num algoritmo de consenso *Proof of Work*, que tem propriedades muito úteis no ambiente em que está inserido, sendo, portanto redes públicas. Novamente é possível contornar esta limitação pelo uso de soluções privativas onde podemos usar algoritmos mais leves, pois o ambi-

ente em que o sistema se vai inserir, não carece de tantos cuidados, pois todos os seus participantes serão corporações conhecidas e identificadas no sistema.
Através do uso de *blockchain* para certificar a origem e percurso de produtos numa cadeia de fornecimento é também interessante explorar os dados recolhidos no processo e como estes podem ser utilizados para tornar a própria cadeia de fornecimento mais eficiente, através de métodos estatísticos e de *machine learning*, devido à inexistência de um *dataset* com dados reais, foi necessário a adaptação de um *dataset* para este propósito. O dataset escolhido foi o registo de viagens entre postos de um *rent-a-bike* para simular as viagens entre as várias entidades na cadeia de fornecimento. No topo deste dataset tiveram que ser gerados dados fictícios para a produção das várias entidades e as ligações entre si.
O objetivo desta dissertação é estudar como a tecnologia *blockchain* pode ser conjugada com uma cadeia de fornecimento para oferecer rastreabilidade de produtos e recolha de informação. Para alcançar este objetivo foi desenvolvido um protótipo de uma aplicação baseada em *blockchain* para recolha de dados numa cadeia de fornecimento, bem como um protótipo de uma aplicação para a visualização remota dos dados inseridos e um protótipo de um módulo de *Machine Learning* capaz de fazer uso da informação recolhida pela *blockchain*.

Abstract

The appearance of blockchain technologies and their growth and development have led to the exploration of applications of the technology in new areas, in addition to the original, cryptocurrencies, areas such as product management and traceability in supply chains are being explored. Initially, this technology was explored with the aim of providing food supply chains with traceability and transparency for the consumer. Currently, solutions for a larger variety of supply chains are being studied and developed. Current studies have proven that the technology has powerful properties to promote traceability and non-repudiation of information related to products in a supply chain, as well as providing liability of entities for damages caused to products, which in the past has been notoriously difficult. The current structure of these supply chains, several different entities located in different physical spaces, is prone to the application of blockchain solutions as it also fits the architecture of the technology itself. All of this leads to a strong interest in applying blockchain technology to supply chains. Unfortunately, all the blockchain based solutions found to solve similar problems in the research phase of this project were developed by private entities, with little to no divulgation about their development and many times not even about how they function. This led to this project being mainly about researching the base technology and developing a solution from scratch.

The problems of currently used traditional solutions are related to the use of non-standardized information registration strategies and ease of repudiation of information, but current consumer demands for knowledge of the origin of products has led to the exploration of new solutions to overcome this. Additionally, it is common for products, at the end of their production cycle, to be damaged and it is practically impossible to locate where the damage occurred in the chain. The idea of adapting blockchain technology as a solution for product traceability in the supply chain presents some points of concern, as blockchains are generally associated with distributed and public systems to maintain a given cryptocurrency, thus making information public. Although this is the initial purpose of its creation, other blockchain technologies oriented to data storage in a business to business model have emerged. These blockchains have access control measures, and are therefore called private. Only allowing access by a select group of entities. Additionally, information stored on a blockchain is also often associated with high costs, and when we refer to public blockchains like Ethereum this is a reality, but by using private solutions we can mitigate this cost. It is also often a concern the computational costs associated with cryptocurrency blockchains like Bitcoin and Etherum. Again, it is possible to get around this limitation by using private solutions where we can use more light weight algorithms, because the environment in which the system will be inserted, does not benefit from the properties of such algorithms.

With the usage of blockchain to certify and record the progress of products as they travel through the supply chain, it is also interesting to explore the collected data, and how it could be used to make the supply chain itself more efficient. The purpose of this dissertation is to study how blockchain technology can be combined with a supply chain to

offer product traceability and information collection. To achieve this goal, a prototype of a blockchain-based application was developed to collect data in a supply chain, as well as a prototype of an application for remote viewing of the data entered and a prototype of a Machine Learning module able to make use of the information collected by the blockchain.

Keywords

Blockchain, Data Storage, Machine Learning, Time Series Analysis, Quorum

Contents

List of Figures

List of Tables

Acronyms

BFT Byzantine Fault Tolerance

CFT Crash Fault Tolerance

ETH Ether

EVM Ethereum Virtual Machine

IoT Internet of Things

LSTM Long Short-term Memory

PoW Proof Of Work

RAFT Role, Audience, Format and Topic.

RNN Recurrent Neural Network

Chapter 1

Introduction

1.1 Problem Description

The supply chain is a complex term referring to the ecosystem of all materials, entities, and personnel involved in the sequence of production and distribution of any goods and services. Currently, due to the demand for production, these ecosystems are incredibly complex. Increased globalization, shorter product life cycle, and rapid technological advancement in manufacturing processes as well as in the service industry necessitate that companies cooperate to meet market demands. This cooperation, be it physical or virtual, makes the supply chain more complex to manage[**?**]. Some difficulties include attributing a cause to potential damage of goods throughout their manipulation flow and transparency in product origin. As such, supply chain management is a critical part of the supply chain and a key factor in dictating company success. One important part of supply chain management is the traceability of goods throughout it. It is a difficult albeit necessary step to localize and minimize mistakes and costs and, additionally, it is becoming a part of legislation and customer demands [**?**].

1.2 General Concepts

1.2.1 Blockchain

A blockchain is a growing collection of transactions organized in blocks linked using cryptography. Each block contains the cryptographic hash of its predecessor, a timestamp, and a chunk of data. It is typically governed by a peer-to-peer network. Each of the peers participating in the network is a node in the system and possesses a copy of the ledger. By adhering to a protocol for inter-node communication and block validation, the nodes in the system work together to form the blockchain network itself. Once data is inserted into a block in the chain it can not be easily altered due to the cryptographic link existing between blocks and its distributed nature.

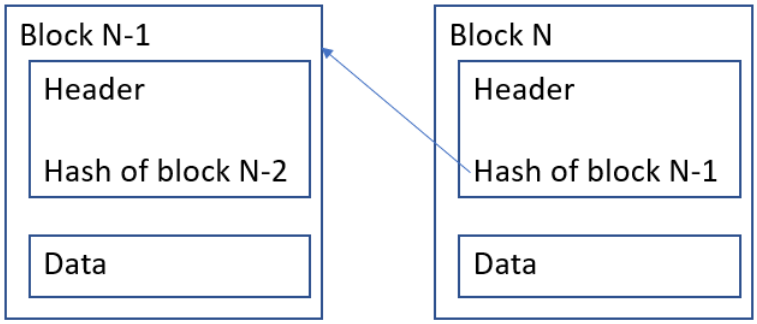

Figure 1.1: Simplified structure of a blockchain, composed by a header and a list of transactions (data). The blocks are linked by each block including the cryptographic hash of it's predecessor. The only exception is the first (genesis) block, common to all clients of the network.

Additions to the ledger are first submitted to the network, where, through a consensus mechanism, they are included in a block which is in turn distributed by the whole network. As such, every participant has a complete and eventually consistent copy of blocks with each transaction in those blocks being at the moment of addition deemed valid by the network consensus mechanism.
Some examples of consensus mechanisms would be proof-of-work, proof-of-stake, or proof-of-authority. For example, Proof-of-work, used in Ethereum is based on the notion that chains with more effort put into them (computational power) are more secure because any entity attempting to overwrite them would need to produce another chain with as much computational power invested in it as the whole Ethereum network. As such, to validate a block, a node must verify the validity of each transaction contained within it (verifying the digital signatures) as well as solve a complex cryptographical puzzle of finding a value that when cryptographically hashed with the block itself results in a hash with a certain set of properties. This facet is mostly seen in public blockchains where anyone has access to the network and can participate in it.

1.2.2 Long Short-Term Memory

Long Short-term Memory (LSTM) is a Recurrent Neural Network (RNN) architecture used in the field of deep learning. RNN unlike other network architectures, which are feedforward, have feedback connections. LSTM distinguishes itself from other RNN because it solves the vanishing gradient problem through the use of LSTM units. These units include a memory cell that is capable of maintaining information in memory for long periods of time, as well as a set of gates that is used to control when information enters the memory when it's output, and when it's forgotten.

1.3 Adopted Approach

As a beginning point, to clearly understand the main advantages and possible issues of using a blockchain solution to perform supply chain traceability, an in-depth study of its

fundamental concepts and state-of-the-art of blockchain usage in the supply chain was needed. Research began by looking into the fundamentals of blockchain, studying its implementation in crypto-currencies, its main properties, and issues. After attaining a good grasp on this, research continued by looking into previous blockchain solutions in the field of the supply chain, more specifically into the food industry as this is where most current solutions are centered.
After understanding the technology and studying previous implementations, of which information was scarce, the next step was to design a system that would take advantage of the studied properties whilst minimizing its disadvantages. Meanwhile, information was collected on several blockchain systems so that a comparative study could be conducted when the system design was complete, we then could compare these systems in order to choose the one that fit and enhanced the system design as much as possible.
With the system architecture and the technology to be used defined it would then be possible to develop the blockchain supply chain traceability system, conduct tests, and refine the system into a functional prototype.
After developing a functional prototype, we looked at how we could add more functionality through the use of machine learning. Research on previous implementations of blockchain and machine learning coupled together was performed. As well as the main issues present in nowadays supply chain ecosystems.
With a problem to be tackled, it was then possible to start looking into a dataset that could be used to represent a supply chain ecosystem, develop and train machine learning and statistical models on it and implement it into the initial prototype.

1.4 Main Objectives

The main objective of this dissertation is to study how blockchain technologies can be used to products through the supply chain, how they are processed and merged to produce a final product. Additionally, the result should be flexible and allow a wide range of information that can be collected to be stored as well, so that other future projects can easily integrate and take advantage of it. To show this after the main work is complete a simple demonstration application where it is possible to view the items being traced will be developed. Additionally, we will also look into integrating a machine learning module to take advantage of the collected information.

1.5 Main Contributions

The contribution of this thesis is the study and implementation of a blockchain-based supply chain management system that brings trust in the product origin to the consumer, providing supply chain-wide data collection, storage, and processing as well as a machine learning module to process that data into useful information.
The first contribution is a comprehensive review of the state-of-the-art of blockchain usage in private use cases, where we go over several blockchain solutions enumerating and

analyzing their main properties. We also compare the blockchain technologies against our use case, a data-heavy consortium style environment.
The second contribution is the proposal and implementation of a modular and containerized blockchain system featuring scripting and other modules that facilitate its setup and management processes, and a smart contract that coordinates and distributes data collection and processing as products are manufactured by the supply chain.
The third contribution is a pluggable machine learning module that makes use of the data stored in the blockchain allowing for the production patterns of the supply chain to be analyzed in order to minimize product transportation time.
Also, a paper which is the summary of this thesis has been submitted to an international conference.

1.6 Document Organization

The remainder of this thesis is organized as follows:

- Chapter 2 - Presents an analysis of several blockchain technologies looked at during the researching phase of the project, with their main characteristics and a comparison between them with regards to our use case.
- Chapter 3 - Discusses the development of the blockchain proof of concept and all the systems developed around it, how the blockchain technology was set up, the API's that work in conjunction with it, and the smart-contract developed.
- Chapter 4 - Discusses the development of the machine learning module developed to work in tandem with Certitex.
- Chapter 5 - Presentation of the final considerations of this thesis work and suggestion of possible research directions for future work.

Chapter 2

Blockchain Technologies Analysis

2.1 Introduction

In this chapter we will analyze a number of blockchain technologies, they will be analyzed in the following aspects:

1. Public or private: can anyone access a shared global network or does a private one have to be created and joined manually;
2. Permissioned: can everyone join any network and participate in it, or are they required to have proper permissions and roles;
3. Smart-Contract capability: is the technology capable of processing data, and not just store transactions;
4. Currency based: is the network-based in a built-in currency through which everything is centered around;
5. Consensus method: what or which consensus methods may be used within this network;
6. Date of Launch;
7. Time per block;
8. Block size.

2.2 Technology Analysis

2.2.1 Ethereum

Blockchain Technology	Ethereum
Public/Private	Both
Permissioned	No
Smart-Contracts	Yes
Currency Based	Yes
Consensus Mechanism	Proof Of Work (PoW)
Date of Launch	July 30, 2015
Open-Source	Yes
Time per block	About 15 seconds
Block size	20-30kb

In Ethereum [**?**, **?**] you can use the globally available and widely used public network or you can start your own from scratch.

