The Pennsylvania State University
The Graduate School
College of Information Sciences and Technology

BLOCKCHAIN SMART CONTRACTS IN MEGACITY LOGISTICS

A Thesis in
Information Sciences and Technology
by
Lawrence Wu

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

May 2018

Abstract

Megacities are becoming a major focus of the global economy, with almost 70% of the world population expected to live in megacities by the year 2050. Logistics is the lifeblood of megacities and retailers need to find efficient and less expensive logistics systems.

This thesis proposes a system of blockchain peer-to-peer smart contracts built with Hyperledger, to build a megacity grocery supply chain where retailers, suppliers, and logistics providers can bid directly for contracts, analyze market trends, trace product flow for food safety, and reduce costs by cutting out the middlemen. A blockchain can supplement or take the place of centralized fourth party logistics providers and reduce the layers of middlemen in the supply chain by connecting suppliers, retailers, and Third Party Logistics Providers (3PLs) directly. The blockchain also provides a distributed ledger that all participants would have a copy of and can contribute to without repudiation.

The decentralization inherent in the blockchain can allow smaller, independent contractors to be more competitive, increase efficiency by reducing the layers of middlemen, and reduce the costs of technology adoption for access to information. Smaller suppliers can be aggregated and matched to retailers through volume orders without a middleman. Logistics providers could be crowdsourced individual contractors that could be called upon anytime for small or rush deliveries, much like how rides can already be hailed on sharing economies such as Uber and Lyft.

In terms of traceability, the cost of sharing data between all participants could be significantly reduced, food safety issues can be mitigated as soon as they are detected, blame would be less likely to fall on the wrong party, and counterfeit products would be impeded by their lack of provenance on the blockchain. Megacities that have to cope with reduced warehouse space, greater demand for direct/fresh delivery will need improvements like these to sustain their growth.

Contents

List of Figures **ix**

List of Abbreviations **xi**

List of Terminology **xiii**

Acknowledgments **xv**

Chapter 1
Introduction **1**
1.1 Problem 1
1.2 Summary of Work and Contributions 2
1.2.1 Key Possibilities 3
1.2.2 Limitations of the Blockchain 4
1.3 Thesis Outline 5

Chapter 2
Problem Definition **7**
2.1 Introduction 7
2.1.1 The Procurement Problem 8
2.1.2 The Multiple Levels of Logistics Providers 10
2.1.3 The Paperwork Problem 11
2.2 The Applicability of the Blockchain 12
2.2.1 Choosing the Type of Blockchain for the Problem 12
2.2.2 Relevance of the Blockchain to the Supply Chain 13
2.2.3 Applying Permissionless and Permissioned Blockchains to the Supply Chain 15
2.2.4 Infrastructure Needs and Costs of the Blockchain 15
2.3 The Blockchain and Smart Contracts 17

2.3.1 Three Combined Applications of Permissionless and Permissioned Blockchains 18

Chapter 3
Background **21**
3.1 Supply Chain Theory 21
3.1.1 Megacity Logistics 21
3.1.2 Types of Supply Chain Inefficiencies 22
3.1.3 Logistics Provider Models 22
3.1.4 Distributed and Collaborative Logistics Planning and Replanning under Uncertainty 24
3.1.4.1 The Procurement Problem 24
3.1.4.2 Satapathy's Research Objectives 26
3.1.4.3 Types of Supply Chains 27
3.1.4.4 Master and Servant Problem Solving Model 28
3.1.4.5 Workflow 30
3.2 A Sharing Economy for the Supply Chain with eHarvestHub 31
3.2.1 The Current State of Supply Chain Middlemen 32
3.2.2 Sources of Markups and Inefficiency 33
3.3 Foundational Concepts of the Blockchain 34
3.3.1 Bitcoin: A Peer-to-Peer Electronic Cash System 36
3.3.1.1 The Byzantine Generals' Problem 37
3.3.1.2 The Distributed Ledger and the Double Spending Problem 37
3.3.1.3 The Proof-of-Work Algorithm 39
3.3.1.4 Unspent Transaction Outputs 42
3.3.1.5 Merkle Trees and Simplified Payment Verification . 42
3.3.1.6 Conclusions 43
3.3.2 Smart Contracts 44
3.3.3 The Bitcoin Blockchain as a State Transition System 45
3.3.4 Applications of Alternative Blockchain State Transition Functions 46
3.3.4.1 Independent Blockchain 47
3.3.4.2 Colored Coins and Metacoins 49
3.3.5 The Ethereum Blockchain 50
3.3.5.1 Smart Contracts: Arbitrary State Transition Functions 50
3.3.5.2 Ethereum Mining Algorithm 52
3.3.5.3 Ethereum in Practice 53
3.3.6 Permissioned/Consortium Blockchain Methodology 54

3.3.6.1 Hyperledger Sawtooth 56
3.3.6.2 Sawtooth Consensus Algorithms 57
3.3.7 Application Paradigms in the Blockchain 58
3.3.7.1 Characteristics of the Blockchain 58
3.3.7.2 Generic Application Patterns 59
3.3.8 Security and Practicality Concerns 60
3.3.8.1 The Blockchain Approach to Ledger Security . . . 61
3.3.8.2 Smart Contract Bugs and Oversights 62
3.3.8.3 Reliance on Network Conditions and Health 64

Chapter 4
Related Work **67**
4.1 Literature Discovery 68
4.2 Formal Papers and Theses 69
4.2.1 Blockchain in Megacity Logistics 69
4.2.2 Aiming for Supply Chain Transparency: Exploring the Potential of Blockchains 70
4.2.3 The Supply Chain has No Clothes 71
4.2.3.1 Achieving Transparency and Traceability in the Supply Chain 72
4.2.4 Distributed Ledger for Supply Chains 73
4.3 General and Informal Knowledge 74
4.4 Smart Contract Applications 74
4.4.1 Initial Coin Offerings: Tokens that do Things 75
4.4.1.1 World Food Program: Cryptovoucher Case Study . 77
4.4.2 Hyperledger: Seafood Supply Chain Traceability 79

Chapter 5
Methodology **80**
5.1 Marketplace 80
5.2 Logistics Matchmaking 83
5.3 Scenario 86
5.3.1 Product 86
5.3.1.1 Build-to-Order Products 86
5.3.1.2 Build-to-Plan Products 87
5.3.2 Order Tickets 87
5.3.3 Geography 88
5.3.4 Transportation Costs 89
5.3.5 Additional Modules 89
5.4 Units 90

5.4.1 Geography and Time 91
5.5 Modeled Entities 92
5.6 Contracts 93
5.7 Computations 94

Chapter 6
Implementation **97**
6.1 Balancing Data Confidentiality and Public Traceability 98
6.2 Hyperledger Composer Overview 99
6.2.1 Business Network Archive 100
6.2.2 Hyperledger Business Network Templates 101
6.3 Model Definitions 101
6.3.1 Participants 102
6.3.2 Assets 102
6.4 Process 102
6.4.1 Product Marketplace 102
6.4.2 Shipping Inquiry 103
6.4.3 Logistics: Shipping Ceremony 104
6.5 Product Bidding Transactions 105
6.5.1 `createProduct` Create a Product 105
6.5.2 `Offer` 105
6.5.3 `CloseBidding` 106
6.6 Contract Bidding Transactions 106
6.6.1 `createContract` Create a Contract for a SOLD Product . . 106
6.6.2 `Bid` 107
6.6.3 `CloseInquiry`: Execute the Contract 107
6.7 Shipment Transactions 108
6.7.1 `CreateShipment` Begin a Shipment for a Contract that is `READY_FOR_PICKUP` 108
6.7.2 `Dock`: Shipper Informs Possessor of Arrival 108
6.7.3 `HandOff`: Change Product Possession to Shipper 108
6.7.4 `PickUp`: Confirm Product Possession and Move Out 109
6.7.5 `Arrive`: Record Shipper Arrival at Retailer 109
6.7.6 `TemperatureReading`: Record Temperature Reading (Shipper) 109
6.7.7 `ShipmentReceived`: Record Shipment Reciept (Retailer) . . 110
6.8 Access Control Rules 110

Chapter 7
Results and Conclusions **112**
7.1 Case Study 112

7.1.1 Food Traceability 112
7.1.2 eHarvestHub: A Marketplace to connect Grocers, Truckers, Farmers Directly 113
7.1.2.1 Future Expansions for eHarvestHub 114
7.1.3 Value-added Products: A Story Told by the Blockchain . . . 115
7.2 Conclusions 117
7.3 Future Studies 118
7.3.1 Possible Expansions of this Implementation 118
7.3.1.1 eHarvestHub Collaboration 118
7.3.1.2 Broadcast a Shopping List with IoT Fridges 118
7.3.1.3 The Blockchain as a Data Source 118
7.3.2 The Unified Theory of Acceptance and Use of Technology . 119
7.3.3 Ontology Driven Supply Chain Provenance 122

Appendix A
Implementation Model Definitions 124
A.1 Product Negotiation Assets 124
A.2 Perishable Assets 126

Appendix B
Additional Online Sources 132
B.1 Videos Watched 132
B.1.1 Bitcoin 132
B.1.2 Ethereum 132
B.1.3 Hyperledger 133
B.1.4 Blockchain Overview Videos 133
B.1.5 Development 133
B.2 Organizations 134
B.3 Foundational Papers 134
B.4 Implementations 134
B.5 Overview Documents 135
B.6 Software 136
B.7 Tutorials 136
B.8 Code Repositories 136
B.9 Courses 137

List of Figures

2.1 The procurement problem as defined by Satapathy (Satapathy 1999). 9
2.2 Flowchart based on info from GSA and Johansson's 2018 thesis for deciding which type of blockchain to use. (Herman 2018), (Johansson 2018) . . . 12
2.3 The methodology applied by this thesis to use the blockchain to solve the procurement problem. . . . 14

3.1 The procurement problem as defined by Satapathy (Satapathy 1999). 25
3.2 Master and Servant Agent Interaction as defined by Satapathy (Satapathy 1999). . . . 29
3.3 A fundamental concept of cryptography is given as the top bullet, and the fundamental concepts of the blockchain that it helps explain is given as the subbullets (Johansson 2018, p. 9). . . . 35
3.4 A transaction on a block of a blockchain's distributed ledger (Limited 2018). . . . 38
3.5 The Merkle Tree as implemented in the blockchain. Where root[i+1].signature = hash(concatenate(root[i].toString(),block[i].toString())) (Limited 2018). . . . 44
3.6 The state transition function of Bitcoin in pseudocode (Buterin, Wood, et al. 2018). . . . 45
3.7 An example of a Domain Name digital asset in the Ethereum Serpent language (Buterin, Wood, et al. 2018): . . . 49
3.8 A smart contract as arbitrary state transition code embedded in a block on the blockchain (Limited 2018). . . . 50

5.1 How a Marketplace can aggregate volume from small suppliers without a middleman. Credits to Alvaro Ramirez of eHarvestHub for assisting with this diagram. . . . 82

5.2 How Logistics Providers can be matched to retailers without a middleman. Credits to Alvaro Ramirez of eHarvestHub for assisting with this diagram. 84
5.3 How Logistics Providers handle and report the delivery on the blockchain. Assistance by Alvaro Ramirez. 85
5.4 A possible idealized city grid for the scenario. 88
5.5 What type of smart contracts could be created and which agents interact with each other using them. 96

6.1 Hyperledger Composer Playground Development Environment (Hyperledger Foundation 2018[c]). 99
6.2 Hyperledger Composer Playground Business Networks (Hyperledger Foundation 2018[c]). 100

7.1 The TOVE Ontology in simple form. (Kim and Laskowski 2016) . . 123
7.2 An example of a TRU trace in a Coffee Supply Chain. (Kim and Laskowski 2016) 123

List of Abbreviations

1PL First Party Logistics Provider. 22,

2PL Second Party Logistics Provider.

3PL Third Party Logistics Provider. iii, 1, 2, 5, 23–25,

4PL Fourth Party Logistics Provider. 1, 2, 5, 23,

AI Artificial Intelligence.

API Application Programming Interface.

ASIC Application-Specific Integrated Circuit. 52, 53,

BTO Build-to-Order. 86,

BTP Build-to-Plan. 87,

CPU Central Processing Unit. 52, 53,

CRVR Collaborative Robust Vehicle Route.

DAO Decentralized Anonymous Organization.

DApp Decentralized Application.

DNS Domain Name System. 47, 48,

ERC Ethereum Request for Comment.

EVM Ethereum Virtual Machine. 50, 53,

GPU	Graphics Processing Unit. 52,
ICO	Initial Coin Offering. 67, 75–77,
IPO	Initial Public Offering. 75, 76,
IT	Information Technology.
KQML	Knowledge Query and Manipulation Language.
LC	Late Customization.
P2P	Peer-to-Peer.
PDP	Stochiastic Pickup and Delivery Problem.
PoW	Proof-of-Work.
RAM	Random Access Memory. 53,
REST	Representational State Transfer.
RFID	Radio-Frequency Identification Tag.
SHA	Secure Hash Algorithm. 40, 52,
SPV	Simplified Payment Verification. 43, 49,
TOVE	TOronto Virtual Enterprise project. x, 122, 123,
TX	Transaction.
UML	Unified Modeling Language.
UTAUT	Unified Theory of Use and Acceptance of Technology.
UTXO	Unconfirmed Transaction Output. 42, 45–47, 58,

List of Terminology

Bitcoin The definitive cryptocurrency as developed by Satoshi Nakamoto (Nakamoto 2008), and the distributed ledger technology described in its whitepaper became known as the blockchain. 35

blockchain A specific type of distributed ledger technology, that uses algorithms on nodes in peer-to-peer networks to reach distributed consensus on a ledger without a trusted third-party. 26

contract As defined in our implementation, a shipping smart contract between a supplier, shipper, and retailer to handoff, transport, and deliver the product for the retailer.. 102

cryptocurrency "A purely peer-to-peer version of electronic cash (that) would allow online payments to be sent directly from one party to another without going through a financial institution" (Nakamoto 2008). 35, 60

distributed ledger A database that is maintained by many participants or across many locations. Consensus is typically guided by a centralized leader or trusted third-party, but the blockchain is an exception.. 26

Ethereum A secure decentralized generalized transaction ledger "that runs smart contracts: applications that run exactly as programmed without any possibility of downtime, censorship, fraud or third-party interference" (Buterin, Wood, et al. 2018). 35, 36

IP address A string identifying a computer on a network that utilizes the Internet Protocol (IP). 47, 48

logistics provider Transports items between suppliers and retailers (and if necessary, return them to sender). Also referred to variously as Shippers, Transporters, and Third Party Logistics Providers, since although there can be first or second party logistics providers employed by the retailer, this thesis aims to link Third Party Logistics Providers directly to retailers.. 102

permissioned blockchain A specific type of blockchain which enforces settings that restrict which nodes observe, participate, or mine, and what type of the ledger data they can access. Such settings can either be decided up through vote by participating nodes, or by central directive from a single party (likely the code developers). Examples are Hyperledger, JP Morgan's Quorum, World Food Program's private Ethereum chain. 35, 55–57, 66

permissionless blockchain A specific type of blockchain which enforces no restriction on participating nodes/miners, and the ledger data they can access. Example are Bitcoin, Ethereum, and most other cryptocurrencies. 55, 58

private permissioned blockchain A specific type of permissioned blockchain with some or all ledger data only verifiable to some or all participating individuals.. 77

product A product to be sold by a supplier to a retailer.. 102

public permissioned blockchain A specific type of permissioned blockchain with some or all ledger data verifiable to the public.. 79

retailer Purchases and receives products, for eventual sale to end customers. Although this process is how this thesis currently conceives of the purchaser, the retailer themselves could possibly be bypassed as the last middleman if customers become comfortable enough buying directly from suppliers.. 102

shipment A derivative asset of the shipping contract, generated when the shipper moves to pick up the product from the supplier. There could possibly be multiple shipments to various different locations, return shipments, or cancelled and renegotiated shipments.. 102

supplier Produces/Procures/Supplies the product for retailers. Could be farmers, food packers, or manufacturers.. 102

Acknowledgments

Thanks to Prof. Dinghao Wu for supporting me over the past 4 1/2 years that I have been working with him on research, starting in undergraduate degree.

Thanks to Prof. Soundar Kumara and his graduate students for providing me with mentorship regarding my graduate career, advisorship of the PSU Blockchain Club, and the supply chain side of this thesis.

Thanks to Prof. Nick Giacobe and his IUG students for working together to break through the deadlines and make this thesis a success.

Thanks to the PSU Blockchain Club for being great friends, journeying across the Northeast to meet other blockchain enthusiasts, and helping me out on the mechanics of blockchain smart contracts.

Thanks to the PSU Cryptocurrency Club for providing a community to discuss the social impacts, profit viability, and economics of blockchain applications.

Thanks to Pablo Eder for connecting me with the startup eHarvestHub, and thanks to them for working together with me to develop this supply chain methodology for the blockchain.

Dedication

This thesis is dedicated to my brother Justin Wu, whom is following in my footsteps in Pennsylvania State University's College of Information Sciences and Technology, and whom I hope is able to reach greater heights in blockchain research.

Chapter 1 Introduction

Seventy percent of the world population is expected to live in megacities by 2050. Logistics is the lifeblood of megacities, making a modern lifestyle possible, but the increase in population density demands increased efficiency, less waste, and lower costs to ensure the sustainability of this growth (MIT Megacity Logistics Lab 2018). Most grocery stores in megacities depend on multiple Third Party Logistics Providers (3PLs) which trade with suppliers or other 3PLs. However, retailers prefer to deal with a single accountable agent. Thus a centralized Fourth Party Logistics Provider (4PL) (typically the developer of the IT information system) is empowered to manage and audit the activities of the 3PLs.

1.1 Problem

In the procurement problem, a retailer wants a product. Knowing that demand exists, the supplier announces product availability and provides a price quote on the market (Satapathy 1999). The total cost to the retailer is the product price plus the estimated shipping cost.

If the retailer does not agree on the supplier's price, they begin a negotiation process, where they make an offer or counteroffer several times until they convergence on price. The supplier and retailer agree to the final price, and a contract is signed.

In the current supply chain, negotiations to solve the procurement problem unfortunately do not tend to occur directly between suppliers (farmers), what we will call "independent logistics providers" (truckers), and retailers (grocery stores). Most retailers depend on multiple large Third Party Logistics Providers (3PLs) which trade with suppliers or other 3PLs. However, retailers prefer to deal with a single accountable agent. Thus a centralized Fourth Party Logistics Provider (4PL) (typically the developer of the IT information system) is empowered to negotiate with, audit, and coordinate the supply chain.

Unfortunately, the 4PL forms an additional layer of bureaucracy as a result, and tends to collude with 3PLs against the retailers by exploiting information asymmetry against the retailer (Polim, Hu, and Kumara 2017). The layers of middlemen in the supply chain also incur tremendous costs, inefficiency and waste (Ramirez 2017). Retailers and in turn, end customers pay large markups to middlemen that aggregate supply. If middlemen are ever pressured by retailers to lower costs, costs are eventually pushed down to the farmers and truckers doing the actual work, reducing their growth. And the layers of middlemen in general brings an increased risk of factors that can go wrong, with every transport needed, more warehouses to store in, all adds up to longer lead time and greater risk of spoilage/depreciation (Ramirez 2017).

1.2 Summary of Work and Contributions

This thesis proposes a blockchain peer-to-peer smart contract system for a megacity grocery supply chain where retailers, suppliers, and logistics providers can bid directly for contracts, analyze market trends, trace product flow for food safety, and reduce costs by cutting layers. Although existing centralized systems work well within organizations that provide complete vertical integration in their supply chain, data transfer between incompatible systems tends to be conducted through paperwork or spreadsheets, which is subject to inefficiencies and nearsightedness.

A blockchain can supplement or take the place of centralized fourth party logistics providers and reduce the layers of middlemen in the supply chain by connecting suppliers, retailers, and 3PL directly. The blockchain also provides a distributed ledger that all participants would have a copy of and can contribute to without repudiation. The blockchain makes a peer-to-peer contract system feasible, although it works best when there are multiple groups that have competing interests and need to have a balance of power to ensure trustworthiness. The competing interests of multiple logistics providers and multiple retailers would reduce the chance of collusion especially with more participants.

The main question explored is, *"Can we use the blockchain to create a sharing* *economy for supply chain logistics and contract negotiation, linking retailers, logistics providers, and suppliers directly?"* A sharing economy is found in transformative albeit centralized IT examples such as Airbnb and Uber.

Related work on the subject does exist of which important insights are described in Chapter 4, but this thesis differs whereby existing distributed solutions to **the procurement problem** as described by Satapathy 1999 and Ramirez 2017 are applied to develop a possible methodology for building a blockchain implementation. This methodology simplifies the core components of this blockchain supply chain, but can be extended with modules to fit the needs of more complex megacities.

The implementation explored in this thesis implements both the negotiation and logistics components of the methodology. Due to time limitations, not all aspects of the methodology described previously could be fully explored in this blockchain implementation, such as volume orders, inventory management, logistics provider tracking, or physical IoT devices.

This thesis simply aims to provide a basic code structure using Hyperledger, to show that it is possible for blockchain smart contracts to be applied to the procurement problem. It is left to future researchers and developers to explore further possibilities, test results, or solutions.

1.2.1 Key Possibilities

Supply chain data could be segregated into public transactions for all to view: such as logistics availability, food traceability, and shipment tracking. Some transactions can remain private to participants and authorized observers: such as nearby transporter pricing and price negotiation between suppliers and retailers. Such transactions can still leave behind a hash fingerprint on the public blockchain attesting that an agreement was reached and that there is a legitimate order for logistics providers to depend on, without including price data irrelevant to logistics providers.

Retailers and Suppliers would be able to see directly what the market has paid for transport and supply, and be aware of upcoming shortages. Retailers and Suppliers are incentivized through the ability to make volume orders to aggregate demand, instead of using the additional transport layers of a middleman. The retailer negotiates with one supplier and after accepting a price, makes a volume order to all other suppliers to see if that fits their minimum price. This process can be conducted in a permissioned blockchain, with only matching suppliers negotiating on demand to retailers.

Logistics providers are incentivized by gaining access to the transaction rates retailer purchases so they can match demand as soon as a shortage appears. This process can be conducted in a public blockchain. Smaller logistics providers would no longer have to depend on the whims of a broker, who can be jealously monopolistic with their relations with truckers, and also withholds cost savings from the retailers for their own gain.

From a traceability standpoint, the cost of sharing and querying data between all participants is significantly reduced, food safety issues can be pinpointed and mitigated as soon as they are detected, blame would be less likely to fall on the wrong party, counterfeit products would be impeded by their lack of provenance on the blockchain.

1.2.2 Limitations of the Blockchain

The blockchain still retains some limitations and considerations to be aware of. For one, although technology significantly extends the analysis capabilities of regulators and participants, the data generated is only as good as the people who check on them. Signatures, licensing, and second opinions form the basis of regulation, with paperwork and bureaucracy utilized to ensure compliance and provenance. And it is people that generate these certifications (of visual inspections, sensor) with their honor, livelihood, and reputations on the line. The blockchain allows these attestations to be recorded in an immutable, distributed ledger, but organizations and regulators must continue to maintain the trustworthiness of those signatures and the data. Another consideration that the smart contract code on the blockchain only has jurisdiction over the data in its ledger. Either people have to actually act on the information, incentives, and contracts given, or IoT devices certified by people would have to be set up to take real-world action upon receipt of a blockchain transaction. The blockchain should not be seen as a transcendental technology in bureaucracies, but as the next stage of a gradual evolution in ledgers with real-world enforcement.

The decentralization inherent in the blockchain can allow smaller, independent contractors to be more competitive, increase efficiency by reducing the layers of middlemen, and reduce the costs for them to adopt technology and get access to information (Ramirez 2017). Smaller suppliers can be aggregated and matched to retailers through volume orders without a middleman (Ramirez 2017). Logistics providers could be crowdsourced individual contractors that could be called upon anytime for small or rush deliveries, much like how rides can already be hailed on sharing economies such as Uber and Lyft. Megacities that have to cope with reduced warehouse space, greater demand for direct/fresh delivery will need improvements like these to sustain their growth (Polim, Hu, and Kumara 2017).

1.3 Thesis Outline

Chapter 2 discusses the Problem Definition of this thesis. The chapter starts with a brief introduction of existing information asymmetry between 3PLs and 4PLs. The existence of these layers of middlemen adds complexity, cost, and inefficiencies to the underlying procurement problem. Existing methods developed for eCommerce are applied to tackle the Procurement Problem in the supply chain with direct negotiations or volume orders. The chapter also considers the relevance of the blockchain to the procurement problem, and if so what form it could take. It continues with a discussion on how the potential utilization of blockchains and smart contracts can enforce a more securely and transparently track of contract bidding in a decentralized digital economy. It also considers what form such a blockchain could take.

Chapter 3 provides an in-depth analysis of the Theoretical Background of the procurement problem, the layers of middlemen, the blockchain, and smart contracts.

Chapter 4 explores related work that explores various areas of *the issue of blockchains in the supply chain*, as well as the applicability of *permissioned blockchain systems in the enterprise in general.* The aim is to pick out and synthesize many insights developed by these papers into this thesis, to push the field further.

Chapter 5 considers the methodology of the procurement problem in this thesis, describing the entities involved and the scenario. This methodology is somewhat idealized but provides a good setup for future supply chain implementations to be built upon.

Chapter 6 describes the implementation of our solution to the procurement problem using a proof-of-concept developed with Hyperledger. This implementation does not fully implement the methodology described due to time constraints, but as a proof-of-concept shows that further development is very feasible in terms of software.

Chapter 7 discusses the case studies that the implementation produced could apply towards, as well as future work worth exploring and a conclusion.

Appendix A contains Hyperledger model code that describe the assets, transactions, and participants involved in our implementation. Additional source code can be found at `https://github.com/lvw5264/hyperledger-megacity`.

Appendix B contains useful additional online links and videos that provide background knowledge regarding the blockchain.

Chapter 2 Problem Definition

2.1 Introduction

The main question explored is, *"Can we use the blockchain to create a sharing economy for supply chain logistics and contract negotiation, linking retailers, logistics providers, and suppliers directly?"* A sharing economy is found in transformative albeit centralized IT examples such as Airbnb and Uber.

eCommerce is defined as "commercial transactions conducted electronically on the Internet" (Polim, Hu, and Kumara 2017). Research into eCommerce as applied to the supply chain is explored to extract existing solutions to the procurement problem, as described by Satapathy 1999 and Ramirez 2017. Both of these existing solutions rely on the existence of a trusted third-party to enforce time, signatures, identity, and transaction order in their ledgers, which has led to the proliferation of Fourth Party Logistics Providers (4PLs) to take that responsibility.

The key insight building upon such research is whether the blockchain and smart contracts can help retailers, suppliers, and logistics providers match offers to inquiries directly. This way, the procurement problem can be solved similarly, just in a decentralized and direct manner without the need for a trusted third-party.

In the procurement problem, a retailer wants a product. Knowing that demand exists, the supplier announces product availability and provides a price quote on the market (Satapathy 1999). The total cost to the retailer is the product price plus the estimated shipping cost.

If the retailer does not agree on the supplier's price, they begin a negotiation process, where they make an offer or counteroffer several times until they convergence on price. The supplier and retailer agree to the final price, and a contract is signed.

If this single small supplier cannot produce enough volume alone, the retailer can initiate a volume order, which provide this agreed upon price to other suppliers. These other suppliers also join in if this price is above their minimum bid. If not enough suppliers joined at that price, the retailer can increase the bid or renegotiate with other suppliers until volume is filled (Ramirez 2017).

Next, to actually deliver the product from the supplier to the retailer, this contract is used to post a market indicating delivery needs. Logistics providers post their location and availability, which is matched by the market to both retailer and supplier pickup/delivery windows. The retailer then inquires for the rates from all logistic providers and negotiates this one on one.

In the current supply chain, such negotiations unfortunately do not tend to occur directly between suppliers (farmers), what we will call “independent” *logistics providers* (truckers), and retailers (grocery stores). Most retailers depend on multiple large Third Party Logistics Providers (3PLs) which trade with suppliers or other 3PLs. However, retailers prefer to deal with a single accountable agent. Thus a centralized Fourth Party Logistics Provider (4PL) (typically the developer of the IT information system) is empowered to negotiate with, audit, and coordinate the supply chain.

Unfortunately, the 4PLs forms an additional layer of bureaucracy as a result. The layers of middlemen in the supply chain also incur tremendous costs, inefficiency and waste (Ramirez 2017). Retailers and in turn, end customers pay large markups to middlemen that aggregate supply. If middlemen are ever pressured by retailers to lower costs, costs are eventually pushed down to the farmers and truckers doing the actual work, reducing their growth. And the layers of middlemen in general brings an increased risk of factors that can go wrong, with every transport needed, more warehouses to store in, all adds up to longer lead time and greater risk of spoilage/depreciation (Ramirez 2017). For megacities to sustain their growth, the burden of the layers of middlemen must be overcome.

2.1.1 The Procurement Problem

In the supply chain, the procurement problem is crucial to the procurement of products and services. Megacity supply chains will tend to utilize Build-to-Order

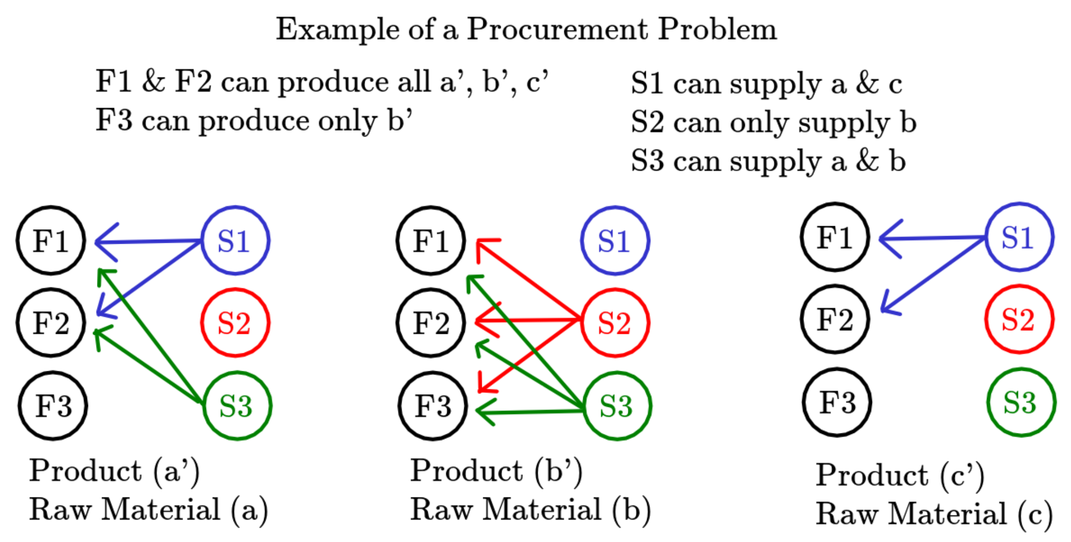


Figure 2.1: The procurement problem as defined by Satapathy (Satapathy 1999).

systems where delivery and production is just in time, especially as warehouse space becomes expensive in higher density cities and eCommerce fragments orders.

The supplier is either a farmer or food preparer that offers food for sale. The logistics provider can be either a single independent trucker as in Uber, or part of a traditionally hierarchical organization like a 3PL. The retailer is either a brick-and-mortar grocery store, a restaurant, an eCommerce platform facilitating deliveries straight to end customers, or even just end customers themselves, perhaps buying in bulk.

1. A retailer wants a product.
2. A supplier announces that they have the product and provides a price quote.
    - Cost includes cost of the product and the shipping cost
3. If the retailer does not agree on the cost, the retailer and supplier begin the negotiation process.
    - The bargaining process takes several rounds until convergence or abandonment.
    - Supplier and Customer agree to the final price.
4. The Supplier contacts logistics providers, announcing the transport task.

   - The transport task is to move an item from the supplier's location to the retailer's location.
   - The bargaining process takes several rounds until convergence or abandonment.
   - Delivery and Pickup windows for both retailer and supplier respectively could be taken into account.
   - Logistics Provider and Supplier agree to the final price.

5. When the logistics provider picks up the product, they conduct a signature "ceremony" to ensure that both the supplier and logistics provider were present, have agreed to the handover, and recorded the terms and starting measurements.
6. When the logistics provider delivers the product, they conduct a signature "ceremony" to ensure that both the retailer and logistics provider were present, agreed to the handover, and recorded the terms and starting measurements.

