



Service Management for P2P Energy Sharing Using Blockchain – Functional Architecture

Mohammad Hossein Abdsharifi
Ripan Kumar Dhar

This thesis is submitted to the Faculty of Engineering at Blekinge Institute of Technology in partial fulfilment of the requirements for the degree of Master of Science in telecommunication systems. The thesis is equivalent to 20 weeks of full time studies.

The authors declare that they are the sole authors of this thesis and that they have not used any sources other than those listed in the bibliography and identified as references. They further declare that they have not submitted this thesis at any other institution to obtain a degree.

Contact Information:

Author(s):

Mohammad Hossein Abdsharifi

E-mail: moab19@student.bth.se

Ripan Kumar Dhar

E-mail: ridh19@student.bth.se

University advisor:

Prof. Kurt Tutschku

Department of Computer Science

Faculty of Engineering
Blekinge Institute of Technology
SE-371 79 Karlskrona, Sweden

Internet : www.bth.se
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57

Abstract

Blockchain has become the most revolutionary technology in the 21st century. In recent years, one of the concerns of world energy isn't just sustainability yet, in addition, being secure and reliable also. Since information and energy security are the main concern for the present and future services, this thesis is focused on the challenge of how to trade energy securely on the background of using distributed marketplaces that can be applied. The core technology used in this thesis is distributed ledger, specifically blockchain. Since this technology has recently gained much attention because of its functionalities such as transparency, immutability, irreversibility, security, etc, we tried to convey a solution for the implementation of a secure peer-to-peer (P2P) energy trading network over a suitable blockchain platform. Furthermore, blockchain enables traceability of the origin of data which is called data provenience.

In this work, we applied a secure blockchain technology in peer-to-peer energy sharing or trading system where the prosumer and consumer can trade their energies through a secure channel or network. Furthermore, the service management functionalities such as security, reliability, flexibility, and scalability are achieved through the implementation.

This thesis is focused on the current proposals for p2p energy trading using blockchain and how to select a suitable blockchain technique to implement such a p2p energy trading network. In addition, we provide an implementation of such a secure network under blockchain and proper management functions. The choices of the system models, blockchain technology, and the consensus algorithm are based on literature review and it carried to an experimental implementation where the feasibility of that system model have been validated through the output results.

Keywords: Distributed Systems, Blockchains, Energy Management, Service Management, Energy Trading, , Energy Sharing, Hyperledger Fabric.

Acknowledgments

Throughout the writing of this dissertation, we have received a great deal of supervision and assistance.

We would first like to thank our great supervisor, Kurt Tutschku, Prof. Dr. at Blekinge Institute of Technology, whose expertise was invaluable in formulating the research questions and methodology and also for sharing his knowledge to improve our research. Your insightful feedback and encouragement pushed us to sharpen our thinking and brought our work to a higher level.

Finally, we would like to thank our great family for their support and encouragement in our entire study life.

Contents

Abstract	i
Acknowledgments	ii
1 Introduction	1
1.1 Introduction	1
1.2 Motivation	3
1.3 Aim and Objective	4
1.4 Research Questions	5
1.5 Contribution to the Thesis	5
2 Methodologies & Related Work	7
2.1 Research Methodologies	7
2.2 Literature Review	9
2.2.1 Formal Experiment	11
2.3 Related Work	11
2.4 Consensus Algorithm	15
2.5 Selection of Blockchain	17
2.5.1 Hyperledger Fabric	19
3 Blockchain & Technologies	20
3.1 Blockchain	20
3.2 Peer-to-Peer Network	24
3.3 Smart Contract and Chaincode	25
3.4 Blockchain Platforms	28
3.5 Security and Privacy	29
4 System Design and Modeling	30
4.1 System Model	30
4.1.1 Proposed Model	34
4.2 System Architecture	34
4.2.1 Secure Market Participation	35

4.2.2	Blockchain and Smart Contract in P2P Energy Trading	37
5	Implementation and Experiment	38
5.1	Implementation Requirements	38
5.2	Implementation Description	41
5.2.1	Sequence Diagram of the System	41
5.2.2	Fabric Transaction Life Cycle	43
5.2.3	Energy Trading Implementation Algorithm	46
5.3	Functions and Commands Description	47
5.4	Implementation Results	52
6	Validation and Results	62
6.1	Validation	62
6.2	Results	65
6.3	Lesson Learned	68
7	Conclusions and Future Work	70
7.1	Conclusion	70
7.2	Future Work	71

List of Figures

1.1	Service Management for P2P Energy Scenarios Using Blockchains [1]	2
2.1	Research methodology [2]	8
2.2	Blockchain frequencies used for p2p energy sharing [3]	17
3.1	The vital blockchain characteristic [4]	21
3.2	Blockchain diagram [4]	22
3.3	Blockchain diagram [5]	23
3.4	Peer-to-Peer Network [1]	24
3.5	Content of each block in the blockchain [6]	25
3.6	Main smart contract [1]	26
3.7	P2P smart contract [1]	27
4.1	Individual energy storage system	31
4.2	Large scale energy storage system	32
4.3	NIST smart grid conceptual model [7]	33
4.4	Market participation [1]	36
5.1	Hyperledger Fabric transaction life cycle [8]	42
5.2	Stepwise to implement a Fabric network	45
5.3	Network participation algorithm	46
5.4	Energy trading algorithm	47
5.5	Bring up the peer using docker compose.	49
5.6	Create the channel.	49
5.7	CouchDB installation YAML file	50
5.8	Network down function	52
5.9	Set up the network	53
5.10	Connect organization one to the channel	54
5.11	Connect organization two to the channel	55
5.12	Add a new peer to the network	56
5.13	Add a new peer to the network	56
5.14	Add database to the network	57

5.15	smart contract function	58
5.16	Installing chaincode - Signing smart contract	59
5.17	Installing chaincode - Signing smart contract	59
5.18	API main function	60
5.19	Transaction creation	61
5.20	Succeed transaction	61
6.1	Hyperledger Fabric transaction life cycle [8]	64

List of Tables

1.1	Contribution to the thesis	6
2.1	Search strings	10
2.2	Consensus used in the blockchain networks	16
5.1	Prerequisites features	39
5.2	Virtual machines specifications	39
6.1	Implementation commands	64
6.2	Public Git repositories	64
6.3	Comparison of different blockchains	66

Nomenclature

API	Application Programming Interface
BC	Blockchain
CPU	Central Processing Unit
DB	Data Base
DERs	Distributed Energy Resources
DPoS	Delegated Proof of Stake
DSL	Domain Specific Language
EMS	Energy Management System
ESCC	Endorsement System Chaince Code
ESS	Energy Storage System
ICT	Information and Communication Technology
IPFS	Interplanetary File System
MEMS	Main Energy Management System
MVCC	Multi-Version Concurrency Control
NIST	National Institute of Standard
P2P	Peer-to-Peer
PBFT	Practical Byzantine Fault Tolerance
PoA	Proof of Authority
PoS	Proof of Stack

PoW Proof of Work

SG Smart Grid

VM Virtual Machine

VSCC Validation System Chaincode

The introductory background information, motivation, and aim of this thesis have been discussed in this chapter. Initially, this chapter talks about the background information about the p2p energy trading system and blockchain technology. Subsequently, research questions and authors' contributions to the thesis are also discussed in this chapter.

1.1 Introduction

Sustainability is the keyword for future energy distribution to meet the demand and supply of energy in an efficient way. For making the system sustainable, there are many concepts that are being adopted for better performance in distribution scenarios. Amongst them, local energy generation and peer-to-peer (p2p) energy trading in the local market can reduce the energy consumption cost as renewable energy sources are used to generate energy at the user's premises and increase the smart grid resilience [1]. However, local energy trading with peers can have trust and privacy issues. A centralized system can be used to manage this energy trading, but it increases the overall cost of the system and also faces several issues such as security, energy reuse, and sustainability [1]. To meet the energy demand, a utility needs to install backup power plants which result in higher production costs, emission of harmful gasses, as it does not have any control over the demand pattern. The smart grid (SG) has emerged as a modern form of the power grid. A p2p energy trading system enables two-way communication between energy consumers and producers for making an efficient energy management system and also reduces the requirement of thermal power plants and conserves electricity [1].

Moreover, the future energy distribution system will be complex [9]. For making the energy distribution system more sustainable, blockchain technology can be used because it is a distributed ledger with the maximum advantage of consensus procedures and cryptography security [9]. The emerging applications of blockchain are highly effective because of its immutability, irreversibility, security, and aptitude to

decentralize markets [4, 10]. Since blockchain is rapidly gaining momentum in this context as an information and communication technology (ICT) platform and referred to as the “Internet of Value” [9]. Figure 1.1 below, shows an overview of the entire scenario.

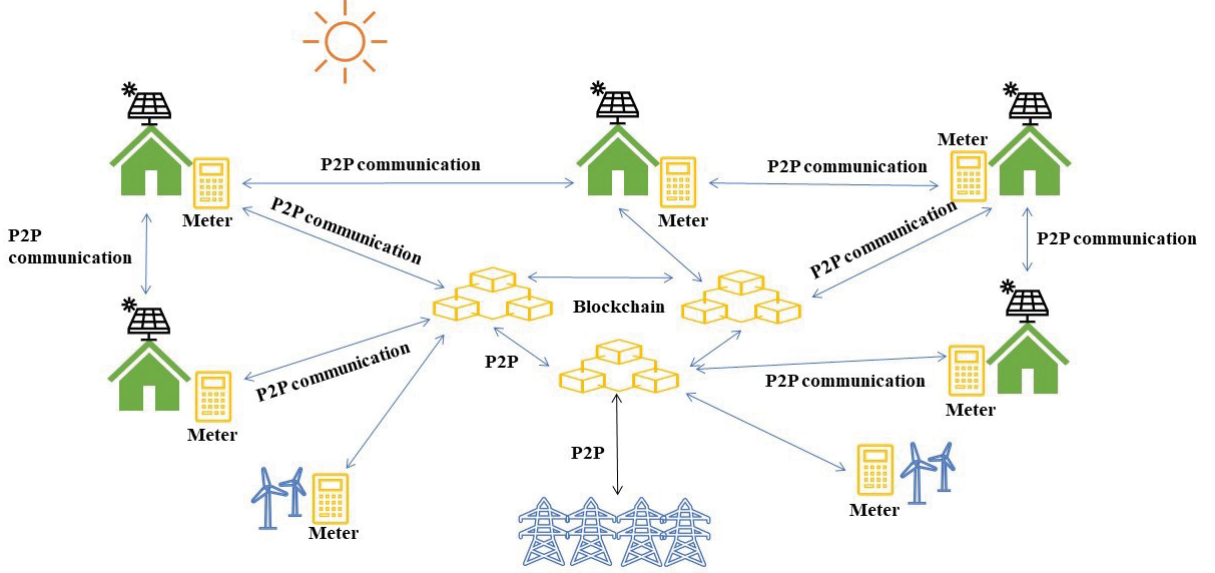


Figure 1.1: Service Management for P2P Energy Scenarios Using Blockchains [1]

Based on network view, blockchain can be classified into different types, such as public permission-less, public permission, private permission-less, and private permission [11]. Blockchain-based concepts are abstracting the network as a connectivity system only and do not consider the enhanced capabilities of cloudified networks in terms of highly secure communication, high level of authentication, and highly secure access to data.

Moreover, it is expected that the analysis and usage of the energy data will increasingly be carried out in service chains, e.g., the usage pattern might be processed and analyzed by AI-based services chains in the cloud. The development and exposure of energy or data services and service chains might enable mutual benefits. Energy providers, individual customers, and network operators can engage in novel forms of business relationships. The exposure of these data services, however, requires an understanding of their confidentiality, trust, and privacy needs. Though the algorithm and data structure of blockchain technology is very precise for energy service management some functionalities e.g. (authentication, authorization, integrity, and security, etc.) should be considered to obtain optimum results for data or energy

storing and retrieving data or energy using blockchain technology [12]. For implementing the distributed service management, private and public ledgers are available only for invited users and a specific transaction might be available to only a specific subset of the users for increasing security [12].

In this work, we will investigate and implement a suitable blockchain and its data structure for p2p energy service management (e.g., authorization, authentication, immutability, security, etc.). Furthermore, we investigate how one can select blockchain techniques from the multiple available technologies. In addition, the comparison of current and available proposals for p2p Energy Sharing using blockchains will be discussed.

This thesis is one of two on service management for p2p energy services. Both the theses are collaborated but focused on different research questions. The other thesis has titled "Service Management for p2p Energy Scenarios Using Blockchain - Identification of Computational Efforts" written by [13], where he is evaluating the performance of the purposed architecture and implementation from this thesis.

1.2 Motivation

Sustainability is a crucial factor for future energy distribution where the p2p energy trading ensures redundancy and security as well as efficiency in the smart grid. Due to the decentralized features, p2p energy trading use blockchain technology which enables the system to be more reliable, scalable, and secure. From the existing literature, we found that there is a lack of trust and security in the p2p energy sharing network, thus we want to implement a framework that can overcome the difficulties in p2p energy sharing. This research will be helpful for the researchers to develop a secure and efficient p2p energy sharing network to conduct their research in many other related scenarios.

The concept of this proposition work starts from digging into the Symphony project [12]. The Symphony project intends to investigate and solve the challenge of exposing services in cloud-native and federated settings via a marketplace, while also fulfilling the security, confidentiality, privacy, and provenance requirements for future digital society applications. The Symphony project aims to provide solutions for securing, monitoring, and accounting of service chains utilizing blockchain-based approaches by analyzing the use-case of data services for managing Renewable Energy Sources (RES) [12]. We have taken the point from the Symphony project and considered the p2p energy trading system using blockchain technology. In addition to that, security, scalability, and reliability are the crucial concern that can be attained through this kind of collaboration [12].

This thesis proposes that blockchain might be utilized to address some of these issues for p2p energy trading use cases. As blockchain works in a distributed manner, it ensures the security of the decentralization of the smart grid and resources over the p2p network. The decentralization guarantees various marks of resumption against security assaults [7]. The blockchain ensures security by a consensus process, hashing, and decentralized storage [14]. It is exceedingly difficult to insert fake data as the blockchain uses the consensus of nodes that consist of fifty-one percent of the network [14]. Considering implementing such a use case, many questions arise in the mind that how anyone can select the specific blockchain techniques amongst many available blockchain technologies and what management functionalities should be considered while implementing such use cases. These questions motivate our thesis work to analyze, discuss and implement blockchain technology in the p2p energy trading networks.

1.3 Aim and Objective

The aim of this research is to identify and implement the most suitable service management functions and data structures for a p2p energy sharing use case in a public cloud environment using blockchain. Furthermore, we aim to research how we can select the best-suited blockchain technique for the p2p energy service management scenario from the set of available blockchain technologies. In order to achieve our aims, following detailed objectives are considered:

- **Objective - 1:** Provide the latest overview of the literature review for the p2p energy management using blockchain technologies and algorithms.
- **Objective - 2:** To categorize and compare in a qualitative way (from the literature review) the available blockchain algorithms in terms of structure and security for the considered use case.
- **Objective - 3:** To provide an initial implementation of the blockchain for p2p energy information sharing usage in the cloud and identify the major requirements for the system.
- **Objective - 4:** Discuss and verify the suitability of the blockchain implementation for meeting the requirements of the use case. To identify the priorities and prerequisite of such implementation and achieve the implementation solution. We also aim to find a suitable model for implementing such a scenario.

1.4 Research Questions

To achieve the objective of the thesis, three research questions are defined:

RQ-1: What are the current proposals for p2p energy sharing using blockchains and how they compare each other?

RQ-2: Given that, there are multiple blockchain implementation technologies available. How can one select the blockchain technique for implementation of the proposed p2p energy management use case of this thesis?