The lack of a permission system is explained by the fact that Ethereum is, by design, meant to be run in its public mode and everyone should be equally able to participate in the network. The main focus of Ethereum is to have a fair, distributed, and public currency that can be traded for processing capacity within the network as well as for goods outside the network. However, if we wish to use Ethereum as a private network the lack of a permission system does not allow us to ensure who participates in the network. This in conjunction with the proof-of-work consensus makes it vulnerable to external attacks such as a sibyl and the 51% attack, where an attacker can gain influence disproportionate to the rest of the network.

Ethereum has smart contract capability. This is necessary in order to process data in a decentralized manner, as well as to establish agreements between two or more non-trusting parties without relying on a third-party (the entire network is the third party).

Ethereum is based on a currency named Ether (ETH). This currency is central to the network as everything either relates to or is paid by it. As such, to make a transaction in the network you transfer ETH to another entity. These transactions have fees that have to be paid to the network in order for it to validate the transaction. To execute a function of a smart-contract you must pay the network for the computer resources needed to process it. This presents advantages when we wish to mediate a public network by preventing transaction spamming or a smart-contract that leads to an infinite loop. To counteract such an infinite loop or over-processing of functions there is a cost limit on transactions. As such, a transaction is terminated once it exceeds the cost limit that was set. The cost of a transaction is calculated by adding the cost of the computer resources needed to perform a function and the fees needed to validate it.

The PoW consensus method is Byzantine Fault Tolerant (this means that the network is capable of continued function even if there are a certain number of malicious nodes on the system). Using this consensus method, all users try to validate blocks by attempting to solve a cryptographic puzzle. The first entity to solve it successfully publishes the block in exchange for a reward. This also means that all the miners of the network are, at the same time, solving a complex puzzle to validate the same set of transactions, resulting in redundancy of work and high computer resource consumption. As such, this method is characterized by high power consumption but a low rate of validated transactions over time [**?**].

Ethereum is open-source software, meaning that it can be modified and adapted to fit different needs.

The time per block and block size of Ethereum is pretty limited at the time of writing and, with the number of transactions happening every single day, this leads to high fees in order to add transactions to the ledger [**?**].

2.2.1.1 Example usage

Ethlance is a hiring platform for freelancers, people who have job offers can post them to *Ethlance*, candidates can then apply with their hourly price and the hirers then select a person to do the job for them, and when the job is done the hiree is paid. This exemplifies the usage of Ethereum perfectly. Both employers and employees are protected by the network because when a contract is made between an employer and an employee it is registered in the blockchain and visible to everyone, if any of the parties act in a dishonest way it is guaranteed that the original terms of the contract are not altered and that both employer and employee can make a case against the other party. Also, the low amount of data required to keep this application running, ensures that the fees are low.

2.2.1.2 Conclusions

Ethereum's public variant is an extremely secure way to host decentralized applications with low data requirements, in which neither of the involved parties trusts each other. Ethereum's private variant can be used to develop the backbone (storage and processing) of enterprise data in a highly decentralized environment. However, the lack of a permissioning system along with a competition-based consensus algorithm makes the blockchain inefficient for this type of usage. This is because in a private environment the proof-of-work consensus and the lack of permissioning allow for another entity with large processing capabilities to overpower the network (51% attack).

2.2.2 Quorum

Blockchain Technology	Quorum
Public/Private	Private
Permissioned	Yes
Smart-Contracts	Yes
Currency Based	No
Consensus Mechanism	Pluggable
Date of Launch	Nov 21, 2016
Open-Source	Yes
Time per block	50ms
Block size	configurable

Quorum is designed to be deployed as a private network. Quorum is a permissioned blockchain, participants must be authorized to join. Also, there is a concept of both private and public transactions. Public transactions are seen and validated by the entirety of the network while private transactions have a key associated with them and only nodes with this key can see the contents of the transaction. However, while private transactions are still present in the blockchain their contents are not, they are replaced by the hash of the encrypted payload.
Quorum has smart contract capability, based on the Ethereum Virtual Machine (EVM) of Ethereum. However, private smart contracts also exist and work similarly to private

transactions, they can only be seen and processed by the nodes with the corresponding key [**?**]. Quorum is not based on a currency. Quorum is distributed with the following consensus algorithms:

- RAFT
- Istanbul Byzantine Fault Tolerance (BFT)

Raft is a non-forking Crash Fault Tolerance (CFT), as long as more than half the network remains healthy.
Istanbul Byzantine Fault Tolerance is a non-forking CFT and BFT as long as there is a fully synchronized network this consensus algorithm can tolerate up to (N-1)/3 Faulty nodes where N is the total number of nodes in the network [**?**, **?**].
Quorum is open-source software, meaning that it can be modified and adapted to fit different needs.
Quorum's performance in terms of data processed per second varies according to consensus mechanism used and the initial configuration of the network but overall is a scalable network [**?**].

2.2.2.1 Conclusions

Quorum is a fork of the ethereum network, designed to be a consortium style network. From the available consensus algorithms, it is possible to make a network with CFT properties or CFT and BFT properties at the cost of block time, both the consensus algorithms used have the capability to scale into the thousands of transactions per second [**?**] if the network is well designed. The network structure (authorized nodes) can be changed at run time and the EVM is included in the project allowing for the implementation of smart-contracts which is useful to apply business logical rules.

2.2.3 IoTA

Blockchain Technology	IoTA
Public/Private	Both
Permissioned	No
Smart-Contracts	No
Currency Based	No
Consensus Mechanism	IoTA consensus
Date of Launch	March 5, 2019
Open-Source	Yes
Time per block	Not applicable
Block size	Not applicable

In IoTA [**?**], you can use the globally available and widely used public network or you can start your own from scratch.
Due to how consensus is achieved in IoTA's blockchain (the Tangle) there is no permissioning in the network, every node is capable of adding new transactions and every node must confirm other transactions to issue transactions.

There is no Smart-Contract capability in IoTA, however, this is planned to be added in the future. In the current state of IoTA, it is not possible due to the nonexistence of time-stamping within the tangle.
While IoTA does have a currency, it is not central to the network. Transactions can still be made without owning any currency. This is because transactions in IoTA are fee-less, you pay for your transaction by confirming other pending transactions.
The consensus mechanism used in IoTA is very different from other blockchain technologies. In it all nodes are required to verify two previous transactions to issue a new one, this new transaction is then connected to the previous two transactions.

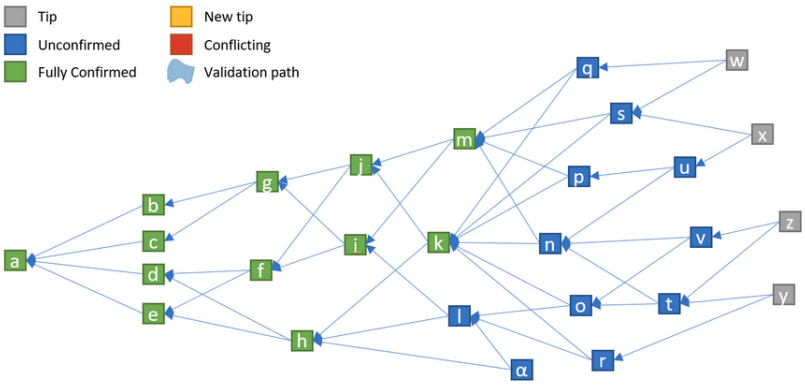

Figure 2.1: Structure of the Tangle [**?**]

A fully confirmed transaction must have connections to every tip.
While time per block and block size does not apply to this network, this network has good scaling capabilities. The more transactions being performed, the more transactions can be processed. This is achieved by the unique consensus mechanism used.

2.2.3.1 Conclusions

IoTA appears to be a capable and scalable network however, contradicting sources make it difficult to discern if some of the characteristics of IoTA are advantageous or disadvantageous [**?**, **?**].
Nevertheless, the lack of time-stamping capabilities along with a generalized disorder of transactions within the tangle and the incapacity of having smart-contracts, makes IoTA difficult to integrate into several fields. IoTA shows a lot of potential for Internet of Things (IoT) devices because of the fee-less transactions, along with fast confirmation rates. However, to add your transaction you must perform a PoW (in order to prevent spam on the network) and you also have to verify two other transactions. This involves verifying all their possible paths back to the genesis transaction, requiring large amounts of computational resources. Also, the IoTA network relies on a centralized system called the coordinator controlled by the IoTA Foundation itself diminishing the decentralized nature of the network. While these issues remain the use of the public network of IoTA is unadvised.
In a setting where there is a trustable central authority to host the central component

mentioned above, a private IoTA network can be advisable for IoT devices. The snapshot system allows periodical truncation of the tangle, which is advantageous for devices with small storage capacity. However, not all nodes in the network need to perform the snapshot so a full history can still be maintained.

2.2.4 HyperLedger Fabric

Blockchain Technology	Hyperledger Fabric
Public/Private	Private
Permissioned	Yes
Smart-Contracts	Yes
Currency Based	No
Consensus Mechanism	Pluggable
Date of Launch	July 11, 2017
Open-Source	Yes
Time per block	configurable
Block size	configurable

In HyperLedger Fabric (from now on referenced as Fabric) it is necessary to maintain your network. Fabric is not a trust-less system and there are nodes with more importance than others.
Fabric is equipped with a feature-full permissioning system. To access the network you must have the proper permissions and an X.509 certificate that identifies you. Besides selective participation there is a rank-based permission system that functions within the network, making certain actors in the network able to do things others can not, such as creating channels. Channels are different ledgers within the same network, Fabric makes use of this feature to allow private transactions within the same systems. For example, a company (ORG1) running Fabric might have a channel with (ORG2) through which decisions in regards to both companies, ORG1 and ORG2, are made. Moreover, ORG1 can have a channel with ORG3 through which decisions in regards to ORG1 and ORG3 are made, in a private manner so that it is not possible for ORG2 to know what happens in this second channel.
Fabric allows for smart-contract capabilities which are called chaincode. Chaincode is instantiated on each peer individually and has a list of endorsers (interested parties of a certain contract). When chaincode is called these endorsers run the code and agree on a final list of transactions. These agreements can be reached in several manners such as majority endorsers agreement.
Fabric does not have a currency system.
Consensus in the Fabric environment is agreed upon on the creation of the network, and it can be a Kafka or Role, Audience, Format and Topic. (RAFT). In both cases the consensus is only CFT and not BFT. This means that the network will continue to operate if at least the majority of nodes continue running. However, the system is vulnerable to malicious nodes that intentionally attempt to cause disagreement on the network.
This technology is open-source and is built in a plug-and-play manner. There are several systems and even versions that may be chosen to fit the needs of a certain network.

Both time per block and block size is configurable. As this network relies on the fact that every node can be trusted to some degree due to having to provide its identity, it has fast confirmation rates.

2.2.4.1 Example usage

TradeLens by IBM is a system built on top of Hyperledger Fabric. The goal is to unite the ecosystem involved in the transportation of goods into a single platform where data may be shared and later audited. The documents involved are standardized and, when issued, they are stored in a cloud and the hash of the document is stored in the blockchain. This guarantees that it cannot be altered without destroying the connection to the blockchain. While not guaranteeing immutability, this adds a layer of protection. For example, if a company alters a document the hash of that document and the hash stored in the blockchain is no longer the same. Therefore, the company can be held responsible for altering the data.

2.2.4.2 Conclusions

HyperLedger Fabric is an example of the capabilities of a permissioned network. The features brought by it over traditional database systems are data immutability, decentralization, and replication. Offering a scalable blockchain system with high throughput due to the lightweight CFT only algorithms and ensuring the health of the system by requiring well-identified users Fabric. However, it is a novel technology with few documentation resources available for the developers and few systems built on top of it.

2.2.5 HyperLedger Sawtooth

Blockchain Technology	HyperLedger Sawtooth
Public/Private	Both
Permissioned	Yes
Smart-Contracts	Yes
Currency Based	No
Consensus Mechanism	Pluggable
Date of Launch	January 30, 2018
Open-Source	Yes
Time per block	configurable
Block size	configurable

In HyperLedger Sawtooth (from now on referenced as Sawtooth), you can design your networks in both a private and public manner. As there is no built-in incentive system (such as Ethereum that awards miners with ETH), additional procedures are needed to implement a public network with Sawtooth.
In Sawtooth, there are several types of nodes:

- Transactors: nodes who can submit transactions to the network;

- Validators: nodes who are allowed to establish a connection to the validator network and decide on changes to the state of the network.