2.1.2 The Multiple Levels of Logistics Providers

Fourth Party Logistics Providers (4PLs) are delegated by retailers the responsibility to manage the activities of, make deals with, and audit the Third Party Logistics Providers (3PLs) in the supply chain (Polim, Hu, and Kumara 2017). Unfortunately, there drawbacks to utilizing the 4PL as a middleman, particularly regarding performance accountability (Polim, Hu, and Kumara 2017). The 4PL is unlikely to disclose all the performance metrics of their 3PLs, making it difficult to identify which 3PL is the root of issues (Polim, Hu, and Kumara 2017). Lacking information, the retailer can do little to find more information to maintain quality, sustainability, and price standards. There is often incentive to collude with the 3PLs against the interest of the retailers they are supposed to serve, where retailers would be left unaware of any deviations or excess margins (Polim, Hu, and Kumara 2017). In general, 4PLs tend to take large margins as the middlemen and increase costs in the supply chain (Polim, Hu, and Kumara 2017).

Disruption can be achieved by enabling a free market to be opened up to upstart 3PLs or individual crowdsourced transporters (Sun, Yan, and Zhang 2016). Instead of having to cultivate a relationship directly with a few retailers and 4PLs, 3PLs or transporters could get access to the entire market by simply entering the blockchain and building up a reputation (perhaps allowed to enter upon some open licensing scheme).

2.1.3 The Paperwork Problem

There are regulatory restrictions with food produce to protect the public that any system must address. In addition, there are meticulous reporting and custody requirements for each step of blood collection and transfer between organizations, which is often accomplished through bureaucratic paperwork or centralized databases by trusted third-party middlemen.

Although technology significantly extends the analysis capabilities of regulators and participants, the data generated is only as good as the people who check on them. Signatures, licensing, and second opinions form the basis of regulation, with paperwork and bureaucracy utilized to ensure compliance and provenance. And it is people that generate these certifications (of visual inspections, sensor) with their honor, livelihood, and reputations on the line.

A proposed blockchain tracking system produced with Hyperledger, aims to significantly reduce the latency involved in regulations that still depend strongly on paper. Despite increased use of centralized computer databases and Enterprise Resource Planning systems within organizations, system incompatibilities between multiple systems tend to force them to transfer data to the next step of the supply chain or to regulators using the lowest common denominator: paper, or sometimes excel spreadsheets.

When the data is in this form, it is not available outside of the organization currently holding the information, so the blood supply chain is near-sighted to supply surplus or excess demand, resulting in waste that puts lives at risk. This nearsightedness also makes tracing provenance difficult outside of a single organization, and only the overworked regulator would have the right to make full audits which may take a very long time.

This blockchain implementation is inspired by IBM's Hyperledger implementation with Maersk (IBM 2018[b]). It significantly reduces the paperwork required for shipping products, from a pile of 200 papers moving back and forth across the world and across multiple countries and companies, to a six-step process on the blockchain where the moment one group such as a customs officer uses their digital signature on the blockchain, the product is immediately certified to proceed in the supply chain, rather than waiting hours or days for the backlog of papers to be delivered (IBM 2018[b]).

2.2 The Applicability of the Blockchain

One of the most important considerations is to decide when the blockchain fits the use case and when it does not. No technology can truly fulfill every use case, or make miracles happen for that matter.

2.2.1 Choosing the Type of Blockchain for the Problem

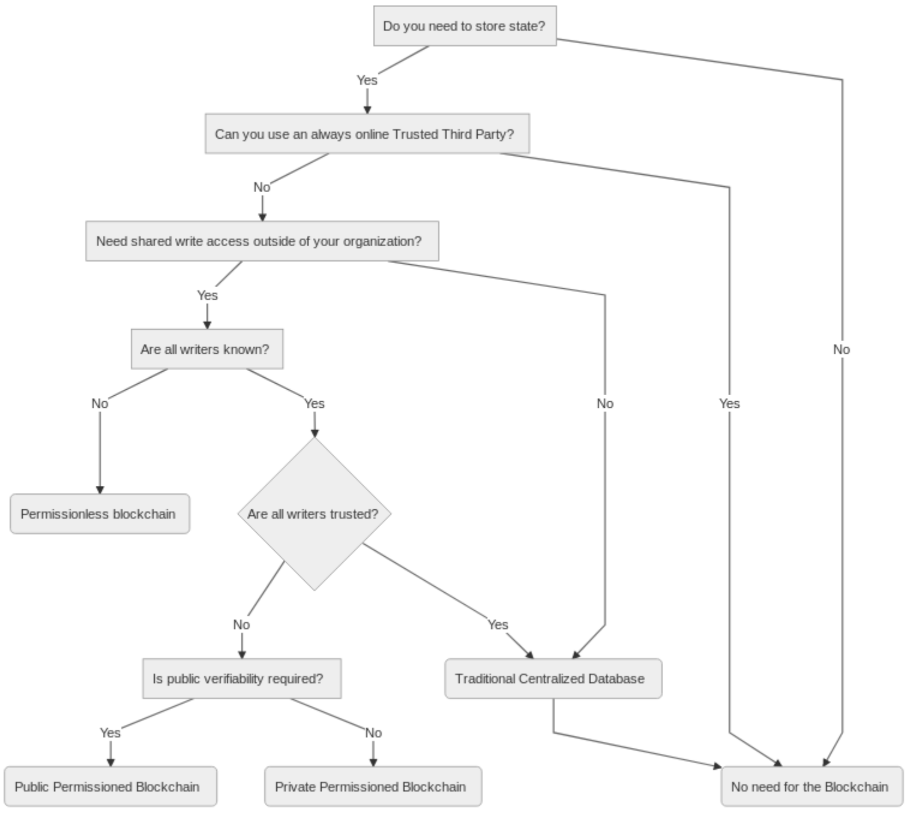

Do you need to store state?
Yes
No
Can you use an always online Trusted Third Party?
No
Yes
Need shared write access outside of your organization?
Yes
No
Are all writers known?
No
Yes
Permissionless blockchain
Are all writers trusted?
No
Yes
Is public verifiability required?
Traditional Centralized Database
Yes
No
Public Permissioned Blockchain
Private Permissioned Blockchain
No need for the Blockchain

Figure 2.2: Flowchart based on info from GSA and Johansson's 2018 thesis for deciding which type of blockchain to use. (Herman 2018), (Johansson 2018)

Multiple types of blockchains exist for enterprise purposes, primarily divided into **Permissionless Blockchains** which enforce no restriction on participating nodes and the data they can access (such as Bitcoin, Ethereum, etc), and **Permissioned Blockchains** which limit participants nodes and restrict data access.

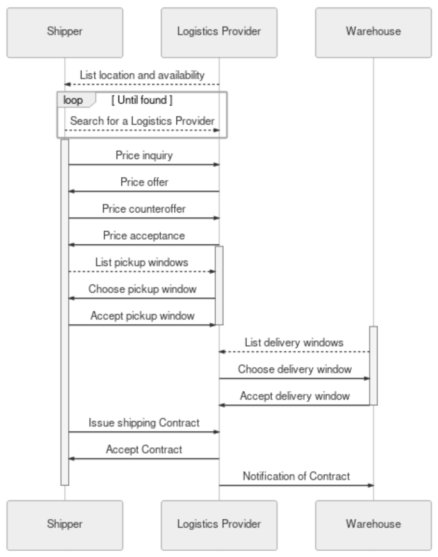
Permissioned blockchains mitigate the effect of having malicious miners enter a blockchain by enforcing strict controls over which miners and nodes are allowed to work on it (ideally, through a private intranet) and who is able to connect to it, perhaps with signed digital certificates. This way, a compromise would have to occur at the highest levels to allow malicious miners even just to get in, let alone establish more than 51% of network hashrate. However, this has the effect of limiting accessibility to a smaller consortium of approved nodes.

Permissioned Blockchains are further subdivided by Public and Private types, depending on whether public verifiability of transactions is needed. However, even private permissioned blockchains can provide all the proofs above for each transaction by posting a hash fingerprint to a public blockchain of some type that does not include any sensitive data.

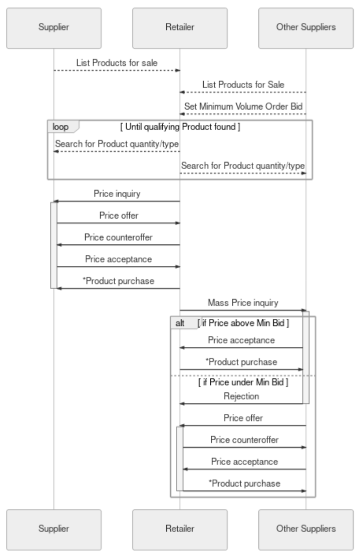
2.2.2 Relevance of the Blockchain to the Supply Chain

A blockchain may be able to supplement or take the place of centralized fourth party logistics providers and reduce the layers of middlemen in the supply chain by connecting suppliers, retailers, and 3rd Party Logistics Providers directly. The blockchain also provides a distributed ledger that all participants would have a copy of and can contribute to without repudiation, taking the place of paperwork and signatures. A blockchain allows all players in a supply chain to access, add, inspect data while restraining these players from changing and deleting existing data. The integration of a blockchain to a supply chain could possibly improve transparency of the transactions inside the supply chain, which is a pressing issue worth researching.

Smart Contracts are code on the blockchain that allows automated action to be taken on the ledger, such as minimum bids, automatic reorders, and volume orders. This allows many of the more mundane and complex elements of contract negotiation that middlemen deal with to be automated. Logistics providers could transparently show their availability to the entire market, including backhauls or



(a) How Logistics Providers can be matched to retailers without a middleman.



(b) Roots of the Merkle Tree in the Blockchain

Figure 2.3: The methodology applied by this thesis to use the blockchain to solve the procurement problem.

multiple hauls, reducing the dependence on a transport broker. Multiple small suppliers fill large volume orders as long as they are above their minimum bid, allowing small farmers to sell direct to retailers instead of relying on a middleman.

Smart Contracts in either Ethereum or Hyperledger are remarkably simple for what they make possible, if challenging to master, since for example a implementing a Domain Name System takes only a few lines to register a digital asset mapping a string to an IP address (Buterin, Wood, et al. 2018), and a voting investment board controlling the cryptocurrency in its smart contract by majority vote can be built in less than 100 lines (Buterin, Wood, et al. 2018).

The blockchain makes a peer-to-peer contract system feasible and automation possible, although it works best when there are multiple groups that have competing interests and need to have a balance of power to ensure trustworthiness. The competing interests of multiple logistics providers and multiple retailers would reduce the chance of collusion especially with more participants.

2.2.3 Applying Permissionless and Permissioned Blockchains to the Supply Chain

In the supply chain, we expect any provenance data required by the Produce Traceability Initiative (PTI) to be made public for verifiability, so a public permissioned blockchain fits this use case. Produce descriptions, locations, handling requirements, and shipment tracking are all public information that each entity in the supply chain are interested in (Produce Traceability Initiative 2018).

We also expect most other transactions such as price offers and other negotiation transactions to only be sent to involved parties and any notaries they both recognize (such as a regulator, arbitrator, or simply a third party record keeper). Such information can be business secrets of a confidential nature since a special deal offered to one retailer due to circumstance might invite jealousy from others.

However, it is expected that a logistics provider that ships the product from the shipper to the retailer would need to be sure that a sale actually was reached between existing retailers and suppliers. Thus, a public product listing should show the minimal listing and its change in state to relevant logistics providers before it would accept the shipping contract. This public product listing would not include price or negotiation data, which tends not to be tremendously relevant to the logistics provider.

2.2.4 Infrastructure Needs and Costs of the Blockchain

The infrastructure that the blockchain will take to build is likely to be less than what one centralized Fourth Party Logistics Providers (4PLs) would typically take on in an IT system. However, there is a larger diffusion of effort and cost required for all main participants, and there are new issues to consider such as market conditions, economic motivations, and ease of deployment of nodes.

In public permissionless blockchains (Ethereum, Hyperledger (configured for public access)), the general public provides the infrastructure to verify transactions, build consensus, and serve the ledger peer-to-peer, incentivized by the ability to produce cryptocurrency or the ability to participate in the consensus. This allows light nodes using Simplified Payment Verification (SPV) that require significantly less space. Thus, a flatter participation mechanism can come about whereby independent contractors competing with can simply operate light wallets on smart phones or IoT devices to communicate with the blockchain, no longer depending on a centralized server. Larger organizations or less organized flocks of “pools” can choose to operate their own full nodes to provide themselves greater access to data and stronger resistance to malicious nodes.

Ethereum has amassed a powerful GPU mining network second only to the strength of Bitcoin, but it would be more difficult for an upstart public permissionless blockchain to accumulate enough mining power to stave off attacks or maintain decentralization. Some caveats are that miners need economic incentive to continue building the consensus or they will simply move elsewhere, reducing the power of the network. As such, most successful public permissioned blockchain smart contract solutions, such as Slock.it and numerous ICOs, have been fully dependent on Ethereum’s blockchain rather than using their own, but this has consequences in that every Ethereum app is subject to longer confirmation times during high traffic periods, which may be unacceptable for other use cases.

Finally, the current dominant consensus algorithm on Bitcoin, Ethereum, and most other public permissioned blockchains is Proof-of-Work, which has a limit of how many transactions it can verify per second. The solution thus far has been to simply incentivize more miners to join with temporarily heightened transaction fees. Unfortunately, mining hashrate often cannot grow exponentially to match the rapid growth of these networks. For one, mining can sometimes be a losing proposition and a money sink. Another issue is that mining consumes a tremendous amount of resources in terms of hardware and electricity for the service provided, which is unsustainable in the long run. Often times if an investor just held the cryptocurrency instead of mining, they would be all the richer.

In private permissioned blockchains (JP Morgan Quorum, Hyperledger when configured for private consensus), each leader of an organization must run a node to participate and receive any data in the first place. Participating organizations can join and get access by various processes such as majority vote from existing organizations, invitation via contract by one, or other processes which will have to explored through future works. Consensus algorithms typically use “voting” consensus instead of proof-of-work style “lottery” consensus, significantly increasing the speed of transactions. The motivations to run a node are getting full access to data, having a stake in any consensus on the ledger, and independent verification of the blockchain using their own copy of the ledger instead of others.

A permissioned blockchain node can be considered much like in Satapathy's thesis (Satapathy 1999) to be a Master Agent, under the control of the leadership of an organization. Servant agents (representing employees accountable to the leadership controlling the node) create an account on this node's centralized database to save transactions (such as regarding trip cost and estimates) to it's node. When Servant agents commit these transactions, the Master Agent's node forwards this transactions to the other Master Agent's nodes, thereby committing it in their shared ledger.Employees under their leader's node would likely operate accounts under this node in this usual centralized manner or in ways integrated with existing centralized systems. Decentralization in permissioned blockchains is for information records between organizations (or between subsidiaries), not necessarily within hierarchical organizations.

Optimization, user interfaces, and full practicality is left to future researchers to consider, though papers exploring some of these topics are provided in the Literature Review section, with one providing the Unified Theory of Use and Acceptance of Technology (UTAUT), which is a particular acceptance metric that is useful to apply (Francisco and Swanson 2018). As Bjorn Johansson states in his MS Thesis, "For organizations who want to get in on the ground floor and help develop the underlying systems and standards this is the time to do so (Johansson 2018)". "For those not interested in doing that or who just wants to use finished software the blockchain technology sector is not yet ready for them. It will likely be years before it is (Johansson 2018)".

2.3 The Blockchain and Smart Contracts

The Blockchain is a mechanism for creating a decentralized trust network, whereby the eponymous Blockchain is a sequential ledger of public/private key signed transactions made from user to user, and where only the key holder can receive or send transactions. In one of the first and most prevalent algorithms, Proof-of-Work, Blockchain secures itself with a network of transaction verifying computers (**Miners**), which are working purely out of self-interest since the Proof-of-Work verification process produces the very tokens that create transactions on the network (Nakamoto 2008).

An example use case of the Blockchain is within its first application, cryptocurrencies such as **Bitcoin**, whose global peer-to-peer network of miners bring in tremendous computing power to ensure the trustworthiness of this decentralized cryptocurrency, adjusts its own interest rate, and protects against fraudulent transactions without centralized authorities or large bureaucracies (Nakamoto 2008). The Blockchain holds the promise of an even greater use case provided by Smart Contracts and Cryptotokens, as seen in the major cryptocurrency Ethereum (Buterin, Wood, et al. 2018).

The Ethereum blockchain innovates upon Bitcoin by allowing code to be embedded into this ledger and executed by miners. This code, dubbed **Smart Contracts**, have two or more parties agree to terms of the code, miners check the conditions, and then execute the actions when the conditions pass (Szabo 1997). The existence of the code on the blockchain provides for the verification, funds transfer, and nonrepudiation of its conditions and actions upon signing parties without needing a central authority to arbitrate upon them (Buterin, Wood, et al. 2018).

Cryptotokens are a derivative product of Smart Contracts that produce units of value much like cryptocurrency, but are verified by the existing miners of the Ethereum blockchain and thus eliminate the need to secure separate Blockchains. These cryptotokens can then be used to function as essentially any kind of financial instrument, from coupons to tickets for goods and services, or used alongside Smart Contracts to produce voting stocks and bonds, software or music licenses, even an entire investment organization that votes on how its funds are used (Vogelsteller, Buterin, et al. 2018). Smart Contracts and Cryptotokens thus allow the Blockchain to sustain the frameworks necessary to sustain a true decentralized digital economy.

2.3.1 Three Combined Applications of Permissionless and Permissioned Blockchains

Three possible approaches can be applied that utilize permissionless and permissioned blockchains, perhaps even in combination, to meet the needs of market participants.

- **All public blockchain** - All entities and smart contracts are on a public permissionless blockchain with almost all information visible and open participation. This is useful when there is to be a decentralized open marketplace, such as in OpenBazaar. It is also very necessary for public accountability, such as in smart contract made investment groups (Decentralized Anonymous Organizations). However, not all use cases benefit from a lack of confidentiality, and not all markets are open to unverified participants.

  - Off-chain Data Transfer - Pure private transfer of signed encrypted data between two participants without a ledger, though still tied to the public blockchain through hash fingerprints recorded publicly. Order tickets do not progress until all valid hashed information has been sent through whatever off-chain means and matched. Nodes under the control of the entities must also stay always on. Decentralization at the data storage level can still be accomplished, but this tends to be up to the discretion participants. However, this results in one entities' deliveries stopping if their single centralized node goes down: although it does not directly affect the others, given that the supply chain is all linked together it can cause delays down the line.

- **Dual Chain** - Both a public permissionless and private permissioned blockchain is used. This way, select suppliers and retailers can keep their supply rates and other transaction data out of the hands of other negotiators and the logistics providers, who are provided order tickets that are created and resolved on the public permissionless blockchain.
  - Private Blockchain - Existing suppliers and retailers vote to allow new entities to join the blockchain as nodes, perhaps based on whether they got certification. Suppliers and retailers then make order contracts on this private blockchain. Once an order is concluded, the order ticket is also sent to the public blockchain by a smart contract on this private blockchain.
  - Off-chain - The private blockchain acts as a stronger, more decentralized off-chain solution. Data can simply be encrypted and delivered only to relevant participants or notaries, with a hash fingerprint of this transfer and signature verification submitted to the public permissionless blockchain.

- **Public Permissionless Blockchain** - For the use of logistics providers, which they can freely access from anywhere, and anyone can participate making an Uber-style sharing economy of independent contractors possible, although registered logistics providers with reputations are more likely to be picked. Only the information pertinent to the logistics provider about items to be shipped is provided while browsing. General locations of where the supplier and customer are provided for browsing, and estimates with a margin of error are built upon this. Once the transporter accepts a contract, the private permissioned blockchain functions as an or

Chapter 3 Background

3.1 Supply Chain Theory

3.1.1 Megacity Logistics

It is expected that 70% of the world population will live in cities by 2050, and make up 14-20% of global GDP (MIT Megacity Logistics Lab 2018). Logistics that brings in goods and services to the megacity increases the quality of life, making this incredible phenomenon of high population density not only possible, but desirable (MIT Megacity Logistics Lab 2018).

However, the rise of the megacity brings in drivers of “increased complexity of urban logistics networks that calls for new thinking (MIT Megacity Logistics Lab 2018)”. For one, a sharp increase in population and its density particularly in developing continues to put pressure on the efficiency and capacity of logistics providers (MIT Megacity Logistics Lab 2018). Next, e-commerce has matured and causes an increase in direct deliveries instead of traditional brick-and-mortar stores, which tremendously increases the complexity of transport but also fragments them (MIT Megacity Logistics Lab 2018). Finally, on top of all this cities are being reconfigured to fit the needs of higher population density by increasing development of public transit, which necessitates reducing road lanes, parking spaces, and space for warehouses that are desperately needed in the traditional supply chain (MIT Megacity Logistics Lab 2018).

To meet this challenge before being consumed by high food costs or needless wastes, the supply chain not only has to produce more to meet growing demand, but has to become more efficient to foster and accomodate increased production.

3.1.2 Types of Supply Chain Inefficiencies

Inefficiencies in the system are a major roadblock and the main source of increased costs.

Data warehousing inefficiencies come about when multiple third-party logistics providers (3PLs) collaborate, since it is very inefficient, difficult, and hard to trust when they all push to one database or even communicate data between entities using paper or other limited data sharing formats.

Operational inefficiencies manifest from the layers of middlemen in between and warehousing the product. These middlemen came into existence due to the distances involved. If there was a way to cut out the middlemen by connecting suppliers directly to transporters and retailers, costs, delay, and thus waste could reduced.

Competitive and inequality inefficiencies come about as a result of centralization and monopolization of relationships. As the pace of globalization progresses and supply chains become more and more difficult for a small business, let alone a single contractor to join in or comprehend. They tend to rely on joining ever larger corporations or outsource complex aspects of their business to middlemen or fourth party logistics providers better placed to add technological efficiency.

These middlemen become their single buyer or seller, but their centralized structure and power often leads to jealously monopolistic or abusive behavior, often passing the costs down to suppliers that have no power to refuse, while extracting large profits from end customers that constantly see prices increase (Ramirez 2017).

3.1.3 Logistics Provider Models

A First Party Logistics Provider (1PL) model has the retailer or manufacturer deal with the entire transport and warehousing on its own. The 1PL could theoretically handle all transport on its own with its own employees, but this is not necessarily the most efficient way to deal with it unless the company reaches a critical mass (Cardinal Commerce 2018). One example is Walmart producing products for its in-house brand, and transporting it to its own stores with its own vertically integrated supply chain infrastructure with every employee paid by the same company or its subsidiaries.

A second party logistics (2PL) model has the retailer or manufacturer outsource the actual transport with trucks, ships, or planes (Cardinal Commerce 2018). However, the retailer or manufacturer is still responsible for the *logistics* of scheduling and coordinating all such transports, production rates, and warehousing, with the 2PL only thinking in the short term of how many transport services they can sell now, since they tend to lack information about how it all fits together (Cardinal Commerce 2018). An example is a local farmer contracting with an independent trucker to ship some locally grown fruit to local restaurants, with both the restaurants and farmer coordinating on the logistics of where, when, and what to ship, and if any inventory should be maintained at warehouses.

A Third Party Logistics Provider (3PL) model has a company that provides a package of both transport and logistics for retailers or manufacturers. The retailer or manufacturer still works closely with the 3PL to judge their performance, negotiate prices, and consider production rates. For example, for a US grocery store to offer imported wine from France, it must contract with a 3PL such as DHL or UPS to coordinate the entire shipping, logistics, and warehouses to bring the wine over from the vineyards to store shelves (Cardinal Commerce 2018). The grocery store or its hired procurers have conduct the whole procurement process, to assess when to use DHL, UPS, Maersk or a combination of 3PLs, manage when they need the product, when to make purchases, whether to keep warehouses and distribution centers in the US or next to the manufacturer, bulk purchases to reduce fees or customs charges, etc.

A Fourth Party Logistics Provider (4PL) model allows retailers or manufacturers to outsource the entire procurement process as stated above, along with any negotiations and management with 3PLs (Cardinal Commerce 2018). The 4PL model has become popular due to the tremendous costs of worldwide supply chains, to deal with customs, multiple types of transport from long distances such as China, and move it to distribution centers. Some 3PLs only do one mode of transportation, but it may need multiple modes in different countries which quickly becomes a total mess. The 4PL takes care of everything, such as operational costs, insurance, administrative costs for managing the supply chain, all paperwork. This saves the tremendous costs of hiring a whole supply chain management division, and the 4PLs would have much more experience and leverage to get discounts.

4PLs however require a lot of trust that they are providing the best value and the best price. however, this trust can be abused. 4PLs might keep around a bad

3PL just out of relationships or kickbacks, and good 3PLs might be stuck outside the system.

3.1.4 Distributed and Collaborative Logistics Planning and Replanning under Uncertainty

Satapathy's thesis on the topic provides a foundational conception of the supply chain as it is seen in this thesis.

Classic supply chains prior to the emergence of e-commerce had information exchange/transactions among participants flow alongside the raw material and products, but in electronic commerce an integrated supply chain emerged with all players able to communicate to each other (Satapathy 1999, p. 9). Satapathy states that the supply chain model became an interdisciplinary problem as a result, not just critical to logistics and inventory management (Satapathy 1999, p. 9). To solve this problem he draws upon research on "distributed information processing, game theory related research on task negotiation and allocation, and research in field of operations research on determining optimal vehicle routes, inventory control policies, and task assignment (Satapathy 1999, p. 9)".

Satapathy states that by extending the results of the research using *real-time sensors*, a real-world integrated supply chain infrastructure for e-commerce can be built (Satapathy 1999). He also states that at the time of writing, real-time monitoring technologies not yet embraced by industries would be needed, such as "secured automated business transaction systems, vehicle monitoring and traffic prediction systems, sensor fusion, and embedded devices (Satapathy 1999, p. 8)".

However, these technologies did become commonplace a decade later, as he predicted. It is thus assumed in this thesis that Satapathy's plan can be taken to represent the current state of the supply chain, and from here we identify possible areas of improvement with the blockchain.

3.1.4.1 The Procurement Problem

Manufacturers in a supply chain often have to choose which suppliers to obtain raw materials from, which manufacturing sites to produce products taking into account the local customer market, and which Third Party Logistics Provider (3PL) to use (Satapathy 1999). Satapathy (1999) (Satapathy 1999) describes this issue as the

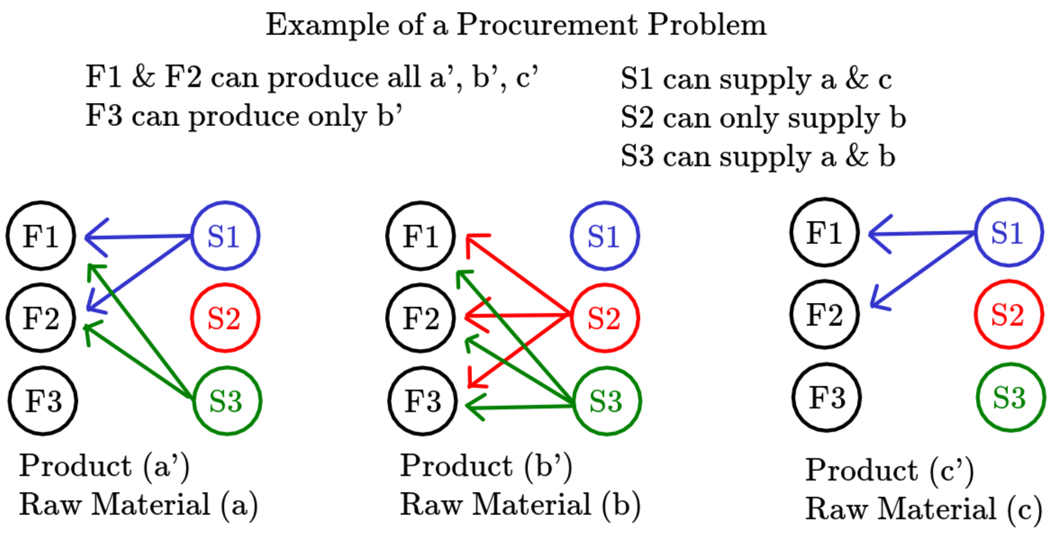


Figure 3.1: The procurement problem as defined by Satapathy (Satapathy 1999).

Procurement Problem. Three principles are defined.

1. "Minimization of bid (transportation cost) for the manufacturing company" (Satapathy 1999).
    - To simplify the problem, Satapathy assumes that the "cost and quality of raw materials and the final product are independent of the suppliers and the manufacturing sites" (Satapathy 1999).
    - Instead, Satapathy focuses on reducing the cost of transport between suppliers, manufacturing sites, and customer locations (Satapathy 1999).
2. "Maximization of payoff to" the 3PLs (Satapathy 1999).
    - In contrast, the freight companies are incentivized to find the the bids with the highest payoff to them, forming a market.
3. "Cost-optimal and cost-reliable estimation of the task-to-vehicle assignments by the freight companies" (Satapathy 1999).
    - It must be possible to estimate distance, time, and cost of the transport assignment.
4. "Reliable fulfillment of the orders by the drivers of the vehicles" (Satapathy 1999).

- Methodologies must be constructed to ensure that the shipping orders do not take too long to fulfill, are fulfilled correctly, and provisions are made to recover if they are not.

3.1.4.2 Satapathy's Research Objectives

Satapathy identifies three research objectives, which are quite relevant to a distributed ledger-based supply chain: just this time with the blockchain to accomplish this without a centralized trusted third-party.

The first objective is *"to develop a distributed (and collaborative) problem solving model, define the decision making entities and formalize their interactions"* (Satapathy 1999) (Satapathy 1999, p. 15).

A multiagent computational model is presented for this purpose. (Satapathy 1999) A *freight company* is defined in the model as being built of two computational entities: master agents, and slave agents (Satapathy 1999). A *master agent* "negotiates with the master agents of other freight companies for transportation tasks, and assigns sub-tasks to the drivers of its vehicles", who are represented by *slave agents* that generate routes for the driver (Satapathy 1999). These agents would be provided by the decision makers they represent with "intelligence, autonomy, persistence, and behavior" (Satapathy 1999, p. 6)

For the purpose of solving the procurement problem, a Belief-Desire-Intention (BDI) logical model was used to model the agents, which are defined in object-oriented Unified Modeling Language (UML) and who interact using "modified Knowledge Query and Manipulation Language (KQML)-based messages" (Satapathy 1999).

The second objective is *"to model a transportation task selection and distribution procedure that minimizes the transportation cost to the manufacturing company and increases the payoff to the freight companies, compared to the current practice"* (Satapathy 1999). In context, in 1999 any type of distributed internet-powered networking for the supply chain was still a new concept, and Amazon was just beginning to be recognized as a possible competitor.

A *discounted bargaining game model with alternating offers*, akin to a "multi-commodity reverse English auction" is provided (Satapathy 1999). The main issues resolved by this model is how a manufacturing company can minimize their bid for transporting raw materials, and how freight companies can increase their profit margin with competitive bidding (Satapathy 1999).

Satapathy conducts a simulation of how master agents can improve payoffs with “strategic bargaining procedure” and different “real-time cost information acquisition systems (Satapathy 1999)”. This way, if master agents “can predict the cost of the tasks accurately and follow a strategic bargaining procedure”, expensive cost information acqusition systems would not always be required (Satapathy 1999).

The third objective is *“how to determine the vehicle route dynamically”* (Satapathy 1999, p. 6), or “to develop a distributed collaborative algorithm to generate reliable vehicle routes under the assumption that there exists an opportunity for the drivers of the vehicles to transfer goods among each other (Satapathy 1999)”. Satapathy introduces the *Collaborative Robust Vehicle Route* (CRVR) algorithm (Satapathy 1999) “for reliable and optimal cost estimation of the tasks in real time”, solving a *single vehicle stochasitic pickup and delivery problem* (PDP) (Satapathy 1999).

The CRVR algorithm determines bottlenecks and feasible routes by factoring in local driver behavior along with real-time measurements of travel and service time (Satapathy 1999). Collaboration is then requested via announcements to delegate some orders to alleviate bottlenecks (Satapathy 1999). The results are that this algorithm “solves a stochastic PDP with ten transportation orders within reasonable time”, and that in some cases collaboration can reduce the cost of tasks (Satapathy 1999).