RQ-3: How can one implement the blockchain and its data structures or the service management for the considered use case of p2p energy sharing? Which management functions can be supported by a blockchain?

1.5 Contribution to the Thesis

From theoretical research through practical implementation and analysis, there is a lot of work in this thesis. This section details how much each of the authors of this study individually contributed to the work presented here. Table 1.1 summarizes the many subjects covered in the study, as well as the author's contribution to this study.

The task was distributed between the authors equally (50 - 50) based on their skills and the work was given to each other in such a way that all important aspects of the research can be addressed precisely. Table 1.1 is showing how we have collaborated in this thesis work.

Research topic	Author 1	Author 2
Introduction	✓	✓
Problem statement	✓	✓
Motivation	✓	✓
Aim and objective	✓	✓
Overview and methodology	✓	✓
Consensus algorithm	✓	✓
Security and Privacy	✓	✓
Related work	✓	✓
Requirements	✓	✓
Selection of Blockchain	✓	✓
System architecture	✓	✓
Implementation environment	✓	✓
Technology	✓	✓
Future work	✓	✓

Table 1.1: Contribution to the thesis

Chapter 2

Methodologies & Related Work

This chapter outlines the important and necessary information by reviewing previous prior studies done in this field. In addition to that, this chapter will provide knowledge about the p2p network, blockchain technology, consensus algorithm, cloud computing, and its different use cases for better understating to the reader. The chapter begins with research methodology and then is followed by the rest of the topics such as literature review, memorization of related works, and finally end with the selection of blockchain and Hyperledger Fabric.

2.1 Research Methodologies

In this research "Qualitative" and "Quantitative" both methodologies have been used for coming up with the solution to the research question mentioned in the above section. The qualitative methods have been used for research questions related to finding the current proposals for p2p energy sharing and the comparison of blockchain implemented to p2p energy sharing. The quantitative method has been used for the implementation of proposed architecture or modeling for p2p energy sharing. The proposed architecture is modeled based on the knowledge accumulated from the literature review. The steps consist of four phases followed by a spiral model below [2]. Figure 2.1 is showing how much was achieved in the work presented here. The aim is to explain the applied engineering methodology and how this contributed to solving the problem at a large scale. Here we followed a single iteration of four phases from the model. The red "X" sign in figure 2.1 is showing the completion of our study.

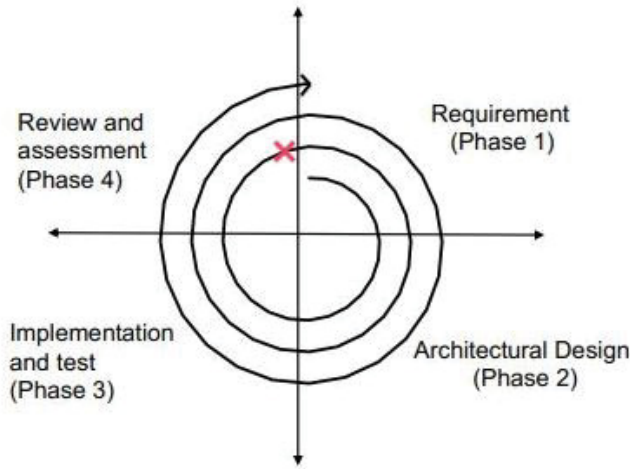


Figure 2.1: Research methodology [2]

Phase - 1: Literature review, analyzing and the documentation of requirements.

- Forming search strategies.
- Applying inclusion and exclusion criteria
- Data extraction

Phase - 2: Investigation of new concepts and technologies and documentation.

The possible technologies processes will be determined for implementation and experimentation for the research. There are several technologies, techniques, and frameworks for the implementation of the research topic, therefore a firm and precise investigation is needed on that technologies and concepts to validate the research.

Phase - 3: Analysis of the implementation methods, verification, finding results, and documentation.

- An analytical model or framework of blockchain technologies will be determined for the p2p network.
- The selected framework will be implemented and verified the results based on research needs.

Phase - 4: Analysis and assessments according to the founded results and documentation.

A review will be done based on the validity of the output results with theoretical findings from the literature review. The documentation process will be done continuously in every phases.

2.2 Literature Review

The main objective of the literature review is to accumulate the background knowledge and data from the related litterateurs to find the solution for the research problems mentioned in the thesis.

Search Strategy

This progression helps us to find the essential exploration of the related papers that are identified with our research point of view. As the research area is immense, we used different search strategies and search engines for accumulating the primary papers which can be useful to provide or find the solution to the research questions. Though all the papers are not obtained through the search strings, we used the snowballing concept to find more relevant papers. Google scholar engine with certain search strings is widely used to find the most frequent papers which led to redirecting authors to IEEE and ACM digital libraries. Furthermore, some more resources such as blogs, official project websites, and web articles are also used in this thesis as follows [12, 15, 16].

Search Strings

One of the most important aspects of any research paper is choosing the proper keywords to discover the most relevant publications. We organized certain strings depending on the thesis emphasis to identify the finest publications in various databases. To get the most complete papers for this study, the operations "AND" and "OR" are employed to merge the strings. Table 2.1 shows the strings used for finding the papers from two databases in this thesis.

Inclusion and Exclusion Criteria

Considering some significant criteria to find the most relevant and updated research papers, we limited our scope to the following. The papers must be related inclusion criteria which is mention in below, because our thesis is focusing on to implement

Search string	Data Base
Blockchain AND p2p energy trading AND smart grid	IEEE and ACM
Blockchain AND security OR privacy OR scalability OR reliability	IEEE and ACM
Blockchain AND cloud AND p2p energy trading OR energy sharing	IEEE and ACM

Table 2.1: Search strings

blockchain technology in p2p network use cases.

Inclusion Criteria

- Paper must be written in English.
- Paper focusing on blockchain, smart contract, p2p energy sharing.
- Paper based on p2p energy sharing or energy trading using blockchain technology.
- Paper focusing on implementation of blockchain in p2p energy sharing scenario.

Exclusion Criteria

- The papers which are not published.
- The paper which does not address to the research questions
- The papers which are not focusing on p2p energy sharing using blockchain technology

Data Extraction

For getting a comprehensive overview of the implementation of blockchain technology in p2p energy distribution, frameworks, system modeling, and functionalities, we went through the related research work. In addition to that, based on our research question we tried to find out the specific information. Furthermore, all the information is saved in a table in the excel file.

The extracted information will help us to find out, which blockchain technology is most commonly used in which area, systems modeling, consensus algorithm, and security features.

2.2.1 Formal Experiment

By this approach, we are implementing an architecture which is modeled based on the knowledge accumulated from the literature review. In addition to that, the literature review has given a deep insight into the implementation of p2p energy sharing or trading using different blockchain technologies and functionalities. We modeled an architecture that provides p2p energy sharing which is distributed approach using blockchain technology. Our approach contains three steps:

- System architecture design and modeling.
- Implantation of that system architecture in the cloud platform.
- Analysis the results and functionalities of that system architecture.

2.3 Related Work

The idea of conducting a literature review is to gather information about existing research and disputes relevant to a specific area of study. It provides a deep understanding of concepts that are required for the analysis of blockchain and its application in cloud resource management. The search strategy, inclusion and exclusion criteria, digital libraries used, related work and latest research area have been discussed. The blockchain studies mentioned here are generic at the beginning and lead to more specific ones at the later stage. Different types of blockchain based frameworks used among cloud service providers for various reasoning have been discussed in this section along with implementation.

In this section, we will briefly discuss the ideas and implementation methods are adopted by the different researchers based on the paper we found in our literature search. Blockchain technology is a relatively new technology, hence it is difficult to find the exact research area that we are heading to conduct. For modeling our architecture, we reviewed related papers to find the most suitable model for the p2p energy sharing using blockchain technology.

P2P Energy Trading

In the papers [14, 17–27], summarizes that, in p2p energy trading model all the nodes are able to buy and sell at the same time. A node in the network is referred to as prosumers and consumers. Due to a lack of flexibility, efficiency, availability, and security, the p2p energy trading model growing rapidly over the centralized energy

distribution model. As the p2p energy trading model works based on distributed network concept thus it provides efficiency, scalability, and security.

Renewable energy sources such as (solar panels, wind turbines, fossil fuels, and plug-in electric vehicles) acting as small-scale energy producers are referred to as prosumers and microgrids. The whole scenario creates a competitive p2p energy trading between microgrids, prosumers, and distribution system operators. Additionally, p2p energy trading provides redundancy to avoid consumers facing power outages. The p2p energy trading requires essential functions such as secure energy bidding, secure energy negotiation, secure p2p payment, and secure energy transmission.

P2P Energy Trading Using Blockchain Technology

In the articles [14, 17, 19, 27–30], summarizes that, in conventional energy trading the flow of transactions is unidirectional whereas the p2p energy trading allows multi-directional trading. The main challenges of p2p energy trading are scalability, availability, security, privacy, and loss of data. To mitigate the shortcomings of the p2p network, blockchain technology is used.

Blockchain has distributed data structure that can keep any information, transaction records, etc in a secured way. Blockchain creates a secure and trustable p2p communication platform. In blockchain technology, each block can be identified by the hash function and connected with another block by the previous hash. Users in blockchain technology are identified by using their cryptographic public keys which enable security in p2p energy trading. By implementing the blockchain technology, the authors tried to store and resume the data between the nodes.

Energy Management Functionalities Using Blockchain Technologies

Through the literature review of papers [17, 23–26, 28], a couple of blockchain-based p2p energy trading system multiple functionalities were found. The authors in these papers mostly demonstrate security as their key functionalities. For increasing security, they have designed and proposed different models. As the smart contract is a simple program stored in the blockchain and it runs when all the conditions are met, thus it faces security vulnerabilities such as Re-entrancy vulnerability, Timestamp dependence, Calls stack depth vulnerability, Transaction ordering attack, Integer underflow, and overflow, Integer underflow and overflow, etc. For checking the vulnerabilities in the smart contracts, they used an open-source tool named Oyente

tool. Most of the security features are inherited from the blockchain such as integrity, trust, decentralization, availability, and non-repudiation.

Cloud Based P2P Energy Trading Using Blockchain Technology

There are several articles and papers that explore the concept of a cloud-based p2p energy trading system. For example, [14, 19, 27, 28, 30–33].

In paper [23], the authors proposed a blockchain and cloud-based manufacturing architecture, which is described in depth by examining platform, security, and performance. The fundamental goal of the architecture is to operate a secure p2p network with various functionalities.

In the article [33], the authors analyzed the security aspects of the energy trading model based on blockchain, smart city, Internet of Things, and cloud computing. They proposed a privacy-preserving distributed energy transaction scheme PP-BCETS by implementing Ciphertext Policy Attribute-Based Encryption (CP-ABE) as a core algorithm in ciphertext form to optimize user privacy. In addition to that, the authors suggested a credibility-based equity proof consensus mechanism to increase the system operation efficiency and make a lightweight distributed transaction model.

The authors in paper [34], proposed a p2p energy trading system using the Hyperledger Fabric and cloud computing to increase the performance in terms of speed, flexibility, and scalability. The authors also analyze the applicability of blockchain and cloud computing for a p2p energy trading system by determining what types of data must be shared in terms of increasing Smart-meter performance.

Decentralized NIST Conceptual Model For Smart Grid Using Blockchain Technology

In the article [7], the authors analyze and integrate the standards. National Institute of Standard and Technology (NIST) conceptual model of the smart grid using three blockchain features as decentralization, trust, and incentive. They also analyzed the key features of building automation and control systems, home energy management systems, grid-wise energy management systems, and demand response management systems with respect to key features of the blockchain. Sub-domains such as metering, energy trading, virtual power plants, demand response, tracking renewable energy, cloud computing, load forecast, electric vehicles, cybersecurity, and micro-grids are also analyzed with respect to blockchain key features. Moreover,

the authors identified the key priorities of each domain and specified a decentralized blockchain-based version of the NIST model.

Implementation Scenarios For P2P Energy Sharing Using Blockchain

In the article [26], the authors proposed and implemented a localized p2p trading model using the Hyperledger Fabric where all the energy and cash transactions are in the variable and immutable way. The model consists of three main entities such as energy nodes, energy aggregator, and energy meter. In addition to that, all the authorized nodes maintain distributed shared databases to keep their records. For traceability, verification, and authentication each node uses a cryptographic hash function.

In the article [28], the authors, heuristic algorithms to solve with an objective to minimize the cycle time of the container packing procedure and constraint that the entire network is patched. A single transaction consists of multiple phases: propose, endorse, broadcast, deliver, commit. This occurs over multiple kinds of nodes: peer, orderer, Kafka, Zookeeper, etc. A flexible and distributed model of trust applies with multiple business participants that collaborate together to complete the transactions. This brings about extreme complexity to the service management operations.

In the paper [31], the authors implemented an energy trading platform on a permissioned blockchain infrastructure to allow energy trading amongst distributed energy resources (DERs). They also implemented the whole scenario in the Canadian micro grid. The proposed system was implemented by using Hyperledger Fabric within the permissioned blockchain framework. the entire system has been segmented into the private channel where all private has their own ledger and peers maintain the ledger states.

In the paper [32], the authors proposed a blockchain-based energy trading framework named it FeneChain. In this framework, the authors tried to leverage anonymous authentication to protect user privacy and make the trading services more secure. A consortium blockchain they build to verify and record energy trading transactions. The performance was also checked by implementing a prototype via a local Ethereum test network and RaspberryPi.

In the article [33], the authors proposed a blockchain-based scheme for secure energy trading systems in a smart grid. The authors proposed two algorithms, one is the cost-aware and other store-aware algorithms. In addition to that, they also examine the performance of algorithms.

In the article, [18], for overcoming the shortcoming of security, privacy, and latency, the authors proposed a scheme ET-Deal which is a smart contract-based secure energy trading for a smart grid system in p2p energy trading or sharing. Ethereum smart contract and IPFS (InterPlanetary File System) is used for the p2p energy trading management system. The authors also tested the ET-Deal on the Mythril which is an open-source tool.

2.4 Consensus Algorithm

In a blockchain network, reaching an agreement is a difficult and important process. Because the new block is validated by all nodes in the network, new transaction records would be added to the blockchain. It is worth noting that after blocks have been validated, they cannot be changed or removed. Blockchains are built with the intention of being valid in a trustless and unstable network with hostile users. As consensus algorithms, many approaches are devised and developed. Based on blockchain development, the number of these algorithms is growing every day. However, in this part, we will go through the most common consensus algorithms utilized in blockchain networks, as well as our desired consensus algorithm in this thesis [35].

In the table 2.2 below, the authors of this thesis are tried to introduce the most important consensus algorithm used in the blockchain networks. The table contains blockchain consensus algorithms overview in terms of characteristics, strengths and weaknesses, application, security issues, and scalability.

Articles	Conse -nsus	Strengths	Weakness	Security issues	Scalability
[36]	PoW	strut Anti-DoS attacks defence Low, impact of stake on mining chances	High energy requirements Wastage of computations and resources Vulnerable to 51 percent attack	DDoS, Sybil, Bribe attack, Selfish mining attack	High
[37]	PoS	Energy efficient, does not require high computational power, decentralize	The Rich become richer	DDoS, Sybil, Bribe attack, Long range attack, Stake bleeding attack	High
[38]	DPoS	Faster execution, does not require sophisticated hardware, energy efficient	easier to launch 51 percent attack, cartels can be easily organized	DDoS, Sybil, Long range attack	High
[39]	PBFT	Reduce energy usage and low latency	Not scalable	Sybil attack	Low.
[16]	PoA	Less computational power requirement, Less transaction latency, High throughput	Faces consistency issues for permissioned Blockchain, Not decentralized , Identities of validators are publicly visible	DDoS, Sybil	High

Table 2.2: Consensus used in the blockchain networks

2.5 Selection of Blockchain

Considering blockchain is still an evolving technology, new discoveries may quickly change which platform is best for specific applications. In the p2p energy trading network, participants must be authorized before joining the network (discussed in chapter 4). Therefore, a private-permissioned blockchain is needed to achieve such authentication.