Sawtooth applies both on-chain and installed smart-contracts. On-chain contracts are similar to Ethereum smart-contracts. There is little information available on installed smart-contracts on Sawtooth's documentation, resulting in the authors being unable to explain them at the time. As such, they will not be considered.
Sawtooth features a pluggable consensus mechanism that can be changed after the network has started. This means that the consensus mechanism can be chosen and altered based on what fits the network at a certain time, without requiring a hard-fork. Sawtooth already features some implemented consensus mechanisms:

- Proof of elapsed time CFT: consensus mechanism that offers crash fault tolerance and scales well.

- Proof of elapsed time SGX: consensus mechanism that offers both crash fault tolerance and byzantine fault tolerance, however, it requires specific intel processors that have SGX capabilities in order to be used.

As with Fabric, the time per block and block size of Sawtooth is configurable, and, as such, confirmation times may vary. However, and similarly to Fabric, if the right consensus mechanism is being used in a private network environment, where validator nodes are trusted, the network scales well. Additionally this new type of consensus algorithms called Proof-of-Elapsed-Time promise to offer Proof-of-Work security level while maintaining the overhead of most crash fault-tolerant consensus like RAFT. The concept behind this innovative idea is to use a protected execution environment present in a certain set of Intel processors called Software Guard Extensions. With it, HyperLedger Sawtooth should be able to run protected code that the owner of the machine can not tamper with, which it can use to fairly determine the leader for each proposing round in a RAFT-like consensus system. While HyperLedger Sawtooth appears to offer all the security of a byzantine fault-tolerant consensus at the scalability of a crash fault-tolerant one, this comes with some issues. Firstly the need for all validating nodes on the system to have a particular processing unit. Additionally, there have been studies regarding the technology that point towards some security flaws [**?**]

2.2.5.1 Conclusion

Much like HyperLedger Fabric, Sawtooth has the features of a capable network for both a public and private environment, providing high throughput, smart-contract capability, and a more comprehensive selection of consensus algorithms this time including both CFT and BFT. However much life HyperLedger Fabric it is a novel technology lacking in terms of documentation and resources for the developers and some issues that need to be ironed out with the innovative consensus mechanisms this technology is trying to feature.

2.3 Textile Industry Supply Chain

Fundamental properties of our supply chain environment:

- Network of known and identifiable nodes;
- Average network size that may scale after system deployment;
- High flow of transactions (a large number of clothing items are being produced);
- Data integrity is a top priority and the main focus of the system;
- System should work as quickly as the production of items, and it can not be a hindrance to current production speeds;

2.3.1 Ethereum

Ethereum consensus (PoW) has some blatant and well-documented scalability issues in terms of power consumption and the possibility of network forks which are undesirable since every transaction is vital to keep the history of an item [**?**]. Also, the network to work in sync, block speeds are somewhat fixed (in the case of Ethereum it is about 15 seconds per block), this means that our network is unable to adapt to different workloads.

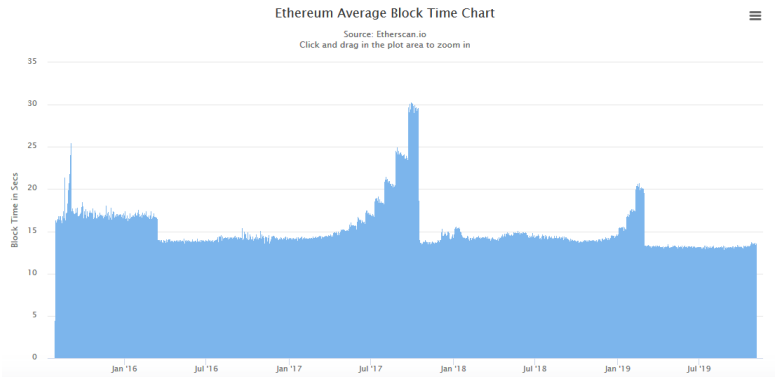

Figure 2.2: Ethereum average block time chart [**?**]

Also, the gas limit per block of Ethereum leads to blocks having a maximum size of around 30Kb.
The network can transact about 2Kb of information per second.
In the private side of Ethereum, this number can be changed but nodes will always be competing with each other to validate blocks. This consensus works in public settings where nodes are neither identified nor trusted, and the access to the network is public. However, in our use case, we have a known and identifiable set of nodes and, as such, we can make use of less demanding consensus algorithms.

2.3.1.1 Ethereum Public Analysis

Ethereum's public network advantages:

- Highly reliable and used system;
- Launched and supported for several years.

Ethereum's public network disadvantages:

- Low transaction processing speeds and concurrent users trying to transact with the system leads to unusable speeds;
- High fee's transaction fees.

2.3.1.2 Ethereum Private Analysis

Ethereum's private network advantages:

- A private network means that only transactions relevant to the system are being processed;
- Capacity to process a larger amount of data than Ethereum public.

Ethereum's private network disadvantages:

- Does not make use of the fact that the nodes in the system are identifiable and are working towards the same goal, leading to less efficiency;
- There is no system in place to limit access of outside nodes to the network.

2.3.2 Quorum

Quorum brings many of the qualities of Ethereum into a private setting. The EVM can be used to run truly decentralized and complex smart-contracts that are audited by the entirety of the network. Also, the IBFT consensus protocol allows users to connect directly to the blockchain without being able to influence the network, to attest information themselves, without relying on a centralized system. In addition, several performance tests [**?**] indicate that Quorum is capable of processing a large number of transactions per unit of time. Quorum advantages:

- Based on Ethereum which is a proven network;
- Private and permissioned network fits our use case and offers performance through a less demanding consensus algorithm;
- No currency;
- Transaction finality;
- Scalable.

Quorum disadvantages:

- No inherent node identification.

2.3.3 IOTA

In IoTA, the way of achieving consensus means that transactions are not processed in blocks, instead, they are added to the network one by one and reference two other previous transactions [**?**]. This means that for every single transaction there is overhead, while in other blockchain systems transactions are arranged into blocks significantly reducing storage used on headers. Also block finality is probabilistic. However, it is worth mentioning that the fee-less transactions and the consensus mechanism make this system very interesting for other use cases.

2.3.3.1 IOTA Public Analysis

IOTA public network advantages:

- Fee-less transactions;
- Fast confirmation times for transactions in a public network;
- Consensus method that scales very well in terms of number of transactions.

IOTA public network disadvantages:

- Relatively new system without too much maturity;
- Huge network size growth rate;
- Incapability to process data in a decentralized manner;
- Currently reliant on a central system managed by IOTA. (security risk).

2.3.3.2 IOTA Private Analysis

IOTA private network advantages:

- A private network means that only transactions relevant to the system are being processed;
- Consensus method that scales very well in terms of the number of transactions.

IOTA private network disadvantages:

- Does not make use of the fact that the nodes in the system are identifiable and are working towards the same goal, leading to less efficiency;
- There is no system in place to limit access of outside nodes to the network;
- Even with the network only processing transactions related to itself, the growth rate of the blockchain in size is still prohibitive.

2.3.4 Hyperledger Fabric

Hyperledger Fabric is a private network solution that relies on a set of trustable nodes and it achieves consensus through a pluggable consensus method that is chosen upon network setup, either Kafka or RAFT.

Kafka ordering [**?**] Relies on a central ensemble called ZooKeeper that is a distributed key-store. It receives requests from applications and forwards them to the appropriate Kafka cluster.

ZooKeeper is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services. Kafka clusters are constituted by peers and orderers:
Orderers: Nodes are responsible for keeping the immutability of the order of transactions to be applied to the blockchain. They do not validate transactions, they just group transactions in blocks of adjustable size. In a network, there is a delegated leader that is responsible for setting the final order of all transactions in any given block, and followers that receive and relay the information to the peers and, in case of leader failure, replace the leader.
Peers: Peers receive transaction proposals from client applications, execute and verify (or endorse) them. They respond to transaction proposals with the original transaction, a response, and their endorsement for that response. Transactions are considered valid if all the peers required by the transaction policy endorse it.
It is notoriously difficult to implement this system as pointed out in Fabric's documentation. Also, this system is not fully decentralized due to the ZooKeeper ensemble.

RAFT Ordering works on a leader-follower basis. For every channel in the network, there is a leader node that upon receiving a transaction replicates it to the followers. If at least half of the ordering nodes on the network respond to the leader the change is considered committed and will be included in the next block. In the case of leader failure, the ordering nodes vote amongst themselves to elect a new leader to keep the ordering service running [**?**].

The Raft ordering service was implemented later into the development cycle of Fabric and is to be the step forward from Kafka [**?**, **?**]. It allows for a fully decentralized network and is the first step towards having a byzantine fault tolerance consensus on the Fabric network. However, not every use case requires BFT properties and, in such use cases, the amount of computational work involved in BFT consensus makes a negative impact on the network's performance. RAFT also massively reduces the number of healthy nodes necessary for the network to continue operating, requiring only 2n+1 nodes to remain healthy, where n represents the number of failing nodes. In our use case, BFT is unnecessary as we have a set of organizations that are working towards the same goal and the nodes in the network are well identified. As such, in case an organization attempts to harm the network, the attack can be pinpointed back.

With the previous analysis in mind, the usage of RAFT best suits our use case.

2.3.4.1 Hyperledger Fabric Analysis

Hyperledger Fabric advantages:

- Private and permissioned network fits our use case and offers performance through a less demanding consensus algorithm;
- The highly modular design of Fabric allows the choice of features needed for our use case;
- The capacity to manage several ledgers with the same network, in case we need to expand the system.

Hyperledger Fabric disadvantages:

- While it is ready for production this relatively new platform is still being heavily updated;
- Impossible to make a public network, as every participant in the network must be registered and well-identified.

2.3.5 Hyperledger Sawtooth

Hyperledger Sawtooth is a blockchain solution designed to run in both private and public environments. The consensus mechanism is pluggable and, by default, is distributed with the following options

- Proof of elapsed time SGX;
- Proof of elapsed time;
- Pratical byzantine fault tolerance.

Proof of elapsed time SGX relies on recent development in Intel processors, as it uses a protected run-time environment in order to run code. In PoET consensus, when there is a block to be added to the blockchain, all validators sleep for a random amount of time. The first one to resume can publish the block. The problem is in guaranteeing that every validator has slept for a random amount of time, this is what the Intel SGX feature attempts to resolve. This would not only make for a very performance-oriented network but also give it CFT and BFT properties [**?**]. However, some sources illustrate issues with the SGX implementation. Briefly, an attacker would only need a small fraction of nodes under his control in order to control the network [**?**]
The other Proof-of-Elapsed-Time consensus mechanism relies on a simulated environment that does not guarantee that it is secure, giving it only CFT properties, making it work well in our use case.
Practical byzantine fault tolerance is a consensus designed for small consortium networks, when a transaction is issued the currently leading node broadcasts it to all other nodes,

the nodes process the transaction and reply to the client, the client must wait for at least f+1 replies where f represents the maximum number of faulty nodes in the network (f = T/3 where T is the total number of nodes) [**?**]. In PBFT the number of messages in transit scales exponentially with the number of nodes of the network, as the network size may be large this consensus will not be used [**?**].

2.3.5.1 Hyperledger Sawtooth Analysis

Hyperledger Sawtooth advantages:

- Private and permissioned network fits our use case and offers performance through a less demanding consensus algorithm;
- Consensus can be changed at any point, even while the network is running, so in case a new and better fitting consensus mechanism is developed it can easily be implemented to the deployed system;

Hyperledger Sawtooth disadvantages:

- Relatively new platform (newer than Fabric);
- Lack of previous usages of this platform on similar use cases;
- Official documentation is far less complete than Fabric.

2.4 Conclusion

After gathering information about several blockchain systems and establishing the requirements for our use case it was still hard to commit to a particular technology. It was clear that a private network would benefit the design however there were still many choices with different features. HyperLedger Sawtooth and Iota were discarded due to the existence of security issues with their innovative consensus algorithms [**?**, **?**]. HyperLedger Fabric was discarded due to the lack of developer resources when compared to Ethereum and its child system Quorum and featuring only CFT consensus mechanisms which would limit future design choices. Leaving us with an Ethereum private network and Quorum, of which, Quorum was the final choice due to the available consensus algorithms which made better use of the properties of our environment and would allow the system to scale better in the future.