3.1.4.3 Types of Supply Chains

Satapathy identifies three types of supply chains “to which the research proposed in this thesis is applicable (Satapathy 1999, p. 7)”.

A **Build-to-Order** (BTO) supply chain, also known as just-in-time (JIT) manufacturing or pull-based systems, has “unpredictable demand, short product life cycle, and high profit margin (Satapathy 1999, p. 8)”. All required material is procured only when the customer makes an order, whereby suppliers send the material down the various levels of the supply chain (Satapathy 1999, p. 8). Examples include online configure-to-order laptops, fresh grocery supply chains, and the Toyota Production System.

A **Build-to-Plan** (BTP) supply chain, also known as push-based systems, has a stable product demand, long product life cycle, and low profit margin (Satapathy

1999, p. 8). Demand is typically predicted one week or multiple months in advance, and depending on how much inventory should be kept, "suppliers, manufacturers, and transporters are chosen for the delivery of product for that period of time (Satapathy 1999, p. 8)". Examples are "beverage, steel, and paper industries (Satapathy 1999, p. 8)".

A **Late Customization** (LC) supply chain produces a standardized product on a push-based system, which is then *localized* to market tastes on a pull-based system (Satapathy 1999, p. 8). Satapathy provides the example of electronic appliances and software products for global export (Satapathy 1999, p. 8).

Satapathy identifies BTO/pull-based supply chains as best placed to benefit from his real-time solution to the procurement problem, as well as the customization pull-based phase of LC supply chains (Satapathy 1999, p. 8). Apparently, Satapathy states that BTP/push-based supply chains would benefit only in some circumstances, such as the procurement problem of a freight company subcontracting to another freight company in a pull-based system (Satapathy 1999, p. 8). As such, in the implementation this thesis will do the same by exploring a supply chain of perishable goods.

3.1.4.4 Master and Servant Problem Solving Model

Satapathy identifies a freight company as composed of a hierarchy of Master and Servant agents (Satapathy 1999, p. 22). [1] This is a method of representing how information flows between multiple "high level decision makers" and down to the "lower level task planners and executors", which can occur non-sequentially (Satapathy 1999, p. 22).

This applies well to implementation using a permissioned blockchain. As decentralization works between organizations, since organizations are not beholden to each other beyond through contracts. Full decentralization does not necessarily have to exist within an organization.

- **Master agents** are authorized to receive and negotiate agreements from other

[1]Satapathy identifies his agents with the common but unwieldy computing term "Master/Slave" agent (Satapathy 1999, p. 21). This is conceptually appropriate for programs or machinery that are ordered around by masters and bound by code to their decisions, but becomes politically charged when sentient beings control these agents, who are employees bound by a wage, not slaves working for nothing. Thus, we will call them Masters and Servants, as servants at least have wages and can quit at the end of their contract like employees.

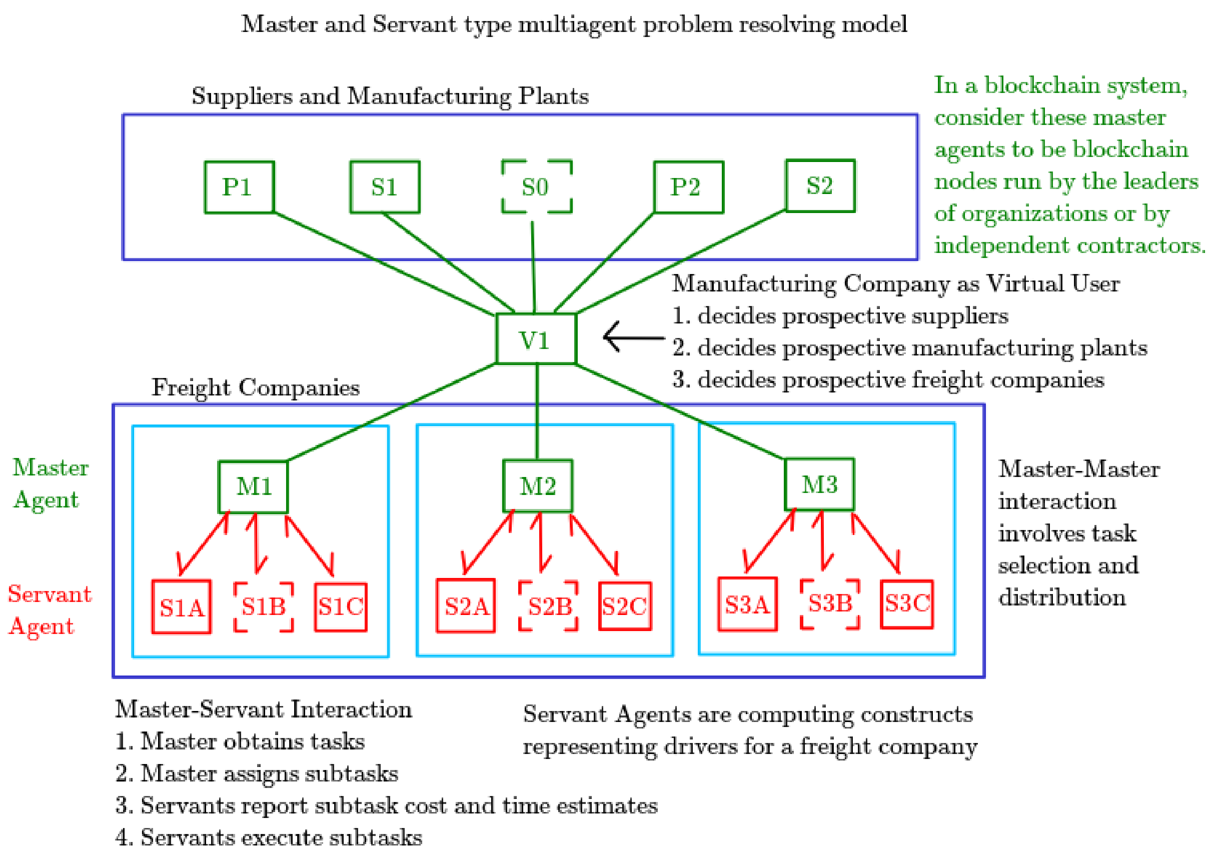

Master and Servant type multiagent problem resolving model
Suppliers and Manufacturing Plants
P1
S1
S0
P2
S2
In a blockchain system, consider these master agents to be blockchain nodes run by the leaders of organizations or by independent contractors.
V1
Manufacturing Company as Virtual User
1. decides prospective suppliers
2. decides prospective manufacturing plants
3. decides prospective freight companies
Freight Companies
Master Agent
Servant Agent
M1
M2
M3
S1A
S1B
S1C
S2A
S2B
S2C
S3A
S3B
S3C
Master-Master interaction involves task selection and distribution
Master-Servant Interaction
1. Master obtains tasks
2. Master assigns subtasks
3. Servants report subtask cost and time estimates
4. Servants execute subtasks
Servant Agents are computing constructs representing drivers for a freight company

Figure 3.2: Master and Servant Agent Interaction as defined by Satapathy (Satapathy 1999).

master agents in other freight companies, or by extension other subdivisions of the same company (Satapathy 1999, p. 22). They then separate the task in the agreement into subtasks and order Servant agents to conduct them (Satapathy 1999, p. 22).

  – Master agents can be considered in permissioned blockchains to be blockchain nodes run by an organization's leaders. The master agent would have to right to forward or refuse transactions made by its servant agents, negotiate new logistics contracts with other master nodes, and participate in the consensus algorithm to decide on the next block or other pertinent settings.

- **Servant agents** represent the freight company's drivers, and generate a *vehicle route plan* for the driver given the transport tasks passed down from the master (Satapathy 1999, p. 22). Servant agents use the plan to compute the expected cost and likelihood of completing the task, the results of which can be passed on to the masters (Satapathy 1999, p. 23). Servant agents within a freight company can interact to produce a collaborative robust plan, which may delegate some tasks to another (Satapathy 1999, p. 23).

  – Servant agents can be considered in permissioned blockchains to represent drivers holding accounts and communicating through blockchain nodes run by an organization. The servant agents create transactions for that organization node to forward using conventional centralized database, and also view transactions using the organiation local node perhaps with a webapp or mobile app interface. Within the organization, servants are still hierarchically structured and accountable to organization masters.

- **Virtual users** represent a manufacturer agent that works with the freight company's master agents to select and distribute transport tasks (Satapathy 1999, p. 23).

3.1.4.5 Workflow

1. The task selection and distribution stage: "A virtual user requesting to solve a procurement problem" (Satapathy 1999, p. 24).

    - Tasks are to be selected and distributed among freight companies by the virtual user (Satapathy 1999, p. 24). Master agents work with servants to determine expected cost, which is the "task presentation" step (Satapathy 1999, p. 24). The master agents come to a consensus on a final agreement as the product of this stage (Satapathy 1999, p. 24).

2. The task and servant assignment stage: "A virtual user approving the final agreement submitted by a group of masters" (Satapathy 1999, p. 24).

    - When the master agents come to the final agreement, they still await virtual user approval. When this occurs, the masters assign subtasks to servants to finish the tasks (Satapathy 1999, p. 27). This is typically done by picking servants that incur the least costs.

3. The task replanning stage: “The subtask failure reported or detected by a” servant (Satapathy 1999, p. 24).

   - If a delivery failure has a master approved contingency plan, the servant agent directly passes the subtask to another servant agent.
   - If there is no alternate plan, the servant agent reports the failure to the master, the master conducts the “task presentation” stage again, and reassigns the remaining subtasks. (Satapathy 1999, p. 27).
   - If no cost effective reassignments are possible, the master agent subcontracts this subtask to other master agents as if it were a virtual user auctioning a procurement problem, eating the cost of additional transport as a loss (Satapathy 1999, p. 24).
   - If other master agents are unavailable to subcontract the subtask in time, the master agent informs the use rand incurs a penalty from the user (Satapathy 1999, p. 24).

3.2 A Sharing Economy for the Supply Chain with eHarvestHub

eHarvestHub is a startup aiming to cut out the middleman in the supply chain. They have a decently implemented app working for a small but large volume group of farmers, as well as a large group of truckers. The startup also aims to produce a blockchain and smart contract implementation using Ethereum smart contracts and cryptotokens to significantly reduce paperwork, increase provenance quality, and allow for blockchain loan financing using the supply chain provenance as credit history (Ramirez 2017). Further future work with eHarvestHub building upon the concepts in this thesis are applicable to see and test the real world effectiveness of a blockchain-enabled supply chain.

As stated in a review of eHarvestHub by Pablo Eder, “The biggest reason that farmers require a middleman, is to help them distribute their produce among hundreds of grocers in multiple regions. In order to solve the transportation problem, the platform also incorporates a reward system for truckers, to move the produce” (Eder 2017).

The eHarvestHub whitepaper explains that supply chain inefficiencies manifest from information asymmetry between middlemen (who deals with the market and paperwork), farmers, and truckers. Despite the fact that farmers and truckers take the majority of the risk and do most of the work, and the fact that food prices have grown ever higher at grocery stores, the grocery stores demand lower costs from middlemen, who then reduce farmers and trucker profits (Ramirez 2017).

In existing systems, grocers could not rely on contracting with many small farmers, due to the need for volume reliability and compliance micromanagement. But grocers and in turn, customers pay large markups to middlemen as a result. And the layers of middlemen in general brings an increased risk of factors that can go wrong, with every transport needed, more warehouses to store in, all adds up to longer lead time and greater risk of spoilage/depreciation.

Therefore, their new app ecosystem matches small farmers, truckers, and customers to each other, empowering them to form actual free markets and direct competition, rather than be subject to monopolizing middlemen that extract profits from all sides. The decentralized information infrastructure of eHarvestHub could thus cut costs for customers not by pressuring actual producers and transporters, but by cutting out the middlemen.

This related work provides a strong social justification for the concept in this thesis. It is one thing to state that fourth party logistics providers could and should be replaced to decrease costs, it is another to hear poignant stories in the whitepaper of small farmers who have no choice to let produce rot in the field due to middlemen jealously guarding their monopoly by inducing such deadweight loss when they are not purchasing. There were independent truckers who saved up for their own truck only to be at the mercy of brokers who take most of the profit margin, ditch prospective truckers as they drive to the destination for the lowest bidder, and wait at least 30 days to pay them.

3.2.1 The Current State of Supply Chain Middlemen

Food imports are crucial to providing all types of fresh produce year round, an esteemed luxury in developed countries and a main export market for developing countries. This demands quality and standards and relationship with foreign grocers, resulting often in foreign dominance and heavy layers middlemen to conduct the price arbitrage for the farmers.

Layers of bureaucracy also came about due to need for certification standards of reliability, quality, availability, and most of all relations with foreign purchasers, all of which was only provided by specialized middlemen that spend their time dealing with paperwork so farmers and truckers could focus on working. However, the result is that these middlemen retain significant information asymmetry, hold a lot of power, all while aiming to protect their own profit margins against downward pressure from grocers, which is thus pushed down to workers who are least able to resist.

Even though many technological improvements to the supply chain have been developed, the majority have been implemented by and to enhance the power, information asymmetry of middlemen or grocers at the expense of actual workers. The whitepaper states that much of the software comes at high costs and high learning curves that farmers simply cannot afford at their low profit margin (Ramirez 2017).

In comparison, the middlemen are best equipped to reduce their own costs and better comply with regulations through technology. These cost reductions can only go so far due to the inherent inefficiencies of multiple layers of middlemen, causing food to "zig-zag" as much as ten times through the supply chain, risking further depreciation and quality loss.

Despite the fact that customers pay ever higher prices for food, grocers still aim to aggressively reduce their costs, which pushes middlemen to pass the costs down to the farmers and truckers. Even after the causes of necessary costs fade, the price reductions remain unchanged or go lower. Thus, despite the high prices, the actual workers do not see higher profits that could contribute to actual expansion to restore supply and bring prices back to equilibrium.

3.2.2 Sources of Markups and Inefficiency

The current situation of high markups and inefficiencies in the food supply chain as described by eHarvestHub is as follows:

1. *Grocers* demand prices as low as the market can bear, even if the customer still ends up paying high prices to the grocer as profit.
2. *Middlemen.* have to satisfy and compete for the grocer's prices, but they do not want to cut their profit margin or risk their own existence. As a result,

they push the costs and risks down to the farmer as much as possible often with little regard for the profit margins of farmers or truckers.

3. *Farmers/Truckers* have no one else to push costs to, and have little leverage against the middlemen since as stated in the prior example the middlemen tend to have little competition. Thus, the middlemen can demand fees even if the product fails to be sold, and the risks of depreciation or delays through the arcane layers in the supply chain are borne ultimately by these groups.

The whitepaper provides the example of two interviews with a farmer and a trucker (Ramirez 2017). An independent small farmer may own their own land, but remain at the mercy of a single middleman, who jealously guard their information asymmetry and monopoly by pressuring farmers not to sell or transport to the end customer or other middlemen, even though this practice leads produce and farmer profits to rot on the ground. As a result, despite improvements in production and technology, farmer profits become slimmer as costs get passed down to them, and quality and production efficiency declines.

As for independent truckers, despite the fact that they own their own trucks and do the actual work of transportation, they have no choice but to rely on brokers to make deals for them while they spend almost all their time with eyes on the road, another relationship ripe for abuse due to lack of competition. Unfortunately, these broker contracts aren't secure for truckers, resulting in wasted money and time for truckers, but cheaper and often lower quality ones are incentivized by the broker's thirst for lower prices. the negative externality persists since the broker puts all the risk on the truckers. All this while truckers get their payments delayed up to 30 days later, due to the slow flow of funds.

3.3 Foundational Concepts of the Blockchain

Before diving into implementations of the Blockchain and Smart Contracts in the supply chain, foundational concepts are explored in this section, drawn from the Bitcoin and Ethereum Whitepapers, and Hyperledger documentation. This was done to define certain terminology and common knowledge that emerged when the Bitcoin whitepaper described blockchains for the first time, and provide the justification for why the blockchain was the right system for its first useful application, cryptocurrency.

However, these concepts themselves are dependent on good understanding of the field of modern cryptography. Bjorn Johansson's master's thesis on "Assessing blockchain technology for Transport Data Logger" provides a great overview of the field. It also provides a guide with contents described below on which cryptography concepts are prerequisites to understanding which elements of the blockchain (Johansson 2018, p. 9).

- Hashing
  - Proof-of-work
  - Signing
- Symmetric cryptography
  - Sharing secret keys
- Asymmetric cryptography
  - Signing
  - Digital identities
  - Permissioned blockchains
- Signing
  - Proving origin/integrity of data
  - Permissioned blockchains
- Digital identities
  - Signing
  - Permissioned blockchains

Figure 3.3: A fundamental concept of cryptography is given as the top bullet, and the fundamental concepts of the blockchain that it helps explain is given as the subbullets (Johansson 2018, p. 9).

When the Bitcoin whitepaper (Nakamoto 2008) was released by an anonymous writer known as "Satoshi Nakamoto", the world encountered a practical solution not only to the *double spending problem* in distributed ledgers and the elusive *Byzantine Generals' problem*, but also how a digital society could be organized in a complex system and establish trustworthy, irreversible transactions without monolithic central authorities. This proposed cryptocurrency would be known a Bitcoin, and the cryptographic trust system outlined by the whitepaper as the **Blockchain**. (Nakamoto 2008).

Following this example, Bitcoin, Ethereum, and various other cryptocurrencies built with the Blockchain have proven themselves as decentralized digital currencies, trusted worldwide as both a store of value and a medium of exchange by hundreds of thousands of traders, investors, merchants, users, and of course, thieves.

Furthermore, the Blockchain can revolutionize aspects of society other than currency. One major blockchain network Ethereum, allows the creation of **Smart Contracts**: whereby code is embedded with transactions on their blockchain, which is then run by miners that monitor the conditions in the code, and execute the action when those conditions are fulfilled, and **Cryptotokens** whose holders are the interested parties in such contracts, able to vote or redeem goods or services from the issuer (Buterin, Wood, et al. 2018).

3.3.1 Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto's foundational whitepaper (Nakamoto 2008) defines a system to create an "electronic coin" using a peer-to-peer network, commonly referred to as **Cryptocurrency**. Since this paper introduces the first practical vision of the Blockchain, various fundamental concepts and definitions spring from this paper.

Bitcoin was not born out of nothing. The elusive writer, Satoshi Nakamoto, made citations to Wei Dai's b-money which proposed a framework of decentralized consensus to build a cryptocurrency, but lacked the implementation details (Buterin, Wood, et al. 2018). Nakamoto was likely also inspired by many concepts in the broader bodies of research, such as Hal Finney's 2005 concept of "reusable proofs of work", that brought together Wei Dai's b-money and Adam Back's Hashcash computation puzzles, but relied on Trusted Computing cryptographic modules that were uncommon at the time (Buterin, Wood, et al. 2018). The influence of the Bittorrent network, which allows decentralized seeds to reliably distribute a single set of data in a peer-to-peer network without corruption or modification, is reflected in Bitcoin's name.

However, the Bitcoin whitepaper (Nakamoto 2008) and Bitcoin implementation melded these concepts elegantly into a decentralized, accessible, yet practically immutable network to facilitate digital cryptocurrency with trustless nodes.

Bitcoin's Proof-of-Work algorithm was a breakthrough development in the field of cryptocurrencies, since it provides a simple and generally effective algorithm for decentralized nodes to come to a *consensus* on updates to the state of the ledger

(Buterin, Wood, et al. 2018). Proof-of-Work also allows any node to participate in forming the consensus in proportion to the amount of computing power they bring to the network. Most of all, Proof-of-Work provides economic incentive for nodes to strengthen the computing power of the network, since nodes that successfully extend the ledger with updated blocks are rewarded with newly minted cryptocurrency value (Buterin, Wood, et al. 2018).

3.3.1.1 The Byzantine Generals' Problem

Leslie Lamport, Robert Shostak and Marshall Pease in their 1982 paper, *"The Byzantine Generals Problem"* (Lamport, Shostak, and Pease 1982), where a group of generals (computers) each controlling an army surround a city, and must unanimously decide whether to attack or retreat (among other complex sequences of strategies), using only slow and unreliable messengers (transmission networks).

If the generals make a united effort, they would most likely succeed through overwhelming force, and would incur no damage if they made a united retreat. However, if some generals fail to join the assault simultaneously, the lone attacking generals could be routed and bear the most damage. Some generals could be traitorous with the intent of intentionally undermining the attack through inaction, and the problem is complicated by the distance between them and the possibility of misinformation by their messengers.

As such, the Byzantine General's Problem establishes a common issue where networked computers must somehow come to a **consensus** on a sequential strategy (or ledger) in a decentralized manner, despite the issues of distance and conflicting individual motivations that can undermine a collective action.

Related to this problem in the specific context of distributed ledgers is the *double-spending problem*, which is the key problem with a proposed solution in the Bitcoin whitepaper (Nakamoto 2008) as described in the next section.

3.3.1.2 The Distributed Ledger and the Double Spending Problem

Public key cryptography ensures that only a single key holder can attest that they are the recipients or senders of a message, through the ability to sign data or encrypt data with their private key. Anyone can verify or decrypt with their public key, as well as the other way around.

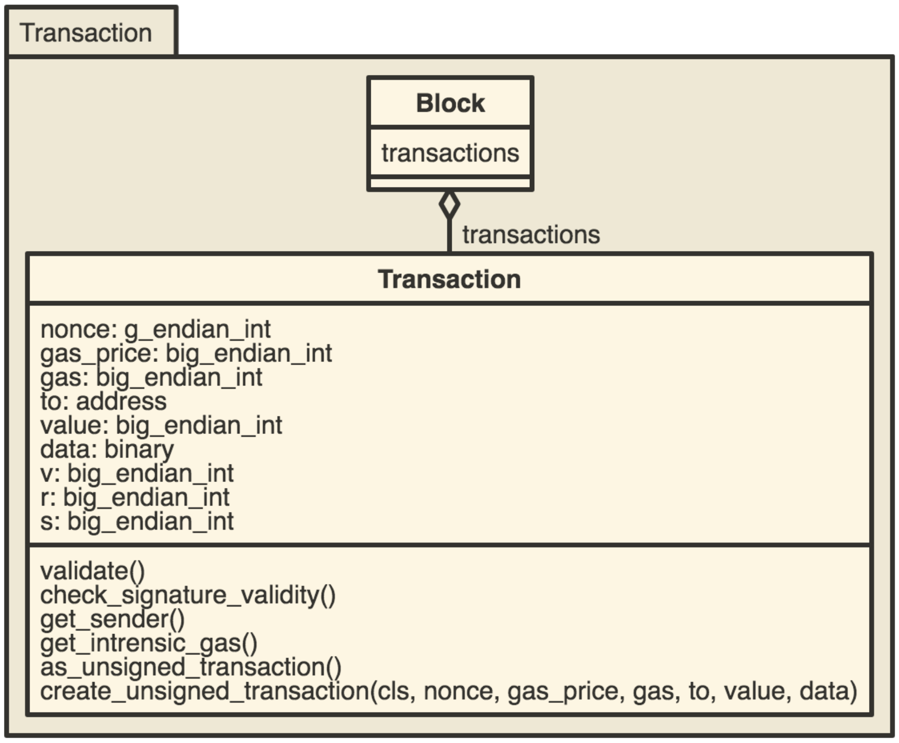


Figure 3.4: A transaction on a block of a blockchain's distributed ledger (Limited 2018).

As such, each owner holds a public/private keypair, (defined as an **account**), which can hold a cryptocurrency value that is continuous and fungible (1 coin, 100 coin, 0.001 coin). An owner of account with a cryptocurrency value can transfer it to a payee (an owner of another account) by initiating a **transaction**, which is a message that holds the value to deduct from their own account and add to a payee's account. The transaction contains a "digitally signed hash of the previous transaction along with the public key of the payee (Nakamoto 2008, p. 2)". The resulting list of transactions and its chain of signatures is *ledger*, where signatures can be verified to check the chain of ownership of that cryptocurrency value.

However, this chain of signatures is vulnerable to the **double spending problem**, whereby there is no way for the payee to verify that the cryptocurrency value was not already spent in other transactions. This problem is commonly resolved by "introducing a *trusted central authority* that checks every transaction for double spending, but then the fate of the entire money system depends on the company running the (authority) (Nakamoto 2008, p. 2)". The trusted central authority would have to be aware of all transactions and timestamp them to determine which arrived first.

3.3.1.3 The Proof-of-Work Algorithm

If there is to be a decentralized solution, transactions would have to be publicly announced, and the majority of nodes on the peer-to-peer network would have to determine that the transaction was the first received before it can be added to their ledgers, a process known as **consensus**. This would produce "a single history of the order in which (transactions) were received (Nakamoto 2008, p. 2)", known as a **distributed ledger**.

In order to decide which transaction was first and arrange them in a sequential ledger, the peer-to-peer network must be able to produce *distributed, verifiable timestamps*. To do so, Satoshi Nakamoto introduces the **Proof-of-Work** algorithm, which implements a distributed system for timestamping transactions in the distributed ledger.

1. *"New transactions are broadcast to all nodes."* (Nakamoto 2008, p. 3)
    - Wallet owners broadcast their transactions to any nodes they are connected to in the peer-to-peer network, who then transmit the transaction to other nodes. Nodes are incentivized to include these transactions based on the amount of the **transaction fee**, which is defined as the difference between the value to send to the payee and the value sent in the transaction, and is kept by the nodes.
2. *"Each node collects new transactions into a* ***block****."* (Nakamoto 2008, p. 3)
    - All nodes on the peer-to-peer network pack together their chosen set of broadcasted transactions into a set of data known as a **block**, which contains a list of transactions the node included, and a reference to a previous block's hash value.
3. *"Each node works on finding a difficult proof-of-work for its block."* (Nakamoto 2008, p. 3)
    - A node must determine a value (defined as a **nonce**) that, when added to the block and hashed, begins with a certain number of zero bits. It

    does this by incrementing this nonce until a hash is found that has a certain number of zero bits (Nakamoto 2008, p. 3).
  - This brute-force process is defined in the paper as *work*.
  - The number of zero bits is defined as the **difficulty**, since “the average *work* required is exponential in the number of zero bits required” (Nakamoto 2008, p. 3).
  - In Bitcoin, the hashing algorithm is SHA-256.

4. *“When a node finds a proof-of-work, it broadcasts the block to all nodes.”* (Nakamoto 2008, p. 3)

   - In a peer-to-peer network, the node broadcasts the block to all nodes that it is connected to, who then transmit the block to all others they are connected to.

5. *“Nodes accept the block only if all transactions in it are valid and not already spent.”* (Nakamoto 2008, p. 3)

   - Once the nonce is determined, anyone can verify that the hash value has the required difficulty by just executing a single hash. (Nakamoto 2008, p. 3) The block is then considered to be validated. This proves that a certain amount of brute-force computational effort had to be expended to generate this hash: a Proof-of-Work.
   - Once this computational effort is expended, the block cannot be changed without redoing the work (Nakamoto 2008, p. 3).

6. *“Nodes express their acceptance of the block by working on creating the next block in the chain, using the hash of the accepted block as the previous hash.”* (Nakamoto 2008, p. 3)

   - Since each following block contains the previous block’s hash, changing any preceding block requires not only recomputing a new hash for that block, but also every block after it (Nakamoto 2008, p. 3).

Each node includes a special transaction (the **Coinbase**) to its block, which deposits the network accepted reward value in their account (as of this writing, 12.5BTC) without deducting from any other owner. As a result, nodes are economically incentivized to generate the proof-of-work hashes for their blocks first and with the most legitimate blockchain, so that their ownership of the reward value is secured. Due to the unpredictability of the hash function, there is a random probability of gaining this reward, but the probability grows substantially higher as the node computes at a greater hashrate (Nakamoto 2008, p. 4). As a result, the system is secured by greed: if a powerful attacker procures a majority amount of hashrate at tremendous expense (e.g. if a single nation-state nationalizes all Bitcoin miners), they must choose between either destroying the trust and value of the cryptocurrency, which wastes their entire investment, or using this hashrate to preserve the trust and value which produces lots of cryptocurrency capital to recoup their investment.

Nodes vote on which blockchain they prefer by mining on it. When a node recognizes the legitimacy of a new valid block, generally by time of receipt, they work to extend it with another block. If the node does not recognize the legitimacy of a block, it does not work on it and if the other nodes use the same code, they are unlikely to do so either. (Nakamoto 2008, p. 8)

As a result, the *longest chain* is also considered by nodes to be the legitimate one. "If two nodes broadcast different versions of the next block simultaneously, (the other nodes) will work on the first one they received, but save the other branch in case it becomes longer (Nakamoto 2008, p. 3)". However, the moment the deadlock breaks and a node receives a longer chain, the nodes will switch to the longer one (Nakamoto 2008, p. 3).

This also mitigates the issue of Sybil attacks that have vanquished other public permissionless distributed ledger systems. In a Sybil attack on the blockchain, an attacker attempts to overpower the network by cheaply bringing in nodes with IP addresses in an attempt to push the network to broadcast only their own blocks. However, valid blocks in the blockchain must not only have the proof-of-work generated per block, but make their blockchain the longest. As a result, the attacker still has to legitimately bring in a majority of expensive hashpower to fool the honest nodes. However, this does have implications for the common light wallets using Simplified Payment Verification (SPV) as described in later sections.

This process of collaboratively generating the longest chain of proof-of-work hashes is known as **mining** by analogy, since monetary distribution of cryptocurrency is decentralized and requires investment of equipment akin to mining gold to bring it into circulation. Nodes that mine are known as *miners*.

The result is that both the payer and payee can watch their transaction on the blockchain network to see whether the block containing their transaction is mined, defined as the **first confirmation**, and to see how many blocks have been mined ahead of their transaction's block, defined as the **# of confirmations** (Nakamoto 2008, p. 7).

Through calculations in the whitepaper, it was determined that when an attacker had a 10% *probability of mining the next block*, 5 confirmations was sufficient to reduce the *probability of an attacker changing a transaction on the blockchain* to 0.1%, exponentially dropping with more confirmations (Nakamoto 2008, p. 8).

3.3.1.4 Unspent Transaction Outputs

A Bitcoin **account balance** is defined by the amount of UTXOs it has on the blockchain (Nakamoto 2008).

The spending of a single UTXO creates a new "change" UTXO. The multiple amounts of change UTXOs form the account's current balance. If a transaction is made to send nearly the whole balance, the transaction will use nearly all the UTXO, and generate one additional "change" UTXO with the remainder.

This is also why paper wallets (containing a printed public and private key) must have the entire amount sent out of it, or the new UTXOs will be lost since paper wallets have no way to be aware of them. As such, in current paper backups, seed words are used instead so they can be imported into live accounts that can handle UTXOs.

3.3.1.5 Merkle Trees and Simplified Payment Verification

One major practical issue with the blockchain is that over time it would reach a size that may be manageable by stationary servers, but not by portable mobile devices, let alone IoT devices on mobile data.

Merkle trees can be used to arrange transactions, such that a "light" node would only have to hold some root hashes of the blocks on the blockchain, so that a certain transaction could be verified to be on the chain. (Nakamoto 2008, p. 4). This hash table can be downloaded to a light node from any or multiple peers on the blockchain's peer-to-peer network, or casual viewers could check online services such as Blockchain.info or Etherscan.io.

The concept of Simplified Payment Verification (SPV) utilizing such Merkle Trees is introduced in the Bitcoin Whitepaper (Nakamoto 2008, p. 4), to allow **light nodes** that keep only a copy of the above root hashes as a hash table, rather than the entire blockchain which can be in the range of hundreds of gigabytes or terabytes. If an SPV node is unsure of the provenance of the latest block it recieved from a peer, it can simply generate a hash of both the previous root hash and the block.

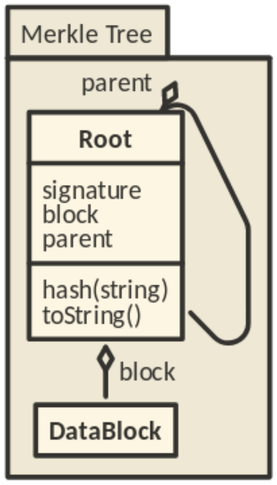
However, such an SPV node would have to ensure the majority of peer-to-peer nodes that they contact are trustworthy by querying multiple nodes, making SPV light wallets vulnerable to Sybil attacks that flood the network with attacker nodes that broadcast their own blocks only (as described prior) where a full node would not be affected. It is best to run a full node rather than use SPV if the service relying on the node deal with lots of transactions often, since the risks of a Sybil attack would be higher than if SPV was used sometimes.