From our literature review, we found many use cases such as smart grid [24], energy sector [26, 27], cloud environment [28], and residential communities [31] implemented Hyperledger Fabric for energy trading or sharing. Furthermore, based on the research on the comparison of different blockchain technologies in the paper [3], numerous papers have been reviewed to find the suitable blockchain for the use case of p2p energy sharing. The frequency of blockchain for this specific use case is seen in the graph below. However, there are additional blockchain technologies that are not covered in this study, like Bitcoin, Cardano, and Tezos, but still, ten different blockchains are studied for such applications.

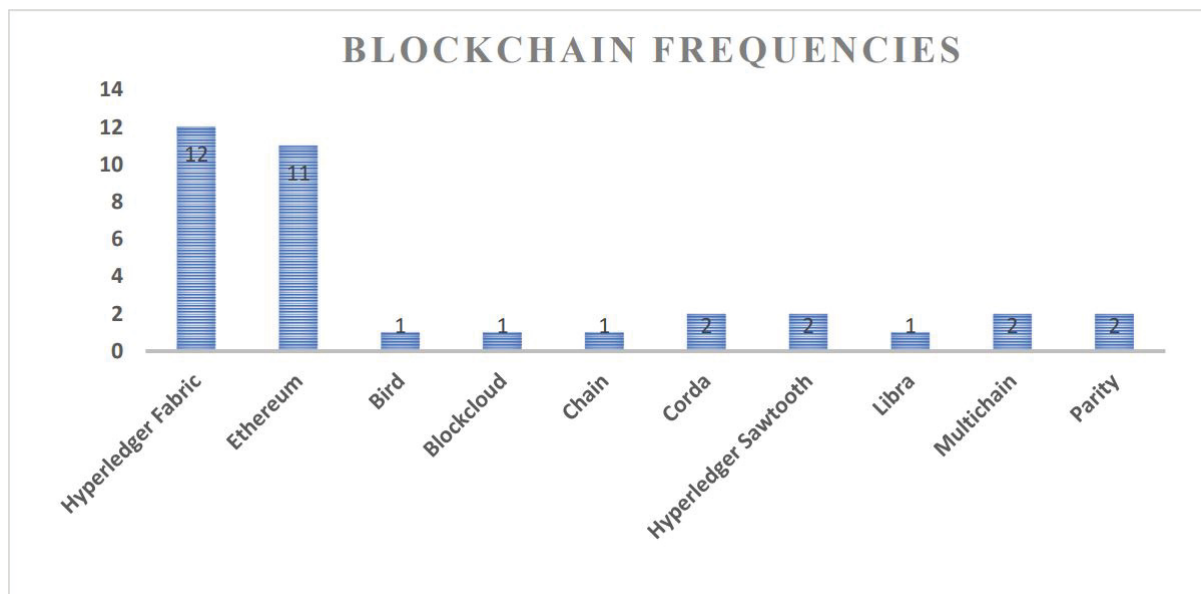


Figure 2.2: Blockchain frequencies used for p2p energy sharing [3]

Figure 2.2 above shows the frequencies of used blockchain technologies for p2p energy sharing scenarios. In figure 2.2, the Y-axis represents the number of papers, and the X-axis represents the frequency of different blockchains. In addition to that, we can also see that the hyper ledger fabric and Ethereum are the most used

blockchain technologies for energy trading purposes so far [3]. Therefore, it will be easier to compare and convert it into academic work. Since Hyperledger Fabric and Ethereum are not similar, the ultimate decision is made based on the features that are most appropriate for this thesis.

Hyperledger Fabric uses different technologies, containers and it is compatible with different programming languages. The entire Hyperledger Fabric infrastructures are open source; therefore, everyone has access to the source codes, and it is possible to match it with different scenarios. It is highly scalable, configurable, and modular that enables innovation, versatility, and optimization for a broad range of industry use cases, for example, energy trading. Fabric is also the first distributed ledger platform to support smart contracts written in multi programming languages such as Java, Go and Node.js [40]. However, the ability of authors in this thesis in Go-Lang is the reason for selecting Go as the Chaincode language.

Furthermore one of the most crucial components of a blockchain is the consensus mechanism, which determines how different parties may agree on the state of the distributed ledger. Ethereum currently uses a Proof-of-Work mechanism, which consumes a large amount of energy, but it has been tested for several years and is widely regarded as very secure and trustworthy in public blockchains. Hyperledger Fabric employs the RAFT consensus method, which is based on voting and requires more extensive communication despite being computationally cheaper. Both the blockchains are planning to improve and develop their consensus to Proof-of-Stake for Ethereum and PBFT (Practical Byzantine Fault Tolerance) for Hyperledger Fabric, but at the time of writing this thesis, their developments are not completed [3].

Addition to that, the privacy and security of p2p network, a private-permissioned blockchain is needed to have such a secure network, therefore, Hyperledger Fabric is the choice landed in this thesis to provide a fully secure environment for energy trading in the market. Hyperledger Fabric minimizes risks and guarantees that transaction nodes are part of the network. Only the parties engaged in the transaction have access to the transaction's record, not the whole network. This ecosystem offers all the benefits of blockchain architecture, including data privacy, immutability, and other features. In comparison with other permissioned blockchains, like Ethereum or Bitcoin, the Hyperledger Fabric is well-performing in terms of latency, execution time, and throughput [26].

Therefore, the main reasons for considering Hyperledger Fabric for the energy trading scenario are security, open-source and availability of Hyperledger, scalability, and ability of Hyperledger to expand it into a large-scale network, and also using efficient consensus algorithm that we rarely found in other blockchains.

2.5.1 Hyperledger Fabric

Fabric is a Hyperledger project sponsored by the Linux Foundation that provides a modular and flexible open-source framework for creating and running permissioned blockchains. Hyperledger Fabric is the first blockchain technology for running distributed applications that is genuinely expandable. It has modular consensus methods that allow it to be customized to certain use cases and trust models. Fabric is also the first blockchain system to run distributed applications built-in general-purpose programming languages without requiring a systemic reliance on a cryptocurrency. Existing blockchain systems, on the other hand, need “smart contracts” to be created in domain-specific languages or rely on a cryptocurrency [40].

Hyperledger Fabric implements the permissioned model with a portable notion of membership that can be combined with industry-standard identity management systems. Fabric provides a completely new blockchain design to allow such flexibility, as well as a revamped approach to dealing with non-determinism, resource exhaustion, and performance assaults [40].

As this thesis is related to energy management and energy trading, and the scope is limited to a private blockchain, therefore, Hyperlegder Fabric performs better than the other blockchain technologies available in the market.

Chapter 3

Blockchain & Technologies

This chapter is the continuation of the previous chapter based on a literature review to discuss the core theoretical concepts that we addressed in our thesis work. The content of the below sections will assist the readers to understand the concept and the methods followed in the thesis. Here, we have discussed the concepts related to blockchain, consensus algorithm, p2p network, and data storage.

3.1 Blockchain

Blockchain technology refers to decentralized, distributed, and digital ledger consisting of records called blocks that are used to record transactions across many computers (which are called nodes) so that any involved block cannot be altered retroactively without the alteration of all subsequent blocks [41].

Blockchain technology was invented in 2008 and implemented in cryptocurrency Bitcoin. The network taxonomy of blockchain is classified based on the different types of network views such as public permission-less, public permission, private permission-less, and private permission [11]. For immutability and fault-resilience made it useful for handling transactions of currency. However, more advanced applications using e.g., smart contracts have been developed since then. According to [42], Szabo visualized the concept of smart contracts in 1994. A smart contract is not a legal contract, but an agreement enforced by the cryptographic code of a blockchain and distributed to each node in the network. After deploying a smart contract runs according to its programmed conditions, monitor some process, and can primarily handle payments paid out according to programmed conditions [5].

According to [43], bitcoin was first proposed by an anonymous developer named Satoshi Nakamoto, has revolutionized the existing financial system in that it enables reliable transactions to occur through a decentralized system even among anonymous parties [43]. As the underlying technological base of Bitcoin, blockchain is expected to create a new economic system [43]. Blockchain is a distributed ledger system

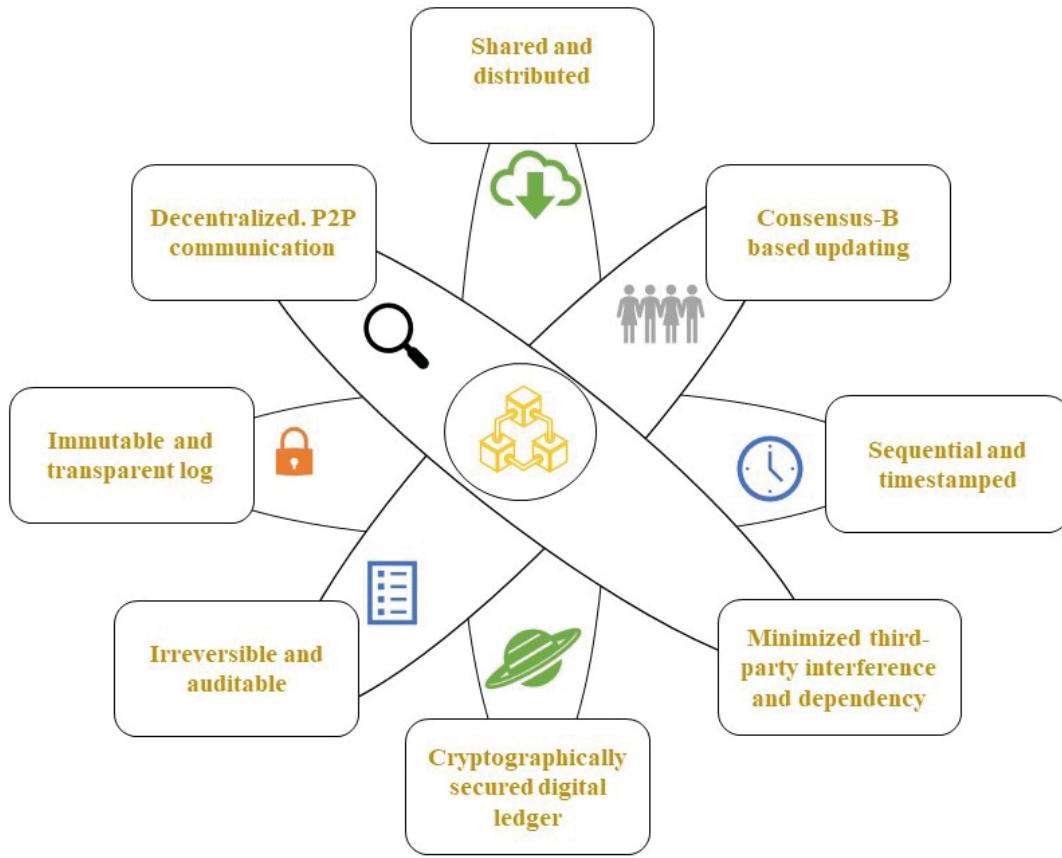


Figure 3.1: The vital blockchain characteristic [4]

where information on transaction details is shared and verified by p2p network participants. The transaction details verified by the network's consensus mechanism can be added to existing blocks and once added into the chain, the block cannot be modified. Because all transactions within a blockchain system are validated and recorded by the consensus of the network nodes, the need for a trusted central entity is eliminated (Rennoek, Cohn, and Butcher, 2018). Bitcoin is a case of the successful application of BC, which is the first global decentralized. Figure 3.1 above shows the vital blockchain characteristics which contain all the mechanisms and contributors in a blockchain.

In a decentralized marketplace, there is no single entity that takes the responsibility for the proper operation of the marketplace. As a result, all guarantees must come to the technical platform. Not only does it need to be properly designed, but it also needs to be properly implemented, deployed, and monitored. Moreover, all these

guarantees need to be verifiable by any of the members of the marketplace, at any time. Finally, as marketplaces store sensitive data that should not be accessible by all the members of the marketplace, data privacy naturally becomes a critical feature. This makes it possible for the data owners to retain control of their data i.e., where it is stored and who has access to it. This can be used, for instance, to restrict access to the details of a contract only to the parties involved [5]. Figures 3.2 and 3.3 are showing the structure of the blockchain algorithm.

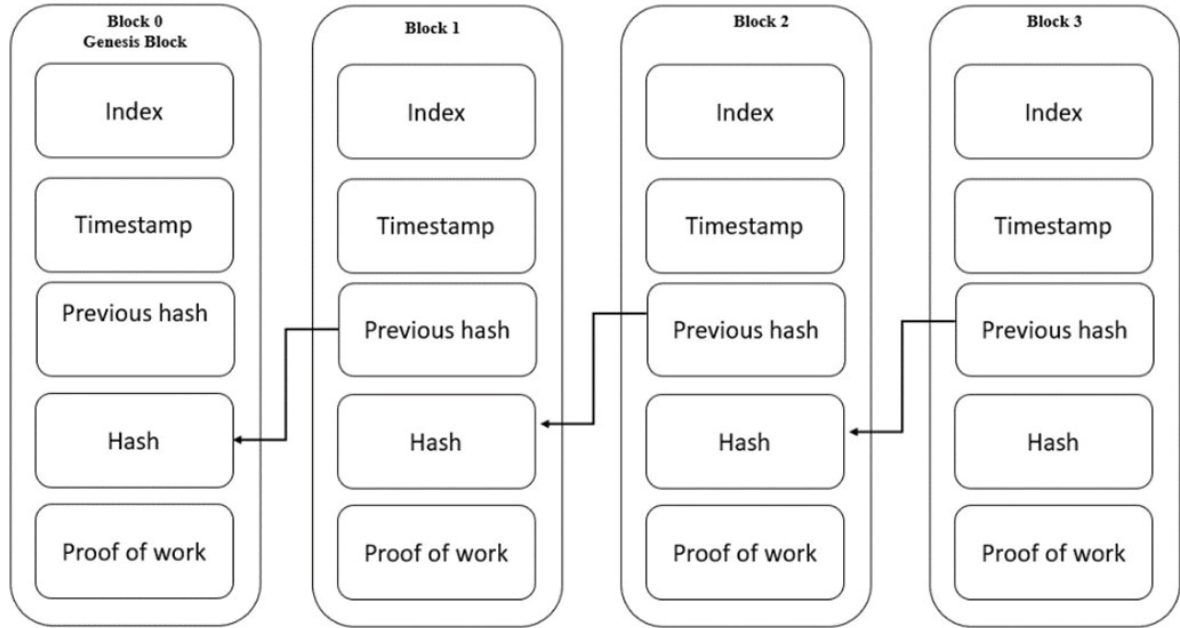


Figure 3.2: Blockchain diagram [4]

For source authentication and identification, each transaction is digitally signed by the owner with a private key. To keep track of transactions occurring simultaneously, multiple transactions are grouped together in a structure called a block, which is uniquely identified by its digital fingerprints (hash) and timestamp. The blocks are immutable cause they always use the previous hash to secure data. The data validation of transactions and the block among potentially distrusted users are done using a consensus mechanism, which means that the state of the shared ledger is updated by the agreement or consensus of the majority of nodes. This updating employs the proof-of-work consensus algorithm, whereby nodes strive to find a special value to achieve the block's hash, less than a target value, which is usually set to avoid any conflicts and establish trust. This target value is set in such a way that nodes compete to find a unique, one-time number (called a nonce) in approximately

10 min—hence, the block generation time is 10 min. This process by which nodes perform rigorous computations, thus devoting their resources (such as their central processing unit and electricity) to find the nonce, is called mining, and the nodes doing so are called miners. Through mining, nodes compute the proof of work, which is a form of achieving consensus among the distrusted nodes [41].