Chapter 3

Certitex Blockchain

This chapter has the main purpose of explaining how the *Certitex blockchain* functions. Certitex uses *Quorum*'s *blockchain*, which is based of Ethereum and additional systems in order to automate certain aspects of the *blockchain* set-up, node connection establishment and *blockchain* interaction.
First, we introduce the *Quorum blockchain*, its properties and components and main modifications to the *Ethereum blockchain*. With this introduction, it is possible to present *Certitex* mainly the modifications applied to *Quorum* and systems developed around it. In this chapter, finally, we will also discuss smart-contract technology and how they were implemented in *Certitex*.
The structure of this chapter is as follows: section 3.1 briefly introduces the main technologies used in the system. Section 3.2 presents a detailed explanation of the *Quorum blockchain*, its software and the main features that made the author choose it as the most fitting *blockchain* technology for the system at hand. Section 3.3 explains *Certitex* in depth namely the configuration changes to *Quorum* and the systems developed around it. Finally, section 3.4, explains the main decisions made with regards to smart-contract implementation in *Certitex*.

3.1 Used Technologies

Docker is an OS-level virtualization software used to deliver software in self-sustained packages called containers. These containers are isolated bundles of their own software, configuration files, and libraries; They are also capable of communications through well-defined channels. Thus they can run on any *linux* machine provided it can install docker. Docker-compose is a tool for defining and running multi-container Docker applications and easily running and stopping them. With the usage of docker, docker-compose, and shell scripting we can simplify system deployment to the press of a single button, which is an important feature for this type of distributed system where the end-user may not have the required technological know-how. Additionally, easier system redeployment allows for a more streamlined developing and testing process.
Istanbul-tools is a software designed to easily generate the required configuration files, given a set of arguments, to set up a correctly configured set of *Quorum* nodes with *IBFT* consensus. It is used to facilitate the generation of configuration files through the multiple restarts required during the process of development and debugging. The quorum node is designed to be a lightweight modified fork of geth, some aspects such as consensus mechanisms, the P2P layer, gas pricing among others have been modified in order to cater to *Quorum's* design. *Quorum* offers two different consensus mechanisms RAFT and

IBFT, much more suited to a private environment, in turn, the P2P layer in *Ethereum* was modified to function with either of these consensus mechanisms, gas pricing has been removed from quorum, *Quorum* has also introduced private transactions, which are transactions that can only be seen by a subset of nodes in the network but are still disseminated throughout the entire network and made private with the usage of cryptography.
Flask is a python library used to prototype web applications, it is used to develop the APIs required to operate and coordinate the deployed *Certitex* containers.

3.2 Quorum Blockchain

In *Quorum*'s white-paper[**?**], JPMorgan Chase co. explains in detail that *Quorum* is a fork from the *Ethereum blockchain* and the main features added to it. Also it is stated that *Quorum* is updated in-line with *Ethereum* releases.
From the previous chapter, the technologies that best fit our use case were Hyperledger Sawtooth, *Hyperledger Fabric*, and *Quorum*, all of these technologies offer *blockchain* accessibility only to a set of known nodes (a private network), and are capable of smart-contracts. The only other constraint is high transaction throughput, which *Hyperledger Fabric* [**?**] and *Quorum* [**?**] are capable of offering. However, *Quorum* has some other features that allow it to stand out above *Hyperledger Fabric* for this particular use case. It is capable of using a Byzantine Fault Tolerant consensus, that unlike Proof of Work does not require solving complex cryptographic puzzles but is instead based in a democratic voting system, which is ideal for smaller supply chain environments because the main constraint in it is the number of voting entities. Additionally, it offers a Crash Fault Tolerant consensus *RAFT* that can be used for larger supply chain environments. IBFT consensus offers the *blockchain* capability to continue operating even in the presence of malicious nodes (Byzantine Fault Tolerance), it can tolerate up to (N-1)/3 faulty nodes, where N represents the number of nodes in the system. On the other hand, *RAFT* consensus does not tolerate malicious nodes but only faulty ones (nodes that have gone offline). While less resilient than IBFT it offers a larger transaction throughput, which may be necessary for large supply chain environments. As such, *Quorum* can adapt to the environment in question if necessary. Additionally, as *Quorum* is a fork from the *Ethereum blockchain* which is a largely established network, there is more accurate information as well as more development effort already in place, with that, it is possible to shorten developing time and ease debugging.
The main goal of *Quorum* is to be a private and permissioned version of *Ethereum* that features stronger privacy solutions as well as boasting more performance. *Quorum* is a completely private network where only a set of known and identifiable nodes are capable of participating. For a node to participate in the network it must be allowed to do so by a majority of the existing allowed nodes, this is achieved locally on a per-node basis where each node has recorded every other that it trusts and will communicate with. Additionally, a set of validator nodes are defined in the genesis file, these are the nodes who are permissioned to validate blocks in the chain. This list of validator nodes can be expanded

and reduced by majority voting of all validator nodes. However, this is not registered on the *blockchain* itself but locally within each node. It is part of *Quorum*s road-map to have on-chain validator node voting and that would allow all participants of the network, both initial and later ones to inspect the set of validator nodes overtime and the votings each note partook in, this information, just like all other information contained within the *blockchain* could not be tampered with.

```
- Support different role-types: Read vs Write nodes/accounts
- Smart Contract-based network permissions
- Authenticated / Protected RPC API access
- Consensus Node Whitelisting
```

Figure 3.1: Quorum road-map [**?**]

The performance increase of *Quorum*s stems mainly from the different approach to consensus mechanisms, in a private and permissioned environment, we can make use of the IBFT consensus. It allows, for larger performance differences in a setting where there are smaller numbers of validators. This consensus mechanism would not be efficient in a public network where all participants would have to be validators. This is due to how IBFT works. In a small consortium setting, where the number of validators is reduced, IBFT works by the set of validators multi-casting messages to each other in three phases, known as *new-round, pre-prepare, prepare* and *commit* over a variable number of rounds. Each round starts by a proposer node constructing a block and broadcasting it to the set of validator nodes (*new-round* phase). Upon receipt of a proposed block validators attest to the validity of that block, if it is valid nodes emit a ”prepare” message (*pre-prepare* phase). Upon reception of majority ”pre-prepare” messages nodes emit the ”commit” message (*prepare* phase). And like the previous step, upon reception of majority ”commit” messages, the block is added to the present validators local *blockchains* and distributed through the P2P gossip mechanism present in *Ethereum*. Each block acceptance message is accompanied by a seal generated by the corresponding validator using public-key cryptography, these seals are added to the final block and can then be audited, guaranteeing at any point in time that a quorum for that block has been reached, ensuring the properties of the PoW consensus without the costly process of validation. This however does not scale as well with the number of validators, each validator must receive and multi-cast messages three times to all others. This exponential growth of messages quickly loses scalability when compared to PoW.

Finally, *Quorum* expands upon *Ethereum* by adding the concept of private transactions, these are transactions that only affect the private state database of nodes who are allowed to see this transaction. While this system could be useful for our use case in very large supply chain environments this was not part of the scope of this project, but can be explored as part of future work.

3.3 Certitex Blockchain

In order to ease system development, deployment, and performance assessment we made use of docker to containerize the application in an environment where all the dependencies are installed in their correct version. These can be incrementally updated after testing the system again with their updated versions, making sure that *Certitex* does not stop functioning through external action. Currently, each node is made up of three different inter-connected docker instances with docker-compose. The *Quorum* node instance, a master API, and a slave API. Each of these fulfills a specific role that can be logically separated, as such, if an issue arrives we can easily pin-point it to a docker instance, lowering the workload of debugging. The *Quorum* node instance is the main component of the *Quorum blockchain* network and it handles everything related to *blockchain* node-to-node communication, connection establishment, *blockchain* synchronization, block validation, block proposing, consensus among other *blockchain* related operations. The master API instance is tasked with the addition and removal of nodes to the authorized lists of the *Quorum* node instance as well as updating the IPs of nodes already existent in the network and automated voting of added nodes as validators through RPC calls to the *Quorum* node instance. Whenever one of these APIs in the network receives an update it broadcasts it to every other master API and applies it, the other master APIs, in turn, re-broadcast the information and applies it itself, giving the system resilience to communication failures through redundancy, re-broadcasting will stop eventually. Once an API applies an update to a particular node the time stamp of that update is recorded with that node. Whenever an update request is received, if the time stamp present on it is smaller or equal to that node's last update timestamp the update request is discarded and is not rebroadcasted, these updates are applied in reverse order of reception in an attempt to process the most recent update only, thus discarding outdated updates increasing performance. This guarantees that every *Quorum* node has an updated list of nodes it is allowed to communicate with. Even though the APIs are a fully connected network, even if the communications are blocked temporarily the update will be applied as long as the not fully connected API has a path of connection to all other APIs no matter the depth, as shown in figure 3.2.
Additionally, if a master API node is temporarily disabled (there is no possible path to communicate to it) it will eventually receive the update when this connection is re-established because APIs will record the transmission failure event and re-attempt it in the future. The entirety of this process is orchestrated by threads, each node has a thread dedicated to applying modifications to the permissioned list of *Quorum*, a thread for each other known node tasked with making requests to that specific node, and a logging thread that records all events to a file on disk for easier auditing.
The slave API or communication API is in charge of providing an interface for interacting with the smart-contract, deploying the smart-contracts, and maintaining the link to the smart-contract through restarts of the system. There are two minor variations of this API, one of these variations is run by the first node to be initialized in the system or the node generated by *god.sh*, if it has no recording of a deployed smart-contract it deploys it and stores its address. The other variables are present in *second-god.sh, third-god.sh* and

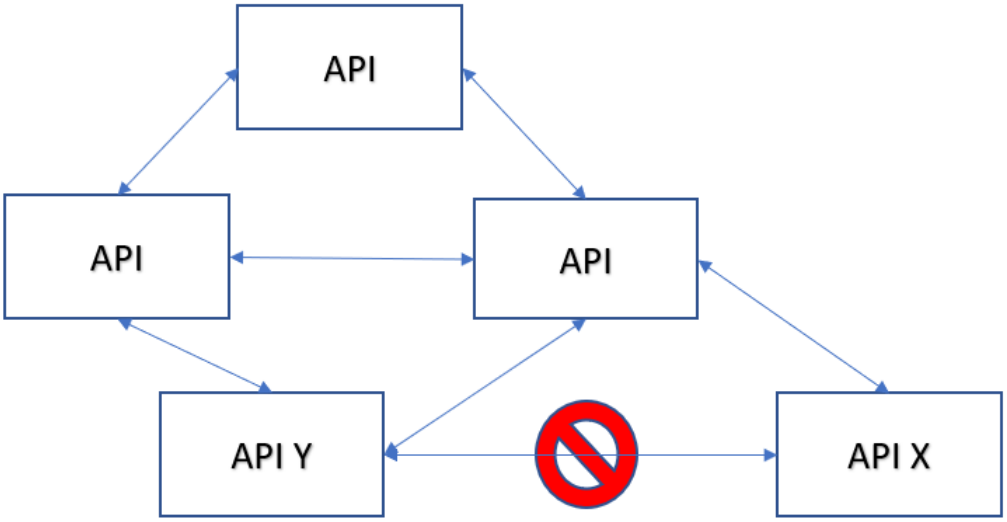

Figure 3.2: API X will eventually receive an update submitted to API Y due to re-broadcasting

prophet.sh in case it does not possess the address of the smart-contract it requests it from the communication API provided on its initialization parameters, and once it obtains it stores it locally in case the system restarts in the future. Once the API is finished setting up it simply waits for requests to arrive through URL, processes the requests checking parameter correctness, if everything is correct it builds a transaction and submits it to the *Quorum* node instance.
In order to automate the process of deploying a node, bash scripting was employed, in a *Quorum* network running the IBFT consensus (our network) it is required that the network starts with at least 3 nodes, this is the main reason for the existence of the four different scripts for setting up a node. *god.sh* generates a network with the minimal (three) number of authorized nodes, and generates *second-god.sh* and *third-god.sh* instantiation files. These later files must be moved to the computers that will be hosting the second and third nodes of the system, and once deployed we will have a functioning *Certitex* blockchain running the IBFT consensus. Lastly *god.sh* also generates the instantiation files for *prophet.sh*. *Prophet.sh* is used to deploy all other nodes after the first three. As such, with this installation setup we allow for a network to easily be set up over time despite the constraints present in *Quorum*, with the only constraint being, the system is only online after the deployment of all *god* nodes.
There are three types of installation scripts, the first one, *god.sh*, receives five parameters:

- RPC port: Remote Procedure Call port - this is the port through which procedures to the *Quorum* instance may be called remotely;
- Geth port: This port is the main *Quorum* instance port - it's the port through which all nodes will communicate to sync blocks, validate blocks, propose blocks, etc, all *blockchain* related communication happens here;
- Master API port: This port is the port that will be used by the master API for its

communications;

- Slave/Communications API port: this port will be used by the communications API for all its communications;
- Name: A string that will be used by docker to name the 3 instances of this node for easier identification.