3.3.1.6 Conclusions

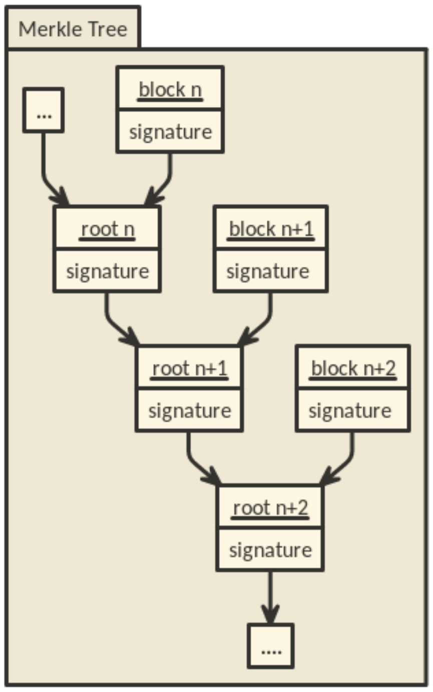
Nakamoto's whitepaper (Nakamoto 2008) describes what would be known as *the Blockchain*, a decentralized peer-to-peer network that maintains and shares a timestamped, immutable, decentralized ledger among them without trusting a centralized third party. Nodes vote on the validity of new blocks with their computing power through the Proof-of-Work algorithm, whereby the longest blockchain is recognized as the legitimate chain, converging to a consensus. *The more blocks ahead of a certain block (confirmations), the more blocks that need to be mined for an attacker to make a malicious change.*

An important note is that the value of a Bitcoin is based primarily upon the trustworthiness and security of this blockchain ledger. The very fact that only the holder of the private key can debit/credit an amount from one Bitcoin account to another on the Bitcoin blockchain is what makes it usable as a store of value and medium of exchange.

The decentralized ledger secured by the Proof-of-Work algorithm described by Satoshi Nakamoto's whitepaper became the blueprints of the first cryptocurrency, Bitcoin, which was proven to be a functional example over the past decade as both a fungible financial instrument and platform for blockchain applications that would inspire many others.



(a) A single element of a Merkle Tree.



(b) Roots of the Merkle Tree in the Blockchain

Figure 3.5: The Merkle Tree as implemented in the blockchain. Where root[i+1].signature = hash(concatenate(root[i].toString(),block[i].toString())) (Limited 2018).

3.3.2 Smart Contracts

Nick Szabo in his 1996 foundational article "Smart Contracts: Building Blocks for Digital Free Markets", describes the smart contract as "a set of promises, specified in digital form, including protocols within which the parties perform on these promises (Szabo 1996)". "New institutions, and new ways to formalize the relationships that make up these institutions, are now made possible by the digital revolution. No use of artificial intelligence is implied (Szabo 1996)".

Szabo had the expectation that *"specification through clear logic, and verification or enforcement through cryptographic protocols and other digital security mechanisms, might constitute a sharp improvement over traditional contract law (Szabo 1997)"*.

This idea proved to be far ahead of its time, and when the blockchain was proven to be practical with the Bitcoin blockchain, efforts were made to combine the smart contract with the blockchain, which led to the development of Ethereum by Vitalik Buterin, Gavin Wood, and many other contributors (Buterin, Wood, et al. 2018).

3.3.3 The Bitcoin Blockchain as a State Transition System

Bitcoin can be thought of as a **state transition system**. **States** on Bitcoin consist of a *ledger*, “the ownership status of all existing Bitcoins (Buterin, Wood, et al. 2018)”. **State Transition Functions** take a state and a transaction, and output a new state.

The Ethereum whitepaper describes Bitcoin in this manner, as developers realized that Szabo’s concept of Smart Contracts could be implemented using the state transition functions of the blockchain.

(Buterin, Wood, et al. 2018):

```
APPLY(S,TX) -> S' or ERROR`
```

Figure 3.6: The state transition function of Bitcoin in pseudocode (Buterin, Wood, et al. 2018).

A “state” in Bitcoin `S` can be defined more precisely as *all Unspent Transaction Outputs (UTXO) (a value on the blockchain with an owner) that haven’t yet been spent.* Each UTXO has an amount and the account owner’s public key (Buterin, Wood, et al. 2018).

A transaction `TX` is made up of a reference to one or more existing UTXO, as well as a signature by the account owner’s private key authorizing the transaction (Buterin, Wood, et al. 2018).

1. Given the State `S` and Transaction `TX` to the function APPLY():
    - If the UTXO in `TX` is not in `S`, return an error.
        - Nonexistent UTXOs cannot be spent.
    - If the signature of the transaction sender was not actually made by the account owner of the UTXO, return an error.
        - Account owners cannot spend a UTXO they do not own.

2. Get the sum of values from all UTXOs in the `TX`. If this sum is less than the sum of values of all the output UTXOs, return an error.
    - Account owners cannot send more than what is contained in their account balance, which is defined as the sum of values from all UTXOs with the account owner's public key.
3. Return `S'` with all input UTXO removed and all output UTXO added.
    - If all the prior checks pass, the transaction sends the amount in the UTXOs and returns any "change" to the sender (output UTXO owned by the transaction sender).

The Ethereum whitepaper also describes the state transition function as a ledger with transactions (Buterin, Wood, et al. 2018):

1. A transaction is "a request to move $X from A to B" (Buterin, Wood, et al. 2018).
2. The state transition function "reduces the value in A's account by $X and increases the value in B's account by $X" (Buterin, Wood, et al. 2018).
3. "If A's account has less than $X in the first place, the state transition function returns an error" (Buterin, Wood, et al. 2018).

3.3.4 Applications of Alternative Blockchain State Transition Functions

Bitcoin's built-in state transition functions are limited to single use case of building a cryptocurrency ledger. Although some custom scripting can be done, it has limited features for noncryptocurrency use cases (Buterin, Wood, et al. 2018).

Users had few methods to define their own state transition functions since Bitcoin's main use case is as a cryptocurrency. Arbitrary code can be dangerous when unrestricted, and cryptocurrencies have little need to do much other than send transactions around on a ledger.

Bitcoin's state transition functions are written in a scripting language that is not Turing-complete. The primary disadvantage is that it lacks the ability to run loops (among other things), resulting in repeated if statements, which are very space-inefficient (Buterin, Wood, et al. 2018).

Next, Bitcoin's state transition functions are embedded in UTXOs, which are consumed in whole to produce new UTXOs. Unfortunately, this means that the function is blind to value, since it has no control over how much will be withdrawn. One inefficient approach is to make certain colored coins like paper currency denominations, which are not fully fungible ($1, $2, $5, $10), but this needlessly defeats an additional benefit of cryptocurrency: fungibility. This dependence of state transition functions on UTXOs also results in a lack of state, where the contract disappears when the UTXO is spent, multi-step contracts cannot exist, and variables cannot be maintained. Therefore, withdrawal limits cannot be implemented (Buterin, Wood, et al. 2018).

Finally, Bitcoin's state transition functions are unable to reference blockchain data such as the timestamp, nonce, and previous block hash, which are valuable sources of random data in game, lottery, or gambling applications (Buterin, Wood, et al. 2018).

As a result, examples of three approaches were provided in the Ethereum Whitepaper that attempt to facilitate alternative state transition functions: single-application blockchains, colored coins on the Bitcoin blockchain, and metacoins on the Bitcoin blockchain (Buterin, Wood, et al. 2018).

3.3.4.1 Independent Blockchain

Independent blockchains either take the Bitcoin source code or something made from scratch, and replace Bitcoin's ledger state transition function and miner network with their own.

Namecoin is one example, an independent fork of Bitcoin that provides a decentralized blockchain-based alternative to the Domain Name System (DNS). (Buterin, Wood, et al. 2018).

DNS provides computers a method to *look up the IP address of a domain name* in a networked, hierarchical database made up of many servers around the world (Verisign 2018). Domain names can be considered to be digital assets, each with one owner that has the right to set the IP address and sell the domain name. Domain names are usually sold in a first-come, first-serve process.

However, domain name registration and sale is a centralized but delegated process, subject to security breaches or simple bureaucratic oversights that have dire consequences for all users and applications relying on them. For example, a cryptocurrency exchange could have its DNS servers hacked to redirect users to a malicious server instead. Another example is that local DNS servers can be under the influence of governments or other central organizations to either censor or manipulate the DNS requests returned. Users would be totally unaware, since there is no ownership provenance or any method to prevent the wrong IP address from being reported (other than SSL which is still not used by all servers, and is also a centralized hierarchical web of trust).

The blockchain fits this application as a domain name is a digital asset with ownership facilitated through central databases. The digital asset is composed of nothing more than a mapping between a textual domain name and owner-specified data, which could be an IP address or other text value. Domain names would be significantly more secure as a blockchain digital asset that can be owned, traded, sold, but not changeable or transferrable without owner consent. Provenance, transaction immutability, the elimination of bureaucracy and centralized actors, and significantly more secure public/private keys for control of the domain name make the blockchain a perfect fit. In fact, the same idea can be used for mapping domain names to public keys for cryptocurrencies or blockchains, which are even more difficult to remember than an IP address, and there are even more dire financial consequences for sending transactions to the wrong address.

As for Namecoin and other independent blockchains in themselves, success has not been as assured. Building an independent blockchain for a single decentralized application is like building a whole network of colocated servers and all its operating systems and software from scratch, simply to run a single website. In addition, these blockchains have to incentivize and bring together a network of their own miners, which is sure to be weaker and vulnerable to attack or volatility from Bitcoin miners or other groups, if they even manage to be organized at all. Most of all, the entire code for the blockchain has to be maintained by the application developers in addition to the application's own state transition functions, often reinventing the wheel or becoming vulnerable to exploits that were already fixed upstream in more developed blockchains.

As a comparison, Ethereum allows a domain name registration system to be implemented easily in a state transition function with just three lines, since it simply sets a key with a value. A more complete version is already in use for Ethereum account addresses, the Ethereum Name System.

```
def register(name, value):
    if !self.storage[name]:
        self.storage[name] = value
```

Figure 3.7: An example of a Domain Name digital asset in the Ethereum Serpent language (Buterin, Wood, et al. 2018):

3.3.4.2 Colored Coins and Metacoins

Colored coins are a basic concept of fungible cryptotokens piggybacking on the security of another blockchain: in this case Bitcoin.

In colored coins, “color” is assigned as attached metadata in the notes section of a Bitcoin transaction, and all following transactions can query its parent to inherit this color. Since typically only the colorer was able to issue colored coins from their account, other Bitcoins without this provenance aren’t accepted (Buterin, Wood, et al. 2018).

The “color” metadata could not make use of Simplified Payment Verification and Merkle Trees (as described in previous sections), forcing users to trace a coin’s provenance using the entire Bitcoin blockchain, instead of using a light client with significantly fewer space requirements. Since most colored coin users could not hold a copy of the whole blockchain, centralized services came about to provide this service, which eventually caused colored coins to devolve into a centralized database with trusted third parties.

By 2016, Bitcoin transactions became too expensive and slow for colored coins to be transferred easily, with USDT Tether being the most potent example. *Metacoins*, offchain solutions that put a protocol above Bitcoin but still checks the Bitcoin in case of conflict, also suffered due to this dependency, and also their inability to use SPV or Merkle Trees. However, Ethereum would later adopt this concept of fungible tokens on its blockchain to full fruition in the ERC20/223 fungible cryptotoken standard (Vogelsteller, Buterin, et al. 2018). This likely sparked development and interest in Ethereum and its eventual Initial Coin Offering boom.

As officially integrated elements in the Ethereum blockchain, ERC20/223 tokens can make full use of Simplified Payment Verification and Merkle Trees instead of relying on centralized colored coin viewers. These tokens can also be offered for purchase with Ethereum cryptocurrency, an Initial Coin Offering. As a result, light clients such as MyEtherWallet and Metamask can easily view, receive, and send ERC20/223 fungible tokens.

The Ethereum whitepaper introduces these examples as a comparison to their innovative arbitrary state transition system, which is Turing-complete and resolves many of the problems of these alternative blockchain applications. (Buterin, Wood, et al. 2018).

3.3.5 The Ethereum Blockchain

3.3.5.1 Smart Contracts: Arbitrary State Transition Functions

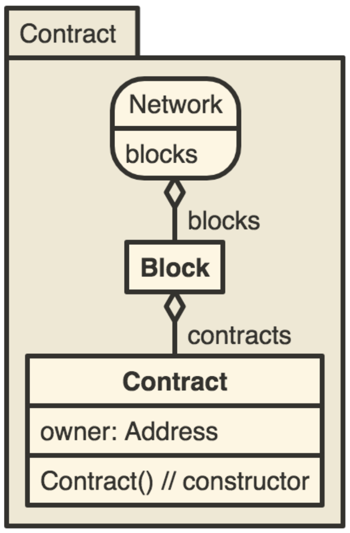


Figure 3.8: A smart contract as arbitrary state transition code embedded in a block on the blockchain (Limited 2018).

In the Ethereum Virtual Machine (EVM), **smart contracts** are defined as Turing-complete code, allowing users to encode *arbitrary state transition functions* that change the state of transactions, variables, and even other state transition functions on the Ethereum blockchain (Buterin, Wood, et al. 2018). (Smart Contracts will be referred to in this thesis as “state transition functions” in technical descriptions below, but as Smart Contracts when describing their use cases)

Turing-complete languages with loops or recursion are vulnerable to the **halting problem**, where there is no way to know ahead of time when a computation will stop or if it will take a very long or even infinite amount of computation.

As a practical workaround, the concept of **Gas** is introduced, where one computation step is one gas. Users specify the maximum limit on computation steps (*Gas Limit*/STARTGAS) and set a price per step (GASPRICE), whereby the maximum **transaction fee** = STARTGAS $\times$ GASPRICE (Buterin, Wood, et al. 2018).

If the state transition function or any descendant functions uses less gas than the gas limit, the user only pays for the amount of computation steps consumed. If the state transition function reaches the gas limit, computation is halted, variables are reverted to their previous state, but the maximum transaction fee is paid, preventing the entire Ethereum network from stalling or resources being consumed without compensation (Buterin, Wood, et al. 2018).

This prevents the following security breaches in Ethereum.

1. Knowing that the language is Turing-complete, an attacker submits **a state transition function with an infinite loop** aiming to cause the miner, and maybe even the entire Ethereum network to *get stuck computing it* (Buterin, Wood, et al. 2018).
    - The gas limit/STARTGAS requires users to specify a limit on computation steps and specify the GASPRICE per computation step. The miner then knows ahead of time how many computations the smart contracts will take at maximum, and how much the user will pay per computation (Buterin, Wood, et al. 2018).
    - The miners pick and choose the state transition functions with the best market rate and amount of computations that they can handle. If an attacker tries to push a low gas price, functions with a gas price lower than market rate are unlikely to be picked up by miners.
2. An attacker submits **one or multiple state transition functions with an infinite loop**, aiming to cause miners to waste so much time computing that they would never finish the block, and thus cause a denial of service on the entire Ethereum network that *never charges the attacker the transaction fee* (Buterin, Wood, et al. 2018).
    - The gas limit/STARTGAS allows miner to know when to halt computation artificially. Miners would see excessive gas limit/STARTGAS and can easily judge whether they are capable of completing the computation in time to generate the new block.

    - This also incentivizes smart contract code to be better written to avoid infinite loops and economize computation steps, since excessive steps will make the code not only fail or be avoided by miners, but also be expensive in gas fees (Buterin, Wood, et al. 2018).

3. The attacker attempts a **incomplete computation attack**, where they set a gas limit/STARTGAS *only sufficient to process the first few steps, but not the rest* (Buterin, Wood, et al. 2018).
    - This does not work, since changes are reverted upon reaching the gas limit. Regardless of success or failure of the function, the attacker has to compensate the miner for resources used, with gas as fees.

Turing incompleteness like in Bitcoin's State Transition Function would not need a gas limit, but the result is that loops would not be possible and contracts could not be launched by other contracts (Buterin, Wood, et al. 2018).

As a result, there is limited benefit for these drawbacks of Turing incompleteness, and ironically the halting problem in Turing complete languages is easily worked around in Ethereum by adding a gas limit with gas price to the transaction (Buterin, Wood, et al. 2018).

3.3.5.2 Ethereum Mining Algorithm

In addition, Ethereum uses a hashing algorithm, Ethhash, designed in response to systemic problems in Bitcoin that were unforeseen at the time of its creation.

Bitcoin uses the SHA-256 hashing algorithm (National Institute of Standards and Technology, 2015), which was originally calculated by commonly available CPUs and GPUs, but eventually Application-Specific Integrated Circuits (ASICs) were developed to compute the SHA-256 hash at ever higher hashrates (Buterin, Wood, et al. 2018). Unfortunately, such equipment can only maximize hashrate at costs of millions of dollars by specialized manufacturers (Buterin, Wood, et al. 2018). Finally, few miners have any incentive to hold and seed the blockchain, since mining does not rely on holding a recenty copy and seeding the blockchain provides no compensation. Worse yet, miners now join mining pools that provide block headers from a single centralized node. As a result, mining pools control 50% of the mining power (Buterin, Wood, et al. 2018).

Ethereum's Ethhash is instead designed to require miners to fetch random data from the blockchain state, compute random transactions and their associated state transition functions from recent blocks, and return a hash of the result. As a result, an Ethereum ASIC would have to be a better CPU (along with associated cache and RAM), as its Turing-complete state transition functions can take forms that a inflexible electrically burned in ASIC cannot provide. In addition, Ethereum miners must have access to the entire blockchain, so every miner must be a node seeding the blockchain, increasing accessibility and decreasing reliance on a single mining pool's node for the next block (Buterin, Wood, et al. 2018).

3.3.5.3 Ethereum in Practice

The Solidity language is the main high level language used to compile to Ethereum Virtual Machine (EVM) low level code (Ethereum Foundation 2017[b]).

With Ethereum Smart Contracts, a buyer can hold money in escrow with the contract, which sends the cryptocurrency to the seller if a tracking number returns a successful delivery, or refunds the cryptocurrency upon failure (Szabo 1997). Smart contracts can also easily create **cryptotokens** as financial instruments that function much like cryptocurrencies, but can be verified using the computational strength of Ethereum's miners rather than having to recruit their own. These cryptotokens can be used as **coupons or tickets** through the issuer's guarantee of redemption for goods or services (Buterin, Wood, et al. 2018).

Cryptotokens can even be programmed to function as **voting shares** where shareholders democratically determine how money put in escrow in the contract are used, as bond instruments where creditors lend funds in exchange for interest (and likely a voting stake in how money gets spent), licenses that users must purchase for software to function or media to be decrypted, really any kind of complex financial instruments that hinge on conditions and actions (Buterin, Wood, et al. 2018).

The organizations produced by such contracts, such as the Decentralized Anonymous Organization, are given power by their ability to hold, collect, and use funds (Foundation 2017). They also exhibit oligarchic or even democratic voting systems for tokenholders, and far in the future could even facilitate the taxation and decision mechanisms of a parliamentary system of government, where tokenholders regulate the actions of an executive by voting upon the transfer of funds.

3.3.6 Permissioned/Consortium Blockchain Methodology

The blockchain works best for the use case it was developed for, open financial markets where all users have a shared interest in knowing the sender, recipient, and value of every single transaction that goes through, as this has direct relevance to each user's coin value. Decentralization can manifest as mining forms a financial motivation to verify transactions on the network, whereas interest in information on the blockchain leads users to operate nodes to "seed" the blockchain that other users can download peer-to-peer akin to Bittorrent.

As a result of the hype in cryptocurrencies and the incredible features of the blockchain's immutable ledger, businesses and consulting firms such as KPMG have identified key benefits of applying the blockchain to the enterprise (KPMG 2018, p. 7).

- "Gain increased efficiencies by automating and streamlining back-office operations, enabling peer-to-peer solutions, removing intermediary services, and reducing post transaction latency and counterparty exposures" (KPMG 2018, p. 7).
  - The key benefit of the blockchain in any enterprise solution is a significant reduction in paperwork and bureaucracy, as well as cutting out the middlemen by reducing information asymmetry and connecting users directly.
- "Increased efficiency in real-time financial settlements worldwide, with greater transparency and increased liquidity" (KPMG 2018, p. 7).
  - Blockchain tokens could be used as financial mediums of exchange within a company or with partners, especially across countries, reducing the amount of transfers and exchange fees.
- "Reduce costs through automation and by leveraging the cloud and eliminating the need for additional servers, simplifying infrastructure and solving significant system integration issues" (KPMG 2018, p. 7).
  - Smart contracts on the blockchain can watch for certain triggers in the ledger or the environment, then execute a programmatic action. This can be extended to the real world or supply chain with signals to workers

    to immediately produce a product upon verified customer purchase, or IoT devices perhaps like a lock on a shared bike that unlocks upon receipt of payment.

- “Increase security with cryptography” (KPMG 2018, p. 7).
    - Blockchain accounts can not only be used for sending transactions, they can also be integrated with a full cryptographic identity solution that can be used to securely sign transactions and send encrypted messages. Although anonymous identity is one possibility with blockchain accounts, a web of trust and hierarchical
- “Enhance Regulatory Compliance by offering detailed, factual evidence and reliable documentation at a lower cost, and creating a concrete audit trail” (KPMG 2018, p. 7).
    - The Blockchain’s consensus algorithm ensures that every transaction on the ledger can be tracked and actors within the ledger can be accountable to their actions without repudiation. Auditors could also operate a blockchain node to obtain a copy of this ledger.

However, not all use cases benefit from public, permissionless knowledge of all transactions, mandated by Bitcoin’s open financial roots. In particular, supply chain contracts are produced by a process of bargaining which takes into account many factors outside of simply price and quality to produce a customized price for each customer. Most of all, transporters and end customers do not even necessarily need detailed granular information to do their jobs. Though there is still a need for some transparency between entities and regulators if not the entire world, there must be segregation yet strong linkage/verifiability between confidential and publicly reported data.

To mitigate the concerns inherent in using a permissionless blockchain that is subject to free-wheeling miners or the developer’s visions, and to keep confidential accounting information within the organization, the concept of permissioned blockchains have been introduced as seen in JP Morgan’s Quorum, Hyperledger, and the World Food Program’s own Building Blocks system. These permissioned blockchains differ from forked blockchains (such as Bitcoin Cash, Namecoin, which are still permissionless blockchains) by only having participants run nodes allowed to connect to the peer-to-peer network. This permissioned blockchain would not be subject to external conditions, and allows accounting transactions to be kept private within the organization. Most of all, it also provides the same smart contract capabilities of Ethereum as mentioned above.

Permissioned blockchains mitigate the effect of having malicious miners enter a blockchain by enforcing strict controls over which miners and nodes are allowed to work on it (ideally, through a private intranet) and who is able to connect to it, perhaps with signed digital certificates. This way, a compromise would have to occur at the highest levels to allow malicious miners even just to get in, let alone establish more than 51% of network hashrate.

The possibility is that much like most organizations own and operate their own internet servers so as not to be subject to the limitations, traffic, data leaks, and costs of cloud services, they may consider running their own blockchains to avoid the market and traffic conditions on public blockchains.

On the other hand, it is apparent that with all the overhead of maintaining a permissioned blockchain, for some organizations and some use cases where trust is not a significant issue or the amount of responsible agents is small, a centralized database might end up being faster and cheaper.

However, many large corporations or organizations are structured human societies unto themselves, with a diffusion of responsibility and anonymity that allows insiders to embezzle funds, management pressures that incentivize them to cook the books, and bureaucratic slowness that chokes the efficiency of the organization with paper pushing. Despite all the advancements of IT in the business world, voluminous paper records that computers print out for offsite storage has been the only thing auditors can trust, despite the tremendous labor required to process and verify them.

Thus, in applications where trust and immutability is a huge concern for large communities or markets, the blockchain allows users to finally apply IT solutions to accounting, where nothing before was trustworthy enough for the purpose.

3.3.6.1 Hyperledger Sawtooth

Hyperledger's **Sawtooth** is an "enterprise" blockchain platform for decentralized ledger applications (DApps) and their associated smart contracts, with a focus on modularity, extensibility, privacy, and security to be flexible enough for all sorts of business needs. (Olson et al. 2018, p. 1)

Like Ethereum, Sawtooth maintains the core blockchain system so that applications built on top can simply specify the relevant business rules in the smart contracts, without needing to fully grasp or maintain the underlying blockchain code.

Sawtooth's motivations for adapting the Blockchain for enterprise needs are to streamline business processes between companies, allow records to be kept in sync without trusted centralized third parties/manual reconciliation processes, and facilitate entirely new ways of doing business. (Olson et al. 2018, p. 1)

Sawtooth smart contracts (or state transition functions) can be programmed in familiar and mature programming languages, from the interpreted languages (Python, Javascript), compiled languages (Rust, C++, Go), and virtual machines (Java, C#). A future Seth compatibility layer for the Ethereum Virtual Machine would add support for existing Ethereum smart contracts in Solidity and Serpent. Future work on Sawtooth would increase throughput and transaction privacy.

Sawtooth aims to adapt the blockchain to the varying needs and changing goals of permissioned blockchains in enterprises.

3.3.6.2 Sawtooth Consensus Algorithms

Various consensus algorithms have been pitched to apply to the Byzantine Generals Problem, each with their pros and cons. Instead of choosing a single consensus to fit all problems, Sawtooth provides applications with the ability to choose the type that fits their use cases, and allows nodes to vote on changing it in the future as necessary. (Olson et al. 2018, p. 6)

Two main consensus algorithms are pitched from two categories: "lottery" consensus, and "voting" consensus. (Olson et al. 2018, p. 6)

"Lottery" consensus is also known as "Nakamoto" consensus, because Bitcoin's Proof-of-Work consensus was the first example of it. Through some process, such as calculating a nonce by brute-forcing a hash algorithm, a node is randomly chosen to lead the consensus by proposing the next block to extend the chain of blocks previously agreed upon (Olson et al. 2018, p. 6).

Sawtooth provides the **Proof-of-Elapsed-Time** algorithm as their "lottery" consensus algorithm to "support large network populations that include byzantine actors (Olson et al. 2018, p. 6)". Much like Hal Finney's original proposal for a cryptocurrency using Hashcash and b-money concepts in 2005 (Buterin, Wood, et al. 2018), Proof-of-Elapsed-Time relies on *Intel SGX Trusted Computing chips* which have become significantly more prevalent since their debut with the Intel Skylake platform in 2015.

"Voting" consensus is influenced by classic "Byzantine fault tolerance" algorithms descended from those proposed by Lamport, Shostak, and Pease (Lamport, Shostak, and Pease 1982), where the nodes explicitly vote according to some process to build consensus. These consensus algorithms tend to consume significantly less electricity for higher throughput than "lottery" consensus (Olson et al. 2018, p. 6).

Sawtooth provides the **RAFT** consensus algorithm (which is still under development) for "high-throughput, low-latency transactions (Olson et al. 2018, p. 6)".

Sawtooth also provides methods for authorized nodes (perhaps managers of each corporation in the consortium) to vote to modify network settings even after the blockchain is deployed, such as access permissions, transaction rules or types (UTXO like Bitcoin, or Ethereum-style states), consensus algorithm modifications or changes, or token inflation rate adjustments. Such changes are common practicalities in enterprises that are tremendously difficult on permissionless blockchains (Olson et al. 2018, p. 7).

3.3.7 Application Paradigms in the Blockchain

To cut through the hype regarding the Blockchain (much like the hype regarding the Internet during the Dot Com Boom), this section will explore what the Blockchain is truly capable of, and what application paradigms are most likely to be used to build systems that utilize its abilities.

3.3.7.1 Characteristics of the Blockchain

Daniel Drescher in his book "Blockchain Basics" defines the basic properties of the blockchain: (Drescher and service) 2017, p. 224)

- Tamper-resistant (Practically Immutable)
- Append-only
- Ordered

- Timestamped
- Open and Transparent
- Secure (identification, authentication, authorization
- Eventually Consistent

Drescher explains that these properties of the blockchain are "independent of the specific data stored in it (Drescher and service) 2017, p. 224)". The blockchain's value is not in storing the data per se, but providing metadata attesting to these properties which legitimatize that data.

As such, it is not strictly necessary for the blockchain to carry the data, as it can tabulate hash fingerprints of the data instead. This is valuable as it can be counterproductive to store large data on the blockchain: every single node on the network would have to hold all data. The large data could be transmitted through other conventional means.

3.3.7.2 Generic Application Patterns

These generic application patterns are basic, proven use cases for the blockchain. Some are dependent on one another, others complement each other. Almost all blockchain applications should make use of at least one of these patterns, and this thesis will refer to these in its application implementation.

- **Proof of Existence** - Store an entry just so it is queriable on the blockchain. The blockchain is able to store data given a transaction fee on public blockchains, or mutual consensus in permissioned blockchains. The ordering and timestamping is not necessarily used, but could be useful metadata.
  - Domain Names
  - patents, registrations, whitepapers
- **Proof of Nonexistence** - "The opposite of Proof-of-Existence" (Drescher and service) 2017). The nonexistence of an entry has meaning as a default case.
  - Complaints, fines, convictions
  - Reputation
- **Proof of Time** - A Proof of Existence taking the timestamp of the entry into account (Drescher and service) 2017). The timestamps in every block on the blockchain record when the block was mined.

  - Delivery or notification tracking
  - tracking of payments
  - tracking of orderly opening and closing of bids
  - management of predictions

- **Proof of Order** - A Proof of Time that considers the relative order of entries (Drescher and service) 2017). The consensus algorithm of the blockchain enforces an order of the blocks, independent of just ordering the timestamp.
  - First come, first serve Applications/tickets - Order is critical for systems that only accept a certain amount of submissions, or rank them by when they were submitted.
  - Auction bids - The person who made the highest bid closest to closing time wins.

- **Proof of Identity** - A Proof of Existence that records the existence of a identity (Drescher and service) 2017). The blockchain can both attest to the existence of identity data or a hash of it on the blockchain, and link the identity to an account on the blockchain.
  - Identity for people or organizations
  - Identity for inventory, live animals, other goods
  - Heirarchical Public Key Infrastructure within an organization
  - Licenses certified by organizations or governments

- **Proof of Authorship** - "A proof that a certain person or institution added certain data to the blockchain" (Drescher and service) 2017, p. 224). If an account has a verifiable Proof of Identity linked to it, each authorized entry on the blockchain the account creates is considered "authored" by that identity.
  - Electronic Publishing
  - Version Control in code or documents
  - Copyright licenses

- **Proof of Ownership** - Brings together Proof of Existence, Proof of Identity, and proof of authorship, the security of which rests upon Identification, Authentication, and Authorization (Drescher and service) 2017).
  - Ownership of digital assets
  - Title deeds to real world assets
  - Cryptocurrencies, or Cryptotokens

3.3.8 Security and Practicality Concerns

Now that the basics of the blockchain system are established, there remain the myriad security and environmental concerns inherent in blockchain decentralized applications, with good footing in recent historical examples. These are issues that affect all smart contract systems today or in the near future, but some of these issues might be overcome with advancements in blockchain technology, such as the development of new consensus algorithms or offchain frameworks. Technology may overcome barriers and increase efficiency, but people and economies will remain subject to patterns from time immemorial.

3.3.8.1 The Blockchain Approach to Ledger Security

There are basic principles that define security in blockchain systems.

- **Integrity** - The ledger of the blockchain cannot be overturned without the consent of more than 51% of all nodes/miners. In a public blockchain, the strength of the Ethereum blockchain is what the app's trustworthiness hinges on, but new miners do not need to be raised. In a private blockchain, all interested entities (such as customers, suppliers, transporters) must run a node and hold a balance of power against each other to prevent each other from colluding against one party. The more entities participating, the lower likelihood of a 51% attack.
- **Availability** - The blockchain ledger is immediately available to all p2p users and can be participated from anywhere there is an uncensored internet connection. Transactions move at 40-50 transactions per second on the public ethereum blockchain (but transactions can move faster given higher fees). Transactions can also move faster on a private ethereum blockchain with all the traffic for the application needs only, as done by the World Food Program in their aforementioned private Ethereum blockchain for refugee food distribution.
- **Authentication** - Identity or a hash fingerprint of identity data can be recorded on the blockchain. For example, transactions reporting identity information signed by authorities and with the consent of the actual owner be transmitted between organizations. Existing Public Key Infrastructures within organizations can be associated with accounts to provide signed identities of each entity. Alternatively, decentralized attestation to an identity can emerge from the transactions the account made on the blockchain (a reputation), or a web-of-trust of signatures by other identified accounts on

the blockchain. This way, this sort of confidential identity information only needs to be shared between organization nodes upon consent to a contract by interested parties. Then, each entity can sign for each other to build a web of trust, a network of asymmetric encryption signatures that blockchain private keys can make. In a web of trust, a supplier is trusted by many customers, and that supplier trusts certain transporters, thus providing each entity an attestation to their reliability and reputation.