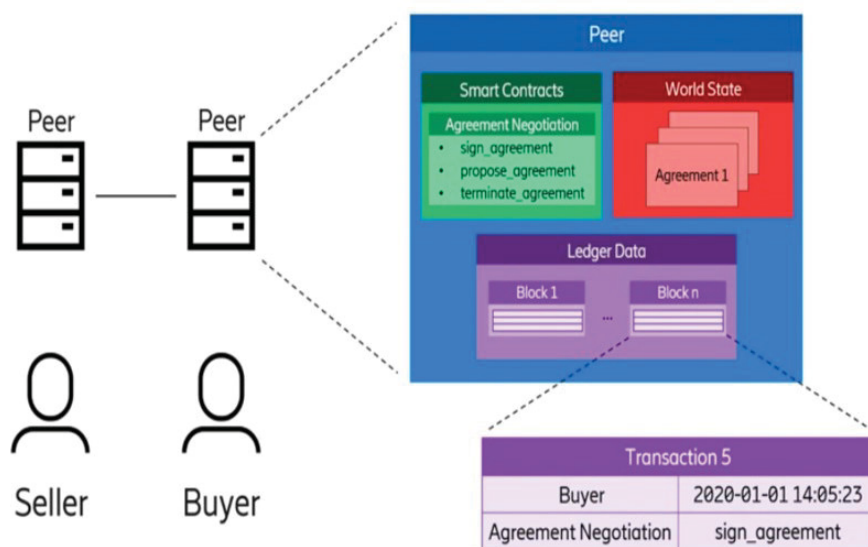


Figure 3.3: Blockchain diagram [5]

In the physical world, trust is intangible, but it is nonetheless central to our interactions with other people and to our consumption of services. Creating an online environment in which people feel secure when interacting and consuming in a similar way requires the development of technologies and protocols that formalize and digitize trust. The current solution to the challenge of facilitating trust online is to rely on trusted third parties such as banks and major internet companies to act as trust anchors, creating and attesting certificates for people or web-based services. Each device, browser and operating system come preconfigured with a list of these trusted third parties and their certificates – their digital fingerprints. By instructing our devices to trust the root certificate of the trusted third party, they can computationally infer trust in all underlying entities [44].

Genesis Block

In a blockchain network, a block is an ordered collection of transactions. It is cryptographically connected to the previous block, which in turn is linked to the next block. The genesis block is the initial block in such a chain of blocks and it does not have a parent block [6]. The ordering service generates blocks, which are subsequently validated and committed by peers. In summary, there is no other transaction before genesis block in a blockchain network. In a Hyperledger Fabric network, the genesis block is created once the first transaction occurs. The blockchain diagram shown in figure 3.2 illustrates the structure of a blockchain as well as block zero which is counted as the genesis block.

3.2 Peer-to-Peer Network

Peer-to-Peer (P2P) is a decentralized network communications architecture that consists of a number of devices (nodes) that jointly store and distribute data, with each node acting as a single peer. P2P communication takes place in this network without the need for a central administration or server, which means that all nodes have the same power and do the same duties [1]. Figure 3.4 indicates a simple p2p network in which there is no central administration.

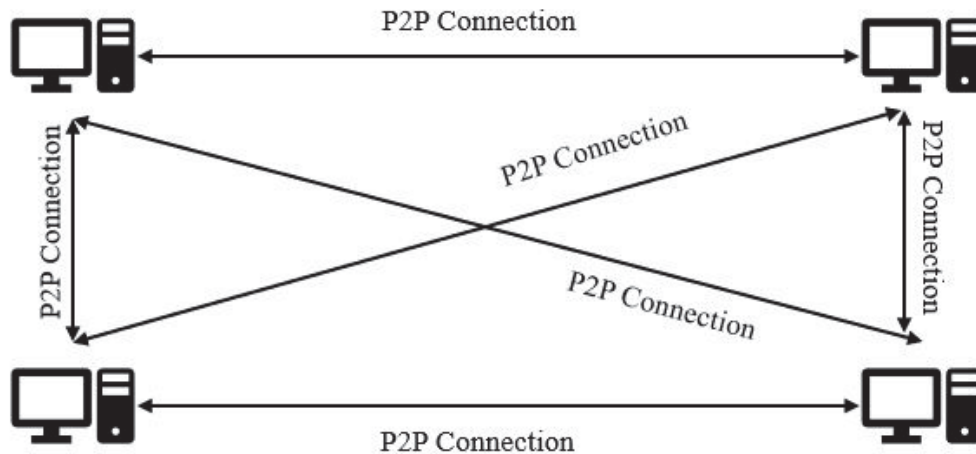


Figure 3.4: Peer-to-Peer Network [1]

3.3 Smart Contract and Chaincode

A smart contract is a blockchain component that contains all the required rules for a transaction to be completed. It is also known as a finite state machine, in which predetermined rules and instructions are carried out whenever a certain event happens. Before committing a transaction, it verifies the preset criteria [1].

The smart contract manages energy transactions between two or more parties in a smart grid energy market by following preset rules. Once a smart contract is launched on the network, its rules cannot be changed. It guarantees that the energy trading market is transparent for all players [6]. Furthermore, market participants trust these contracts with their energy and payments, removing the need for central intermediaries to oversee these transactions. Once the smart contract is signed, it is impossible to revoke or change it and it will be added to the chain to make the entire blockchain [1]. The figure 3.5 below shows the content of each block inside the blockchain in which the smart contract is added to each block.

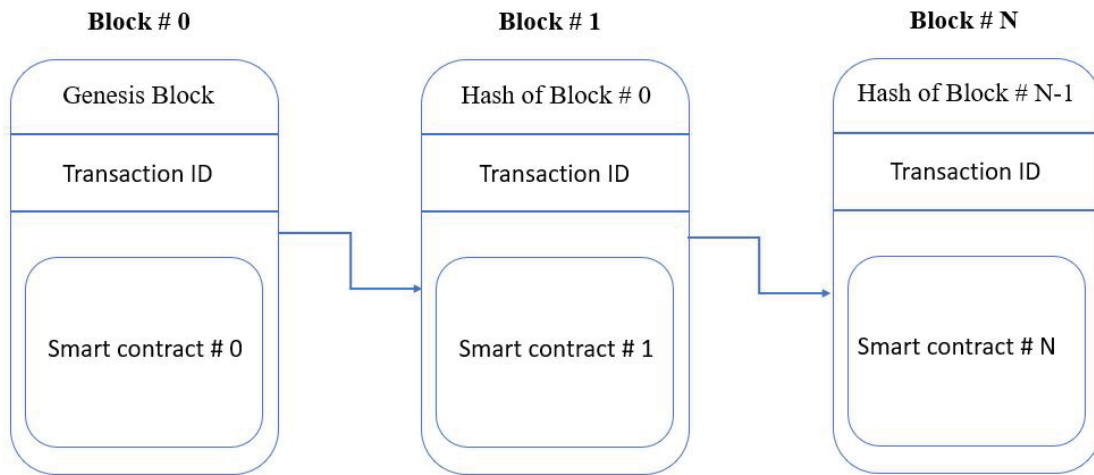


Figure 3.5: Content of each block in the blockchain [6]

In Hyperledger Fabric, term "smart contract" is known by chaincode. A chaincode is an executable code that has a specific set of rules for each transaction.

In this thesis, we employed two smart contracts for secure p2p energy trading in a smart energy market. The next subsections will go through the details of these two smart contracts.

Main Smart Contract

The main smart contract is developed to register the parties to the network. This contract will be between the participants and the network provider. The network provider is the party who is responsible to provide the entire infrastructure for p2p energy trading. The network provider is also responsible for the maintenance of the network. Furthermore, the main smart contract is also defined as the rules for any communication and transaction between the parties and the network provider [1].

In case a transaction wants to be set between two or more parties in a different region, the main smart contract will handle the entire process to register the parties to their local Energy Management System (EMS) and prepare the transaction and energy trading between two or more different regions. This kind of transaction is only happening if a region wants to trade energy with another region. The complete explanation for the terms EMS, Energy Storage System (ESS), and Main Energy Management System (MEMS) is provided in chapter four of this study.

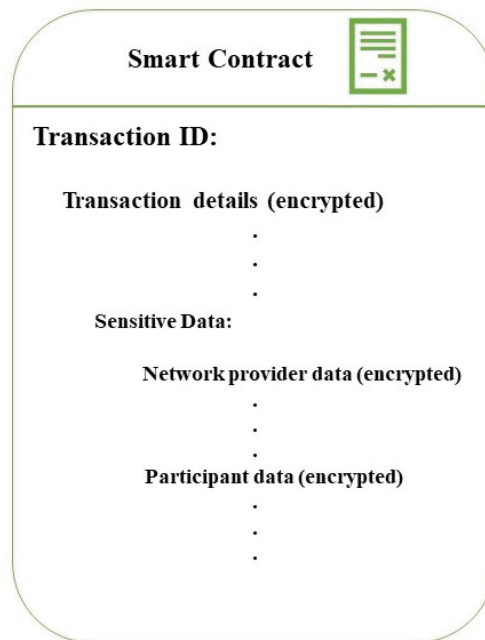


Figure 3.6: Main smart contract [1]

The main smart contract can control all the transactions and operations of energy trading in the local energy market. It verifies the user's identity before allowing them to trade. When a market participant sends a sell or buy request to its local EMS, the EMS authenticates the user by checking the validity of the user if it is

registered in the market or not. If a participant is registered with the required information, the main smart contract will give further permission to the participant to trade securely. The main smart contract is also stored all the transactions within the local area [1]. Figure 3.6 depicts some important data that should be stored in the main smart contract.

To start energy trading securely between two participants, a p2p smart contract should be signed between seller and buyer in the network which we discussed in the next part.

P2P Smart Contract

In a p2p energy trading network, the whole trading mechanism between two participants is controlled by the p2p smart contract. If the main smart contract is not added to the chain, market players cannot do any transaction in the network. The essential data from the main contract and information on energy consumers and prosumers are received from the main smart contract and will use it in p2p smart contract. This smart contract's seller function keeps the seller's relevant data, while the buyer function stores the buyer's data to identify them in the network. Once both the parties in the network sign the p2p smart contract, it will be added to the chain, and it is not possible to change [1].

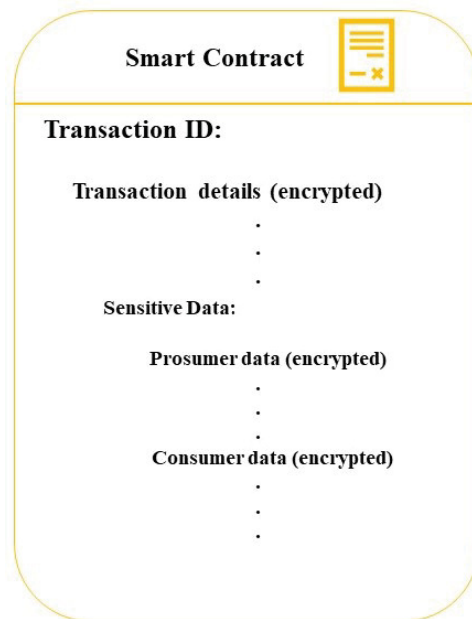


Figure 3.7: P2P smart contract [1]

It is always preferable to trade with participants from the same region. Trade

between nearby regions is recommended when trading inside the same region is not possible, for example, all the participants in a region are consumers due to high energy consumption in this region. The entire market is cleared in this manner, and the transaction is transmitted back to the main smart contract [1]. Figure 3.7 shows the p2p smart contract and some of the key features of this contract.

3.4 Blockchain Platforms

One of the major problems that individual companies who are seeking to use blockchain to make transactions between entities are the lack of standards. However, not all platforms are appropriate. In this section, we compare different blockchain platforms to select the most suitable platform for secure p2p energy trading using blockchain [45].

Hyperledger Fabric, Ethereum, Bitcoin, MultiChain, and Quorum are the five most popular platforms now in use. The authors are describing five available platforms to select the best platform for the use case of p2p energy trading using blockchain.

Hyperledger Fabric: Hyperledger Fabric is an open-source blockchain technology designed for corporate use and created by the Linux Foundation. Hyperledger Fabric allows smart contracts to be created using general-purpose scripting languages like Go, Java, and Node.js rather than restricted domain-specific languages (DSL). Hyperledger Fabric is classified under private permissioned distributed ledger [46].

Ethereum: Ethereum is a blockchain platform that has had a significant impact on blockchain technology's growth in recent years. Ethereum has established itself as the most well-known platform for smart contract support, thanks to a built-in programming language called Solidity. The ease with which Ethereum smart contracts can be created allows blockchain technology to be used not only for cryptocurrencies but also for a variety of other applications [46].

Bitcoin: Bitcoin is the first and most widely used blockchain platform, providing a secure, low-cost, and quick way to conduct digital financial transactions without the need for a central bank. Using a programming language, Bitcoin allows for the creation of smart contracts. Bitcoin, on the other hand, is an unsuitable candidate for creating smart contracts due to its programming language limitations [46].

MultiChain: MultiChain is a blockchain platform that allows users to quickly build up private blockchains in businesses. It provides a command-line interface for network interaction and uses a basic API to expand the fundamental capabilities of the Bitcoin API. Through the JSON-RPC (remote procedure call protocol) API,

MultiChain allows multiple clients such as Go, Java, Node.js, PHP, Python, and Ruby to communicate with the network [46].

Quorum: Quorum is an Ethereum-based blockchain platform that was created to make developing Ethereum blockchain applications in businesses easier. Quorum is an excellent choice for situations where transaction processing speed and throughput are critical. Quorum has the same functionality as Ethereum. It does, however, include a few changes, such as network and peer permissions management, better transaction and contract privacy, voting-based consensus methods, and increased speed [46].

3.5 Security and Privacy

As blockchain works in a distributed manner it ensures the security of the decentralization of the smart grid and resources over the p2p network. The decentralization guarantees various marks of resumption against security assaults [7].

The blockchain ensures security by a consensus process, hashing, and decentralized storage. It is very difficult to insert fake data as the blockchain uses the consensus of nodes that consist of fifty-one percent of the network [14].

Each block of the blockchain is dependent on the previous block and uses the hash key of the previous block to add a new block to the network that makes traceability possible. It also uses the electronic contracts known as smart contracts which are self-executable computer programs that automatically induce terms and conditions between two or more parties for establishing the communication and transactions [14].

Chapter 4

System Design and Modeling

This chapter explains and sets the requirements for the implementation of blockchain technology in a p2p energy trading system in a theoretical context. Three different models have been discussed in this chapter. Considering their advantages and disadvantages we have selected our system model where we will implement blockchain technology. The system models carry information about the structure and the behavior of that system to the reader.

4.1 System Model

Based on goals, creating a blockchain-based p2p energy trading needs to define a set of technical design and rules. There are many studies worked on system modeling and designing the p2p energy trading over blockchain. Most papers are worked on the positioning of the Energy Storage System (ESS) in the network. Although the main goal of the blockchain system is to decentralize the system as much as possible, there are some reasonable reasons provided by different researchers to have a main ESS for implementing p2p energy trading.

In addition to having sustainable energy resources, prosumers prefer to have stable power distribution lines connected to an existing utility company to make sure that they have enough energy when all the nodes are in their peak consumption.

In all three models, the network provider is a party who provided all the infrastructures to enable safe and secure p2p energy trading. the network provider is responsible for all the maintenance as well. All the parties can trust it using a predefined smart contract at the time of joining the network.

In this thesis, the authors decided to elaborate on three different system models that focused on positioning the ESS in the network. In the end, the authors expressed their purposed model based on the merits and demerits of each model considering the security and scalability of each model. In the next subsections, three different system models are discussed.