These are the ports on the main OS, not the docker instance. let's say that the chosen RPC port is *8500* when there is a connection to the host machine in this port, the host machine knows that it needs to map it to the docker *Quorum* instance on port *8545*, *8545* is the value that all RPC ports are mapped to within docker so the mapping for this particular case would be *8500* on the host machine, would map to the docker *Quorum* instance on port *8545*, this same process happens to all other ports and can be seen in the figure 3.3.

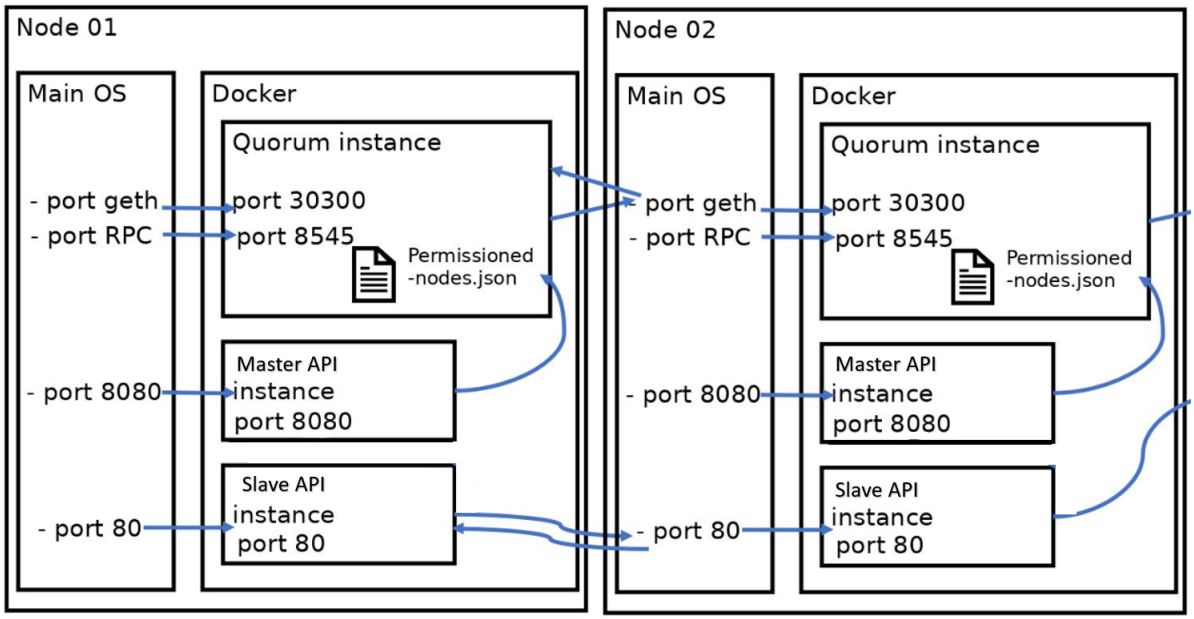

Figure 3.3: System communications architecture between two nodes.

With all the provided parameters, *god.sh* can instantiate this first node through the following steps.

1. Checking for previous installation folders of itself, and deletes them if it finds any.
2. Adds rules to the system IPtables to allow communication through all ports received as parameters
3. Runs *instanbul-tools* for 3 nodes, thus generating:
    - a genesis.json file that needs to be used by every node that joins the network;
    - 3 folders, each containing a nodekey, these folders are named "0", "1", and "2" the contents of folder "0" must be used by node 0, the contents of folder "1" must be used by node 1, etc;

- a static-nodes.json file, that is a list of all nodes each node is present in the following format: ”*Ethereum*NodeId”@”ipaddress”:”port”?discport=0. *Ethereum* node ID directly relates to each aforementioned nodekey, and additionally the IP address and ports are merely placeholders that must be updated later (0.0.0.0:30300).

4. Edits the genesis.json file to a different chainID, this step is not necessary however it may avoid collisions with other *Quorum* networks running on the default chainID.

5. Edits the static-nodes.json file to update it's own entry, correcting the IP address and ports with its own.

6. copies into a temporary folder all the files required for docker to initialize the node: genesis.json, static-nodes.json, nodekey, Master API's executable, the docker-compose file, etc.

7. It populates the other installations, the Second and Third god's (thus allowing the 3 node network to be fully set up) as well as the Prophet installation folder that allows the network to be expanded on demand. For all installation folders, it provides them the edited genesis.json, the edited static-nodes.json, and specifically for the *second-god* and *third-god* their respective nodekey.

8. Finally it runs the docker-compose file. Once this is complete the node is running.

The second type of bash script is the Second and Third *god's*. They receive the same parameters as the first *god* (for their own set up), and the IP address and port of another node that is already running the network (in the case of the Second *god* it must be the First *god's* IP and port, in the case of the Third *god* it can be either the First or the Second *god's*). Both these scripts execute exactly the same as the first one however steps 3,4 and 7 are skipped. and before it executes the docker-compose it produces a request to the provided node's Master API to update its entry with the correct IP and ports, thus, making all online nodes update their lists to include it. Note that this is not an ”addition” request but rather an ”edition” request, since this node already exists in static-nodes.json from Istanbul-tools it only has incorrect values for IP and Port.
The third and final type of bash script is the *prophet* script, this script adds a new node to the network with the validator status, it receives the same parameters are the Second and Third *god's*:

1. Adds rules to the system's IPtables to allow communication through all ports received as parameters

2. runs Istanbul-tools for one node

3. copies into a temporary folder all the files required for docker to initialize the node: genesis.json, static-nodes.json, nodekey, Master API's executable, the docker-compose file, etc.

4. edits static-nodes.json to have it's IP and Port corrected.

5. Makes a request to the provided node's Master API to be added to the network and voted as a validator.
6. Runs the docker-compose file. Once this is complete the node is running.

Note that after step 5, all nodes in the network now have their list updated to contain this new node, additionally, whenever the Master API receives an addition request (which all nodes will due to the gossip mechanic) they also make a request to the node they are adding to add them and broadcast it to the network. This whole process ensures that eventually, as long as no node is completely isolated from the network all nodes will have a consistent list containing all nodes in the network including the new node currently being instantiated.
The docker-compose file is responsible for instantiating the three components of the *Certitex* node, it starts by instantiating the *Quorum* node, to do this it begins by creating the usual folder structure of a geth node.
mkdir -p setup/data/geth

Afterwards, it copies the files provided by it's creator ".sh" file
cp /datastore/0/nodekey setup/data/geth/nodekey
cp /datastore/static-nodes.json setup/data/permissioned-nodes.json
cp /datastore/static-nodes.json setup/data/static-nodes.json
cp /datastore/genesis.json setup/genesis.json

It initiates the data directory with the provided genesis files and begins running as a node.
geth –datadir setup/data init setup/genesis.json
nohup geth –datadir setup/data
–permissioned
–verbosity 5
–networkid 1337
–rpc
–nodiscover
–rpcaddr 0.0.0.0
–rpcport 8545
–rpcapi admin,db,eth,debug,miner,net,shh,txpool,personal,web3,istanbul,Quorum
–emitcheckpoints –istanbul.blockperiod 1 –mine –minerthreads 1 –syncmode full
–port 30300

Additionally, it maps out main OS ports to a fixed corresponding port within docker.
ports: - $PORT_RPC: 8545
- $PORT_GETH: 30300

The remaining two docker images follow a similar process, map the provided ports, install dependencies, run the python web-app. All 3 images for the node are connected to the

same data volume, meaning they operate in the same directories as each other, this is how the master API is capable of updating the lists used by the *Quorum* node.

3.4 Smart-Contract Implementation

A very large concern on this project is storage space. This concern is reflected in our implementation choices when it comes to smart-contracts. Initially, the idea was to have one contract per item, as with this approach implementation would be simpler and ownership could easily be fine-grained. However, through testing, we quickly realized that deploying smart-contracts is costly storage-wise. As such, our aim was directed at having a single smart contract governing the entirety of the network.

This smart contract will define what an item is within our system, what operations can be performed on it, and dictate the conditions that must be met for a certain operation to be performed. This allows for complex rules and restraints to be applied. In our proof-of-concept, the smart contract built is a simplified skeleton to demonstrate functionality. In a production environment, a more developed version that pays attention to the specifics of the given supply chain should be implemented.

An item within the system is represented by a set of fields that can be seen in Table 3.1:

Table 3.1: Structure of the data representation of each product in the system, with detailed information on composing fields.

Name	Variable Type	Description
ProductID	Integer	A unique identifier of the product linking it to it's real world counterpart
ProductType	String	A short descriptor of the product (Shirt for example)
ListHolderID	Ethereum Addresses List	A list of *Ethereum* that identify the holders of the product (in order) as it travels through the supply chain
ListMessages	String List	A list of messages corresponding one to one with ListHolderID for any notices related to the transferring of the product
Exists	Boolean	A boolean dictating whether or not the item is still being tracked in the system (allows for soft deletion of items)
Quantity	Integer	An integer counting how many products are in this entry
isBulk	Boolean	A boolean dictating whether or not the item is being bulk traced
ListComponentID	List of Integer	A list of product ID's that were components to the creation of this entry enabling tracing through product transformations

The smart contract is also able of performing operations on the items, the list of operations can be seen in Table 3.2:

Table 3.2: Supported operations of the system, along with their purpose and functionality.

Name	Functionality
Create Entry	Creates a product entry with initial holder as the account that issued the transaction, when executed through a slave **API!** (**API!**) (as it should) the holder will be the node that is hosting the slave **API!** that is being used.
Transfer Entry	Transfers a product entry to another holder, appending to its holder's list and messages list.
Merge Entries to new non-bulk Entry	Merges one or more entries into a new single traced product. [1]
Merge Entries to new bulk Entry	Merges one or more entries into a new bulk traced product. [1]
Merge Entries to existing bulk entry	Merges one or more entries into an existing bulk traced product. [1]
Merge bulk entries to singular	Merges two bulk traced entries into one
Search for entry	Searches for an entry in the list and returns it

[1] If entries being used are bulk traced, a quantity must also be provided, entries fully used during this process are marked as deleted or have their quantities reduced.

Entry searching makes use of dictionaries to find the correct index within the full list of entries, making its complexity O(1).
Safeguards have been implemented regarding item merging, ensuring that the items that are being merged are in sufficient quantity and that they still exist within the system. When an item is fully consumed during the merging process (quantity reduced to zero) it is soft deleted, by making its Exists flag false.

3.5 Certitex Performance Testing

In this section, the experiments performed in order to gauge the performance and feasibility of the system will be presented and discussed.
To assess the performance of the system, three main metrics must be considered: how many transaction requests should be sent to the system, how many transaction requests are actually sent to the system, and how many of these transaction requests are validated. Regarding the first metric mentioned, it is expected that a perfect system can receive and validate any number of transaction requests received. This is difficult to implement in a real system as several constraints limit performance, such as network costs and available computational resources. As such, the performance of the system in a real use case is limited and the limits of the system must be assessed in order to compensate for possible constraints.
The testbed used in these tests consists of four machines with the same deployment of the system in a local network, connected by a 1 Gbps switch. Each of these machines acts as both a peer and a client in the system and is equipped with a CPU with two cores at 4.2 GHz and 16 Gb of RAM, running on Debian GNU/Linux 9.11 (stretch) in x86_64 architecture with kernel 4.9.0-8-amd64.
As can be seen in the results of the experiments in figure 3.4, the testbed is able to sup-

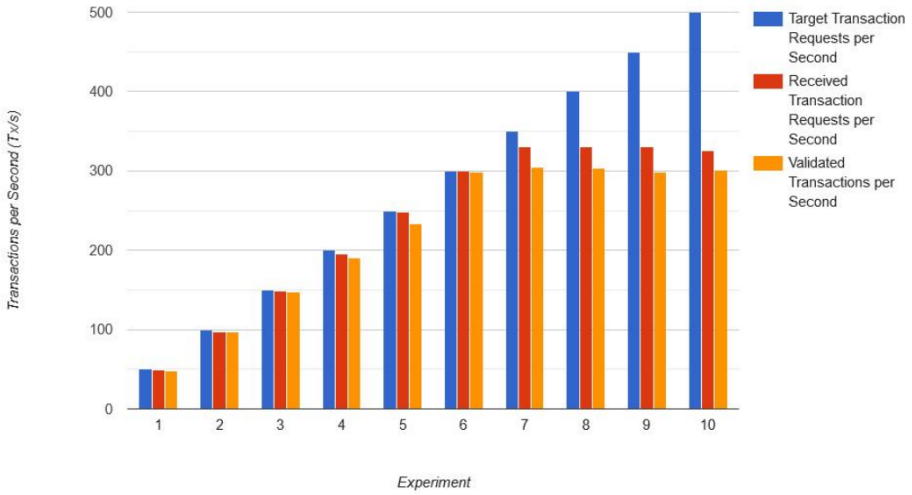

Figure 3.4: Performance graph of the capabilities of handling write transactions by the blockchain system. The blue bars represents the target sent transaction requests per second. The red bars represents the number of transactions that are being requested to be inserted on the blockchain system per second. The orange bars represents the number of transactions that are inserted into the blockchain system, and therefore, validated.

port generating approximately 315 transactions to be validated by the system per second and supports validating approximately 300 transactions per second. The main constrain appears to be on the testbed itself, as it is not capable of generating loads that surpass the system capacity by a large margin, indicating that if the testbed had more computational resources, the blockchain throughput would likely see an increase as well.