  – **Off-chain Authentication** - Although identities are signed and verified using the blockchain, if the identity information only needs to be known by entities party to a transaction, a blockchain identity record could contain only hash fingerprints or truncated versions of the information needed (city, zip code), signed by an authority or a web-of-trust. For example, a supplier stores a truncated version of its general location info on the blockchain, but only when an order is agreed by the customer and the transporter agrees to move the product, does the supplier send a double signed encrypted message to both of their actual location and relevant information about the items.

3.3.8.2 Smart Contract Bugs and Oversights

Although Ethereum Smart Contract code is incredibly simple compared to many other revolutionary technologies such as AI or data mining, security as always is difficult to master.

For one, the entire agile development model of "build fast, fix fast" falls flat against the wholesale immutability of the blockchain. For once a smart contract is issued and people come to rely on it, it cannot easily be recalled or updated at the same address without agreement from the users. The best that can be done is to have expiration dates on contracts which allow a period of time that users can prepare and developers can issue an updated contract. However, this still is not conducive to immediate security patches, which simply cannot be implemented to industry standards in the blockchain.

An example is seen in the Parity multi-sig wallet contract cancellation. The Parity multi-sig wallet was an Ethereum smart contract whose code was imported by subinstances of it that users create, allowing multiple users to control the amount of one single wallet. This design was a godsend to startups that wanted their funds to not be controlled by just one person, but by multiple employees or even applications. Thus, millions of dollars of Ether were put into instances of these multi-sig wallets. Unfortunately, the oversight was that rather than calling the head contract as an immutable library of functions that would never disappear: the primary Parity wallet was a standard smart contract which the owner reserves the right to rescind with the "suicide" function. While it was common sense at the time to include this function to kill the contract when it was no longer necessary or was superseded (so that it could be pruned from the blockchain), the fact that other subinstances depended on importing functions from this contract made it a central point of failure for all of them.

Thus, when an unrelated exploit allowed a meager blockchain "pentester" to make themselves the owner of the Parity smart contract that all other instances depended on, this "pentester" now had the right to use the suicide function on that smart contract. The result was that, likely believing that all the money in every Parity wallet would be recalled to the "owner", the "pentester" used the suicide function to kill the main Parity contract. Unfortunately for them, the main contract did not have any rights to the Ether in the other Parity wallet subinstances, which was a proper security measure and the reason behind this unusual function call structure. However, this did make the hundreds of milions of dollars of Ether in every single Parity wallet permanently inaccessible to users, because when the users attempted to use the pay or recieve functions on their subinstances, their contract code queried for the head instance, which was now "dead", and the transaction crashed. Since the subinstances cannot update the dependency on the "dead" Parity main contract to a fixed one, the users could only beg the Ethereum developers for a bailout, with the moral quandrary of either damaging the immutability of the Ethereum blockchain or scaring smart contract applications away due to the fragility of the code against bugs. The situation remains unresolved as of this writing, though it may have temporarily increased the buying power of all the remaining ethereum in the market.

In addition, economics is as central to the eventual performance of a market built upon a smart contract as advanced mathematics is to AI or data mining. The general dilemma for a smart contract platform that holds an ICO is that the cryptotoken must provide value to both speculative investors and the users of that platform.

For example, Storj coin aims to provide a decentralized economy where individuals or companies with surplus storage on their servers can offer it up to users as a cloud service through their proof-of-storage blockchain system. Users pay providers the Storj coin in compensation: so the fees per GB cannot be too onerous and should at least be on par or cheaper than Amazon S3: not out of the question, since it is pretty expensive. However, the initial investors and speculators that come after want to see the price of the Storj coin skyrocket, and thus engage in pumping and marketing schemes that increase the value of the coin beyond its actual fair market price. Although this would normally incentivize more providers to jump in and thus provide more storage at a lower rate, the reality is that the demand was manufactured by holders rather than real users. Thus, the providers discover when "mining" that very few users actually take them up on the offer since the high cost of tokens has actually made it more expensive than Amazon S3, killing the value for them and reducing the profits made by storage miners. Thus, the speculative action of the investors has actually reduced the real market value of the Storj coin, despite pumping up the value. A compromise must be made between these two interests, or else an ICO may be doomed to be little more than a pump and dump scheme.

Thus, a different approach to building smart contracts with test driven development and proactive mitigation of damage from exploits or oversights is required, for the consequences of a minor bug could be economic loss in the millions of dollars, or in bad situations, wholesale economic meltdown. On the flip side, blockchain testnets that function very similarly to the real market can be made with simulated traffic and situations to test against a smart contract. This opens up powerful opportunities for developing experimental economic designs against significantly more accurate models, since this is a unique case where the model functions almost exactly like the production (cryptocurrency) market.

3.3.8.3 Reliance on Network Conditions and Health

As of this writing, just about half of all popular "cryptocurrencies" are actually Ethereum-based ERC20 tokens (as seen on Coinmarketcap), which run entirely on the public Ethereum Blockchain, with examples such as EOS, TenX, Status.im, ARK, and many others often raised and distributed through initial coin offerings

(ICOs).

This is due to the relative ease, functionality, and safety of creating an Ethereum cryptotoken compared to building an entire blockchain and mining network from scratch. For one, Ethereum provides a convenient token building platform that takes care of all the complexities of actually running the blockchain network, and leaves to the developer only the logic directly relevant to their application in the smart contract code. If a developer attempts to make their own blockchain, tremendous amounts of code would have to be maintained and a large fleet of miners, neither of which are directly applicable to their mission. Such a blockchain would also be vulnerable to hashpower attacks from large Bitcoin or Ethereum miners, who could easily overpower the meager amount of miners that the developer can muster on their own.

Unfortunately, all users of these large blockchains are faced with the negative externalities of these networks, much like how weather or market conditions affect supply chains. For one, the current most popular consensus algorithm, Proof-of-Work, has a limit of how many transactions it can verify per second depending on how many miners are active. The solution thus far has been to simply incentivize more miners to join with temporarily heightened transaction fees. Unfortunately, mining hashrate often cannot grow exponentially to match the rapid growth of these networks. For one, mining can sometimes be a losing proposition and a money sink. Another issue is that mining consumes a tremendous amount of resources in terms of hardware and electricity for the service provided, which is unsustainable in the long run. Often times if an investor just held the cryptocurrency instead of mining, they would be all the richer.

Ethereum's blockchain has hit this transaction processing ceiling many times: first during hyped up ICOs where users scramble to purchase cryptotokens at a pegged value, but today in part due to continuing traffic load caused by the blockchain speculation game Cryptokitties, where users raise, breed, and trade digital kittens. Just this simple game has produced traffic loads on the network equivalent to major ICOs, Cryptokitties shows little sign of ending its traffic backlog. The result is that transaction fees have jumped from just 2 cents to 50 cents, and transaction times have stretched to peaks of 40 minutes from just 5-10 minutes. If a ticketing system was built upon Ethereum, users might be paying unexpected fees and also may not be able to confirm their bids for a superior seat in time before another person takes it. If the World Food Program's Building Blocks blockchain used the public Ethereum blockchain, their supply chain could be negatively affected by transaction-induced delays caused by events outside of their control (which is why they use a permissioned blockchain).

Thus, much like the dial-up Internet and its paltry servers lacked the bandwidth to handle national, let alone global levels of traffic without huge technological advancements, the Blockchain is at an immature stage where instability and volatility is paramount. This caveat should be observed in any major deployment of a blockchain application.

Chapter 4 Related Work

The background literature below explores various areas of *the issue of blockchains in the supply chain*, as well as the applicability of *permissioned blockchain systems in the enterprise in general.* The aim is to pick out and synthesize many insights developed by these papers into this thesis, to push the field further.

Research pertaining to the Blockchain space is new, but many more quality papers have popped up in the past year as a result of the hype on Ethereum and ICOs. IBM Hyperledger has also found industry acceptance and some government support.

As a starting point for more formal research in the space of the Blockchain as applied to the supply chain, the Blockchain in Megacity Logistics paper provided a base outline for how some sort of private permissioned blockchain, using "voting" consensus, would work to eliminate some roles of the fourth party logistics provider (Polim, Hu, and Kumara 2017). This paper provided a very rough proof of concept put together in Google Forms, but the field has advanced significantly since 2017 when it was published, whereby public and private smart contract blockchains have started to show promise in early implementations that have matured.

The viability of public Ethereum smart contracts has been tested through practical user participation in top Initial Coin Offerings (ICOs) and adoption of Ethereum decentralized apps that use smart contracts, such as Cryptokitties, Slock.it, or Etherdelta. Many of these smart contracts have hit traffic, trust, and security problems along the way, which are described in several ICO analyses below.

Permissioned blockchains using smart contracts have also found viable implementations. In particular, the concept of a private permissioned election algorithm blockchain ended up being very prescient, since implementations now exist such as JP Morgan's Quorum or Hyperledger which make this sort of blockchain far more than just a proof of concept. The World Food Program's Syrian refugee food distribution system utilizes a private Ethereum blockchain and smart contracts to generate cryptotokens which are tied to biometrics, significantly reducing the risk in distributing cash or ration coupons. Walmart and IBM's Hyperledger has shown promise in increasing transparency through the supply chain through the blockchain's irreversible provenance.

4.1 Literature Discovery

The Penn State Library's Lionsearch system was used to find research documents and all relevant literature regarding the topics "Blockchain AND Supply Chain". For papers and theses, there were less than 500 results, and much fewer that were relevant to blockchains as applied to the supply chain.

Google Scholar was also utilized with the search terms "Blockchain AND Supply Chain" and the stipulation of publications after 2017, which discovered many of the same results but had a few different papers of interest. Unfortunately, the author's ability to investigate some papers was hampered by lack of university access to many of them.

Papers regarding the topic of the alternative applications of the Blockchain in fields such as the supply chain appear to have emerged quite recently after 2017. Research in Blockchain applications remains an immature field of study, and many technologies such as Ethereum smart contracts and Hyperledger seems not to have been widespread knowledge or considered usable until recently 2016-2017. However, papers about the supply chain or the blockchain in general were still worth reading as basic principles still hold true.

The papers and articles discovered had references that were worth investigating and adding to the bibliography. However, redundant or older sources were avoided. Of particular significance were theses regarding the blockchain as applied to the supply chain by Lund University in Sweden, one of which had a detailed analysis from the organizational side of how a blockchain provenance system could be applied to IKEA's supply chain.

4.2 Formal Papers and Theses

4.2.1 Blockchain in Megacity Logistics

In this paper, a proposal to apply the blockchain to logistics contracting systems is discussed (Polim, Hu, and Kumara 2017). The current problem is an information asymmetry between logistics providers and grocery store retailers in a grocery supply chain.

Most grocery supply chains depend on a 3rd party logistics providers which trade with suppliers or other 3rd party logistics providers. However, retailers prefer to deal with a single accountable agent.

As a result, a centralized 4th party (typically the developer of the IT information system) is empowered to manage and audit the activities of the 3rd party logistics providers. Unfortunately, the 4th party forms an additional layer of bureaucracy as a result, and tends to collude with 3rd parties against the retailers by exploiting information asymmetry against the retailer.

The thesis of the paper was to develop a blockchain data structure to work as a decentralized peer to peer audit system for supply chain contracts, replacing the 4th party and eliminating information asymmetry. This way, retailers can see what others have paid for transport and supply, while logistics providers are incentivized by gaining access to the transaction data of retailer purchases.

The blockchain makes a peer-to-peer audit system feasible, but it works best when there are multiple groups that have competing interests and need to have a balance of power to ensure trustworthiness. The competing interests of multiple logistics providers and multiple retailers would reduce the chance of collusion especially with more participants.

A supply chain based on the blockchain would allow all parties to facilitate logistics contracts and statistically infer market demand, supply, and a fair price. All retailers would be able to make available their delivery needs, and all logistics providers can post their service levels and rates for retailers to pick from. The 3rd party supplier that is the source of a production problem could be easily and transparently discovered, as well as any affected products that could have came from the same factory. Grocery stores could even purchase longer term futures at a discount which allow the suppliers to have more capital on hand.

At the time it was developed (early 2017), smart contract infrastructure was not mature enough to make this practical, so as it was fiendishly difficult to implement and would have needed to use an independent blockchain, which require miners to be raised on their own.

This paper built an proof-of-concept blockchain in Python on Google Forms, with consensus enforced by miner voting, but it was in its very early stages. Despite this, it was very forward thinking as this is essentially how "voting" consensus algorithms such as Quorumchain in JP Morgan's Quorum, or RAFT in Hyperledger Sawtooth work (Olson et al. 2018). Quorum makes that sort of voting private blockchain possible with smart contracts, enforcing the transactions and building Decentralized Apps through code on the blockchain.

However, the idea is significantly more practical now that Ethereum Smart Contracts are in the spotlight. Ethereum allows decentralized applications to be easily built atop its blockchain by taking care of the mining, actual blockchain implementation, and the cryptocurrency, so the developers can focus on the application code and make them immediately usable to the public. Two Ethereum smart contract standards also help facilitate this, ERC20/223 fungible tokens and ERC721 nonfungible tokens.

4.2.2 Aiming for Supply Chain Transparency: Exploring the Potential of Blockchains

This thesis states that "supply chains lack transparency due to complexity and fragmentation". However, the blockchain is expected to provide trustworthiness, nonrepudiation, and transparency to the supply chain. With IKEA, the authors explored areas in their mostly opaque supply chain that urge transparency, and truly fulfill IKEA's corporate mission of providing quality, sustainability, and authenticity.

Sustainability in particular was found to be an emerging driver of customer decisions without incurring significant costs, but customers grow weary of promises that reveal themselves to be empty. IKEA's lack of oversight or even information about the lower levels of the supply chain has been a public relations disaster, and the authors investigate the power of the blockchain to fulfill this need for provenance in the blockchain, since traditional centralized systems are bound by information asymmetry (Lutzenburg 2017).

4.2.3 The Supply Chain has No Clothes

In this paper, the authors state that the blockchain can allow the full history of an item to be seen to all nodes on the network (public or private to participants). This makes it a single audited ledger of all recorded transactions that cannot be repudiated by a single dishonest entity. Transparency and traceability is achieved with the blockchain with an immutable and distributed custody record of each item's journey through the blockchain. Final customers and regulators have strong interests in transparency and traceability to ensure quality, which then influences participants. Thus, participants can track, narrow down, and predict issues in the supply chain to a greater and faster extent than could have been done with traditional IT systems and paper accounting (Francisco and Swanson 2018).

Thus, the authors use the Unified Theory of Use and Acceptance of Technology (UTAUT) to investigate the question of how the blockchain could be utilized and accepted in the real world (Francisco and Swanson 2018). The UTAUT is further described as applicable to future work in Chapter 7.

Before, organizations could only trust large cumbersome paper bureaucracies to produce a trustworthy ledger. Even IT systems for accounting still rely on auditors that laboriously review printed paper archived off-site, quarterly, since the centralized computer systems were run by the company and could be tampered with after.

The authors acknowledge that the blockchain can bring transparency to a new level, but currently "academic and managerial adoption of blockchain technology is limited by our understanding." (Francisco and Swanson 2018)

Scholars must first prove the value of blockchains in supply chains before it can be pitched and adopted by business. The effectiveness and best use cases of the blockchain is unknown due to its infancy and the reliability of real time data that enters it depends on many other factors. The hardware and software frameworks of the Internet of Things and the Blockchain would enhances each other, but neither of them are commonplace yet in the enterprise due to technological limitations or other factors. As such, it might be tough to pitch the blockchain to established companies, but disruptive startups building brand new sharing economies may accomplish this. (Francisco and Swanson 2018)

4.2.3.1 Achieving Transparency and Traceability in the Supply Chain

Three definitions of the levels of transparency and traceability on the supply chain are provided.

- **Opaque** - For whatever reasons, "no information shared between parties, even day-to-day information is obscured." (Francisco and Swanson 2018, p. 3)
  - Opaque and Translucent levels tends to manifest from the use of paperwork, which makes it difficult and expensive to share or move large amounts of data. Thus, organizations become reluctant to share unless forced to by regulators. Much of the supply chain remains in this state.
- **Translucent** - Only Outline information is shared, such as software interfaces or snippets of data, akin to black box design. (Francisco and Swanson 2018, p. 3)
  - Even after the dawn of the Information Age eliminates much of the cost of data sharing, other considerations such as competitive advantage and privacy considerations come into play. Permissioned blockchains can help businesses provide translucence by limiting the visibility of the data, though regulators may often be party to it and there may at least be a public footprint.
- **Transparent** - "Information is shared on a selective and justified basis." (Francisco and Swanson 2018, p. 3) This allows participants to collaboratively develop the information further, significantly reduce needless reinvention and make connections and insights that one would never have realized on their own.
  - The public blockchain systems rely on full transparency for transactions on the blockchain, as influenced by their use in financial markets. However, in the supply chain full transparency of information is not necessarily desired for all transactions, such as price bargaining.

Given this, the authors introduce three archetypes that facilitate transparency and traceability in the supply chain.

- **Product Segregation model** - “Certified materials and non-certified materials are not mixed” (Francisco and Swanson 2018, p. 4).
  - Organic apples can be certified and packed seperately from non-organic apples, to prevent fraud.
  - The Everledger Project, partnered with Barclays, traces the origin and processes involving mined diamonds, using their carbon structure as an identifier. This is an early example of the possibilities of supply chain provenance on the blockchain.
- **Mass Balance model** - Certified and non-certified materials can be mixed where segregation is very difficult or impossible to achieve (Francisco and Swanson 2018, p. 4).
  - Cotton yarn cannot easily be checked for its provenance, so certified yarn can be mixed in with noncertified yarn.
- **The Book and Claim Model** - Instead of having traceability at every stage of the supply chain, it instead enforces the “volume of certified material produced at the beginning (Francisco and Swanson 2018, p. 4).
  - Seafood Quotas - To stop the practice of overfishing, companies or regulators could set quotas on the maximum amount of fish that licensed fishermen can catch. Only the amount of fish that meet the quotas can proceed forward in the supply chain for packing, while any overcatch is not accepted forward in the supply chain and thus disincentivized.

4.2.4 Distributed Ledger for Supply Chains

This paper establishes a possible solution to balance the need for some information asymmetry, but provide transparency in other supply chain segments (Wu et al. 2017).

It proposes that there could be a private blockchain for contract negotiations which could involve special prices or quantities that are bargained and of unique value to each customer and thus would be worth keeping under the table. However, transport and shipping information should remain public much like UPS does. Much like how in cryptocurrencies all transaction information is relevant to all participants, logistics information is just as crucial to everyone in the supply chain. A delay in one area could cause delays elsewhere.

To unify the two without leaking confidential information, a hash of the private transaction is put on the public blockchain, so that it can be matched with the contract that the supplier and customer signed. This is valid as the transporters are only interested in basic information about the product (Wu et al. 2017).

4.3 General and Informal Knowledge

The authors of this paper began with long background knowledge and firsthand experience about the public peer-to-peer decentralized blockchains, Ethereum, Solidity, and the power of decentralized apps and cryptocurrencies that the blockchain made possible. This drew from informal community discussion over the course of several years, which resulted in us witnessing many important posts and case studies on the possibilities of the blockchain.

Certain less formal or unwritten sources such as whitepapers, blog posts, videos, talks, and others were also used, but mainly from primary sources such as the creator of Ethereum, conferences from the Ethereum Foundation, IBM promotional videos, development resources, and explanations and glosses by developers or creators to bring their knowledge to the general public. These sources are referenced in the appendices.

4.4 Smart Contract Applications

The following applications represent the some of the most common and successful use cases of blockchain smart contracts in general. In particular, most smart contracts rely on the issuance of cryptotokens perhaps through Initial Coin Offerings or other distribution mechanisms to represent digital assets such as vouchers, tickets, or voting power.

4.4.1 Initial Coin Offerings: Tokens that do Things

Initial Coin Offerings (ICOs) are currently the most important and popular use case of Ethereum Smart Contracts. These investment vehicles are built on Smart Contracts that issue a token (often ERC20/223 compatible (Vogelsteller, Buterin, et al. 2018).) that provides certain existing or proposed features, anything from profit dividends, to voting shares, to vouchers for future products, to units of value on the proposed platform. These tokens are made available to the public by developers in such eponymous Initial Coin Offerings, which resemble existing Initial Public Offerings (IPOs) in that a fixed amount of tokens are fielded for sale at a flat rate (Ethereum Foundation 2017[a]).

The basis for the value of these tokens are explained on websites with Whitepapers, which typically describe a platform that the group is aiming to develop, their reasons for raising funds through an ICO and their promise that such funds will be used to develop the company and the product, and any future proposed features that the tokens will gain value with. Interested investors simply send Ethereum cryptocurrency to the stated contract, and are sent back the amount of tokens promised. As a result, if the ICO gains popularity it can raise a significant amount of cryptocurrency for the company.

Already, many ICOs have raised incredible amounts of money in the tens of millions of dollars, far more than Amazon, Google, or Facebook gained in their first investment rounds. Lisk, a modular cryptocurrency, collected 15K of Bitcoin, which in 2016 prices is around $9 million USD ($600 ETH * $15,000). The Golem Network raised around $9.6 million in Ethereum in just 30 minutes (Coinstaker 2018).

As for examples of ICOs that have been successful, look no farther than Ethereum, where Vitalik Buterin began a proto-ICO which issued Ethereum to initial investors in return for (Bitcoin) funds for the development of the technology. The result was that an unconventional, high risk, but groundbreaking expansion of the blockchain into Smart Contract technology was developed, with incredible gains for those who invested first: from a few cents to $10, and from $10 to $300 as of this writing.

Much like with the Initial Public Offering (IPO)-fueled dot-com boom of the 1990s, given that the barrier to entry is so low and that investors are desperate to park their money into these speculative vehicles, ICO-based scams have inevitably manifested. Some ICOs do not necessarily set out to defraud their investors, but lose their money from poor design, hacks, or poorly considered economic decisions on tokens. Other ICOs simply fizzle out upon the failure of the team to meet the promises, which is an accepted risk in such investment schemes. Unfortunately, there are ICOs that are outright money grabs with no good faith intention of ever delivering what they promise, or even making such vague promises that there is no value that can be expected at all. Thus, some satirical ICOs have launched such as the Useless Ethereum Token, which comes clean and tells investors not to expect any profit other than the speculative value of this ERC20 compatible token, that any funds raised in the ICO will immediately go towards the organizer's personal benefit, and that the organizer reserves the right to manipulate the token amount or settings at will.

As a result of the rise of ICO scams, the US and China have reacted swiftly to the phenomenon with differing reactions. The US Securities and Exchange Commission has determined that these ICOs are securities just like IPOs are, and need to be licensed. They came to this conclusion while investigating the Decentralized Anonymous Organization (DAO), the most famous smart contract investment vehicle, which was ignominiously hacked: they elected not to prosecute this specific case, and acknowledged that ICOs would be warily accepted as just another form of security. More detailed guidelines for how registration would work remains in progress (Securities and Exchange Commission 2017). The People's Bank of China had a very different reaction: they immediately enforced a moratorium on certain ICOs they believed to be scams, and ordered them to refund all holders immediately. After the liquidation of such "scam" ICOs and the dismantlement of some involved exchanges, they worked on establishing guidelines for how ICOs should be regulated and held. This is in line with their typical policy on blockchain solutions: to first enact a blanket ban until they can control their undesirable effects, and then allow it with more stringent restrictions such as licensing (People's Bank of China, 2017).

While the "dot-com boom" of the 1990s caused a serious economic slowdown and the loss of huge amounts of investment money in dubious ventures, it is clear that this influx of money provided the very few successful and honest startups with the resources to compete against entrenched big businesses. Although Amazon was a mere online bookseller, and barely survived the “dot-com crash”, it would end up putting other brick-and-mortar booksellers out of business and even accomplish Jeff Bezo’s lofty dream from the start of producing an Everything Store. Of course, this is tempered by failures such as Pets.com or even Enron, where greed and mismanagement got the better of people when they encountered easy money. But the significantly greater investment involved in ICOs could have similar effects on the startups that manage to live through a future crash: it would break some and fool some, but also make give some blockchain solutions the potential to revolutionize the way we use technology.

The example of ICOs over the past two years and more that it has been active in operations outlines all the risks and rewards of Smart Contracts in themselves, and is living proof that despite all its flaws, many sorrows and scams, Smart Contracts can function and can make incredible digital organizations possible. Governments have begun to establish legal precedent in this wild west of investment, adding more bedrock to the legitimacy of these systems as a factor in the global economy.

4.4.1.1 World Food Program: Cryptovoucher Case Study

As an example of the power of private permissioned blockchains, the World Food Program has developed a blockchain ticketing system using an Ethereum-based private permissioned blockchain to quickly and effectively deliver food vouchers to Syrian migrants in Jordan. According to a report on the World Food Program website, the initiative was developed in 2016 as a proof of concept that was then tested in simulations in January 2017, and then showed great promise. The system was then put into pilot in Azraq Refugee camp in Jordan and was so successful that it has not been phased out, serving 10000 refugees to date (WFP Innovation 2017).

It appears that the system is designed to take the place of direct cash transfers that use donations to purchase food locally, or paper vouchers that have little tracking and could be subject to counterfeiting, or centralized payment systems that depend on slow bureaucracy, could be subject to insider accounting fraud, and is a single point of failure.

Unlike normal cryptocurrencies/cryptotokens, the value of the cryptovoucher is guaranteed directly by the World Food Program and its associates, forming the first significant use case of a cryptovoucher in practice. This significantly speeds up accounting across various state borders and also cuts down on bureaucratic waste or corruption. It also cuts out middlemen such as banks or other institutions that would take a cut, where fees of 30% on millions of dollars are not uncommon. Finally, insiders within the organizations or their partners themselves cannot simply siphon money out without defeating at least half the miners (WFP Innovation 2017).

The cryptovouchers are apparently linked to their recipients in combination with the existing IrisGuard system in place for distributing benefits. IrisGuard is a biometric identification system that is ideal for refugees who rarely have valid papers on hand and may lose issued vouchers. This may mean that the biometric signature is actually utilized as the secure element for a private key of the account, thus ensuring that each individual gets their fair share (WFP Innovation 2017).

Even though the address information and transactions are stored on the blockchain and fully accountable records for the World Food Program, they are effectively anonymized as hashes on the Blockchain such that sensitive data (which here could make the difference between life and death) never needs to be recorded on any database, and for this purpose may be counterproductive anyway. This illustrates the powerful duality of the Blockchain: it can be fully accountable in transactions and authentication of users, but can remain otherwise fully anonymous to facilitate secret ballots or prevent personal information from needing to enter a central database (WFP Innovation 2017).

However, although this tracking is used in a positive manner to cut down on bureaucracy, corruption, and waste without leaking identities, it can also be a powerful tool for governments to eliminate anonymity for good in finance. The same methodology could be used by a central government such as in China's envisioned cryptocurrency to provide a cash alternative wallet that can be linked without a doubt to each individual's national identity card. Thus, transfers of money from one person to another can be seen immediately by the central government, allowing them to stamp out visible corruption or tax evasion easily, among other things. Of course, this sort of surveillance in financial transactions is already the case with bank transfers and very popular mobile payment systems such as Alipay and Wechat pay, but it seems like the goal is to eliminate paper cash itself: the last bastion of anonymity in finance.

4.4.2 Hyperledger: Seafood Supply Chain Traceability

Hyperledger Sawtooth provides a tantalizing seafood supply chain model that closely tracks the provenance of very perishable seafood where it is extremely crucial to ensure that it was stored correctly, obtained lawfully, and rejected when expired. A public permissioned blockchain is used so that any customer or regulator can check the provenance of the fish they consume. A permit of perhaps 1000 fish is provided to fishermen, who are then only able to sell 1000 fish down to the largest consumer, the supply chain, whereas any overcatch is not saleable (only doable through less scrupulous means that have significantly lower profit margins) (Hyperledger 2018).

One major benefit of the blockchain is to not just facilitate trade, but also restrict resource consumption at least to only what is demanded or legally allowed.

In sustainable seafood, there must be a cap and trade system enforced whereby laws obligate fishermen not to catch more than a certain amount of fish to ensure that fish stocks have enough population to replenish. This is particularly difficult to do with paper which often ends up with excruciatingly slow regulatory red tape or poor enforcement, and facilitates the rise of middlemen that take huge markup to deal with the provenance paperwork while the fishermen are away doing whatever.

Since the tokens cannot be forged by any individual in the system, many of which (such as either the fishermen or the middlemen) have strong motivations to do so, the blockchain provides the regulator with the assurance that no one can at least sell to the market fish catches over the limit, and the end retailer/restaurant with the assurance that neither of their purveyors can forge the provenance of their stocks (Hyperledger 2018).

Chapter 5 Methodology

Existing distributed solutions to **the procurement problem** as described by Satapathy 1999 and Ramirez 2017 are applied to develop a possible methodology for building a blockchain implementation. This methodology simplifies the core components of this blockchain supply chain, but can be extended with modules to fit the needs of more complex megacities.

5.1 Marketplace

One major problem with the typical bargaining process is that retailers give a price range to the middlemen, who then push suppliers to accept a lowered price with commission, or risk delaying or losing the sale if the suppliers bargain. The retailer also forfeits a price margin to the middlemen in exchange for the service of reducing the amount of agents to bargain with and to aggregate inventory.

Instead, in this bargaining process, to provide the volume out of otherwise small and divided farmers, an order book is given to fill orders much like an exchange but with the setup of a marketplace. The retailer sets up an inquiry by filtering out by quantity, delivery date, and whether the retailer or supplier arranges shipping. The retailer is then given a selection with prices, and they either accept the face value of one, or bargains with any one supplier directly until both parties find a price that works for them.

This single supplier might not be likely to be able to fill the volume needed by most retailers, but as a collective they can. As such, if other suppliers provide an equivalent product and set a minimum price that is under the given price, they can join this volume order on the order book automatically. If the retailer still does not have enough quantity needed,they can fill the rest by bargaining again with a higher price with a supplier in the remaining group that has a higher minimum price or shipping cost. Once this supplier accepts the new price, the retailer makes the volume order again until their quantity is filled.

This way, the retailer only has to bargain a few times with small suppliers, but still get their volume orders filled since if a concluded bargain is above the other suppliers' own minimum price, they automatically join the order.