Model 1 - Individual Energy Storage System

Based on [47, 48], in a decentralized energy storage system, smart homes are equipped with their own energy storage system and renewable power generator. In this model, the surplus generated energy from each node is stored in its own ESS. Meanwhile, the status of ESS will send to the Energy Management System (EMS) using a smart energy meter and the extra energy will be ready for sale. If the produced energy is less than its nodes demand, the smart meter will communicate with EMS to purchase required energy from other prosumers. In a decentralized energy storage system, each node is responsible for the maintenance of its own ESS system.

In the decentralized storage system, although the nodes have a considerable number of options for selecting the amount of their own ESS, due to the price of having a big storage system, the scalability of ESS might not be flexible. In terms of security, a decentralized ESS has enough security and safety. The figure below gives a clear vision of such a model where each node hosts its own ESS.

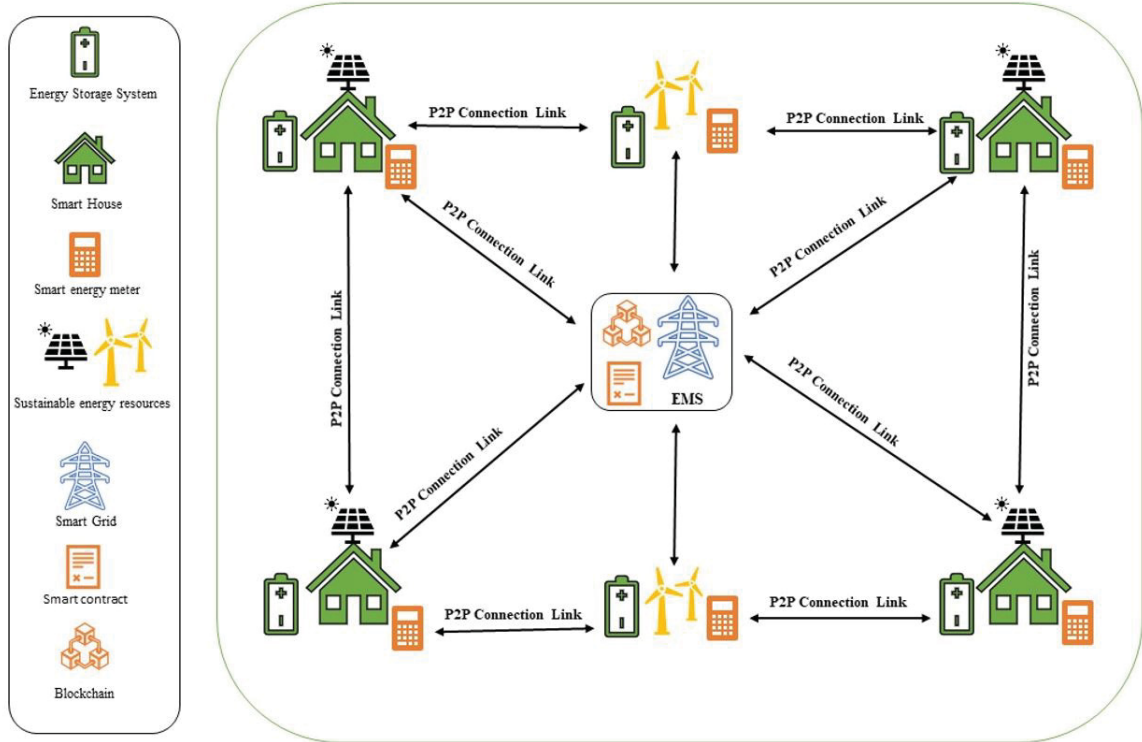


Figure 4.1: Individual energy storage system

Model 2 - Large Scale Energy Storage System

This market design applies rules to a system with a single large ESS that is owned by the network provider. In contrast to model 1, this model has only one large storage entity that is strategically situated and controlled by the community. The battery may only be charged using renewable energy generated by prosumers in the community, and this will be compensated. At a little higher rate than the charging compensation rate, discharge is available to everyone – producers and consumers [49].

According to [50], the storage systems are belonging to the service provider who is responsible for the maintenance and development of the network. each area can be divided into smaller regions where each region hosts a large energy storage system. All the participants in a region are connected to their local storage system via a secure p2p connection link and all the regions are connected to a Main Energy Management System (MEMS). The role of having a MEMS is to have redundancy for the whole network. For example, if a storage system in a region failed, the EMS of that region sends a request to MEMS for purchasing energy from other regions, by this way, a power outage for the end-users will never occur or will be in the minimum state. A clear example of a large-scale ESS model is shown below.

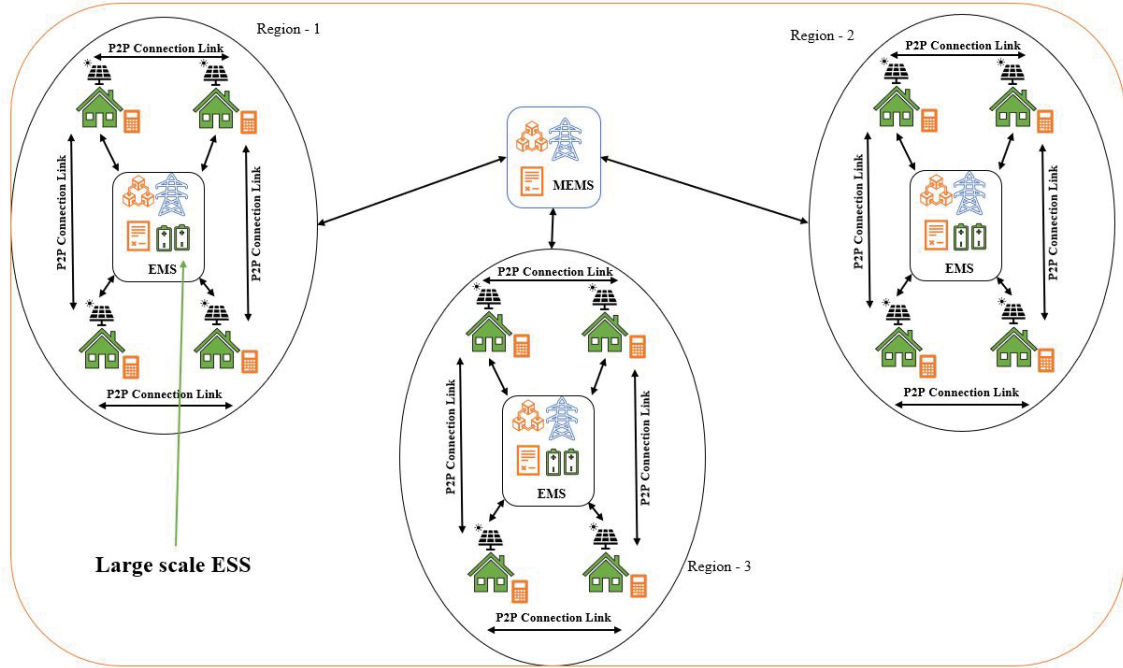


Figure 4.2: Large scale energy storage system

Model 3 - NIST Conceptual Model

Under the NIST conceptual model [7], actors, applications, and concepts are divided into seven domains and sub-domains. customers, marketplaces, service providers, operations, generation, transmission, and dispersion are examples of these areas. Actors, applications, and concepts are divided into seven domains and sub-domains according to the NIST's conceptual model.

The smart grid's consumers domain is made up of the smart grid's ultimate power users. Participants in the electrical market make up the market domains. Customers, prosumers, and utilities use cloud and edge computing services provided by service providers. The operations domain is made up of independent system operators that create smart grid apps that make use of edge and cloud computing services and run various revenue models. The generators, carriers, and distributors of electricity in bulk amounts are the bulk generation, transmission, and distribution domains, respectively.

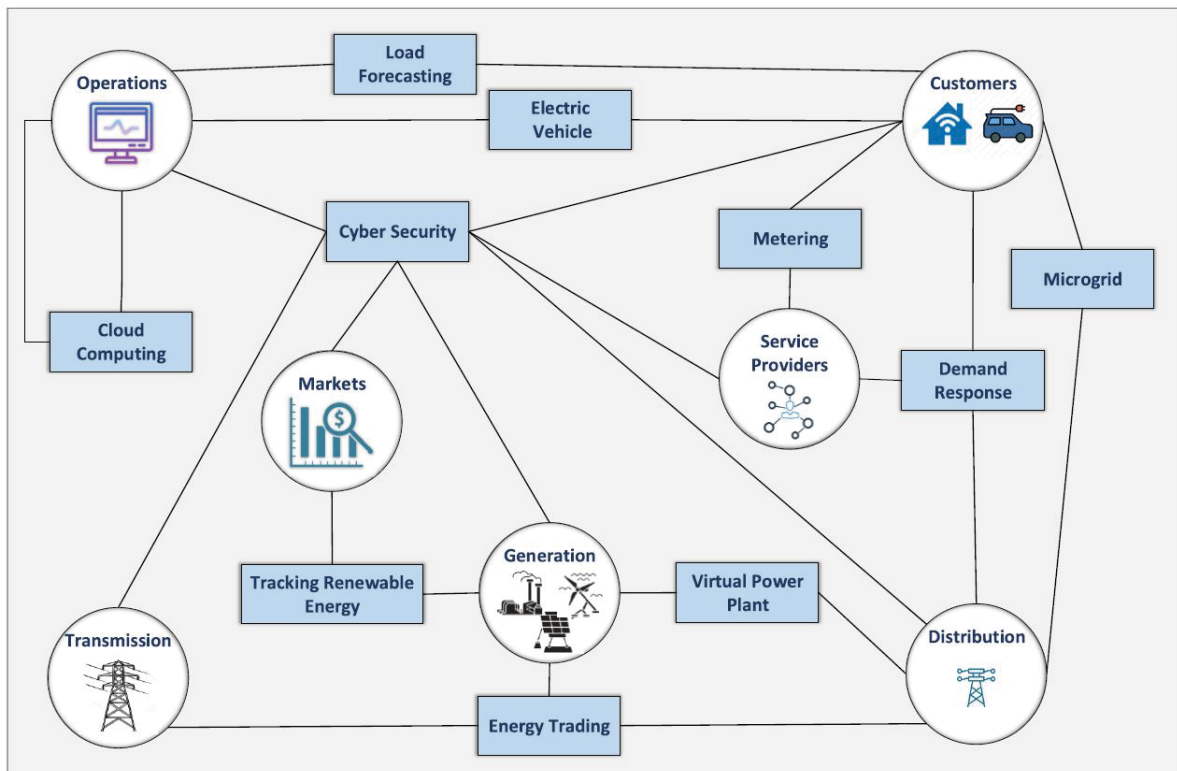


Figure 4.3: NIST smart grid conceptual model [7]

Deregulation of the electrical power infrastructure has resulted in the global decentralization of electricity markets. As a consequence, customers may purchase energy at lower costs on the spot and in future markets. However, energy trading system designs differ due to the wide range of energy generating systems, which can range from residential distributed energy resources (DER) to commercial-grade power plants. As a result, today's energy trading systems employ a variety of centralized and decentralized trading techniques [7].

The NIST smart grid domains and their relationship with the identified sub-domains for prospective inclusion of decentralized applications are depicted in the figure 4.3.

4.1.1 Proposed Model

The authors selected their proposed model without any prejudice on a specific model and tried to do a fully scientific survey on the advantages and disadvantages of different system designs.

In this thesis, three different system model (two technical model and one conceptual model) is discussed in which all the three are structurally and technically are implementable. After having a review of different studies, the authors decided to choose model-2 (large scale energy storage system). Although model-1 has its own merits, the scalability of model-2 is the main reason for choosing this model. In the selected model, the network provider is able to develop the capacity of ESS at any time. Therefore, it is possible to increase the capacity of the storage systems in all regions very quickly.

The main purpose of having a blockchain p2p energy trading system is nothing but security and safety. Since all the nodes can only take advantage of the system just after the verification and authentication process mentioned in section 4.2.2, both the models are fully secured under the blockchain. The whole implementation process is dependent on the model chosen.

4.2 System Architecture

In this part, we will go through the system or network architecture necessary to set up a completely decentralized p2p energy trading system based on blockchain. Several hardware and software technologies will be used to enable such trading in a genuine peer-to-peer energy trading network. The following discussion will elaborate on the most critical requirements for this network in a highly secure blockchain network.

System Requirements

To implement the p2p energy trading scenario using blockchain, some technologies and infrastructures are required. Here, we will only express the most essential requirements. It means all these items must be available for both running and joining a p2p energy trading using blockchain.

- **Smart node:** A node in a p2p network can use p2p communication to access information on the network. In a p2p energy trading network, a node could be either an energy consumer or a prosumer who owns a copy of the blockchain ledger.
- **P2P link connection:** In a p2p network, a group of nodes is linked together with equal permission and responsibilities for trading energy using a p2p link connector.
- **Renewable power generators:** Various renewable electricity generators can be used to produce energy in a p2p energy trading market. The most common use of such generators is solar photo-voltaic panels and wind turbines.
- **Smart energy meter:** A smart meter is an electronic device that keeps track of data like electric energy consumed and/or produced, voltage levels, current, and power factor. In such a p2p energy trading network, the smart meter is responsible for making all the transactions and storing a ledger of blockchain.
- **Energy storage system (ESS):** The energy storage system is a device that stores electric energy for later use. In this scenario, the battery is used to store energy.
- **Energy management system (EMS):** An energy management system is a computer-based tool that power system operators use to monitor, regulate, and manage energy efficiently. In our case, EMS is just offering a safe p2p energy trade transaction utilizing blockchain.
- **Application programming interface (API):** API allows blockchain nodes to communicate with one another. API is implemented on EMS and smart meters in this case.

4.2.1 Secure Market Participation

Participants who are actively consume and produce energy in the network are required for the deployment of p2p energy trading using blockchain. Although trading

energy is not a new concept in the twenty-first century, security has always been one of the most important concerns for any market.

The most important aspect of any secure trade is that both the seller and the buyer are aware of the parties involved and they are not malicious. As a result, the way that the members join the network is important since they must be known to one another. To continue, we will explain how the members may join the network safely.

Each network participant has a public and private key pair in the p2p energy trading network. Participants' public keys are exchanged with other network users, and a user cannot mathematically guess another user's secret key using his public key. Participants' public keys are utilized to produce unique addresses using a hash function. For trading on the network, these unique addresses are used to identify them. Their true identities are kept hidden in this way. As a result, malicious nodes are unable to access the sensitive data of legitimate network members, ensuring privacy [1].