While the system is not over-capacity, transaction delay (from sent to validated) is, on average, 515 ms as was expected with a 1 second block time. As the system gets overloaded and transactions are placed in a queue, delays begin increasing and get progressively worse until the load becomes lower than the system capacity. However, no transactions were dropped during testing.

3.6 Conclusion

In this chapter, we discussed the key aspects of the Quorum and its main advantages over its base technology Ethereum for our use case. We also overviewed all the technologies used in the Certitex blockchain. As well as its development process, implementation, and design. Each node in Certitex contains three modules, the blockchain module itself powered by *Quorum*, where a copy of the ledger is stored and maintained. A connectivity module that communicates with other nodes broadcasting information regarding itself and other nodes it knows thus automating the connectivity process of the peer-to-peer network that is a blockchain. A smart contract interface module allowing users to interact with the smart contract that holds the logical rules required to create digital representations of products and combine, transform, and transfer them to other entities, this module ensures that the smart contract is deployed and coordinates with its peers to ensure that all participants in the system are interacting with the same smart contract. Additionally,

we go over the performance testing of the *Certitex* blockchain.

Chapter 4

Machine Learning

4.1 Introduction

The goal of this chapter is to explain how the machine learning module, which was developed to function in tandem with *Certitex*, works. The main goal of this module is to add prediction functionality based on all the data *Certitex* can collect over time.
In section 4.2 we introduce the used data set and how it was initially processed.
In section 4.3 we apply some statistical-based methods in order to predict optimal paths for products based on our initial processing of the data set. Additionally, we go over why this approach would not work.
In section 4.4 we discuss how the data set was enhanced with additional data and further processing in order to achieve a working machine learning module that would yield useful information. In it, we go over improving the simulated environment with further processing of the data set, generating additional data, how results were extracted, and explain the machine learning model developed.

4.2 Data set

Name	New York City Bike Share Dataset
Licensing	CC BY-NC-SA 4.0
Dataset Size	126MB 735503 entries by 17 features

This dataset contains information about several trips users of a public bike renting system in New York have taken. Because we do not possess real-life information about supply chain logistics this was a good approximation to the transportation of goods between the several entities involved in a supply chain. The dataset contains the following features and in parenthesis how those features were adapted to our problem:

- Trip Duration (Time of travel for a particular transportation of goods between entities)
- Start Time (Date of the trip, used for seasonal feature extraction)
- Stop Time (discarded)
- Start Station ID (ID of the start entity in this transportation)
- Start Station Name (discarded)

- Start Station Latitude (Latitude of this entity)
- Start Station Longitude (Longitude of this entity)
- End Station ID (ID of the end entity in this transportation)
- End Station Name (discarded)
- End Station Latitude (Latitude of this entity)
- End Station Longitude (Longitude of this entity)
- Bike ID (discarded)
- User Type (discarded)
- Birth Year (discarded)
- Gender (discarded)
- Trip_Duration_in_min (discarded)

To adapt this data set to the environment of a supply chain we made use of *NetworkX* to create a directed graph. The nodes of the graph represent the several stations present in the data set (or our entities in the supply chain) and the edges of the graph represent the existence of at least twenty-five trips between those stations a simplified version of the graph can be seen in Figure 4.1.

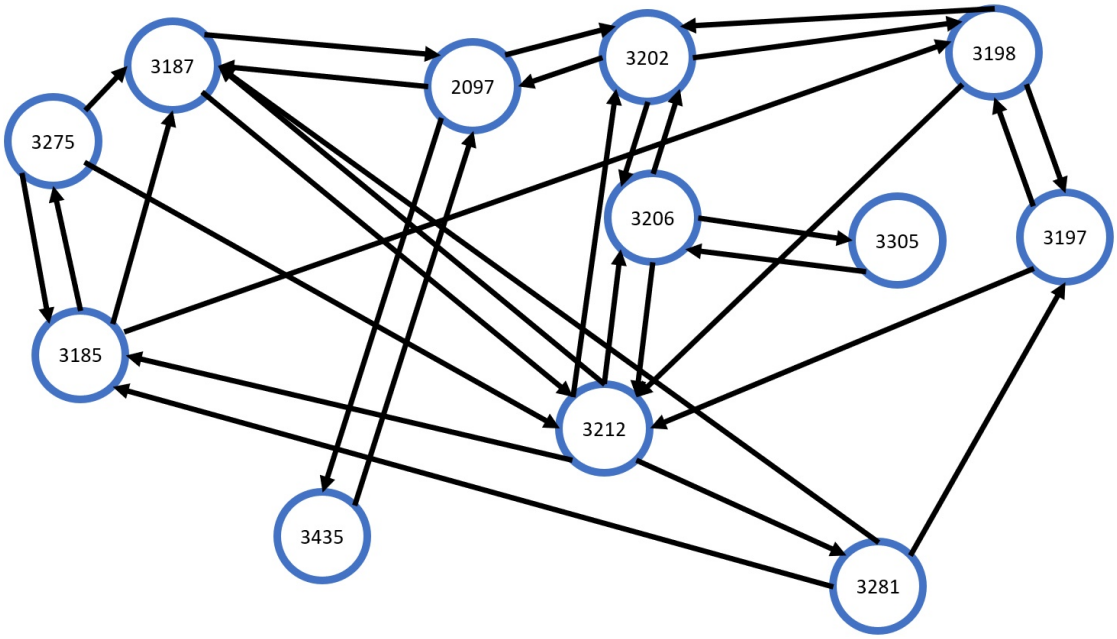

Figure 4.1: Time and distance analysis of 50 paths for shirt production. The X-axis represents time in seconds and the Y-axis represents the distance in kilometers.

The nodes contain information relative to the station's latitude and longitude in order to be able to produce an accurate image where the nodes' position and distance are accurate to real life. The edges between those nodes contain a count of the number of trips

between them and represent the several rows of our data set. We removed all edges with less than 25 trips, to eliminate connections that the data set has too few entries about. Next, we attributed to each node randomly an ”entity type” from among the following: factory, storage, and retailer. After this we calculated all simple paths, this is paths with no repeated nodes, from possible start positions of a product or component, possible start positions are factory or storage, to all other nodes. This is to represent as many diverse and complex supply chain production cycles, while in some cases the production cycle may involve only transportation between a factory and a storage facility, in other cases these production cycles may involve many trips between different factories. To each of these paths, a final product was randomly attributed as well. So we created a list of data structures that look like the following:

[[StationID,StationID,...] , [Product]]

4.3 Statistical Prediction of Optimal Paths

Using the data set created in 3.2 we may begin training a model that finds the most suitable routes for producing any particular item. To achieve this we started analyzing this list of paths statistically. By iterating over the path we incrementally calculate the total distance traveled using the Haversine formula. This formula calculates the great-circle distance between two points – that is, the shortest distance over the earth’s surface from the initial station’s latitude and longitude to the end station’s latitude and longitude. This is a quick approximation of the actual travel distance between the stations. Another option was explored to give us the actual distance traveled by road.
Using google maps however this option is free of charge for only a limited number of uses.
To begin extracting useful information from the data set, a statistical analysis function was built, that given a product ”X” would return a Cartesian graph where each dot represents a path in the system to produce ”X”, the value where this dot falls in the X-axis represents the time in seconds and the Y-axis the distance in Kilometers. A sample of this graph can be seen in Figure 4.2.
With this initial form of data processing, we attempt to give the user of the end system a rough answer to the question ”What is the faster way to produce product *’X’*?”
However, this prediction was rather simple and did not take into account a lot of variables present in real-world situations such as:

- The model only takes into account travel time between stations, it does not consider production times as such, which makes storage facilities just an unnecessary point in a path and mostly under-used;
- The model only considers very simple production paths, where raw material journeys through a series of factories and storages until it finally reaches a retailer;
- The model does not improve over time, it only averages the times of the trips it has records of, this means that it does not take into account important factors of cloth production such as seasonality or even simpler things such as weekends or days off.

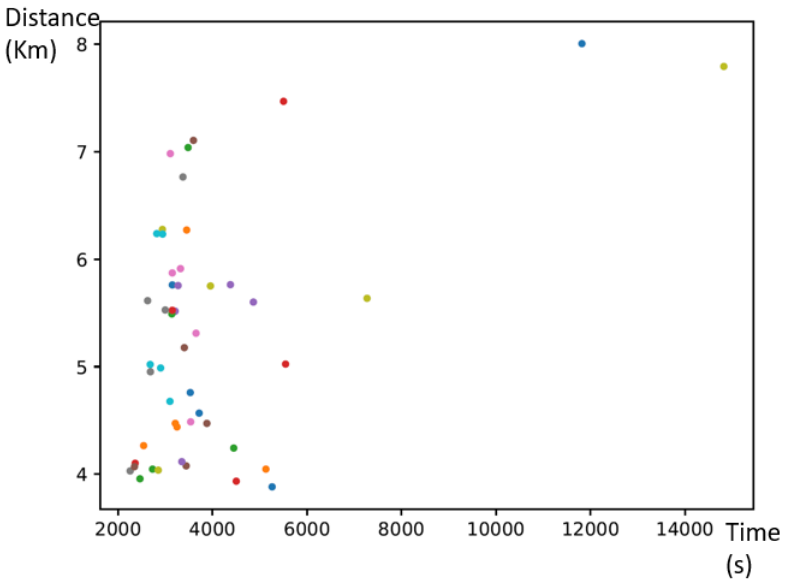

Figure 4.2: Time and distance analysis of 50 paths for shirt production. X axis represents time in seconds and Y axis represents distance in kilometers.

4.4 Improved Real World Simulation

4.4.1 Improving the simulated environment

In order to improve on the simple model built-in section 4.3 we began by improving the simulation to make it more similar to reality. We began by generating item production patterns. For example, a pattern of production would be raw material ”a” in conjunction with raw material ”b” would make finished product ”A” (mind the capitalization). These production patterns are generated automatically and randomly, can have varying depths and heights. A pattern with depth 2 and height 3 would look like the 4.3

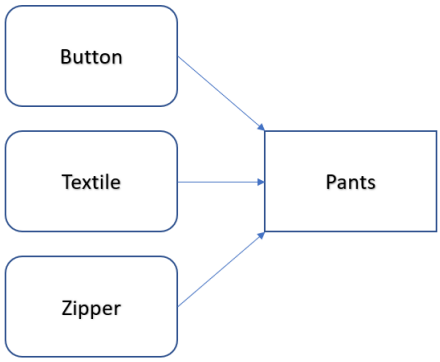

Figure 4.3: Production pattern example of a real world pants manufacturing process.

While generating the production patterns mentioned previously we also determine the set of items that we will be working with. Production patterns will repeat items to simulate as many complex supply chain environments as possible. For example, as seen in picture 4.3, the raw material ”textile” is used in the production of pants, but it may also be used in producing many other products.

We then divided factories and storages into two, the raw materials section, and the finished product section. Members of the first group will have a predisposition to work with non-finished goods (from now on denoted with a non-capital letter) and the second group with finished goods (denoted with capital letters). After their alignment is set we finally attribute to each node their set of products. For storage facilities, this list will determine the items they are capable of storing, for factory facilities it determines the items they are capable of producing, and for retailers, the finished products it sells. Each node's respective list varies in size and is randomly determined.
After tweaking the parameters over multiple runs in order to reach a good balance of node types as well as item and production pattern variety while still allowing the data set to provide a vast amount of paths per production pattern, we ended up with a graph with the following properties, which will be used for the rest of the explanation:

- Number of nodes: 55
- Average connections per node: 27
- Number of factories / storages / retailers: 15/32/8
- Number of raw materials/ finalized products: 4/3

In figure Figure 4.4 we can visualize the production patterns we will be working with.