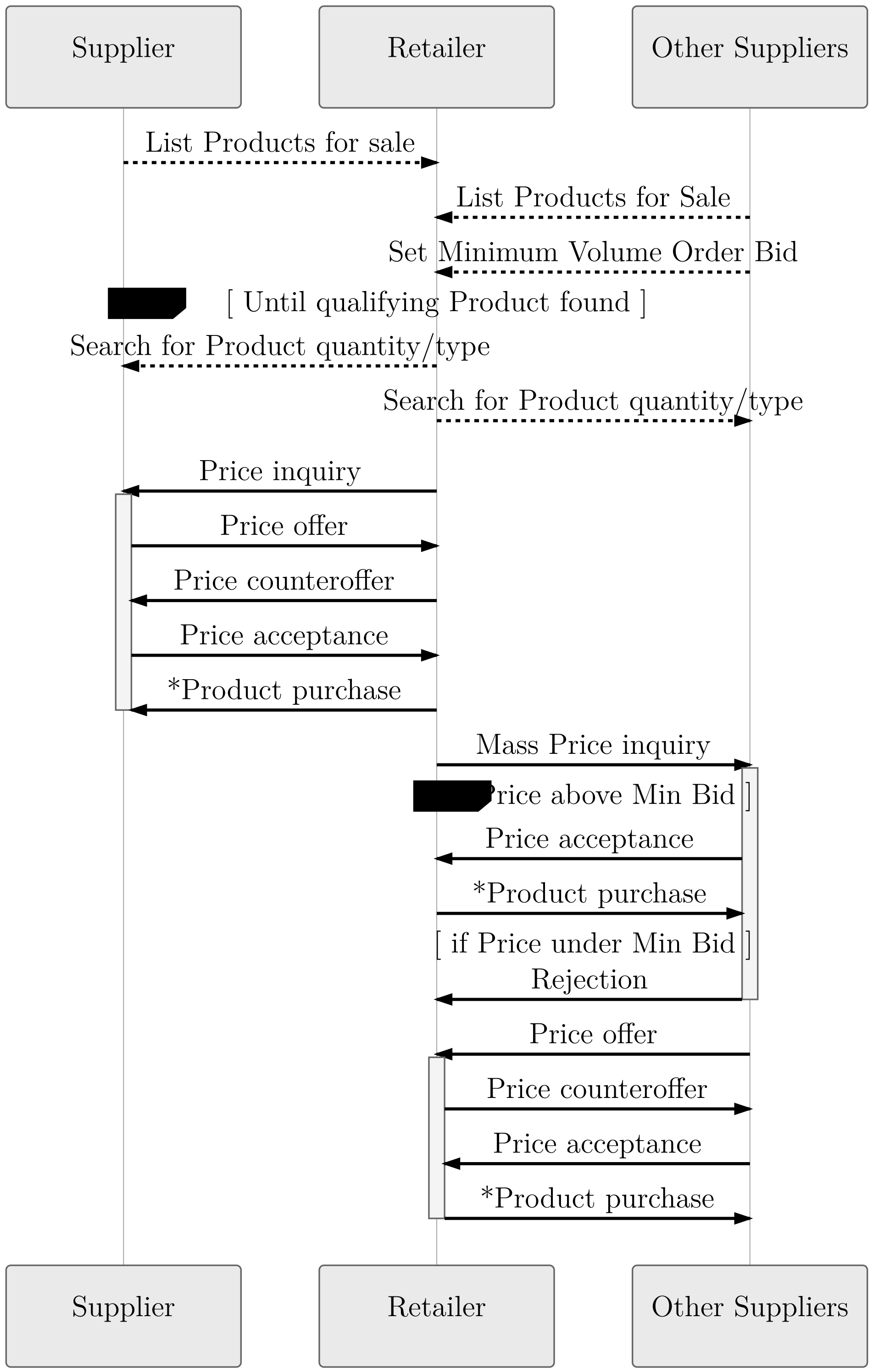


Figure 5.1: How a Marketplace can aggregate volume from small suppliers without a middleman. Credits to Alvaro Ramirez of eHarvestHub for assisting with this diagram.

5.2 Logistics Matchmaking

Transportation service can be either paid by supplier or paid by retailer. In a 1PL model, the retailer could elect to provide their own transport whereby the retailer uses their own procedures for documentation and regulatory purposes, but which can be integrated into this blockchain system.

If the supplier provides their own logistics provider, due to visibility, regulatory, and liability issues pertaining to the retailer, such employed logistics providers should utilize this blockchain system to communicate progress to the retailer.

Typically though, the orderer (either supplier or retailer paying for shipping) will seek out a 3PL that can transport the goods, and record their progress on it.

Some benefits of an individual logistics provider using this system instead of relying on a broker are that orders can be made ahead of time, and it is clear from the records on the blockchain where the trucker will be. The trucker can thus easily see when they would be idle and then offer it up also as well. As orders get added, retailers can hire truckers in advance too to fill backhauls (shipments with the trucker's aim of returning home).

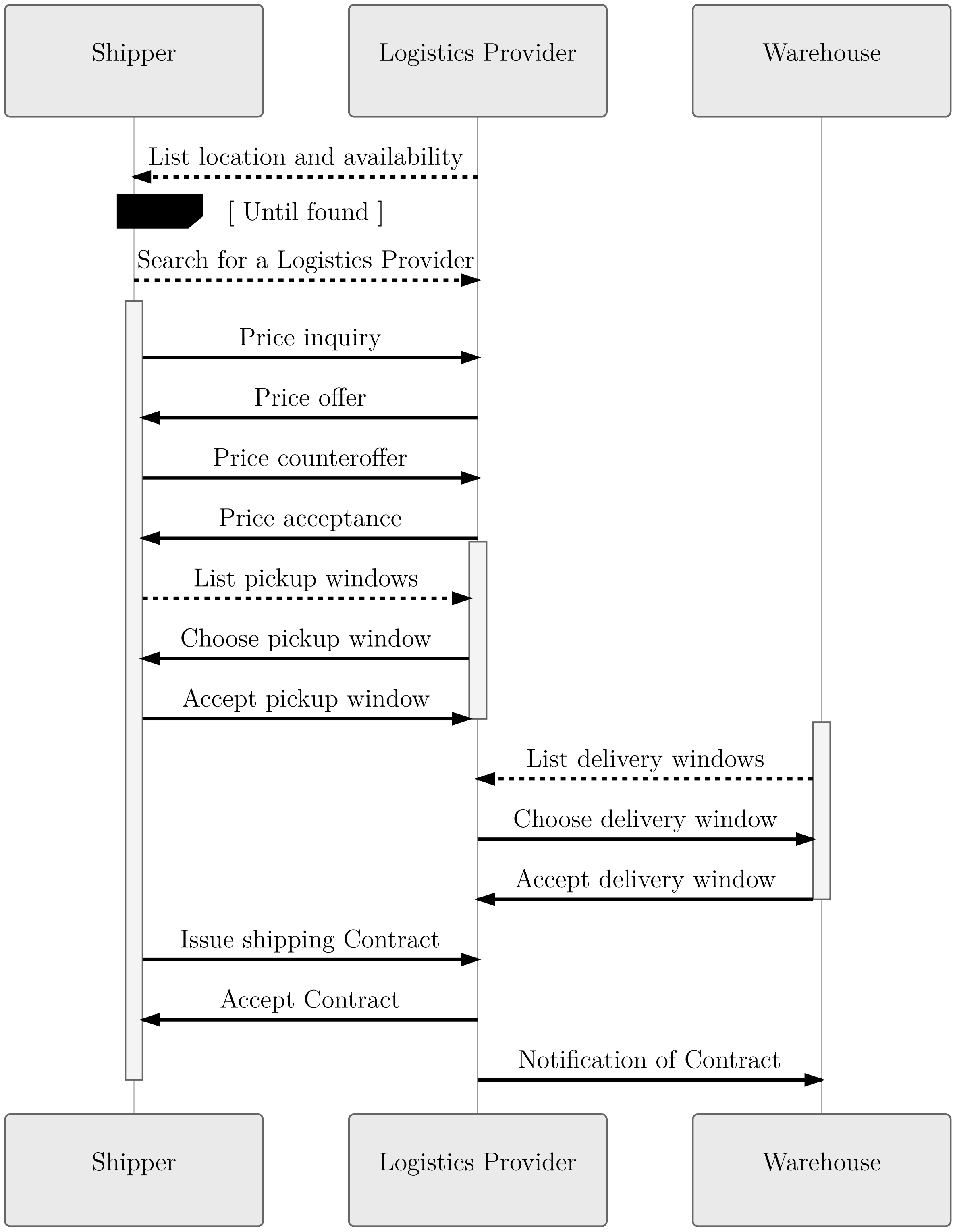


Figure 5.2: How Logistics Providers can be matched to retailers without a middle-man. Credits to Alvaro Ramirez of eHarvestHub for assisting with this diagram.

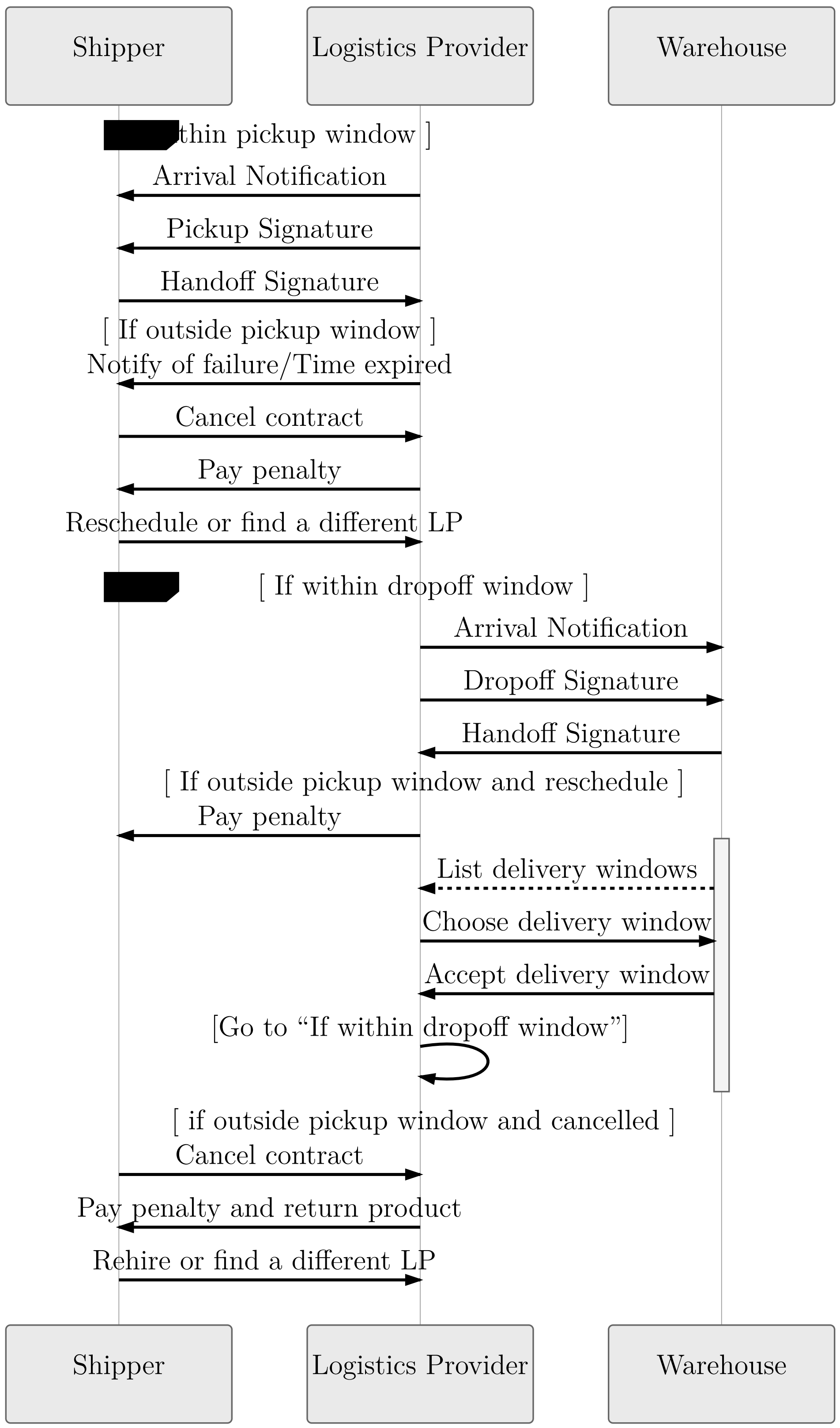


Figure 5.3: How Logistics Providers handle and report the delivery on the blockchain. Assistance by Alvaro Ramirez.

5.3 Scenario

Let there be three entities in this supply chain:

- **Suppliers** - Produces/Procures/Supplies the product for retailers. Could be farmers, food packers, or manufacturers.
- **Logistics Providers/Shippers** - Transports items between suppliers and retailers (and if necessary, a return to sender).
- **Retailers** - Purchases and receives products.

It will be assumed that all entities accept shipping or deliveries only during business hours (9am - 9pm). All entities should take these limited hours into account with travel times, to account for the need to return the package if delivery is not possible.

This study assumes that the suppliers and retailers are already familiar with each other. However, the Logistics Providers could be either traditional transport providers, or even independent contractors forming a sharing economy, so they represent the largest source of uncertainty in the system.

5.3.1 Product

Let there be a set of products that retailers demand in varying amounts, as in the sample list below.

It is assumed that either the supplier or the retailer will take care of moving the boxes into the transporter's vehicle if necessary.

5.3.1.1 Build-to-Order Products

In the Build-to-Order (BTO) supply chain (Satapathy 1999), products suffer an expiry date, need to be kept at certain temperatures to reduce spoilage, and the retailer is acutely aware of differences in quality. Some bto consume each other as dependencies.

- Bananas
- Apples
- Cherimoyas

- Grapes
- Milk
- Pears
- Peaches

5.3.1.2 Build-to-Plan Products

In the Build-to-Plan (BTP) supply chain (Satapathy 1999), products do not have a significant expiry date, should be commoditized (whereby the retailer cares not for the differences in quality), do not consume each other as dependencies, and can be made of any material such as paper or plastic, which introduces cost and supply differences.

- Cups
- Plates
- Napkins
- Forks
- Spoons
- Knives
- Chopsticks - Less common, but very necessary for noodles and picking up sushi.

5.3.2 Order Tickets

Let there be order tickets that go through various stages through the supply chain, as described below.

1. Supply Stage - Produced by the supplier to indicate their inventory to retailers (or alternatively, inventory futures).

- Supply Futures Stage - If the supplier is confident that a delivery or production will be produced by a given date, they can offer a supply future to retailers to be delivered at a certain date.

2. Order Stage - When a retailer makes a purchase of a product, the ticket shifts to the order stage, becoming available for logistics providers to see and bid on.

3. Transport Stage - When the transporter accepts an order ticket, they move to fulfill the product delivery from supplier to retailer.
4. Receipt Stage - When the transporter delivers the product to the retailer, the ticket enters the final receipt stage and is kept as a record.

5.3.3 Geography

The following grid provides a simple idealized vision of the city grid. There are four suppliers, four retailers, one gas station in the center, and four transports, equally distributed across the city grid.

```
        T
  | S | C | S |
T | C | G | C | T
  | S | C | S |
        T
```

Figure 5.4: A possible idealized city grid for the scenario.

Let there be a city represented by the above diagram that has a regular grid street pattern and no traffic.

- **Immobile** - In this scenario, immobile entities (Suppliers, retailers, gas stations) will never move.
  - [S]uppliers - Suppliers must factor in the location of retailers and logistics providers into the cost to the retailer.
  - [C]ustomers - Retailers must factor in the location of suppliers and logistics providers into the cost of delivery.
  - [G]as Stations - Logistics Providers need to fill up here before they run out of gas. This can be done anytime as long as the logistics providers get to their destination on time.
- **Mobile** - In this scenario, mobile entities can linger at any point in the grid.
  - [T]ransporters - Logistics Providers have a certain amount of gas which must be purchased, and must return to the gas station to get it. This can be done off the chain on their own time.

5.3.4 Transportation Costs

Transportation cost are dictated by Speed, Distance, and Gas Price. The transporter's app should calculate the total cost and the total profit for them for each ticket. Each transporter should be able to select the best one for their needs, but each retailer should also be able to select the transporter with the least costs.

- Gas Price - The current gas prices should be factored in at least indirectly. This is one of the basic predictors of the cost of transport.
- Distance - The distance estimate between two locations can be provided by an oracle, such as Google Maps.
- Profit Margin - The lowest profit margin that the logistics providers are willing to take for the trouble taken. It would converge towards a constant wage per hour at market rates (or a government mandated price floor).

One issue is that even when the path between two locations can be estimated, there may not necessarily be a resolvable or fast path, due to weather or road conditions. This issue is akin to the "halting" problem in Turing machines where it cannot be calculated when calculation will stop.

To deal with the "halting problem" of transport costs, the retailer should set a "gas limit", a maximum amount that the retailer is willing to pay for transport. It must be able to pay for both the transport from supplier to retailer and a return journey if the retailer cannot be found.

In short range journeys, since the cost of going there and back is low, a gas limit of two times the one way transport cost should be provided, given that if package is returned a sender the costs and time wasted by having to redeliver is not as high. However, longer range journeys will probably need to be insured and reputation visible to manage the risk of having to return to sender, which can be provided by an Insurance module, which is described below.

5.3.5 Additional Modules

The system should be designed to be a modular system. There can be many-to-many input and output attached onto each other, such as a production add-on attached to this logistics network, which then has reputation and insurance attached on it.

- Reputation - Only those who actually took the service are allowed to rate entities.
- Trustworthiness - A module could assess the likelihood that the transporter has either brought the items to the retailer or back to the supplier safely.
- Availability - A module could assess the likelihood that the Transporter has delivered the goods on time, and the likelihood that the Supplier and Retailer are available at their appointed times.
- Quality Standard/Reliability - A rating module specifically for Suppliers. The quality of the product is rated against some industry standard criteria, and the reliability of filling those orders (for futures) can also be rated. Cost could be additionally weighed against quality standard/reliability. A real optimization algorithm could be applied here.
- Insurance - Insurance can be set up to provide payouts given a certain amount of risk. Insurance providers thus judge the risk for retailer selections and push the retailer to avoid risky logistics providers or suppliers.
- Kanban Production - A production module can be set up in the factories themselves to enforce a pull system where demand enforces production, rather than the typical push system.
- Long Range Inter-network Delivery - Long range rush delivery via Uber between cities can also be done.

5.4 Units

- **Standard Coin Unit**, expressed as `coin = 1 value coin` - Coin is the medium of exchange token in this system, and all services and goods are denominated in it. All entities seek to maximize the amount of coins they have, thus Suppliers and Logistics Providers seek the highest prices and lowest costs, while retailers seek the lowest prices.
- **Gasoline** - Used as the ultimate basis of cost for transportation, passed down to the retailer in the price of the burden. The transporter is also required to have enough fuel to act upon orders, or to reach the gas station, which gives them an additional movement cost that is factored in the cost.
- **Standard Gas Unit**, expressed as `gas` - The standard gas unit is defined as a certain amount of gas. It is fungible (can be obtained and used in fractions).

- **Price of Standard Gas Unit**, expressed as `gas.price = coins` - The cost of one standard gas unit. The gas price varies based on the markets and is outside of the control of the entities in this model.
- **Standard Cargo Unit, Cargo**, expressed as `cargo(weight, size) = weight * size` - The standard cargo unit based on weight and size. A transporter should assess their vehicle to see the maximum amount of items in Cargo units that can fit.
- In this model, the effect of cargo on gas consumption is considered to be negligible.

5.4.1 Geography and Time

Given a city grid, there are two points for which there is a non-diagonal path, thus determining the distance. Also, because of a city-wide speed limit, moving one block will cost at least 1 turn.

- **Standard Distance Unit, Block**, expressed as `block = 1 city block` - The standard unit of distance between two locations, delineated by blocks.
- Properties
  - **Location**, expressed as `location = (x, y)` - Each entity is on a block in the grid city. There can only be one immobile entity on each block, but mobile entities such as logistics providers can move through or rest on any block.
- Functions
  - **Path**, expressed as `path(location1, location2)` - The shortest non-diagonal path between two locations, and how to get there (calculated from openstreetmap or Google maps).
  - **Distance**, expressed as `distance(path) = blocks` - The shortest non-diagonal path between two locations, as a numerical value.
- **Standard Turn Unit**, expressed as `turn = 1 game turn` - To simplify the model, time is packaged into board game style turns.
- Speed Limit, expressed as `speed = 1 block / 1 turn` - All logistics providers are subject to at a city speed limit of 1 block per 1 turn. Thus, logistics

providers cannot exceed this speed. This speed limit models the effect of stoplights and intersections that cause cars to stop and idle, so logistics providers would be unlikely to exceed the speed limit even if they could. However, this will become a larger factor in long distance transportation.

5.5 Modeled Entities

- Transporter, expressed as `tp()` - The transporter has certain properties that determine its own transport cost, the profit margin it is looking for, and thus the cost to the retailer.
- Properties
  - **Location**, expressed as `tp.location = location` - The current location of the transporter on the grid.
  - **Capacity**, expressed as `tp.capacity = cargo` - The capacity that the transporter can carry, measured in Cargo units.
  - **Fuel**, expressed as `tp.fuel = gas` - The amount of gas that can be held as fuel inside the transporter.
  - **Minimum Profit Margin**, expressed as `tp.min_profit = coin` - The minimum profit that the transporter will accept to both pay for labor costs and get additional cash. Although the transporter can set this to any value, competition pushes the logistics providers to abide by a market price, since a minimum profit that is too high will hide all possible orders. There is no maximum profit limit.
  - The transporter can choose to develop a profit margin based on percentage or unit, but that is their own deal.
- Functions
  - **Path**, expressed as `tp.path(tp.location, location1, location2) = path**` - Given the current location, return all the paths that need to be taken to both pick up the cargo from the supplier and reach the retailer.
  - Loading and unloading will each take up one turn. However, no gas will be consumed in this process.

  - **Distance**, expressed as `tp.distance(tp.path) = blocks` - The total distance that to be driven given a path.
  - **Speed**, expressed as `tp.speed(distance, time) = distance/time` - The ratio of distance driven to the turns taken to do so. Since it is in the city, all logistics providers must remain under the speed limit of 1 cell per 1 turn.
  - However, slow logistics providers can require more turns to move, such as 1 cell per 2 turns.
  - **Efficiency**, expressed as `tp.efficiency(distance, gas) = distance/gas` - Since driving x blocks requires y gas, efficiency is the ratio of distance to gas (similar to MPG). This efficiency can differ from transporter to transporter.

5.6 Contracts

Only two contracts involve logistics providers, though supply contracts and reciept contracts may be useful as information records of where possible orders can go.

- Supply Contract - The Supply contract represents a product available for sale.
- Properties
  - **Supplier Location**, expressed as `contract.sp.location = location` - The location of the product supplier.
  - **Quantity**, expressed as `scontract.quantity = cargo` - The amount of cargo in Standard Cargo Units that must be moved.
  - **Unit Price**, expressed as `contract.unit_price = coin/cargo` - The product unit cost of the cargo. This already includes the supplier's profit margin.
- Order Contract, expressed as `ocontract(tp)` - A contract in the order stage. Available for logistics providers to pick up.
- Properties
  - **Retailer Location**, expressed as `ocontract.cu.location = location` - The location of the retailer.

  - **Load**, expressed as `ocontract.load = cargo` - The amount of cargo in Standard Cargo Units that the retailer orders. Can be less than quantity, but cannot exceed it.
  - **Deadline**, expressed as `ocontract.deadline = turns` - A deadline: the amount of turns left to complete this delivery.

- Functions
- Transport Contract, expressed as `tcontract()` - A contract in the transport stage. The main difference between this and the order contract is that a transporter has already picked it up, and it is tracking their progress.
- Transporter - The transporter assigned to this contract.
- Path

  - ETA - Turns needed to deliver to retailer. Derivative of path.
  - Gas Needed - Gas that is predicted to be used.

- Cost - See formula below.
- Profit - See formula below.

  - Total Cost to Retailer - See formula below.

- Receipt Contract, expressed as `rcontract()` - After the transport is completed, returned to sender, or fails, the retailer recieves a receipt Contract
- Status - The result of the transport: Delivered, Returned to Sender, Lost.
- Return - A function that can be used make a return claim to the supplier.

5.7 Computations

These computations are functions of the above contracts. They help the logistics providers to find the most profitable orders, and Retailers to find the cheapest logistics providers.

- **Base Transport Cost**, expressed as `cost(location1, location2, tp, gas.price)` - The expected base transport cost not including labor costs. It takes into account total trips taken, the distance per trip, the estimated efficiency (km/L) for the transporter, and the gas price. Labor costs are to be recouped at the user's leisure through profit.

- Total Trips Needed, expressed as `ttt(ocontract.quantity, tp.capacity) = ceiling(quantity / capacity) = cargo` - The total amount of trips needed to move all the supplier's cargo to the retailer with a given transporter.
- `cost = (ttt(ocontract.quantity, tp.capacity) * distance(path(ocontract.sp.location, ocontract.cu.location)) * gas.price ) / tp.efficiency = coin`
- total trips needed = ceiling[{Q/C}]' - The total amount of trips needed to move all the supplier's cargo to the retailer with a given transporter.
- **Profit Margin**, expressed as `profit = ocontract.reward - cost = coin` - Since profit is renevue subtracted by cost, the transporter seeks only contracts that are above their minimum profit margin `tp.min_profit`.
- The profit margin will vary but tends to stabilize at whatever the majority of logistics providers will want in exchange for doing the transport, since competing logistics providers will push profit margins down to get retailers.
- **Total Price to Retailer**, expressed as `total = cost + profit` - The total price that the retailer pays to the transporter for their service, based upon an agreed profit margin.

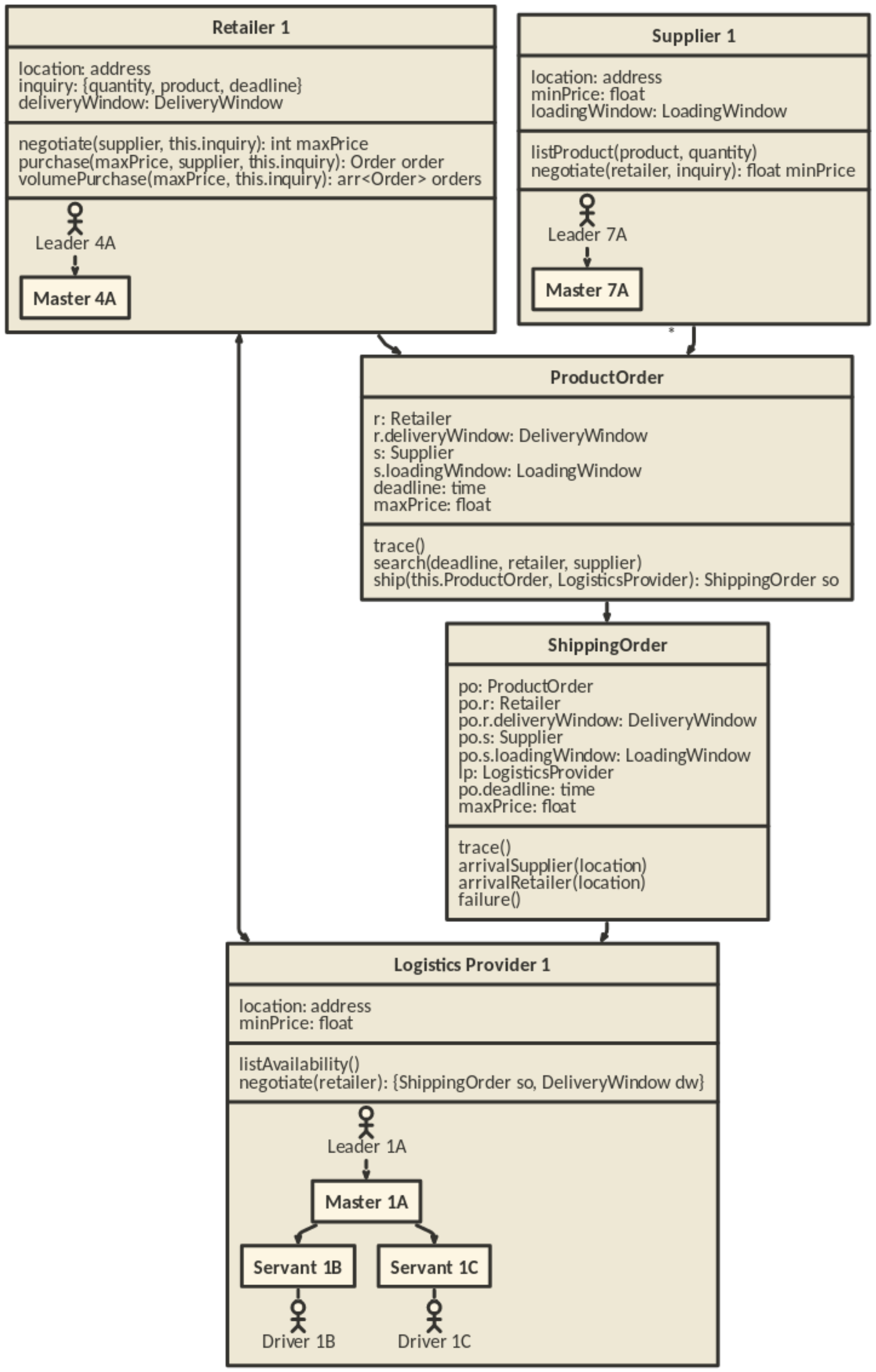

Retailer 1
location: address
inquiry: {quantity, product, deadline}
deliveryWindow: DeliveryWindow
negotiate(supplier, this.inquiry): int maxPrice
purchase(maxPrice, supplier, this.inquiry): Order order
volumePurchase(maxPrice, this.inquiry): arr<Order> orders
Leader 4A
Master 4A
Supplier 1
location: address
minPrice: float
loadingWindow: LoadingWindow
listProduct(product, quantity)
negotiate(retailer, inquiry): float minPrice
Leader 7A
Master 7A
*
ProductOrder
r: Retailer
r.deliveryWindow: DeliveryWindow
s: Supplier
s.loadingWindow: LoadingWindow
deadline: time
maxPrice: float
trace()
search(deadline, retailer, supplier)
ship(this.ProductOrder, LogisticsProvider): ShippingOrder so
ShippingOrder
po: ProductOrder
po.r: Retailer
po.r.deliveryWindow: DeliveryWindow
po.s: Supplier
po.s.loadingWindow: LoadingWindow
lp: LogisticsProvider
po.deadline: time
maxPrice: float
trace()
arrivalSupplier(location)
arrivalRetailer(location)
failure()
Logistics Provider 1
location: address
minPrice: float
listAvailability()
negotiate(retailer): {ShippingOrder so, DeliveryWindow dw}
Leader 1A
Master 1A
Servant 1B
Servant 1C
Driver 1B
Driver 1C

Figure 5.5: What type of smart contracts could be created and which agents interact with each other using them.

Chapter 6 Implementation

The implementation explored in this thesis implements both the negotiation and logistics components of the methodology. Due to time limitations, not all aspects of the methodology described previously could be fully explored in this blockchain implementation, such as volume orders, inventory management, logistics provider tracking, or physical IoT devices.

This thesis simply aims to provide a basic code structure and methodology to show that it is possible for blockchain smart contracts to be applied to the procurement problem. It is left to future researchers and developers to explore further possibilities, test results, or solutions.

Inventory would be a system for farmers to record their product inventory specifically with the purpose of offering them to the market for sale. In this implementation, inventory is handled in a simple manner of recording product assets on the blockchain that can then be made available for sale. The eHarvestFarm app from eHarvestHub (Ramirez 2017) can be considered to be a vision of what such a UI would look like to the farmer, so in our implementation it is assumed that the app exists and pushes information to a local blockchain node;s REST API to record their inventory and offer products to retailers.

The **Marketplace** would post and matching orders. Two marketplaces would exist, a market for products and a market for shipping contracts.

Based on this information, the matchmaking system gathers suppliers that fit the requirements by location and filter criteria first, then all of those nodes are ordered to report their price (if they don't report they are not included).

The finer details of negotiation of price and relationship building can be done off chain through traditional means of communication. The blockchain in these negotiations instead takes the place of formal inquiries or offers that are expected to be recorded on paper and signed to hold both parties to its promises.

These transactions should probably be transmitted using a permissioned blockchain, since only the parties of the transactions and any optional notaries need to have a copy. The logistics provider that picks up the product and competitors only need to know of the existence and resolution of these deals, not their confidential intricacies.

Logistics in this implementation draws upon the Hyperledger Perishable Goods Network template with a public logistics system for shipment tracking and temperature monitoring (Hyperledger Foundation 2018[d]). Once product purchase negotiations are concluded by suppliers and retailers, their product listing status changes initiate order bids on this public logistics blockchain. This logistics provenance system can be further refined to comply with the Product Traceability Initiative.