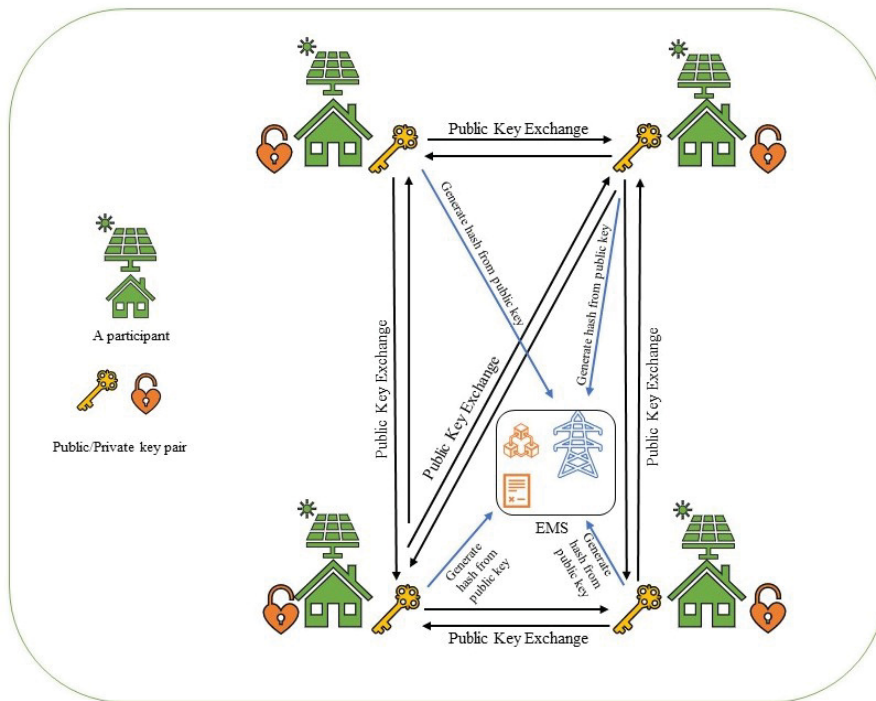


Figure 4.4: Market participation [1]

Once the public key is shared with participants inside the network, a smart contract contains predefined rules between participants, and the network provider will sign, and a transaction will store in the database. The participant public key will use

for signing and encrypt the confidential data inside the smart contract. Once the smart contract is settled and approved by the majority of members, it will add to the chain and all participants will have a copy of the entire chain. The header of the current block contains the hash of the previous block, in this way the whole chain of blocks is maintained. These unique addresses are stored in the energy management system (EMS) and used to make the block of blockchains and also to identify them while trading on the network. The key condition for joining the blockchain-based p2p energy trading network is that the member does not have to be an absolute consumer. It means that at least one renewable energy generator must be installed along with all the smart devices provided by the network provider. A graphical picture shows the key exchange between parties and the way that a new participant wants to join the network securely in figure 4.4.

4.2.2 Blockchain and Smart Contract in P2P Energy Trading

The purpose of blockchain energy trading is not only for sustainability but also for increasing the privacy and security of trading energy in the market. Therefore, taking the advantage of blockchain and smart contracts, third-party supervisors are no longer required.

When a prosumer has some surplus energy, it sends a sell request message to EMS using its smart meter device for selling its surplus energy. This communication will encrypt using the prosumer public key. In reply to this request, a private message from EMS will send to the prosumer to verify the prosumer ownership. Once the validity of the prosumer in the network is confirmed by the EMS, the prosumer can transfer its extra energy to ESS. Meanwhile, the status of the EMS panel will be updated, and the energy will be ready to be sold in the market. The amount of energy transferred to the EMS will store in the database [51].

Once a consumer sends a buy request to the EMS, the EMS will do the same security procedure did for the prosumer to authenticate the consumer in the network.

Since the authentication is completed for both the consumer and prosumer, they can deal with predefined rules and prices in the smart contracts. Once the smart contract is signed using both prosumer and consumer keys, it cannot be changed and will be added to the chain and store in the database.

Chapter 5

Implementation and Experiment

In this chapter, we illustrated the experiment utilizing several technologies to implement p2p energy trade using blockchain, which addresses RQ-3 of this study. We tried to implement a simple p2p energy trading network that tries to communicate securely under the Hyperledger Fabric blockchain platform in a remote Ubuntu machine. Subsequently, we have gone through the technology that has been used in this experiment as well as a brief description of each technology. The chapter begins with implementation requirements and implementation description followed by a sequence diagram of the system, Hyperledger Fabric transaction life cycle, energy trading implementation algorithm, and finally concluded with the implementation result.

5.1 Implementation Requirements

This section provides a comprehensive knowledge of p2p energy trading needs for Hyperledger Fabric blockchain. For example, certain features, virtual machines (ideally on the cloud system), distributed databases, programming languages, containers, and virtualization systems. Below, we detailed precisely how all technologies for a Hyperledger Fabric blockchain energy trade are being employed in this thesis to demonstrate our approach.

Prerequisites Features

Some qualities are required for virtual machines to replicate the intended environment. Table 5.1 below lists the prerequisites that must be installed on virtual machines (VMs) in order to execute the simulation used in this thesis. A git repository was provided for downloading all the general prerequisites commands. A '.md' file contains all the commands that need to be executed. You need to make sure that you are downloading the version specified in table 5.1 below.

Git <https://github.com/msnband/Thesis-prerequisite-Hyperledger-Fabric>

Features	Version
curl	7.68.0
Docker	20.10.7
Docker-compose	1.25.0
go	go1.13.8
nodejs	v10.19.0
node	v10.19.0
npm	6.14.4
pip	20.0.2
python	2.7.18
Ansible	2.9.6
Hyperledger fabric	2.2
consensus type	solo
Distributed database	CouchDB

Table 5.1: Prerequisites features

Cloud Computing

Cloud computing service that allows end-users to access a wide range of network and application services. For example, offering virtual computers, storage, and a variety of other cloud services.

In this thesis, due to the availability and accessibility of Microsoft Azure for BTH students, we decided to use this service for the implementation environment. To enable VMs to implement the network, all the requirements mentioned in table 5.1 should be satisfied.

In the table 5.2, the specification of the VM used in this experiment is shown.

Operating system	RAM (GiB)	Storage HDD (GiB)	Accessibility
Linux (Ubuntu Server)	16	30	SSH

Table 5.2: Virtual machines specifications

State Database - CouchDB

There are two types of peer state databases supported by Fabric. The LevelDB database is the peer node's default state database. LevelDB only allows key, key range, and composite key searches and stores chaincode data as basic key-value pairs.

CouchDB is an optional alternative state database that lets you represent data on the ledger as JSON and run sophisticated searches against data values rather than keys. CouchDB support also allows you to install indexes with your chaincode to improve query efficiency and query big datasets [52].

To make use of CouchDB's advantages, such as content-based JSON queries, the data must be formatted as JSON. Before setting up the network, the network manager must select whether to use LevelDB or CouchDB. Due to data compatibility problems, switching a peer from LevelDB to CouchDB is not supported in Hyperledger Fabric. Therefore, all network participants must use the same database type [52].

Programming Language

To simulate a true peer-to-peer energy trading network in a virtual environment, the authors decided to use multiple programming languages in which they can implement blockchain more easily. Therefore, various programming language is used to satisfy the requirements.

To install and implement the Hyperledger Fabric environment, we used the test case provided by, IBM, Hyperledger, and Linux foundation. In order to connect all the codes and software, a Shell script is used. For example, a Shell script to Dockerize the network. Nodejs is used to add the blocks to make the entire blockchain and also communicate with the database. Go-language is used as the language for smart contract deployment and Ansible automation tool is used to automate the configurations. Finally, a Python code is used to apply energy trading algorithm, automate and connect the entire scenario. Although, there are other powerful programming languages that can be used for this scenario, the ability of authors in these programming languages is the reason to use them.

Virtualization and Containerization

To build a peer in a network virtually and add all the required components to it, container orchestration is needed. Docker and Docker-compose are used to create such peers in the network. Taking the advantage of containers, users can make a large number of peers in the network to observe the behavior of the network and each node in a blockchain network. Docker is primarily concerned with a single container or image, however, docker-compose can handle multiple docker containers.

Docker: In simple word, Docker is a container virtualization technology. It is similar to an extremely light virtual machine. It is a collection of platform-as-a-service

solutions that provide applications in containers via OS-level virtualization.

Docker-Compose: Compose is a Docker application that allows you to define and execute multi-container Docker applications. To configure our network with Compose, we utilized a YAML file, then we may use a single command to build and execute numerous container images.

5.2 Implementation Description

To implement a p2p network under Hyperledger Fabric blockchain, some steps should be followed. In this thesis, we used a test case environment provided by Hyperledger and Linux foundation. We organized the test environment to implement a network consisting of three peers authorized by an authenticate organization in a docker image. We enabled a p2p connection between peers by allocating the same IP (localhost) but in a different port. We also enabled CouchDB as a state database for each node to record the transactions. In the final, we have written a pre-defined rule to apply and enable energy trading in the Hyperledger Fabric network using the smart contract. Each node has its own ID (docker image ID) for making transactions in the network using the smart contract. Docker composed is used to define and run multi-container Docker images. For example, each docker image is a peer and a YAML file is used to make and set up the configurations for each peer.

In addition, an Application Programming Interface (API) written in Nodejs is defined to see the scalability and the capacity of the network. Using this API, we are able to run N number of energy trading transactions in the network and store all the transactions in the CouchDB. The entire implementation is publicly available in a git repository addressed in this chapter below.

5.2.1 Sequence Diagram of the System

The Hyperledger Fabric is a permissioned blockchain technology with several unique characteristics that make it ideal for organization-class applications. It can execute smart contracts (chaincode) written in Go, Java, or Node.js. To mention a few features, it includes a pluggable consensus protocol and an application-specific trust model for transaction validation [8].

A Fabric network is made up of many entities, such as peer nodes, ordering service nodes, and clients, all of which belong to different organizations. A membership service provider, for example, the network infrastructure provider, provides each of

these with a network identity. All entities in the network have access to and may verify the identities of all organizations. In figure 5.1, we showed how peers interact together in a Fabric network. In the following subsections, we described all the Fabric key components concisely [8].

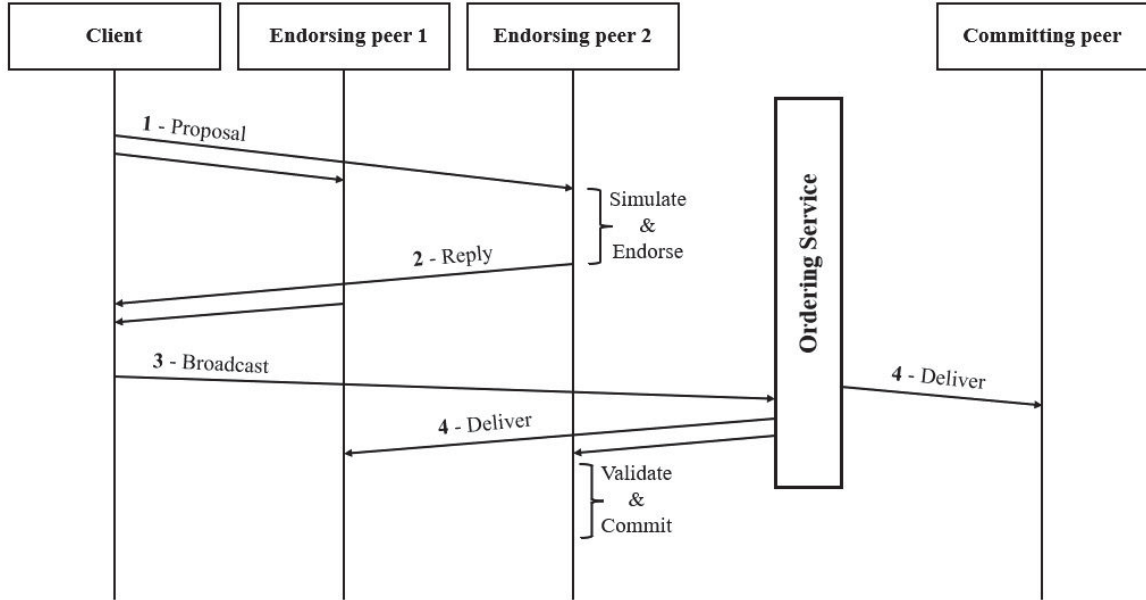


Figure 5.1: Hyperledger Fabric transaction life cycle [8]

Types of Peers In Hyperledger

In a Hyperledger Fabric p2p network, while all peers functionally behave the same, they can assume multiple roles depending on how the network is configured [53]. In below, we described two types of available peers in a Fabric network.

- **Committing peer:** In a channel, every peer node is a committed peer. It accepts blocks of produced transactions, which are then verified before being committed as an append operation to the peer node's copy of the ledger [53].
- **Endorsing peer:** Every peer with a smart contract can be an endorsing peer if one is deployed. However, in order to be an endorsing peer, a client application must employ the smart contract on the peer to create a digitally

signed transaction response. The phrase endorsing peer refers to this reality explicitly. A smart contract endorsement policy specifies which organizations' peers must digitally sign a produced transaction before it can be accepted onto a committed peer's copy of the ledger [53].

Fabric Key Components

- **Channel:** Fabric provides the idea of channel, which functions as a "private" subnet of communication between two or more peers to enable isolation. Only the peer members and participants in a channel may view the transactions. Each channel has its own immutable ledger and chaincodes [8].
- **Peer:** A peer node maintains the ledger of a system by executing a chaincode, which implements a user smart contract. A well-defined ledger API provides chaincode access to the shared state. A peer can be classified as either an endorsing peer, which possesses the chaincode logic and executes it to approve a transaction, or a committing peer, which does not [8].
- **Ordering Service:** An ordering service node participates in the consensus process by cutting blocks of transactions that are given to peers using a gossip communication channel [8].
- **Client:** The client application is responsible to create a transaction proposal. The client sends the transaction proposal to one or more peers at the same time in order to gather proposal replies, in this case, for example, selling or purchasing energy from the network. The transaction is then published to the order to be included in a block and sent to all peers for validation and commitment [8].

5.2.2 Fabric Transaction Life Cycle

Unlike other blockchain networks, the Fabric uses a simulate order-validate and commit transaction mechanism instead of an order execute transaction architecture [8]. The transaction flow is depicted in figure 5.1, which consists of three steps:

First, endorsement phase - which involves implementing the transaction on a subset of peers and capturing state changes.

Second, ordering phase - here is where the transactions are ordered using a consensus process.

Third, validation phase - validation and commitment to the ledger. The network must be booted in conjunction with participating organizations, their membership service provider and peer identities before transactions may be submitted on Fabric [8].

1. **Endorsement phase:** Initially, the client signs the proposal with his or her credentials and sends it to one or more endorsing colleagues at the same time. The chaincode's endorsement policy specifies which organization peers and client must submit the proposal for simulation. Each endorsing peer first validates that the applicant is entitled to invoke on the channel transactions.

Secondly, the peer runs the chaincode that can access the current pair's ledger status. The outcomes of the transaction comprise the answer value, read-set, and writing-set. Everyone reads the actual status of the ledger, but all writings are intercepted and alter the workspace of a private transaction [8].

Third, the endorser invokes the Endorsement System Chaincode (ESCC), which signs the transaction response with the identity of the pair and answers the client with a proposal reply.

Finally, the client examines the proposal response to ensure that it carries the peer's signature. The customer gathers a sufficient number of proposal replies from various peers and double-checks that the endorsements are the same [8].

2. **Ordering phase:** The client transmits to the ordering service a well-formed transaction message. The transaction includes read-write sets, peer signatures, and the Channel ID. For the functioning of the transaction, the ordering service does not have to check the contents. It accepts transactions from multiple customers on different channels and sticks them to each channel. It produces transactions blocks per channel, signs the block, and sends it to peers via a gossip message protocol [8].
3. **Validation phase:** The network sends blocks to all peers on a channel, both endorsing and committing peers. The peer checks the orderer's signature on the block first. Each valid block is decoded, and all transactions within a block are validated using Validation System Chaincode (VSCC) before moving on to Multi-Version Concurrency Control (MVCC) [8].

VSCC Validation: A validation system chaincode compares transaction endorsements to the endorsement policy defined for the chaincode. The transac-

tion is deemed void if the endorsement policy is not met [8].

MVSCC Validation: The multi-version concurrency Control check guarantees that the versions of keys read during the endorsement phase by a transaction match their present state in the local ledger at commit time. This is comparable to a read-write conflict check used for concurrency management, and it is carried out sequentially on all legal transactions in the block (as marked by VSCC validation). The transaction is declared invalid if the read-set versions do not match, indicating that a concurrent preceding (as in earlier in this block or before) transaction changed the data read and was successfully committed since its endorsement [8].

4. **Ledger update phase:** The ledger is updated by adding the block to the local ledger as the last stage of transaction processing. With the write-sets of valid transactions, the CouchDB, which stores the current state of all keys, is updated (as marked by MVCC validation). These CouchDB changes have been completed atomically for a block of transactions, and the updates have been applied to return the CouchDB back to the state after all transactions in the block have been processed [8].