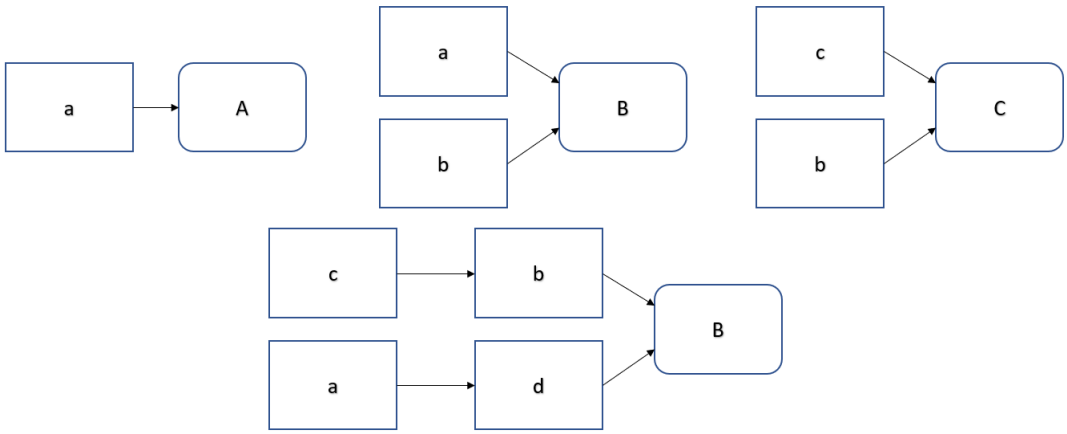

Figure 4.4: Generated production patterns that will be considered.

4.4.2 Generating Production Values

Now that we added more real-world properties to how items are produced and which node works with which product, as well as how those items are combined into finalized products, work begun on adding realism to the nodes themselves. Each factory at any given point will only have a certain amount of items it can produce, that will vary over time, additionally, storage facilities will have some products already in stock throughout the year that were produced previously and remain unused, this will solve the previously

mentioned problem where storages were only adding one more step to the production chain rendering them useless.
This data is artificially generated based on realistic factors such as factory production capacity correlates with the number of nodes it is connected to. Factories have preferred seasons and days of the week. A random modifier is applied to each day that further adds variability to the results and is meant to represent a good and productive day versus a bad less productive day.
The node's Base amount is determined via the addition of a set of numbers repeatedly drawn from a normal distribution with a mean of 10 and standard deviation of 5, the number of draws is based on the node's neighboring nodes.

$BaseProduction = min(\sum_{n=0}^{N} \mathcal{N}(10, 5), 10)$
where N represents the number of nodes direcly connected to this one.

In addition, we add temporal variation via an extra variable "X" that varies from 0.3 to -0.3 based on season and day of the week. So in a given day, a factory's production will be determined by the following formula:

$$BaseAmount + \mathcal{N}(BaseAmount * X, BaseAmount * 0.05) + BaseAmount * Modifier$$

Generally, a factory will have a stable production throughout the year, with slight deviations due to the modifier and day of the week and noticeably larger seasonal variability. Additionally, a default value of two hundred items was added to each storage. This process was applied to the graph for a large length of time (ten years) and the results were saved. With this new and improved simulation, we have fixed many of the issues listed before as well as add many new features such as seasonal variability and flexible patterns of production. With this as a base, it would now be interesting to build a model that, while based on this semi-fictitious data, would be able to, in the future, use real data and extract useful information. The problem to be solved was: Given any particular finalized product and a retailing facility, what are the possible ways to get any amount of the finalized product to this retailer with a detailed journey (all nodes and paths used and for what reason), ranked by time.

4.4.3 Extrapolating results

In order to extrapolate useful results a prediction function that receives as input the final product, the target node (a retailer), an amount of that product, and a snapshot of the previously generated data was made. The algorithm looks for solutions going through the production patterns in a reverse fashion. To exemplify this, let's use the production patterns for product "A" present in Figure 4.4 and explain the algorithm step by step.
As stated before the algorithm knows that we are trying to produce "A", it also knows that we have a target retailer to deliver the products to. Let's also assume that we need to deliver 400 of "A" to this target retailer. Initially, the algorithm creates a list of all production

patterns that allow us to produce the target item, so a) and b). As aforementioned, we iterate through the production patterns in a reverse fashion so from maximal depth to depth zero. As such, maximal depth for all production patterns is the finished product ”A”, in this step, the algorithm tries to transport already produced ”A” that is present in storage facilities to the target retailer. A helper graph is constructed where it's node's are the target node and all storage facilities that work with the product ”A”, the edges for this graph are created if we have entries in the New York City Bike Share Data set of trips between a storage facility and the target retailer node. In other words, the graph will be a collection of all storage facilities of ”A” and a retailer (our target retailer) where these storage facilities may or may not have an edge leading to the retailer depending on data present on our dataset. The edges also have two additional attributes ”time” and ”distance”. The time is the average number of time taken from all trips registered from one node to the other target node and the distance, is the distance calculated by the Harsevenian formula of the GPS coordinates of the two nodes. Now to calculate the results for depth zero, with this new graph, all that we need to do is calculate the weighted paths (in this case our weight is the time attribute), from all nodes to the target node and append each calculation to the results in the following format:
[CapacityForThisLine, time, distance, [pathAndProduct]]

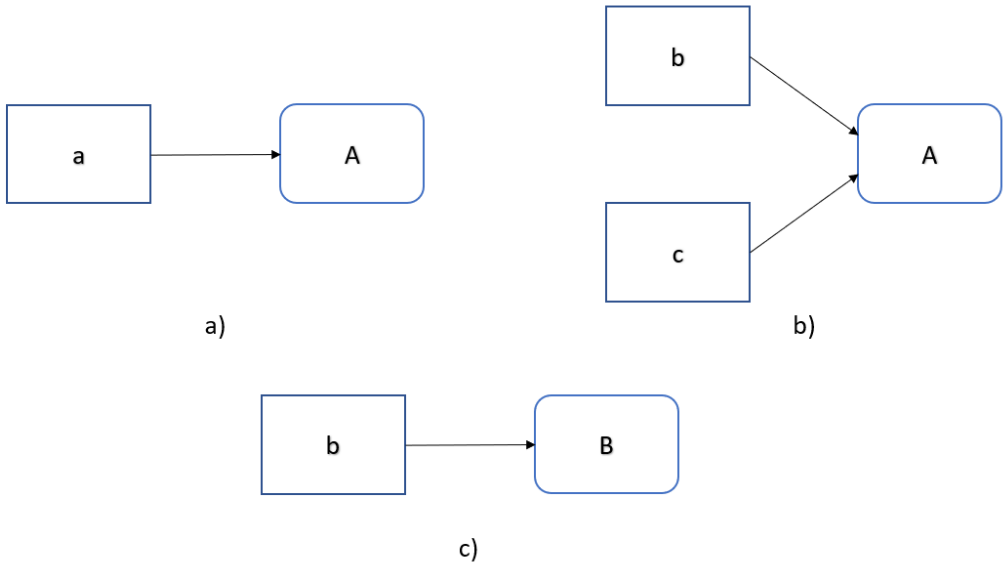

Figure 4.5: Small sample of production patterns.

With the depth zero results calculated we may move on to the more complex next step of calculating depth one, in this depth we need to consider the fact that each product may be the conjunction of several linear production patterns, looking at b) in Figure 4.5 this production pattern is in reality constituted by the two simpler production patterns of:

$b \rightarrow A$
$c \rightarrow A$

Additionally, we must consider the fact that there may be more than one production pattern per end product, this is in place to simulate the fact that some factories resulting intermediate products are incompatible with other factories' manufacturing processes. For example, a factory's resulting intermediate product may be more processed than another which leads to some factories being able to complete it into a finished product whilst others can not. As such the algorithm must, for each production pattern with the desired end product calculate the fastest route to get the required intermediate product(s) to a factory, this factory will transform the intermediate product(s) into the desired final product and finally, this product must be transported to the target retailer. To exemplify this we will use the set of production patterns present in Figure 4.5.

For final product A we begin with production pattern a), we build a graph containing all storage facilities of product a and all factories that can transform this product into final product A. After this we must connect all the added storage facilities to all the factories, this is done through the dataset, if we have recordings of travels between the nodes the link is created, it is always a one-way edge from storage to a factory, this is because this is the only way products will flow, and each edge will have a weighting and a distance attribute calculated from the average recordings present. Finally, the target retailer node is added and all factories connect to this node, just like before with the same attributes through the use of the dataset. With the completed graph we can use NetworkX to calculate the weighted paths starting from storages of a to the retailer node. Due to how the graph was built the results of this calculation are always valid paths, they start with intermediate product a, transport it to a factory that is capable of transforming it into final product A and are then transported to the desired retailer. The result of this production pattern is, just like before:

[CapacityForThisLine, weight/time, distance, [pathAndProduct]]

This same process is repeated for production pattern b) with some key differences, in this case, as stated before this production pattern is the union of two simpler production patterns, each of these simpler production patterns is calculated individually yielding a list of intermediate results. These intermediate results contain paths that only include product c, or only product b, or both. They are then merged among themselves so that the pattern is fully completed. The result must contain both intermediate products b and c, In this merging process the time attributed to each path is the largest time out of all the sub-products and the smallest capacity. For example, if we have these two intermediate results being merged:

[100, 360, 200, [[[3191, 3267, 3186], ['b']]]]
[150, 400, 250, [[[3270, 3267, 3186], ['c']]]]

would result in the final path: *[100, 400, 250, [[[3270, 3267, 3186], ['c']], [[3191, 3267, 3186], ['b']]]]*

Telling exactly how each item should be transported, for c, from storage 3270 to factory 3267 and b from storage 3191 to the same factory 3267, where they are merged into final product A and then finally transported to the target retailer 3186, and this process would take (considering only transportation times) 400 seconds. It is a small value due to the

dataset selection that depicts travel times anywhere between 30 seconds and 10 minutes, however, if real-world data were to be used the results would become realistic as well.

4.4.4 Machine Learning

The only part of the result that has not been explained it's the value at index 0 of all the shown results, "CapacityForThisLine". This value is the result of a machine learning model developed. In the previous section, we talked about how we generated artificial data, where each factory would have a production capacity per day. This data is then used to train a long short-term memory neural network to predict the production capacity for any given day based on the generated data. Allowing us to attempt to predict the future production capacity of all factories in the system.
In the Figure 4.6 we can see a graph where the production capacity of a factory is plotted over time, we can also notice the seasonality trend for this particular factory, in this case there is an increase in production during autumn:

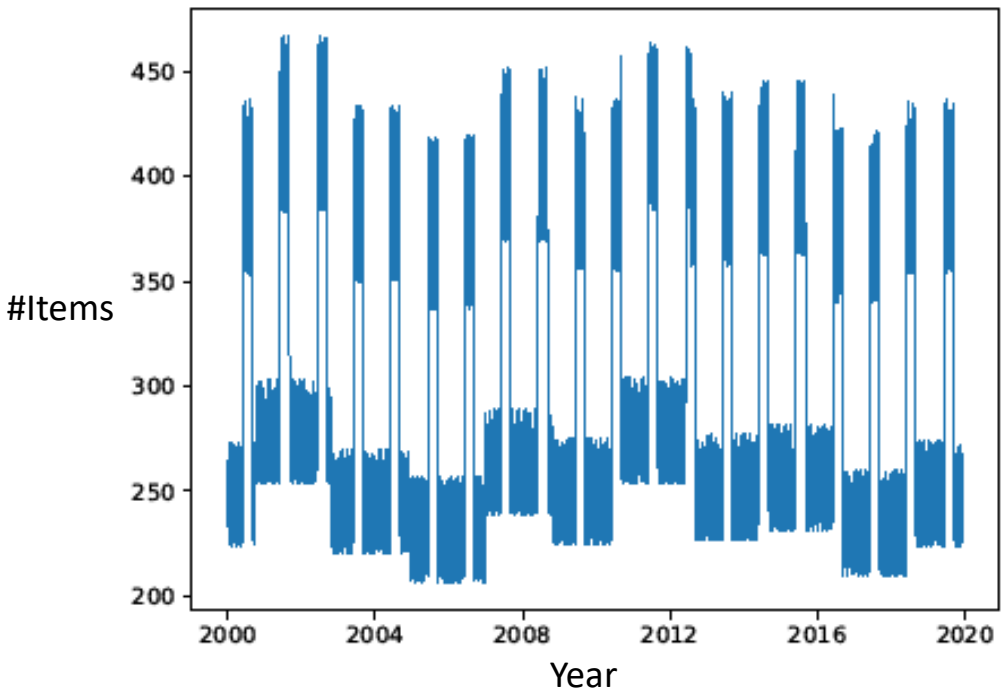

Figure 4.6: Factory production variation over 10 years with one reading per day

To train a model on this data it was first processed in order to reduce the magnitude of variation between values. To do this we first calculate the average number of items for the whole set and take it away from each value. after this we re-map the values into a zero to one scale. where the largest value of the set would be one, and the smallest zero. Starting with an initially simple 4 layer network consisting of the input layer, a long short-term memory layer, a dense layer, and finally the output layer we began by testing different configurations until we achieved satisfactory results.
Firstly we began by splitting the data into three disjoint sets, train (80%), validation (10%),

and test (10%) set. The training set was used to adjust the weights of the neural network, the validation set was used to test the evolution of the network throughout its training, and to stop training it if it constantly did not improve during twenty training epochs and the test set was used to score the neural network's performance at the end.
We first began by tweaking the look-back value of the LSTM layer. This change provided the most impact on the results. If there is at least one year of data stored, the model improves significantly. The results can be seen in Table 4.1. Nevertheless, the obtained results were always superior to statistical approaches such as the naive approach and the moving average approach. This comparison can be seen in Table 4.3

Table 4.1: Root Mean square error in the test group for neural networks trained with a varying number of lookback in the long short-term memory layer

Look back	Test Root Mean Square Error
7	20.33
31	18.35
62	19.59
93	15.30
186	15.69
365	9.11

Then we continued by tweaking the LSTM layer by changing the number of neurons, starting with two, incrementing by two, and finishing at eighteen, the results can be seen in Table 4.2, The number of present neurons did not impact the results significantly so the layer was left with two neurons. Keeping this value low will significantly decrease training times.

Table 4.2: Mean square error in the test group for neural networks trained with a varying number of neurons in the long short-term memory layer

Neurons	Test Root Mean Square Error
2	9.11
4	9.12
6	9.44
8	9.10
10	9.35
12	9.23
14	9.14
16	9.50
18	9.80

Table 4.3: Moving averages calculated from the x previous values, x in parenthesis

Ours	9.11
Naive	18.23
Moving Average(2)	24.96
Moving Average(5)	24.96
Moving Average(10)	32.54

With an optimized number of neurons and a look back for that number of neurons that on average yields a result only nine products away from the actual value in the testing set, we have achieved results twice as accurate as of the best statistical model. The model

takes on average sixty-eight seconds to train using CPU on an Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz. The results of the training can be seen in Figure 4.7, and the network architecture can be seen in Figure 4.8.

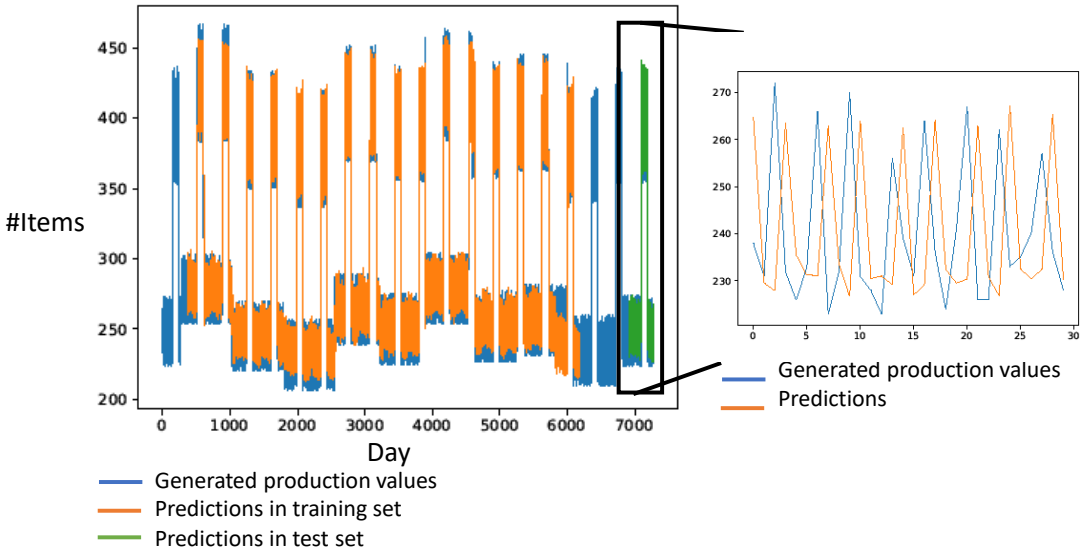

Figure 4.7: Raw production values, train set/validations set predictions and test set predictions of the neural network

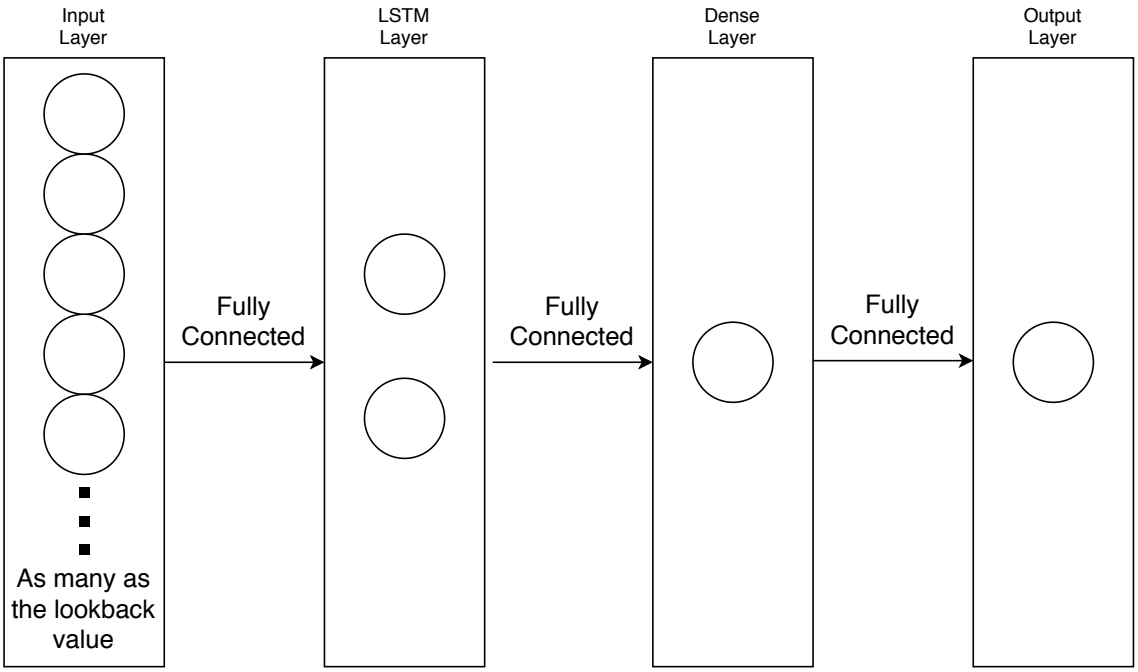

Figure 4.8: Final network architecture.

With the production capacity for each line now being predictable, the algorithm is now capable of providing a list of production paths for each product to a specific target retailer, ordered by the expected duration of the path as well as giving us the information as to how many items it predicts that path will be capable of outputting at any given day. The results

will be exemplified in the following paragraphs for transporting a finalized product ”C” with regards to the aforementioned production patterns in Figure 4.4.

1. `[200, 232.61594989561587, [[3213, 3186], ['C']]]`
2. `[200, 332.4891025258255, [[3209, 3186], ['C']]]`
3. `[200, 393.0569482440958, [[3203, 3186], ['C']]]`
4. `[182.65, 1692.6125829034636, [[[3192, 3267, 3186], ['b']], [[[3191, 3267, 3186], ['c']]]]]}`
5. ...

In this case, there was only one pattern for producing item C as seen in Figure 4.4 intermediate product b and c together are transformed into C. The results can be interpreted as follows:

1. There is a possibility to transfer 200 (index 0) of product C (index 4) taking 232.6 units of time (index 1) starting from Storage facility 3213 (index 2) and ending in retailer 3186 (the target).
2. There is a possibility to transfer 200 (index 0) of product C (index 4) taking 332.4 units of time (index 1) starting from Storage facility 3209 (index 2) and ending in retailer 3186 (the target).
3. There is a possibility to transfer 200 (index 0) of product C (index 4) taking 393.1 units of time (index 1) starting from Storage facility 3203 (index 2) and ending in retailer 3186 (the target).
4. This is the case where we will produce product C without directly transporting it from storage facilities. In this suggestion, we produce it at factory 3267 which is estimated to be able to produce 182 products at this point (index 0) based on the aforementioned machine learning model. The production path advised is to transfer product b from storage 3192 to factory 3267 and product c from storage 3191 to factory 3267, where it will be turned into product C and then transferred into the target retailer

4.5 Conclusion

This chapter described the creation of a pluggable module for Certitex. From an initial statistical prototype module based on randomized pathing to an LSTM-based machine learning model that, with some adaptations to the data set, closely resembles a real-world supply chain network. This was achieved through the use of the ”New York City Bike Share Dataset” with some adaptations and data generation. In the end, we have a complex graph of nodes whose connections are dictated by the data set. Nodes will have a type randomly attributed from the set ”factory”, ”storage” and ”retailer” and a set of products that they

work with. Additionally, randomized production patterns are simulated that dictate how items are merged throughout the supply chain in order to obtain finalized products. Factory nodes also have generated their daily production quantities based on factors such as season, number of connections to other nodes, and other modifiers. Using this generated data, the machine learning model was built. It is capable of predicting production capacity for any factory on a given day, with an estimated error of about 1.63%. This machine learning module serves to feed an algorithm capable of outputting the estimated delivery travel of all possible paths present in the supply chain capable of producing or transporting an item to a retailer based on the generated production patterns, and also the list of products each node works with and the connections between nodes given by the data set.

Chapter 5

Final Considerations and Future Work

5.1 Main Conclusions

Recent advances in blockchain technology in recent years has led to several breakthroughs and paradigm changes in the possible uses of the technology. Developing solutions that allow for blockchains to be managed privately are the aim of a lot of criticism due to contradicting one of the blockchains' main design features. However, it has allowed blockchains to serve as a middle ground between fast and secure in environments where there are many stakeholders, such as a supply chain. To illustrate this, a blockchain-based conceptual system is proposed and prototyped, and tested. Alongside this prototype, other modules were developed to demonstrate the flexibility and capabilities of the prototype itself. This includes a demonstration application providing an interface through which data can be inserted, processed, and extracted from the system. Additionally, a machine learning module was developed to demonstrate how data that could be inserted into the system can be used to provide valuable information to the users. If a blockchain system following the guidelines that were used to develop the proof of concept were to be introduced in a real-world supply chain, it is believed that many of the issues stated in this dissertation would be reduced or even eliminated, while providing a framework for many other useful features to be implemented.
The main objective of this thesis was to study how to use blockchain technology in supply chain environments, introducing the advantages commonly associated with blockchain and to tackle the challenges associated with more traditional supply chain management strategies. To achieve this a study on blockchain technology was conducted and thoroughly analyzed in order to create a conceptual system proposal. Based on this proposal, a proof of concept prototype was developed and tested, and its performance was accessed. The integration of This project into a real world situation is the main focus of future work for this thesis. As such, all of the proposed objectives for this dissertation work were successfully completed.

5.2 Future Work

To conclude this work, suggestions of research directions for future work will now be presented:

- Integration of Certitex into a large scale supply chain ecosystem study it's impacts and possible flaws, as well as using Certitex to collect data;

- Perform quality of life improvements such as migrating the API engine from its current Werkzeug WSGI to a production-oriented variant allowing for the API's to keep up with the increase in data input. As well as adapt the system to any other issues that may arise from real-world implementation;
- With the real-world data collected develop a module that would translate this data into a format usable by the machine learning module into this project and access it's performance;
- Develop other modules that much like the machine learning module already developed, can make use of the collected information to produce meaningful output for the user.