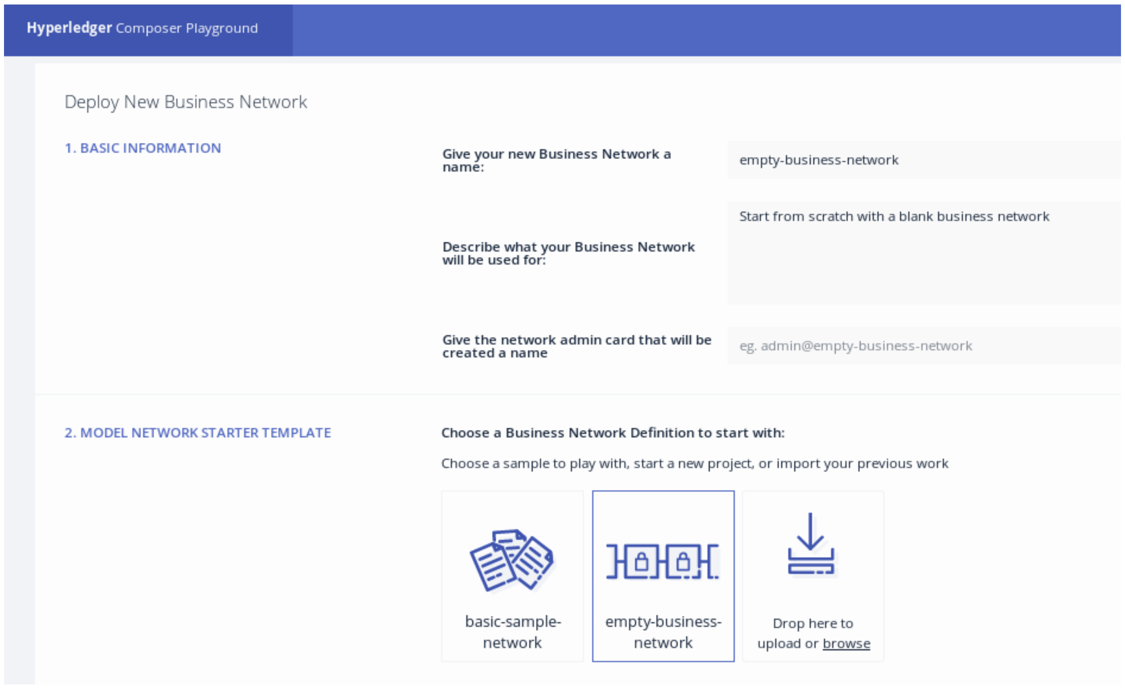
6.1 Balancing Data Confidentiality and Public Traceability

In the supply chain, we expect any provenance data required by the Produce Traceability Initiative (PTI) to be made public for verifiability, so a public permissioned blockchain fits this use case. Produce descriptions, locations, handling requirements, and shipment tracking are all public information that each entity in the supply chain are interested in (Produce Traceability Initiative 2018).

We also expect most other transactions such as price offers and other negotiation transactions to only be sent to involved parties and any notaries they both recognize (such as a regulator, arbitrator, or simply a third party record keeper). Such information can be business secrets of a confidential nature since a special deal offered to one retailer due to circumstance might invite jealousy from others.

However, it is expected that a logistics provider that ships the product from the shipper to the retailer would need to be sure that a sale actually was reached between existing retailers and suppliers. Thus, a public product listing should show the minimal listing and its change in state to relevant logistics providers before it would accept the shipping contract. This public product listing would not include price or negotiation data.

6.2 Hyperledger Composer Overview


Hyperledger Composer Playground

Deploy New Business Network

1. BASIC INFORMATION

Give your new Business Network a name: empty-business-network

Describe what your Business Network will be used for: Start from scratch with a blank business network

Give the network admin card that will be created a name: eg. admin@empty-business-network

2. MODEL NETWORK STARTER TEMPLATE

Choose a Business Network Definition to start with:

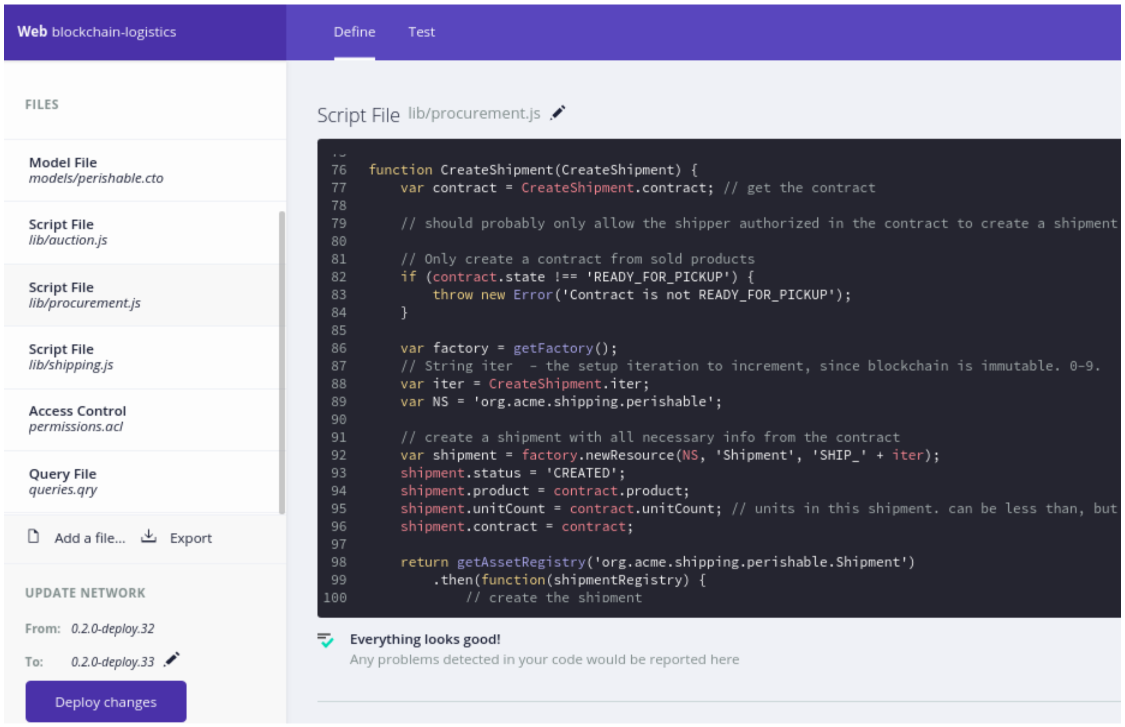
Choose a sample to play with, start a new project, or import your previous work

Figure 6.1: Hyperledger Composer Playground Development Environment (Hyperledger Foundation 2018[c]).

Hyperledger Composer is a development framework for building Business Networks on Hyperledger blockchains. **Hyperledger Composer Playground** is an integrated development environment (IDE) for developing, testing, and debugging blockchain concepts and smart contracts (Hyperledger Foundation 2018[c]). This way, a business architect and a developer could work together to define and test strategies for applying the blockchain to business needs. The infrastructure of a blockchain solution can also be built and tested atop the business network specification. It allows REST API endpoints to be defined through Queries, so that webapps, mobile apps, or even conventional IoT devices can be built to interact with their local blockchain node (Hyperledger Foundation 2018[c]).

Hyperledger Composer Playground runs on IBM Bluemix, and has a decent user interface and set of developer tools. It exports to packages that can then be deployed to production Hyperledger blockchain systems on IBM Bluemix or through personally operated nodes (Hyperledger Foundation 2018[c]).

6.2.1 Business Network Archive



```
function CreateShipment(CreateShipment) {
    var contract = CreateShipment.contract; // get the contract

    // should probably only allow the shipper authorized in the contract to create a shipment

    // Only create a contract from sold products
    if (contract.state !== 'READY_FOR_PICKUP') {
        throw new Error('Contract is not READY_FOR_PICKUP');
    }

    var factory = getFactory();
    // String iter  - the setup iteration to increment, since blockchain is immutable. 0-9.
    var iter = CreateShipment.iter;
    var NS = 'org.acme.shipping.perishable';

    // create a shipment with all necessary info from the contract
    var shipment = factory.newResource(NS, 'Shipment', 'SHIP_' + iter);
    shipment.status = 'CREATED';
    shipment.product = contract.product;
    shipment.unitCount = contract.unitCount; // units in this shipment. can be less than, but
    shipment.contract = contract;

    return getAssetRegistry('org.acme.shipping.perishable.Shipment')
        .then(function(shipmentRegistry) {
            // create the shipment
```

Everything looks good!
Any problems detected in your code would be reported here

Figure 6.2: Hyperledger Composer Playground Business Networks (Hyperledger Foundation 2018[c]).

Business Networks in Hyperledger consist of the following files:

- **Model File** (.cto) - “Defines the structure and relationships between model elements (Hyperledger Foundation 2018[b])”. The model file is written in Composer Modeling Language to define structures of digital assets, transactions, and participants that would be recorded on the ledger, much like for an object oriented interface to a database.
- **Script File** (.js) - Defines the business rules around what transactions are possible and what they do, using Javascript (Hyperledger Foundation 2018[c]). Akin to Smart Contracts in Ethereum.

- **Access Control** (.acl) - Defines access controls for what type of participant roles exist, and what participants are allowed to create, read, and write under those roles (Hyperledger Foundation 2018[c]).
- **Query File** (.qry) - Queries resemble SQL statements and object relational database views. These can generate REST API endpoints for participants to query transactions and their data out of the blockchain. Queries abide by existing transaction access control limitations, so even though queries are available to everyone transactions are not just plain accessible to everyone.

6.2.2 Hyperledger Business Network Templates

The Hyperledger Perishable Goods Network template was used as a basis, which provides a good basis that establishes how simple shipping contracts are issued between three participants, and how penalties are meted out depending on what data inputs are provided (Hyperledger Foundation 2018[d]). However, it provides no price negotiation methodology.

To integrate the negotiation of these contracts and product listings as digital assets, the Hyperledger Car Auction Network template was drawn upon with its blind auction price negotiation mechanism, integrated into the Perishable Goods Network template, and our own methodology produced to form the implementation in this thesis (Hyperledger Foundation 2018[a]).

One more template that is useful for future work as an example of compliance in food regulations from IBM is a traceability system that models FDA Foreign Supplier Verification Program (IBM 2018[a]).

The templates are licensed under the Apache 2.0 license, so the code snippets here and in the associated repository are licensed accordingly by the author, Lawrence Wu.

6.3 Model Definitions

This business network defines participants and digital assets.

6.3.1 Participants

- Supplier - Produces/Procures/Supplies the product for retailers. Could be farmers, food packers, or manufacturers.
- Logistics provider/Shipper - Transports items between suppliers and retailers (and if necessary, return them to sender). Also referred to variously as Shippers, Transporters, and Third Party Logistics Providers, since although there can be first or second party logistics providers employed by the retailer, this thesis aims to link Third Party Logistics Providers directly to retailers.
- Retailer - Purchases and receives products, for eventual sale to end customers. Although this process is how this thesis currently conceives of the purchaser, the retailer themselves could possibly be bypassed as the last middleman if customers become comfortable enough buying directly from suppliers.

6.3.2 Assets

- Product - A product to be sold by a supplier to a retailer.
- Contract - As defined in our implementation, a shipping smart contract between a supplier, shipper, and retailer to handoff, transport, and deliver the product for the retailer.
- Shipment - A derivative asset of the shipping contract, generated when the shipper moves to pick up the product from the supplier. There could possibly be multiple shipments to various different locations, return shipments, or cancelled and renegotiated shipments.

6.4 Process

In the process below there are **actors**, *assets/records*, ‘functions()‘, ‘variables‘, and ‘STATES‘ as defined in the Business Network code.

6.4.1 Product Marketplace

1. There is assumed to exist at least three types of business entities: **Suppliers** (farmers or food producers), **Shippers** (Logistics Providers), and **Retailers** (grocery stores, restaurants).

2. The **supplier** creates an inventory of *Products* with the `CREATED` status. The **supplier** then lists them for sale publicly with the `FOR_SALE` status, and sets their desired price floor `reservePrice`.
3. A **retailer** makes an price offer `offer()` to the supplier for their desired product.
4. If the **supplier** is satisfied with the *offer*, it can `closeBid()` to sell the *product* to the highest bidding *retailer*, entering the `SOLD` state.

    - An improvement that can be made in this process is to add counteroffers, as well as volume orders, to reduce the amount of negotiation needed by allowing suppliers to join a coalition in volume orders if requirements are met and the retailer's offer is above their minimum bid.

6.4.2 Shipping Inquiry

1. To begin the transportation negotiation process, the **retailer** publicly creates a `Contract` in the `INQUIRY` state using the `listContract()` function. The retailer sets a `maxPrice` that it will pay per unit for shipping. The contract also stipulates *temperature requirements*, *deadlines*, and *penalties* for tardiness or temperature mishaps.
2. Prospective shippers observe new *Contracts* and make a `Bid()` with a certain `unitPrice` to a promising shipping *Contract*.

    - An improvement that can be made here is to send out a notification of new *Contracts* to all prospective **shippers** meeting the requirements.

3. The **retailer** considers these bids and if they remain under their `maxPrice`, the **retailer** accepts the lowest bid using `closeBidding()`. When bidding closes, the *Contract* enters the `READY_FOR_PICKUP` state.

    - An improvement that can be made here is to do the shipping cost inquiries from available shippers alongside any product price inquiries. There could also be an `ACCEPTED` state in between when the retailer purchases and is ready for pickup, though the shipper must also require enough buffer time for the contract to actually be fulfilled.

6.4.3 Logistics: Shipping Ceremony

A **Ceremony** as explained below refers to the transaction process that must occur on the blockchain for the supplier, shipper and retailer to record a successful pickup and handoff of the item's possession, also known as a *Shipment*.

Proper conduct within this ceremony in terms of records, temperature, deadlines, and provenance results in payment. Misconduct, missed deadlines, or failure to report results in penalties or cancellations. The ceremony also follows a certain order whereby the next transaction cannot be made without successful resolution of the previous transaction.

1. When a *Contract* enters the `READY_FOR_PICKUP` state, the **shipper** generates a *Shipment* once it begins heading to the **supplier**.
2. During transport, the **shipper** reports the *temperature conditions* using any methods the retailer required in the contract description, which is recorded periodically by the **shipper** to the blockchain. The data source can either be manual records, offline sensors, or **retailer** certified IoT devices.
3. Upon arrival to the **supplier**, the **shipper** records that they are ready to load with the `DOCKED` state. When the **supplier** has finished loading the *product* on the **shipper's** vehicle and the **shipper** is on the move, the shipment enters the `IN_TRANSIT` state. The **supplier** and **shipper** both confirm this state from their account by scanning the product label to set the **shipper** as the `possessor` of the *product* on the blockchain.
4. Upon arrival to the **retailer**, the **shipper** records that they are ready to unload with the `ARRIVED` state. When the **retailer** has finished unloading the *product* from the **shipper's** vehicle, the shipment enters the `DELIVERED` state. The **retailer** and **shipper** both confirm this state by scanning the product to set the **retailer** as the new `owner` and `possessor` of the *product* on the blockchain. The optional `buyer` variable on the contract becomes null again.
5. When the *shipment* enters the `DELIVERED` state, the *contract* runs the `payOut()` function, which gauges the shipper's *temperature records* and *delivery times* to assess any penalties. The total price is `unitPrice` is multiplied by the amount of product units transported, and the balance is subtracted from the **retailer** and deposited to the **shipper**.

6.5 Product Bidding Transactions

Transactions on the blockchain relevant to the product bidding process.

6.5.1 `createProduct` Create a Product

Generate a `Product` in either the `CREATED` state (if not selling yet) or `FOR_SALE` state for retailers to bid on, with a reserve price.

```
{
  "$class": "org.acme.product.auction.Product",
  "productId": "PRD_001",
  "description": "Bananas from Panama.",
  "type": "BANANAS",
  "supplier": "resource:org.acme.shipping.perishable.Supplier#031farmer",
  "owner": "resource:org.acme.shipping.perishable.Supplier#031farmer",
  "possessor": "resource:org.acme.shipping.perishable.Supplier#031farmer",
  "unitCount": 5000,
  "state": "FOR_SALE",
  "reservePrice": 1000
}
```

6.5.2 `Offer`

If a Product is in the `FOR_SALE` state, the retailer posts an offer for it. The product is then updated to include this offer in its state.

```
{
  "$class": "org.acme.product.auction.Offer",
  "bidPrice": 1000,
  "listing": "resource:org.acme.product.auction.Product#PRD_001",
  "retailer": "resource:org.acme.shipping.perishable.Retailer#001market"
}
```

6.5.3 `CloseBidding`

Once the `closeBidding` Transaction is made by the `Supplier`, the `Retailer` with the highest `Offer` becomes the the product `buyer`, their highest `bidPrice` is used, and the `Product` enters the `SOLD` state, ready for a Contract to be made upon it at the retailer's leisure. The transaction then updates any modified assets in their registries.

> An improvement that could be made is to take product quality into account, as well as product and retailer location into account asynchronously by putting out an Inquiry, then deciding on inquiries to make a final Contract.

```
{
  "$class": "org.acme.product.auction.CloseBidding",
  "listing": "resource:org.acme.product.auction.Product#9381"
}
```

6.6 Contract Bidding Transactions

Transactions on the blockchain relevant to the contract bidding process.

6.6.1 `createContract` Create a Contract for a SOLD Product

Given a `Product` in the `SOLD` state, generate an open `Contract` in the `INQUIRY` state for shippers to bid on, with a reserve price.

```
{
  "$class": "org.acme.shipping.perishable.Contract",
  "contractId": "CON_001",
  "supplier": "resource:org.acme.shipping.perishable.Supplier#001farmer",
  "shipper": "resource:org.acme.shipping.perishable.Shipper#001shipper",
  "retailer": "resource:org.acme.shipping.perishable.Retailer#001market",
  "arrivalDateTime": "2018-04-02T03:49:09.707Z",
  "unitCount": 5000,
  "unitPrice": 0.5,
  "minTemperature": 2,
  "maxTemperature": 10,
  "minPenaltyFactor": 0.2,
  "maxPenaltyFactor": 0.1,
  "product": "resource:org.acme.product.auction.Product#PRD_001",
  "state": "INQUIRY",
  "maxPrice": 0.5
}
```

6.6.2 `Bid`

The Shipper makes a `Bid` for a `Contract` at a certain bidPrice.

```
{
  "$class": "org.acme.shipping.perishable.Bid",
  "bidPrice": 0.1, // shipper’s offer for unitPrice per unit transported
  "contract": "resource:org.acme.shipping.perishable.Contract#CON_001",
  "shipper": "resource:org.acme.shipping.perishable.Shipper#SHIP_001"
}
```

6.6.3 `CloseInquiry`: Execute the Contract

Once the `closeInquiry` Transaction is made by the `Retailer`, the `Shipper` with the lowest Bid is assigned to the contract, their lowest `bidPrice` is used, and the `Contract` enters the `READY_FOR_PICKUP` state. The transaction then updates any modified assets in their registries.

> An improvement that could be made is to somehow take product and retailer location into account in the contract.

```
{
  "$class": "org.acme.shipping.perishable.CloseInquiry",
  "inquiry": "resource:org.acme.shipping.perishable.Contract#CON_001"
}
```

6.7 Shipment Transactions

Transactions on the blockchain related to the shipment process.

6.7.1 `CreateShipment` Begin a Shipment for a Contract that is `READY_FOR_PICKUP`

Given a `Contract` in the `READY_FOR_PICKUP` state, generate an open `Shipment` in the `CREATED` state to notify all participants that the shipper is moving to pick up the product.

```
{
  "$class": "org.acme.shipping.perishable.Shipment",
  "shipmentId": "SHIP_001",
  "product": "resource:org.acme.product.auction.Product#PRD_001",
  "status": "CREATED",
  "contract": "resource:org.acme.shipping.perishable.Contract#CON_001"
}
```

6.7.2 `Dock`: Shipper Informs Possessor of Arrival

The shipper docks at the possessor (most likely the supplier or their warehouse), which emits a notification to the supplier and retailer.

```
{
  "$class": "org.acme.shipping.perishable.Dock",
  "shipment": "resource:org.acme.shipping.perishable.Shipment#SHIP_001"
}
```

While in the `DOCKED` state, the shipper waits for the possessor to make them the new possessor of the product using the `HandOff` transaction.

6.7.3 `HandOff`: Change Product Possession to Shipper

If the Shipper has a `DOCKED` shipment, product possessor hands off the product to the recipient by changing the `Product`'s `possessor` to the recipient. This is the equivalent of a digital signature confirming their handoff.

> An improvement that could be made for the product's label to be scanned by the possessor to confirm their conscious approval of this handoff.

```
{
  "$class": "org.acme.shipping.perishable.HandOff",
  "shipment": "resource:org.acme.shipping.perishable.Shipment#SHIP_001"
}
```

6.7.4 `PickUp`: Confirm Product Possession and Move Out

Given that the shipment status is `DOCKED`, and the product was transferred by the possessor to the shipper, the shipment status changes to `IN_TRANSIT`.

```
{
  "$class": "org.acme.shipping.perishable.PickUp",
  "shipment": "resource:org.acme.shipping.perishable.Shipment#SHIP_001"
}
```

6.7.5 `Arrive`: Record Shipper Arrival at Retailer

The shipper records that they have arrived and the time as provided by the transaction, which notifies the Retailer to take receipt of the shipment.

```
{
  "$class": "org.acme.shipping.perishable.Arrive",
  "shipment": "resource:org.acme.shipping.perishable.Shipment#SHIP_001"
}
```

6.7.6 `TemperatureReading`: Record Temperature Reading (Shipper)

The `TemperatureReading` allows shippers to record the temperature of the shipment on the `Shipment` as it goes forward:

```
{
  "$class": "org.acme.shipping.perishable.TemperatureReading",
  "centigrade": 8,
  "shipment": "resource:org.acme.shipping.perishable.Shipment#SHIP_001"
}
```

The retailer will provide a penalty if any temperature reading is outside of minimums and maximums as specified by the contract.

> An improvement that can be made is to mandate minimum reporting requirements, or have the retailer certify the reporting hardware used. The possessor could also do a temperatureReading here prior to the shipper taking custody of the product.

6.7.7 `ShipmentReceived`: Record Shipment Reciept (Retailer)

The retailer confirms their pickup of the product by sending a `ShipmentReceived` transaction on the `Shipment` to begin the payout process based on stipulations in the `Contract`.

If the shipment is late, whereby both the shipper's own recorded arrival time `Shipment.arrival` and the retailer's pickup time (of the `ShipmentRecieved` transaction) is after the deadline `Contract.arrivalDateTime`, the shipper will not be paid.

> An improvement that can be made is to require the shipper and the retailer to physically and consciously verify that they have met, perhaps through scanning each other's QR Code or simply tapping NFC smartphones together.

```
{
  "$class": "org.acme.shipping.perishable.ShipmentReceived",
  "shipment": "resource:org.acme.shipping.perishable.Shipment#SHIP_001"
}
```

6.8 Access Control Rules

Access Control Rules were not yet implemented here due to time constraints, as in this implementation for debugging purposes, a network owner has access to arbitrarily change all assets.

However, a permissioned blockchain can be implemented using Access Control Rules to enforce object-oriented programming style encapsulation, where changes can only be made through public methods and not arbitrarily by unauthorized means or participants.

Useful access control settings could be utilized as below. See the Hyperledger Access Control Language documentation for further information (Hyperledger Foundation 2018[b]).

The following Access Control Rule allows all businesses (suppliers, shippers, retailers) complete read access to all shipping info.

```
rule BusinessRead {
    description: "Allow a business read access to all shipping info"
    participant: "org.acme.vehicle.auction.Business"
    operation: READ
    resource: "org.acme.shipping.perishable.*"
    action: ALLOW
}
```

The following Access Control Rule only allows the Product's owner to make any change to their product.

```
rule ProductOwner {
    description: "Allow the owner total access to the product,"
    participant(m): "org.acme.shipping.perishable.Business"
    operation: ALL
    resource(v): "org.acme.product.auction.Product"
    condition: (v.vehicle.owner.getIdentifier() == m.getIdentifier())
    action: ALLOW
}
```

Chapter 7 Results and Conclusions

7.1 Case Study

Three case studies explored below provide possible applications of the implementation that was made. Due to time constraints, the true applicability and increased effectiveness of the blockchain could not be fully tested, but the possibilities are recognizable in these case studies.

A general vision in Food Traceability is provided as the first case study, where the blockchain could provide distributed interorganizational data availability digitally, compared to existing systems that trust only paper copies or centralized third-party technology providers.

Next, the applicability of this implementation to the startup eHarvestHub is explored, in its aim to both reduce food prices to end customers and increase profit margins for farmers and truckers by flattening the supply chain, cutting out the middlemen.

Finally, provenance can be conveyed as a selling point in the story of a luxury item's production. It is the evidence and justification to customers for the quality or uniqueness of a luxury brand, which counterfieters cannot provide without the ability to modify the blockchain.

7.1.1 Food Traceability

Before each food handling entity can join the supply chain, they must apply for a license and comply with regulatory frameworks such as the Produce Traceability Initiative. In our system, we consider the regulator to certify licensees, where licensees then get the right to deploy a Hyperledger node using docker to join the network. Employees of the licensees then use a webapp or mobile interface communicating over a REST API to their local node to interact with the blockchain.

The food product also has to be labeled with a machine readable barcode containing all relevant identifying data about the blood, which the supplier should generate and place onto the food pallet, or RFID temperature sensor. This provides an element for the following entities on the supply chain to view the product information from the label and additional provenance on the blockchain, record additional measurements on the blockchain, and prevent counterfeiting of provenance.

Next, if the temperature of the food is detected by a reporter to be below legal levels or if the product is past its expiration date, the logistics provider would suffer a penalty as agreed upon in the smart contract. If the product is at a critical temperature, it must not continue onwards to the patient in the supply chain.

With this blockchain system, when an employee in a warehouse in the supply chain receives the likely spoiled food, the scan would register its expired state. As such, the food cannot continue onwards for delivery to the patient since a licensed transporter would not be able to scan it in, and even if the food somehow reached the retailer anyway, upon a scan by the expired state would still be revealed. This makes the use of tamper-evident seals or RFID crucial, as is already prevalent.

The logistics of moving perishable items by a certain time and minimal cost do also need to be considered. The retailer would have to weigh the estimated costs of transport from the supplier to their location with the cost of the item.

7.1.2 eHarvestHub: A Marketplace to connect Grocers, Truckers, Farmers Directly

eHarvestHub is a key collaborator with the work of this thesis, as they have existing customers and practical experience with the supply chain. A blockchain application for the procurement problem and other aspects is directly applicable to their visions.

eHarvestHub's proposed app ecosystem provides a sharing economy for farmers to "list their fresh produce and aggregate volume across multiple small farmers at profits up to 70%" using the eHarvestFarm app (Ramirez 2017). This allows farmers to collectively fill large volume orders from grocers shopping in the eHarvestHub app, who would thus enjoy “cost savings up to 17%” (Ramirez 2017). Connecting the farmer directly to the grocer, cutting out the middleman, allows the farmer to gain much of the markup that would have otherwise been skimmed off by middlemen, while grocers and their customers can enjoy lowered prices (Ramirez 2017).

Farmers or grocers could also directly hail independent truckers the moment produce is ready using the eHarvestHaul app, akin to Uber, so truckers can truly function as contractors on their own terms, without the need for brokers.

On eHarvestFarm, where our implementation’s **Inventory** system is applicable, farmers could meet traceability requirements much more easily through the use of Blockchain timestamping at each change of custody, as obligated by the US Product Traceability Initiative. An inventory management system is provided to reduce overselling or underselling, and monitors of quality control to provide provenance and faster traceability of issues.

On eHarvestHub, where our implementation’s **Marketplace** system is applicable, the grocer can either accept a product at face value, choose to bargain directly with one farmer, or make a large bulk order through a smart contract that numerous small farmers could fill (much like an exchange) if the price is above their minimum bid.

On eHarvestHaul, where our implementation’s **Logistics** system is applicable, truckers can directly market their services to farmers or grocers and react to immediate market needs. They can then sign smart contracts which provide immediate payment transfer upon successful delivery. The app allows for accurate forecasting of delivery times, faster response to farmer/grocer demand for service, and increased profit for truckers. Currently, brokers provide truckers with unstable contracts, little opportunity for double loads or backhaul, and significant downtime during artificial lulls in demand. Instead, truckers can have more opportunities to double load, increase capacity for the supply chain at lower costs, or choose to take a break on their own terms much like Uber.

7.1.2.1 Future Expansions for eHarvestHub

One justification eHarvestHub gave for using the blockchain for the supply chain in an interview was for the feature of supply chain finance. The key feature is immediate and low fee payment upon the satisfaction of conditions (such as temperature, deadlines, all trackable in this implementation). An industry specific credit rating could emerge based on how reliable the trucker or farmer was with their transactions and procurement, that could allow eHarvestHub or other participating entities to provide small loans.

In this case, a cryptotoken could be used, and that is probably another application where 1 cryptotoken represents \$1 stored in a bank account, especially to avoid the effects of speculative volatility on the users themselves. eHarvestHub could take the role of using these tethered cryptotokens akin to credits that could be cashed out and deposited at market rate anytime, so that the moment the trucker completes their contract, they can go to the app and cash out without having to stay on the hook with the broker for 30 days or more.

7.1.3 Value-added Products: A Story Told by the Blockchain

Value-added products could make such provenance a selling point in the story of its production.

With the wealth of information on the internet, there's a good quote about marketing that matches (Lutzenburg 2017):

"Marketing used to be about creating a myth and selling it; now it's about finding a truth and sharing it." - Marc Mathieu, Unilever (Lutzenburg 2017)

The blockchain that powers cryptocurrencies like Bitcoin are able to bring customers the story of their product. It could be safety related as in Wabi, that infant formula is certified not to be counterfeit or melamine tainted.

In Barclay's Everledger, it aims to make sure diamonds have a heartwarming story of helping a miner make a living for his family in Kimberly, and not a horrid story of conflict and slave labor, thereby disincentivizing the latter.

Another would be with Champagne, as only certain regions in France are allowed to name their wines with this brand, but counterfeiting is common. They aim to use the blockchain to provide drinkers the story of every farmer, grape and wine barrel that made the genuine wine they paid good money for.

At the end of the day, counterfeit Champagne can be difficult to detect, no matter the method of production. Champagne houses were notoriously bad at record keeping, at least before World War II, and were inconsistent in the bottles, labels and corks they used (Jenks 2018). Also, it can be difficult for a casual drinker to say what a correct bottle is supposed to taste like, so counterfeit wine slips through where there is a lack of experienced tasters (Jenks 2018).

There are four approaches that were identified by Tyler Jenks from the startup Very (Jenks 2018).

The existing strategies are to lobby importers to stop counterfeiting and put manpower and lawyers towards the enforcement of the brand (Jenks 2018). A passive strategy is to use certificates of authenticity, special labels and seals that are then government certified through the use of paper regulation (Jenks 2018).

Thus, two alternative strategies for applying the blockchain emerge as identified by Jenks. Blockchain provenance to enhance authenticity tokens at every stage of the supply chain, from the farm to the warehouses to overseas and to the retailers (Jenks 2018). This thesis's implementation can be directly applied for this purpose, since this process would normally be accomplished through the use of paper signatures or electronic communication enforced by a government regulator. One key issue is at each step of the the supply chain, the blockchain cannot stop counterfeiting beyond the seals on the bottle, much like how paper contracts require a regulatory bureaucracy to make them more than just words.

The other strategy is to enhance the data of the blockchain supply chain with IoT devices that track the materials used to make the grapes, and capture a full provenance that cannot easily be counterfeited and reject those that don't have it (Jenks 2018). This also provides special value to the customer, who may be much more willing to look for the complete story in every bottle and every grape of wine that went into it. Jenks provides a vision that Amazon Alexa could be used to narrate a cute story about family owned farms, quality grapes, the region in particular (Jenks 2018).

GPS devices could put on pigs searching for truffles, on wagyu cows throughout their lifetime, on ships searching for alaskan salmon that record how it got transported (Jenks 2018). There can be soil samples taken by inspectors and recorded to the chain, leather along the supply chain to the luxury bags: everledger to keep out blood diamonds, where the added value to the customer is the story and authenticity in the bag.

Faking this provenance becomes an order of magnitude more difficult as the ledger accumulates more data. This is the inverse of a problem with paperwork, whereby the more paper and communications collected the more difficult those records are to access.

7.2 Conclusions

With almost 70% of the world population expected to live in megacities by the year 2050, logistics is the lifeblood of megacities and retailers need to find efficient and less expensive logistics systems.

This thesis proposed a blockchain peer-to-peer smart contract system for a megacity grocery supply chain where retailers, suppliers, and logistics providers can bid directly for contracts, analyze market trends, trace product flow for food safety, and reduce costs by cutting layers. Although existing centralized systems work well within organizations that provide complete vertical integration in their supply chain, data transfer between incompatible systems tends to be conducted through paperwork or spreadsheets, which is subject to inefficiencies and nearsightedness.

The main question explored is, *"Can we use the blockchain to create a sharing economy for supply chain logistics and contract negotiation, linking retailers, logistics providers, and suppliers directly?"* A sharing economy is found in transformative albeit centralized IT examples such as Airbnb and Uber.

The implementation explored in this thesis implements both the negotiation and logistics components of the methodology. Due to time limitations, not all aspects of the methodology described previously could be fully explored in this blockchain implementation, such as volume orders, inventory management, logistics provider tracking, or physical IoT devices.