In the following part, we provided figure 5.2 to show all the Hyperledger Fabric network processes. Building a network, for example, consists of adding peers, creating channels between peers, and executing a smart contract (chaincode) between peers. A clear Hyperledger Fabric work cycle diagram will also demonstrate how clients (peers) interact together using all the technologies we explained at the beginning of this chapter.

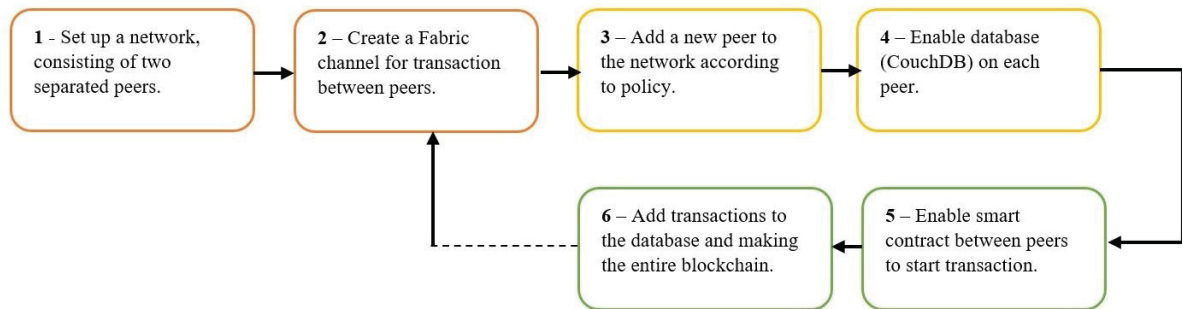


Figure 5.2: Stepwise to implement a Fabric network

5.2.3 Energy Trading Implementation Algorithm

Since reaching an agreement is a complex process in all blockchain networks, and because a new block must be validated by all the nodes before added to blockchain, the process is overly complex. The following figures show the entire energy trading algorithm implemented in the Azure cloud. In the algorithms shown in figure 5.3 and 5.4, we briefly described all the steps of the entire implementation phase. In 5.3, authors depicted all the steps for network participation, for example, crypto material generation, key exchange between peers and signing the main smart contract to register a new peer to the network. A Yaml file is defined to configure all the configurations for the peers, make crypto materials and generate public/private keys.

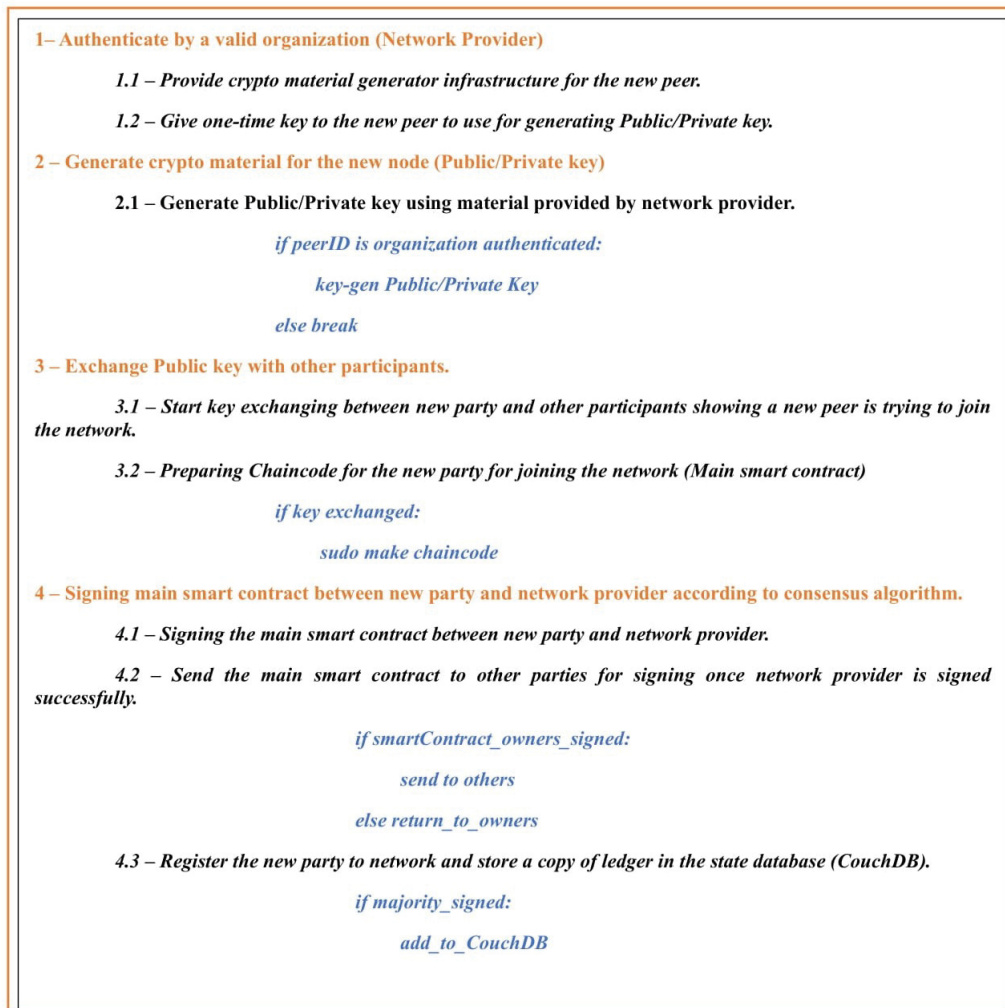


Figure 5.3: Network participation algorithm

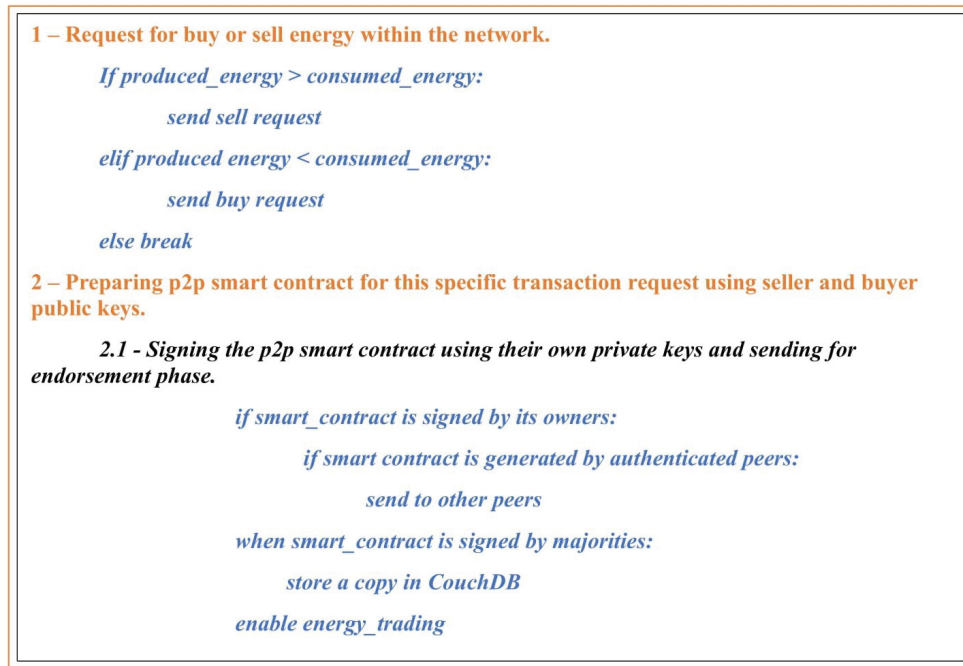


Figure 5.4: Energy trading algorithm

In the figure 5.4, the algorithm showing how one party can send sell or buy request within the network. In addition, we showed the energy trading algorithm cases along with conditions. For this step, a random generator function is used to implement the produced and consumed energy for each peer. The main smart contract is written in Go is defined for any energy transactions and to add the transaction to CouchDB. A public git repository containing the entire simulation code is provided as well. The code is written in Python and is responsible for doing all the steps of the algorithm consequently.

Git <https://github.com/msnband/ThesisEnergyTradingSimulation>

5.3 Functions and Commands Description

To implement such a blockchain network, multiple programming is conducted. The implementation environment is configured in a way to be automated as much as possible. For running and configuring several settings parallelly, a script written in Shell contains a series of commands to run sequentially to carry out some operations and running YAML files for making Docker images. There are many configurations

to set up a p2p network, but to avoid complexity, we tried to configure and implement the most important characteristics of each component in the network. To download the latest version of Hyperledger Fabric, the following command should execute:

```
$:- curl -sSL https://bit.ly/2ysbOFE | bash -s
```

Once it is downloaded successfully, our implementations should be replaced with the default implementations. In this case, should be replaced with the folder "fabric-samples". A git repository containing all the necessary codes and dependencies is provided to download and execute the commands to see the results, download and replace it with the "fabric-samples" folder.

```
Git https://github.com/ragadeep-maker/MasterThesis-P2P
```

In below, we will elaborate concisely on the most important commands and functions are used in this work.

Set-Up The Network

The first command is used to set up the network is "up". This command will call an in-built function to create a network consisting of three participants. Two peers and one orderer. The task for the orderer is already explained in this chapter. Before bringing up the network, each peer needs to generate the crypto material that will define that peer on the network. The crypto material will generate automatically during the process using three YAML files located in the "fabric-samples/test-network/organizations/cryptogenic" directory. The following commands should be executed in the directory "HyperledgerFabric_v1/HyperledgerFabric/fabric-samples/test-network". The command is used for setting up the network is below:

```
$:- ./network.sh up
```

Tip: It is important to "down" the network at the end or once you do not like to continue. This implementation will impose some changes in the network configuration of your Linux machine. Using "down" commands, will return your network configuration to the default.

```

# Bring up the peer and orderer nodes using docker compose.
function networkUp() {
    checkPrereqs
    # generate artifacts if they don't exist
    if [ ! -d "organizations/peerOrganizations" ]; then
        createOrgs
    fi

    COMPOSE_FILES="-f ${COMPOSE_FILE_BASE}"

    if [ "${DATABASE}" == "couchdb" ]; then
        COMPOSE_FILES="${COMPOSE_FILES} -f ${COMPOSE_FILE_COUCH}"
    fi

    DOCKER_SOCKET="${DOCKER_SOCKET}" docker-compose ${COMPOSE_FILES} up -d 2>&1

    docker ps -a
    if [ $? -ne 0 ]; then
        fatalln "Unable to start network"
    fi
}

```

Figure 5.5: Bring up the peer using docker compose.

Figure 5.5 showing the used function for bringing up the network using docker-compose.

Create The Channel Between Peers

The second command is calling relative functions in figure 5.6 to establish the channel between peers. The command is below:

\$:- ./network.sh createChannel

```

# call the script to create the channel, join the peers of org1 and org2,
# and then update the anchor peers for each organization
function createChannel() {
    # Bring up the network if it is not already up.

    if [ ! -d "organizations/peerOrganizations" ]; then
        infoln "Bringing up network"
        networkUp
    fi

    # now run the script that creates a channel. This script uses configtxgen once
    # to create the channel creation transaction and the anchor peer updates.
    scripts/createChannel.sh $CHANNEL_NAME $CLI_DELAY $MAX_RETRY $VERBOSE
}

```

Figure 5.6: Create the channel.

Add Database To The Peers

Since the network participants need a database for storing the transactions, a database needs to be installed on each peer. we have to re-install all the networks to enable an active database. Therefore, the following command is used to do all the setups from scratch and download and install CouchDB on each Docker image. This command is calling a YAML file shown in figure 5.7 to download, install and configure CouchDB on all the images.

```
$:- ./network.sh up createChannel -s couchdb
```

```
version: '3.7'

networks:
  test:
    name: fabric_test

services:
  couchdb0:
    container_name: couchdb0
    image: couchdb:3.1.1
    labels:
      service: hyperledger-fabric

    environment:
      - COUCHDB_USER=admin
      - COUCHDB_PASSWORD=adminpw

    ports:
      - "5984:5984"
    networks:
      - test

  peer0.org1.example.com:
    environment:
      - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
      - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb0:5984
      - CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=admin
      - CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=adminpw
    depends_on:
      - couchdb0

  couchdb1:
    container_name: couchdb1
    image: couchdb:3.1.1
    labels:
      service: hyperledger-fabric
```

Figure 5.7: CouchDB installation YAML file

Add a New Peer To The Network

So far, we have created a network consisting of two peers with a secure channel between them. The CouchDB is also installed on each peer, and they are ready to store the transactions. In this step, we are trying to add a new peer to the network according to the policy we mentioned in section 4.2.1 of chapter four in this thesis. To add the new peer, we should change the working directory to the directory where the configurations are located. In this case, we put the configurations in */HyperledgerFabric/fabric-samples/test-network/addOrg3*. The implementation code is written in a bash script and the file named *addOrg3.sh*. To add the third peer to

the same network and same channel, the following command should be executed in the mentioned path:

```
$:- ./addOrg3.sh up -c mychannel -s couchdb
```

Main Smart Contract

So far, we have created a network consisting of three members who are not allowed to communicate together using available channels and networks since they did not sign the main smart contract yet. The role of the main smart contract is described in chapter three in section 3.3 of this thesis. Now, we are trying to call the chaincode function to enable peers in the network to communicate according to the policy mentioned in section 3.3. Here is the command is used to deploy the chaincode in Go-language in the network:

```
$:- ./network.sh deployCC -ccn basic -ccp ../asset-transfer-basic/chaincode-go -ccl go
```

Down The Network

At any stage of the implementation, if we would not like to continue, we should down the network to return the network status to its default one, for example, IP configurations and some firewall rules we applied for this implementation. The command below and the function shown in figure 5.8 are provided below:

```
$:- ./network.sh down
```

```

# Tear down running network
function networkDown() {
  # stop org3 containers also in addition to org1 and org2, in case we were running sample to add org3
  DOCKER_SOCKET=$DOCKER_SOCKET docker-compose -f $COMPOSE_FILE_BASE -f $COMPOSE_FILE_COUCH -f $COMPOSE_FILE_CA down --volumes --remove-orphans
  docker-compose -f $COMPOSE_FILE_COUCH_ORG3 -f $COMPOSE_FILE_ORG3 down --volumes --remove-orphans
  # Don't remove the generated artifacts -- note, the ledgers are always removed
  if [ "$MODE" != "restart" ]; then
    # Bring down the network, deleting the volumes
    # Cleanup the chaincode containers
    clearContainers
    # Cleanup images
    removeUnwantedImages
    # remove orderer block and other channel configuration transactions and certs
    docker run --rm -v "$(pwd):/data" busybox sh -c 'cd /data && rm -rf system-genesis-block/*.block organizations/peerOrganizations organi
    ## remove fabric ca artifacts
    docker run --rm -v "$(pwd):/data" busybox sh -c 'cd /data && rm -rf organizations/fabric-ca/org1/msp organizations/fabric-ca/org1/tls-c
    docker run --rm -v "$(pwd):/data" busybox sh -c 'cd /data && rm -rf organizations/fabric-ca/org2/msp organizations/fabric-ca/org2/tls-c
    docker run --rm -v "$(pwd):/data" busybox sh -c 'cd /data && rm -rf organizations/fabric-ca/ordererOrg/msp organizations/fabric-ca/orde
    docker run --rm -v "$(pwd):/data" busybox sh -c 'cd /data && rm -rf addOrg3/fabric-ca/org3/msp addOrg3/fabric-ca/org3/tls-cert.pem addO
    # remove channel and script artifacts
    docker run --rm -v "$(pwd):/data" busybox sh -c 'cd /data && rm -rf channel-artifacts log.txt *.tar.gz'
  fi
}

```

Figure 5.8: Network down function

Note: It is strongly recommended to down the network at any time you are not going to continue. This implementation has applied some changes in the root configurations of your Linux machine network. Using the down command will return the configurations to the default.