This thesis simply aims to provide a basic code structure using Hyperledger, to show that it is possible for blockchain smart contracts to be applied to the procurement problem. It is left to future researchers and developers to explore further possibilities, test results, or solutions.

The decentralization inherent in the blockchain can allow smaller, independent contractors to be more competitive, increase efficiency by reducing the layers of middlemen, and reduce the costs for them to adopt technology and get access to information (Ramirez 2017). Smaller suppliers can be aggregated and matched to retailers through volume orders without a middleman (Ramirez 2017). Logistics providers could be crowdsourced individual contractors that could be called upon anytime for small or rush deliveries, much like how rides can already be hailed on sharing economies such as Uber and Lyft. Megacities that have to cope with reduced warehouse space, greater demand for direct/fresh delivery will need improvements like these to sustain their growth (Polim, Hu, and Kumara 2017).

7.3 Future Studies

7.3.1 Possible Expansions of this Implementation

7.3.1.1 eHarvestHub Collaboration

The author is likely to continue to develop this system as a solution and mobile application with Alvaro Ramirez's eHarvestHub. With their customer base, it would be possible to conduct experiments in market studies with actual participants and markets, and also grow a business from the grassroots.

7.3.1.2 Broadcast a Shopping List with IoT Fridges

A possible concept is that an IoT smart fridge could broadcast a set of orders as items go below a certain threshold, such as milk or eggs. It can also inform users when food is going bad or inform of any recalls. This allows product demand to be forecast better by suppliers for just-in-time manufacturing that only manufactures and sends out products when customers show a clear need for them.

7.3.1.3 The Blockchain as a Data Source

The blockchain provides an unprecedented source of large data that is easily accessible to all, public, and of high transaction integrity. This makes data mining to obtain useful information and knowledge about the ticketing system feasible.

Examples include the turnover rate of resale, real time population flow through checkpoints, and to help determine the ideal market price for the tickets, as well as many more possibilities.

The blockchain also functions as a useful economic model for simulation of traffic in itself, a unique case where the model is also the literal implementation of the market. Thus, the code can be stress tested, and generate enough data to test these data mining techniques. In this research proposal, the data generated from the performance testing provided in the previous section will unlock the ability to attempt this.

Thus, a blockchain economy could be simulated by using a simulated blockchain to gather a large amount of easily mined and valueless digital assets, generate a large amount of wallets for each actor, and simulate each actor in parallel to consider whether to buy or sell in this model economy.

Once the economy has been simulated, perhaps even multiple times or with different setups (such as larger demand of interested customers against the fixed supply), the ideal market price for each type of projected demand on digital assets can be calculated. An AI algorithm could possibly be put together learning from this data.

7.3.2 The Unified Theory of Acceptance and Use of Technology

Francisco and Swanson brings in the Unified Theory of Acceptance and Use of Technology (UTAUT) justification and framework to be used for judging likelihood of adoption, particularly in blockchain traceability applications. The authors conclude that the UTAUT framework can be used "as a lens to understand the adoption of blockchain in the supply chain", as it provides a "robust conceptual framework to explain these relationships", of motivators and barriers could encourage or discourage companies to adopt blockchain technologies for this purpose (Francisco and Swanson 2018, p. 11). It assumes that the main driver of adoption is the practical usefulness of the system, though it does not necessarily provide steps to increase its adoption (Francisco and Swanson 2018, p. 11).

Six major factors in the UTAUT are paired to independent and dependent variables, where the following conclusions are drawn by the authors:

1. "Performance expectancy positively impacts the behavioral intention of using blockchain technology for supply chain traceability" (Francisco and Swanson 2018, p. 9).
    - "**Performance expectancy** is the degree the usage of a new technology can provide consumers the expected benefits for performing specific activities" (Francisco and Swanson 2018, p. 6).
    - "The blockchain offers a solution for a trusted single-source of distributed information with improved information accuracy and efficiencies" (Francisco and Swanson 2018, p. 6).

2. “Effort expectancy positively impacts the behavioral intention of using blockchain technology for supply chain traceability” (Francisco and Swanson 2018, p. 9).

   - “**Effort expectancy** refers to a technology’s ease of use. Individuals are less likely to use technology if it is sensed to be more difficult to use and require effort than existing methods” (Francisco and Swanson 2018, p. 6).
   - “Information Sharing Methodologies such as vendor managed inventory (VMI) create(s) efficient replenishment models without the need for traditional orders” (Francisco and Swanson 2018, p. 6).
   - The “blockchain enables the use of “smart contracts” that are based on user defined rules requiring little to no human intervention” (Francisco and Swanson 2018, p. 6).

3. “Social influence positively impacts behavioral intention to use blockchain technology for supply chain traceability” (Francisco and Swanson 2018, p. 9).

   - “Social influence is defined as the extent an individual perceives important others believe the new system should be used.” (Francisco and Swanson 2018, p. 6)
   - Although blockchain are social technologies by design, it needs to reach “a critical mass of users to realize increased network effects”, which could lead to “higher intentions to use” (Francisco and Swanson 2018, p. 6).

4. “Facilitating conditions (i.e., technical resources and organizational support) positively impact behavioral intention to use blockchain technology for supply chain traceability” (Francisco and Swanson 2018, p. 9).

   - Since the blockchain is highly networked and demands participation to be part of that society, good coordination and the availability of technical resources and intuitive tools will increase the likelihood of adoption. A lack of these resources will reduce it. (Francisco and Swanson 2018, p. 6)

5. “Behavioral intention will positively influence the use of blockchain traceability applications” (Francisco and Swanson 2018, p. 9).

- Blockchains are social technologies powered by Cryptoeconomics, where if incentive is provided in the form of cryptotokens to take a certain action (such as secure the network with Proof-of-work, make a retweet or facebook like, wear hats in game), users are not only likely to join in, they would spread the word.

6. "Trust in technology positively moderates the relationship between Performance Expectancy and Behavioral Intention" (Francisco and Swanson 2018, p. 9).
7. "Trust in technology positively moderates the relationship between Effort Expectancy and Behavioral Intention" (Francisco and Swanson 2018, p. 9).
8. "Trust in technology positively moderates the relationship between Facilitating Conditions and Behavioral Intention" (Francisco and Swanson 2018, p. 9).

    - Although the blockchain does provide a strong cryptosystem, if the users do not at least marginally see and understand how it works getting them to trust it can be hard. On the other hand, the more experience users have with it and the more successful case studies, the more likely that the technology will appear trustworthy (Francisco and Swanson 2018, p. 9).
    - Even then the devil is in the details. With HTTPS/SSL, incorrect time allows bad certificates to become good again, and its expense of certification has led to the perseverance of unencrypted HTTP.
    - GPG public/private keys offer end-to-end encrypted email, but its difficulty and inflexibility makes adoption rare outside of large corporations that provide a significant amount of support to users.

9. "Inter-organizational Trust positively moderates the relationship between Performance Expectancy and Behavioral Intention" (Francisco and Swanson 2018, p. 9).
10. "Inter-organizational Trust positively moderates the relationship between Effort Expectancy and Behavioral Intention" (Francisco and Swanson 2018, p. 9).
11. "Inter-organizational Trust positively moderates the relationship between Social Influence and Behavioral Intention" (Francisco and Swanson 2018, p. 9).

12. “Inter-organizational Trust positively moderates the relationship between Facilitating Conditions and Behavioral Intention” (Francisco and Swanson 2018, p. 9).

    - Especially in supply chains, inter-organizational trust is particularly crucial as all entities are interdependent, and delays or errors in one can negatively affect everyone down the line. There has to be a way for the culprit to be held accountable and the innocent to be unaffected.

7.3.3 Ontology Driven Supply Chain Provenance

Ontologies provide a formal definition of how entities and objects are related to each other, how axioms are defined, aiding in knowledge transfer and understanding across different software implementations. In terms of traceability, it can be immensely beneficial to define an ontology for doing tracebacks regarding what processes and items were consumed to make up the final product.

The basic principles of the Traceability Ontology can be drawn upon to conduct a more structured provenance trace of products in our implementation (Kim and Laskowski 2016).

The paper Ontology Driven Supply Chain Performance introduces, analyzes, and implements the TOVE traceability ontology in the Ethereum blockchain as a Solidity-language smart contract, for the purpose of improving customer knowledge about the provenance, traceability, and authenticity of products in supply chains. This blockchain implementation is made possible by the emergence of IoT sensors that provide key information about items at every point in the supply chain, and the blockchain which is a distributed, shared interorganizational ledger that provides trustworthiness, nonrepudiation, and transparency (Kim and Laskowski 2016).

The authors explain that in existing supply chains, although some centralized systems such as shipment tracking have managed to provide accurate provenance to form a digital footprint accessible to customers, these open tracking systems are the exception rather than the rule, as they generally limited to a single corporation’s logistics and does not coordinate across the entire supply chain. The result is that in the supply chain, information silos form between each corporation that foster information asymmetry, which becomes a liability rather than an asset once the retailer and its customers demand trackable provenance to fulfill quality, sustainability, and authenticity requirements (Kim and Laskowski 2016).

The authors introduce "ontology-based enterprise modeling", which provides formal specifications for data to ensure common interpretation across shared databases in organizations. This would represent the state of the supply chain, but would assist with enabling automated inference and verification. The features of ontology-based enterprise modeling is especially pertinent in a blockchain-enabled supply chain, since "a modeling approach based on formal ontologies can aid in the development of smart contracts that execute on the blockchain" (Kim and Laskowski 2016).

A specific model for ontology-based enterprise modeling, the TOVE Ontology is introduced, specifically adapted to ensure food safety through the food supply chain. In the figures below, a simple vision of the ontology is depicted.

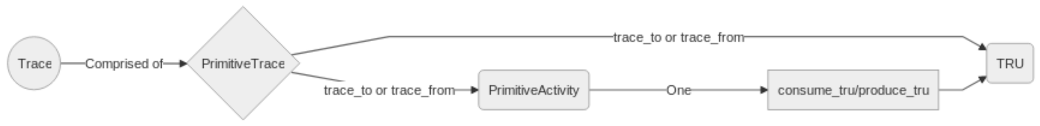


Figure 7.1: The TOVE Ontology in simple form. (Kim and Laskowski 2016)

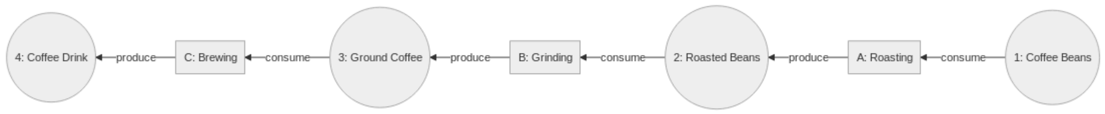


Figure 7.2: An example of a TRU trace in a Coffee Supply Chain. (Kim and Laskowski 2016)

The TOVE Ontology as expressed in Ethereum Solidity-language smart contracts by these authors is a simple but effective ontology for representing the state of the supply chain and an item's provenance through various processes done by various participants. Ontologies also promote interoperability, since PrimitiveActivities can be made that function at higher levels on TRUs, such as shipping that can transport any TRU within limits. The ability for the blockchain to tokenize and coordinate the supply chain utilizing tried and true ontologies is what future implementations should draw upon (Kim and Laskowski 2016).

Appendix A Implementation Model Definitions

The digital assets, participants, and transactions defined in the implementation of this thesis. Source code can be found at `https://github.com/lvw5264/hyperledger-megacity`.

A.1 Product Negotiation Assets

```
namespace org.acme.product.auction
import org.acme.shipping.perishable.*

/**
 * The type of perishable product being shipped
 */
enum ProductType {
  o BANANAS
  o APPLES
  o PEARS
  o PEACHES
  o COFFEE
}

/* Business Logic inherited from Car Auction */
```

```
asset Product identified by productId {
  o String productId
  o String description
  o ProductType type
  --> Supplier supplier
  --> Business owner
  --> Business possessor

  // auction data, merged from productListing
  o Long unitCount
  o ListingState state
  o Double reservePrice
  --> Retailer buyer optional
  o Offer[] offers optional
}

enum ListingState {
  o CREATED // created but not yet up for sale
  o FOR_SALE // supplier offers the product up for sale
  o RESERVE_NOT_MET
  o SOLD // sold products are ready to have shipping contracts made
}

// an offer for the product from a retailer for the
// supplier to consider
transaction Offer {
  o Double bidPrice
  --> Product listing
  --> Retailer retailer
}

// close bidding in a final transaction, with the closebidding function
// determining the outcome
// This also should make it available for shipping negotiation
```

```
transaction CloseBidding {
  --> Product listing
}
```

A.2 Perishable Assets

```
namespace org.acme.shipping.perishable
import org.acme.product.auction.Product
/* Business Logic inherited from Perishable */

/**
 * The status of a shipment
 */
enum ShipmentStatus {
  o CREATED
  o DOCKED
  o IN_TRANSIT
  o DELIVERED
}

/**
 * The status of a contract
 */
enum ContractState {
  o INQUIRY // generated shipping inquiry
  o RESERVE_NOT_MET // offers did not meet maxBid
  o READY_FOR_PICKUP // bids closed and ready to purhcase
  o SHIPMENT_CREATED // update from shipping transactions
  o SHIPMENT_PICKED_UP
  o SHIPMENT_IN_TRANSIT
  o SHIPMENT_ARRIVED
}

/**
```

```
 * An abstract transaction that is related to a Shipment
 */
abstract transaction ShipmentTransaction {
    --> Shipment shipment
}

/**
 * An temperature reading for a shipment. E.g. received from a
 * device within a temperature controlled shipping container
 */
transaction TemperatureReading extends ShipmentTransaction {
  o Double centigrade
}

/**
 * The shipper docks at the possessor, which emits a notification
 * The shipper waits for the possessor to make them the new
 * possessor of the product
 */
transaction Dock extends ShipmentTransaction {
}

/**
 * Recipient picks up from possessor, with their signature equivalent
 * being a change in state to IN_TRANSIT
 */
transaction PickUp extends ShipmentTransaction {
}

/**
 * Possessor hands off to recipient, with signature equivalent
 * being change of possessor to the recipient.
 */
transaction HandOff extends ShipmentTransaction {
}

/**
 * The shipper arrives at the retailer, which emits a notification
 * The shipper waits for the retailer to take possession of the product
 */
transaction Arrive extends ShipmentTransaction {
}

/**
 * A notification that a shipment has been received by the
 * retailer and that funds should be transferred from the retailer
 * to the supplier to pay for the shipment.
 */
transaction ShipmentReceived extends ShipmentTransaction {
}


/**
 * A shipment being tracked as an asset on the ledger
 */
asset Shipment identified by shipmentId {
  o String shipmentId
  --> Product product
  o ShipmentStatus status
  // optional field. if not provided, product.unitCount will be used.
  o Long unitCount optional
  o TemperatureReading[] temperatureReadings optional
  // time of shipment arrival by the shipper.
  // technically required in ARRIVAL state.
  o DateTime arrival optional
  --> Contract contract
}
```

```
/**
 * Defines a contract between a Supplier and an Retailer to ship using
 * a Shipper, paying a set unit price. The unit price is multiplied by
 * a penality factor proportional to the deviation from the min and max
 * negociated temperatures for the shipment.
 */
asset Contract identified by contractId {
  o String contractId
  --> Supplier supplier
  --> Shipper shipper optional
  --> Retailer retailer // the buyer
  o DateTime arrivalDateTime
  o Long unitCount // unitCount is set from Product
  o Double unitPrice
  o Double minTemperature
  o Double maxTemperature
  o Double minPenaltyFactor
  o Double maxPenaltyFactor

  --> Product product optional
  // auction data, merged from car auction
  o ContractState state
  o Double maxPrice // maximum price the retailer will accept
  --> Retailer buyer optional
  o Bid[] bids optional
}

// supplier makes a lower shipping bid to the retailer to consider.
transaction Bid {
  o Double bidPrice
  --> Contract contract
  --> Shipper shipper
}
```

```
// create a contract given a product
transaction createContract {
  o String iter // for debugging purposes, ID padding
  o DateTime arrivalDateTime
  o Double maxPrice // sets initial unit price
  o Double minTemperature
  o Double maxTemperature
  o Double minPenaltyFactor
  o Double maxPenaltyFactor
  --> Product product
}

// close a contract to bids given offers
transaction CloseInquiry {
  --> Contract inquiry
}

// create a shipment
transaction CreateShipment {
  --> Contract contract
  o Long unitCount
}

/**
 * A concept for a simple street address
 */
concept Address {
  o String city optional
  o String country
  o String street optional
  o String zip optional
}

/**
```

```
 * An abstract participant type in this business network
 */
abstract participant Business identified by email {
  o String email
  o Address address
  o Double balance
}

/**
 * A Supplier is a type of participant in the network
 */
participant Supplier extends Business {
  // in auctions, supplier is an auctioneer
}

/**
 * A Shipper is a type of participant in the network
 */
participant Shipper extends Business {
  // in shipping service auctions, shipper is an auctioneer
}

/**
 * An Retailer is a type of participant in the network
 */
participant Retailer extends Business {
  // in auctions, retailer is a buyer
}
```

Appendix B
Additional Online Sources

These additional sources are not explicitly cited, but provide valuable background knowledge.

B.1 Videos Watched

B.1.1 Bitcoin

- BBC News. Bitcoin explained: How do cryptocurrencies work? Feb 12, 2018. https://www.youtube.com/watch?v=SzAuB2FG79A
- Squirrel Monkey. If bitcoins were around in the 90s. Mar 21, 2014. https://www.youtube.com/watch?v=ZW-oUXqomT0

B.1.2 Ethereum

- Techcrunch. Decentralizing Everything with Ethereum's Vitalik Buterin | Disrupt SF 2017. Sep 18, 2017. https://www.youtube.com/watch?v=WSN5BaCzsbo
- Ethereum Foundation. DEVCON1: Understanding the Ethereum Blockchain Protocol - Vitalik Buterin. Jan 4, 2016. https://www.youtube.com/watch?v=gjwr-7PgpN8
- Ethereum Foundation. DEVCON1: Ethereum for Dummies - Dr. Gavin Wood. Dec 11, 2015. https://www.youtube.com/watch?v=U_LK0t_qaPo
- Ethereum Foundation. Devcon2: Ethereum in 25 Minutes. Oct 10, 2016. https://www.youtube.com/watch?v=66SaEDzlmP4

B.1.3 Hyperledger

- IBM Blockchain. IBM and Maersk demo: Cross-border supply chain solution on blockchain. Mar 15, 2017. https://www.youtube.com/watch?v=tdhpYQCWnCw
- IBM Blockchain. Walmart's Food Safety Solution built on the IBM Blockchain. Aug 22, 2017. https://www.youtube.com/watch?v=SV0KXBxSoio
- IBM Watson Internet of Things. Genius of Things: Blockchain and Food Safety with IBM and Walmart. Feb 16, 2017 https://www.youtube.com/watch?v=MMOF0G_2H0A
- IBM Watson Internet of Things. Genius of Things: golden state foods transforms the supply chain with IoT and blockchain. Dec 14, 2017. https://youtu.be/hRDgZSUaW9s

B.1.4 Blockchain Overview Videos

- Anders Brownsworth. Blockchain 101: A Visual Demo. Nov 5, 2016. https://www.youtube.com/watch?v=_160oMzblY8
- Ethereum Foundation. Programmable Incentives: Intro to Cryptoeconomics. Nov 26, 2017. https://www.youtube.com/watch?v=-alrVUv6E24

B.1.5 Development

- Ethereum Foundation. Intro to Solidity 2017 Edition. Nov 26, 2017. https://www.youtube.com/watch?v=KkN1O8TChbM
- Ethereum Foundation. Dapp Development using Remix, Mist, and Geth. Nov 26, 2017. https://www.youtube.com/watch?v=HE4oREd1y20
- Ethereum Foundation. MetaMask: Dissecting the fox. Nov 26, 2017. https://www.youtube.com/watch?v=P6Bosg9QKnw
- Ethereum Foundation. Real-World Smart Contract Development Lessons. Nov 26, 2017. https://www.youtube.com/watch?v=pGh7UAiKTGo
- Ethereum Foundation. Web3.js 1.0. Nov 26, 2017. https://www.youtube.com/watch?v=92pdrRH_VGA
- Decypher Media. Ultimate Introduction to Ethereum DApp Development. Nov 17, 2017. https://www.youtube.com/playlist?list=PLV1JDFUtrXpFh85G-Ddyy2kLSafaB9biQ

- EIP Breakdown: The ERC 223 Token Standard. Jul 9, 2017.
  https://www.youtube.com/watch?v=GS62VNyPVHs

B.2 Organizations

- Ethereum Foundation. Ethereum. Retrieved February 23, 2018.
  https://ethereum.org
- IBM Blockchain. Retrieved February 23, 2018.
  https://www.ibm.com/blockchain/

B.3 Foundational Papers

Landmark papers regarding the blockchain and cryptocurrency.

- Nakamoto Institute. Literature. Retrieved February 23, 2018.
  http://nakamotoinstitute.org/literature/
- Szabo, Nick. Formalizing and Securing Relationships on Public Networks. 1997. http://nakamotoinstitute.org/formalizing-securing-relationships/
- Szabo, Nick. Smart Contracts Glossary. 1995. http://nakamotoinstitute.org/smart-contracts-glossary/
- BigChainDB

B.4 Implementations

Initial Coin Offerings or corporate implementations of the blockchain relevant to us.

- Slock.it Blockchain managed IoT locks
- Hyperledger Sawtooth Seafood Case Study
  https://www.hyperledger.org/projects/sawtooth/seafood-case-study
- https://blog.propy.com/technical-overview-the-first-real-estate-deal-on-the-blockchain-18a34979403
- Blockchain Design Patterns: https://github.com/kuip/dlt-design-patterns

B.5 Overview Documents

Articles that summarize foundational papers regarding the blockchain for a wider audience.

- https://spectrum.ieee.org/computing/software/bitcoin-the-cryptoanarchists-answer-to-cash
- https://www.bitcoin.com/guides/bitcoin-white-paper-beginner-guide
- Eliazar. Poster summary of Satoshi's original bitcoin whitepaper, on occasion of the 6th anniversary of its publication. Oct 31, 2014. https://redd.it/2kxwrk
- Anders Brownsworth. Blockchain 101: A Visual Demo. Nov 5, 2016. https://anders.com/blockchain/
- https://medium.com/crypto-currently/lets-build-the-tiniest-blockchain-e70965a248b
- https://bitcoinmagazine.com/articles/smart-contracts-described-by-nick-szabo-years-ago-now-becoming-reality-1461693751/
- https://medium.com/@FolusoOgunlana/cracking-the-ethereum-white-paper-e0e60c44126
- The Economist Staff. 2015. "The Great Chain of Being Sure about Things," The Economist.
- http://usblogs.pwc.com/emerging-technology/understanding-the-ico-infographic/
- https://medium.com/crypto-currently/the-anatomy-of-erc721-e9db77abfc24
- Best of ICOs. Eharvesthub Review. Nov 20, 2017. https://hackernoon.com/eharvesthub-review-61dd04ba8cc4
- https://blog.bigchaindb.com/how-automakers-can-use-blockchain-adab79a6505f
- https://unblock.net/nick-szabo/
- Institute for the Future. Bitcoin is the Sewer Rat of Currencies. https://medium.com/institute-for-the-future/bitcoin-is-the-sewer-rat-of-currencies-b89819cdf036

Note: Andreas Andropoulos takes a dim view on permissioned "intranet" blockchains. However, even so much of our technological infrastructure does use intranets in some form.

B.6 Software

- JP Morgan Chase. Quorum. Retrieved February 23, 2018. https://github.com/jpmorganchase/quorum
- LINUX FOUNDATION, 2017: Hyperledger [Online]. Available: https://www.hyperledger.org/ [Accessed 14 March 2017].
- Consensys. EthOn. Retrieved February 23, 2018. https://github.com/ConsenSys/EthOn
- https://media.consensys.net/ethon-introducing-semantic-ethereum-15f1f0696986
- https://solidity.readthedocs.io/en/develop/
- http://nomnoml.com/

B.7 Tutorials

- Solidity Tutorial
- https://learnxinyminutes.com/docs/solidity/
- https://monax.io/docs/solidity/solidity_1_the_five_types_model/
- http://truffleframework.com/tutorials/
- http://truffleframework.com/boxes/
- http://truffleframework.com/tutorials/robust-smart-contracts-with-openzeppelin
- http://truffleframework.com/tutorials/debugging-a-smart-contract
- https://consensys.github.io/smart-contract-best-practices/recommendations/
- http://truffleframework.com/tutorials/building-dapps-for-quorum-private-enterprise-blockchains
- https://sawtooth.hyperledger.org

B.8 Code Repositories

- Ethereum Wiki. ERC20 Token Standard. Retrieved February 23, 2018. https://theethereum.wiki/w/index.php/ERC20_Token_Standard
- https://github.com/ethereum/EIPs/issues/20
- ProfessorMarek. Ethereum application for traceability ontology. Retrieved February 23, 2018. https://github.com/professormarek/traceability
- http://truffleframework.com/tutorials/pet-shop

- https://github.com/truffle-box/metacoin-box
- https://github.com/jpmorganchase/quorum-examples

B.9 Courses

- https://www.coursera.org/learn/cryptocurrency
- Narayanan, Arvind. Bitcoin and Cryptocurrency Technologies. 2017. http://bitcoinbook.cs.princeton.edu/
- https://www.edx.org/course/blockchain-business-introduction-linuxfoundationx-lfs171x

Bibliography

Major Sources

Francisco, Kristoffer, and David Swanson. 2018. “The Supply Chain Has No Clothes: Technology Adoption of Blockchain for Supply Chain Transparency”. *Logistics* 2 (1): 2.

Kim, Henry M., and Marek Laskowski. 2016. “Towards an Ontology-Driven Blockchain Design for Supply Chain Provenance”.

Lutzenburg, Benedict. 2017. “Aiming for Supply Chain Transparency: Exploring the Potential of Blockchains” [**inlang**eng]. Student Paper. MA thesis, Lund University. `http://lup.lub.lu.se/student-papers/record/8927100`.

Olson, Kelly, et al. 2018. *Sawtooth Whitepaper*. Tech. rep. Linux Foundation. `https://www.hyperledger.org/wp-content/uploads/2018/01/Hyperledger_Sawtooth_WhitePaper.pdf`.

Polim, Rico, Qianyu Hu, and Soundar Kumara. 2017. “Blockchain in Megacity Logistics”. *IIE Annual Conference. Proceedings*: 1589–1594.

Ramirez, Alvaro. 2017. *eHarvestHub Whitepaper*. Tech. rep. eHarvestHub. `https://www.ehhico.com/whitepaper/en.pdf`.

Sun, Jianjun, Jiaqi Yan, and Kem Z. K. Zhang. 2016. “Blockchain-based sharing services: What blockchain technology can contribute to smart cities”. *Financial Innovation* 2 (1): 1–9.

Wu, Haoyan, et al. 2017. “A Distributed Ledger for Supply Chain Physical Distribution Visibility”. *Information* 8 (4): 137.

Foundational Works

Buterin, Vitalik, Gavin Wood, et al. 2018. “Ethereum Whitepaper”. `https://github.com/ethereum/wiki/wiki/White-Paper`.

Lamport, Leslie, Robert Shostak, and Marshall Pease. 1982. “The Byzantine Generals Problem”. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 4 (3): 382–401.

Nakamoto, Satoshi. 2008. “Bitcoin: A Peer-to-Peer Electronic Cash System”. Published at Bitcoin.org by the author.

Szabo, Nick. 1996. “Smart Contracts: Building Blocks for Digital Free Markets”. *Extropy* 16. `http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_2.html`.

— . 1997. “Formalizing and Securing Relationships on Public Networks”. Archived by the Nakamoto Foundation.

Vogelsteller, Fabian, Vitalik Buterin, et al. 2018. “ERC-20 Token Standard (Draft)”. `https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md`.

Implementations

Hyperledger. 2018. “Seafood Case Study in Supply Chain Traceability Using Blockchain Technology”. `https://www.hyperledger.org/projects/sawtooth/seafood-case-study`.

WFP Innovation. 2017. “Building Blocks”. `http://innovation.wfp.org/project/building-blocks`.

Documentation

Ethereum Foundation. 2017(a). “Create a Crowdsale”. `https://www.ethereum.org/crowdsale`.

— . 2017(b). *Solidity Documentation*. Ethereum Foundation. `http://solidity.readthedocs.io/en/develop/index.html`.

Foundation, Ethereum. 2017. “Create a Democracy Contract in Ethereum”. `https://www.ethereum.org/dao`.

Hyperledger Foundation. 2018(a). “Car Auction Network”. `https://github.com/hyperledger/composer-sample-networks/tree/master/packages/carauction-network/README.md`.

— . 2018(b). “Hyperledger Composer Documentation”. `https://hyperledger.github.io/composer/`.

— . 2018(c). “Hyperledger Composer Overview”. `https://www.hyperledger.org/wp-content/uploads/2017/05/Hyperledger-Composer-Overview.pdf`.

— . 2018(d). “Perishable Goods Network”. https://github.com/hyperledger/composer-sample-networks/blob/master/packages/perishable-network/README.md.

IBM. 2018(a). “Applying blockchain to customs declarations”. https://developer.ibm.com/code/patterns/implement-fda-food-supplier-verification-program-on-hyperledger-composer/.

— . 2018(b). “Applying blockchain to customs declarations (at Maersk)”. https://www.youtube.com/watch?v=LeKapqAQimk.

Jenks, Tyler. 2018. “DC Blockchain Summit Keynote: A Blockchain for Champagne?” https://www.youtube.com/watch?v=20-IjyuocW8.

Limited, Kuip. 2018. “Distributed Ledger Technology Design Patterns”. https://github.com/kuip/dlt-design-patterns.

National Institute of Standards and Technology. 2015. *FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION (FIPS PUB) 180-4 - Secure Hash Standard*. Tech. rep. National Institute of Standards and Technology. Visited on 03/17/2018. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf.

Produce Traceability Initiative. 2018. *Heilmeier’s Catechism*. Tech. rep. Produce Traceability Initiative. https://www.producetraceability.org/documents/faqs%5C%202011%5C%20062011%5C%20final.pdf.

Online Sources

Cardinal Commerce. 2018. “1PL 2PL 3PL 4PL”. https://www.cardinalcommerce.com/startups/distribution-outsourcing/1pl-2pl-3pl-4pl.

Coinstaker. 2018. “Initial Coin Offering - A killer introduction to ICOs and an awesome list”. https://www.coinstaker.com/initial-coin-offering/.

Eder, Pablo. 2017. “eHarvestHub Review”. Visited on 11/20/2017. https://hackernoon.com/eharvesthub-review-61dd04ba8cc4.

MIT Megacity Logistics Lab. 2018. “Better logistics for cities. Better cities for logistics.” http://megacitylab.mit.edu/.

Additional Sources

Drescher, Daniel, and SpringerLink (Online service). 2017. *Blockchain Basics: A Non-Technical Introduction in 25 Steps*. Berkeley, CA: Apress. ISBN: 9781484226049.

Herman, Justin. 2018. “The Blockchain at the GSA Emerging Citizen Technology Office”. DC Blockchain Summit.

Johansson, Bjorn. 2018. “Assessing blockchain technology for Transport Data Logger” [**inlang**eng]. Student Paper. MA thesis, Lund University. `http://lup.lub.lu.se/student-papers/record/8934190`.

KPMG. 2018. *Seizing New Potential: KPMG’s Digital Ledger Services*. Tech. rep. KPMG. `https://home.kpmg.com/content/dam/kpmg/us/pdf/seizing-new-potential.pdf`.

People’s Bank of China. 2017. *CIRC Notice on Preventing Financing Risk of Initial Coin Offerings*. Tech. rep. People’s Bank of China. `http://www.pbc.gov.cn/goutongjiaoliu/113456/113469/3374222/index.html`.

Satapathy, Goutam. 1999. “Distributed and collaborative logistics planning and replanning under uncertainty: a multiagent based approach”. PhD thesis.

Securities and Exchange Commission. 2017. *Investor Bulletin on Initial Coin Offerings*. Tech. rep. Securities and Exchange Commission. `https://www.sec.gov/oiea/investor-alerts-and-bulletins/ib_coinofferings`.

Verisign. 2018. *How the Domain Name System Works*. Tech. rep. Verisign. `https://www.verisign.com/en_US/website-presence/online/how-dns-works/index.xhtml`.