5.4 Implementation Results

In this section, we will show the implementation results related to RQ-3. The results will provide all the steps of a p2p energy trading network over Hyperledger Fabric blockchain. Authors tried to present all the steps according to the Hyperledger-Fabric life-cycle shown in chapter five, picture 5.1 and also energy trading algorithm showed in figure 5.3 and 5.4 of this thesis. In the initial step, we suppose to set-up a p2p network with two peers and establish a secure channel between them. Then we will show how a new peer can join the network securely according to the policy we mentioned before. Finally, we will apply energy trading algorithm to enable peers to trade energy securely using Hyperledger Fabric blockchain. two smart contracts (main smart contract and p2p smart contract) written in Go-language are defined for any transaction within the network. The roles for each smart contract is described on chapter three section 3.3 of this work. The results are clearly showing that energy trading using Hyperledger Fabric blockchain is technically possible.

Set Up The Hyperledger Network

In the initial step, we are trying to bring up the network securely. The network is consisting of two separate organizations and each organization has one peer, as well as an orderer peer. The complete explanation of the orderer is explained in section five of this thesis. According to the Hyperledger official documentation [15], peers are not able to communicate without having a secure network. To create and prepare such secure network, the Hyperledger-Fabric blockchain source code is used. The figure below shows that the system is trying to make a secure network for further purposes.

```

et2599user2@ET2599VM2:/thesis/Hyperledger/fabric-samples/test-network$ ./network.sh up
Starting nodes with CLI timeout of '5' tries and CLI delay of '3' seconds and using database 'leveldb' with crypto from 'cryptogen'
LOCAL VERSION=2.3.2
DOCKER IMAGE VERSION=2.3.2
/thesis/Hyperledger/fabric-samples/test-network/./bin/cryptogen
Generating certificates using cryptogen tool
Creating Org1 Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-org1.yaml --output=organizations
org1.example.com
+ res=0
Creating Org2 Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-org2.yaml --output=organizations
org2.example.com
+ res=0
Generating Orderer Org Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-orderer.yaml --output=organizations
+ res=0
Generating CCP files for Org1 and Org2
Creating network "fabric_test" with the default driver
Creating volume "docker_orderer.example.com" with default driver
Creating volume "docker_peer0.org1.example.com" with default driver
Creating volume "docker_peer0.org2.example.com" with default driver
Creating orderer.example.com ... done
Creating peer0.org1.example.com ... done
Creating peer0.org2.example.com ... done
Creating cli ... done
CONTAINER ID    IMAGE                                COMMAND          NAMES          CREATED          STATUS          PORTS
6753c4ae3elf    hyperledger/fabric-tools:latest     "/bin/bash"      cli            2 seconds ago    Up Less than a second
c26aa48256e6    hyperledger/fabric-peer:latest      "peer node start" peer0.org2.example.com 6 seconds ago    Up 2 seconds    0.0.0.0:9051->9051/tcp,
1/tcp, :::19051->19051/tcp
bcc912ee023c    hyperledger/fabric-orderer:latest   "orderer"        orderer.example.com 6 seconds ago    Up 3 seconds    0.0.0.0:7050->7050/tcp,
53->7053/tcp, 0.0.0.0:17050->17050/tcp, :::17050->17050/tcp
51393c40f6b5    hyperledger/fabric-peer:latest      "peer node start" peer0.org1.example.com 6 seconds ago    Up 2 seconds    0.0.0.0:7051->7051/tcp,
17051->17051/tcp

```

Figure 5.9: Set up the network

Establish Channel In The Network

To enable the peers to communicate securely, a private channel should be established between peers which decided to make a transaction. For example, communicating for signing the smart contract and trading energy. The channels are a private communication layer between individual members of the network. Channels can only be utilized by organizations invited and invisible to other network members. A unique blockchain ledger is available for each channel [15]. Once the channel is established for the first time in the network, the peers must exchange their keys and sign the main smart contract to register in the network. Since it is the first transaction, we are creating the genesis block in this step since the first transaction is completed and added as the first chain. To sign the main smart contract by peers according to Hyperledger life cycle mechanism, ordering and endorsement phase must be satisfied. The ordering and endorsement phase is also visible in the screenshot below.

```

Creating channel 'mychannel'.
If network is not up, starting nodes with CLI timeout of '5' tries and CLI delay of '3' seconds and using database 'leveldb'
Generating channel genesis block 'mychannel.block'
/thesis/Hyperledger/fabric-samples/test-network/bin/configtxgen
+ configtxgen -profile TwoOrgsApplicationGenesis -outputBlock ./channel-artifacts/mychannel.block -channelID mychannel
2021-09-05 11:43:41.356 UTC [common.tools.configtxgen] main -> INFO 001 Loading configuration
2021-09-05 11:43:41.367 UTC [common.tools.configtxgen.localconfig] completeInitialization -> INFO 002 orderer type: etcdraft
2021-09-05 11:43:41.368 UTC [common.tools.configtxgen.localconfig] completeInitialization -> INFO 003 Orderer.EtcdRaft.Options unset
heartbeat_tick:1 max_inflight_blocks:5 snapshot_interval_size:16777216
2021-09-05 11:43:41.368 UTC [common.tools.configtxgen.localconfig] Load -> INFO 004 Loaded configuration: /thesis/Hyperledger/fabric
2021-09-05 11:43:41.375 UTC [common.tools.configtxgen] doOutputBlock -> INFO 005 Generating genesis block
2021-09-05 11:43:41.375 UTC [common.tools.configtxgen] doOutputBlock -> INFO 006 Creating application channel genesis block
2021-09-05 11:43:41.376 UTC [common.tools.configtxgen] doOutputBlock -> INFO 007 Writing genesis block
+ res=0
Creating channel mychannel
Using organization 1
+ osnadmin channel join --channelID mychannel --config-block ./channel-artifacts/mychannel.block -o localhost:7053 --ca-file /thesis
ons/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem --client-cert /thesis/Hy
/ordererOrganizations/example.com/orderers/orderer.example.com/tls/server.crt --client-key /thesis/Hyperledger/fabric-samples/test-n
om/orderers/orderer.example.com/tls/server.key
+ res=0
Status: 201
{
  "name": "mychannel",
  "url": "/participation/v1/channels/mychannel",
  "consensusRelation": "consenter",
  "status": "active",
  "height": 1
}
Channel 'mychannel' created
Joining org1 peer to the channel...
Using organization 1
+ peer channel join -b ./channel-artifacts/mychannel.block
+ res=0
2021-09-05 11:43:47.845 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
2021-09-05 11:43:48.105 UTC [channelCmd] executeJoin -> INFO 002 Successfully submitted proposal to join channel
Joining org2 peer to the channel...
Using organization 2
+ peer channel join -b ./channel-artifacts/mychannel.block
+ res=0
2021-09-05 11:43:51.239 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
2021-09-05 11:43:51.499 UTC [channelCmd] executeJoin -> INFO 002 Successfully submitted proposal to join channel
Setting anchor peer for org1...
Using organization 1

```

Figure 5.10: Connect organization one to the channel

```

+ configtxlator proto encode --input config_update_in_envelope.json --type common.Envelope
2021-09-05 11:43:52.436 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
2021-09-05 11:43:52.473 UTC [channelCmd] update -> INFO 002 Successfully submitted channel update
Anchor peer set for org 'Org1MSP' on channel 'mychannel'
Setting anchor peer for org2...
Using organization 2
Fetching channel config for channel mychannel
Using organization 2
Fetching the most recent configuration block for the channel
+ peer channel fetch config config_block.pb -o orderer.example.com:7050 --ordererTLSHostnameOverride orderer.example.com -c
doer/fabric/peer/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.c
2021-09-05 11:43:52.890 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
2021-09-05 11:43:52.901 UTC [cli.common] readBlock -> INFO 002 Received block: 1
2021-09-05 11:43:52.901 UTC [channelCmd] fetch -> INFO 003 Retrieving last config block: 1
2021-09-05 11:43:52.904 UTC [cli.common] readBlock -> INFO 004 Received block: 1
Decoding config block to JSON and isolating config to Org2MSPconfig.json
+ configtxlator proto decode --input config_block.pb --type common.Block
+ jq '.data.data[0].payload.data.config'
Generating anchor peer update transaction for Org2 on channel mychannel
+ jq '.channel_group.groups.Application.groups.Org2MSP.values += {"AnchorPeers":{"mod_policy": "Admins","value":{"anchor_pe
rsion": "0"}}' Org2MSPconfig.json
+ configtxlator proto encode --input Org2MSPconfig.json --type common.Config
+ configtxlator proto encode --input Org2MSPmodified_config.json --type common.Config
+ configtxlator compute_update --channel_id mychannel --original original_config.pb --updated modified_config.pb
+ configtxlator proto decode --input config_update.pb --type common.ConfigUpdate
+ jq .
++ cat config_update.json
+ echo '{"payload":{"header":{"channel_header":{"channel_id":"mychannel","type":2}},"data":{"config_update":{"channel_id
{"groups":{"Application":{"groups":{"Org2MSP":{"groups":{"mod_policy":"","policies
ll,"version":"","Endorsement":{"mod_policy":"","policy":null,"version":"","Readers
{"Writers":{"mod_policy":"","policy":null,"version":"","values":{"MSP":{"
{"version":"","mod_policy":"","policies":{"values":{"version":"","
{"version":"","write_set":{"groups":{"Application":{"groups":{"Org2MSP":{"groups
mins":{"mod_policy":"","policy":null,"version":"","Endorsement":{"mod_policy":"","po
_policy":"","policy":null,"version":"","Writers":{"mod_policy":"","policy":null,"versi
mod_policy":"","Admins":{"value":{"anchor_peers":{"host":"peer0.org2.example.com","port":9051}}
{"value":null,"version":"","mod_policy":"","policies":{"
{"policies":{"values":{"version":"","
+ configtxlator proto encode --input config_update_in_envelope.json --type common.Envelope
2021-09-05 11:43:53.476 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
2021-09-05 11:43:53.519 UTC [channelCmd] update -> INFO 002 Successfully submitted channel update
Anchor peer set for org 'Org2MSP' on channel 'mychannel'
Channel 'mychannel' joined
et2599user2@ET2599VM2:/thesis/Hyperledger/fabric-samples/test-network$

```

Figure 5.11: Connect organization two to the channel

Add a New Peer To The Network

In this part, a new participant authorized by an organization will add to the network securely. In the first step of energy trading algorithm in figure 5.3, we shown the step to step mechanism for joining a new peer to the network securely. The below picture is showing that the crypto materials for new peer is generating and a new peer is added successfully. The full process for joining a new participant in the network is discussed in section 4.2.1 of this study.

You will observe the creation of the crypto material for Org3. The creation of the organizing definition and finally the modification, signature, and submission to the channel configuration is shown below.

Add stateDB (CouchDB) To The Network

CouchDB works separately with the peer as a distinct database process. In this step, we will update the peer's docker images to bootup with a stateDB. To enable and install a distributed database on peers, we installed CouchDB as a database to store the transactions. A core.yaml file provided by Hyperledger foundation is used to install CouchDB on each peer. In the final step of each transaction, a copy of the smart contract will send to all peers and store their database. Once it is stored, it is not possible to make any change on that. Below screen shot is showing that the CouchDB is installed for each peer successfully.

```

Creating channel 'mychannel'.
If network is not up, starting nodes with CLI timeout of '5' tries and CLI delay of '3' seconds and using database 'couchdb'
Bringing up network
LOCAL_VERSION=2.3.2
DOCKER_IMAGE_VERSION=2.3.2
/thesis/Hyperledger/fabric-samples/test-network/bin/cryptogen
Generating certificates using cryptogen tool
Creating Org1 Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-org1.yaml --output=organizations
org1.example.com
+ res=0
Creating Org2 Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-org2.yaml --output=organizations
org2.example.com
+ res=0
Creating Orderer Org Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-orderer.yaml --output=organizations
+ res=0
Generating CCP files for Org1 and Org2
Creating network "fabric_test" with the default driver
Creating volume "docker_orderer.example.com" with default driver
Creating volume "docker_peer0.org1.example.com" with default driver
Creating volume "docker_peer0.org2.example.com" with default driver
Creating couchdb0 ... done
Creating orderer.example.com ... done
Creating couchdb1 ... done
Creating peer0.org1.example.com ... done
Creating peer0.org2.example.com ... done
Creating cli ... done

```

CONTAINER ID	IMAGE	NAMES	COMMAND	CREATED	STATUS	PORTS
e67f9841d097	hyperledger/fabric-tools:latest	cli	"/bin/bash"	2 seconds ago	Up Less than a second	
a3583db7bbe8	hyperledger/fabric-peer:latest	peer0.org2.example.com	"peer node start"	5 seconds ago	Up 2 seconds	0.0.0.0:
05a4654ff59b	hyperledger/fabric-peer:latest	peer0.org1.example.com	"peer node start"	6 seconds ago	Up 3 seconds	0.0.0.0:
ff58531c3dd6	couchdb:3.1.1	couchdb1	"tini -- /docker-ent..."	9 seconds ago	Up 5 seconds	4369/tcp
07965cc5b4f9	hyperledger/fabric-orderer:latest	orderer	"orderer"	9 seconds ago	Up 6 seconds	0.0.0.0:
::17050->17050/tcp	orderer.example.com					
79e392284dd7	couchdb:3.1.1	couchdb0	"tini -- /docker-ent..."	9 seconds ago	Up 6 seconds	4369/tcp

```

Generating channel genesis block 'mychannel.block'
/thesis/Hyperledger/fabric-samples/test-network/bin/configtxgen
+ configtxgen -profile TwoOrgsApplicationGenesis -outputBlock ./channel-artifacts/mychannel.block -channelID mychannel
2021-09-05 16:01:18.936 UTC [common.tools.configtxgen] main -> INFO 001 Loading configuration

```

Figure 5.14: Add database to the network

Apply Smart Contract

After creating the network and channel, participants can start using smart contracts to interact within the network. Seller or buyers must use a smart contract for any transaction, in this case, trading energy. In this case, we used Go as the smart contract implementation language. The chaincode should be installed in all participants in order to enable peers to interact using smart contracts.

```
type Asset struct {
    ID          string `json:"id"`
    Energy      string `json:"energy"`
    RequiredEnergy int    `json:"requiredenergy"`
    Owner       string `json:"owner"`
    Price       int    `json:"price"`
}

// InitLedger adds a base set of assets to the ledger(list of assets)
// Here the data is stored in the json format in the CouchDB
func (s *SmartContract) InitLedger(ctx contractapi.TransactionContextInterface) error {

    assets := []Asset{

        {ID: "asset1", Energy: "Electricity", RequiredEnergy: 40, Owner: "prosumer", Price: 10},
        {ID: "asset2", Energy: "Electricity", RequiredEnergy: 60, Owner: "consumer", Price: 15},
        {ID: "asset3", Energy: "Electricity", RequiredEnergy: 100, Owner: "utility-grid", Price: 25},
        {ID: "asset4", Energy: "Electricity", RequiredEnergy: 40, Owner: "prosumer", Price: 12},
        {ID: "asset5", Energy: "Electricity", RequiredEnergy: 60, Owner: "consumer", Price: 16},
        {ID: "asset6", Energy: "Electricity", RequiredEnergy: 100, Owner: "utility-grid", Price: 18},
    }

    // creating a readable object for the database
    for _, asset := range assets {
        assetJSON, err := json.Marshal(asset)
        if err != nil {
            return err
        }
    }
}
```

Figure 5.15: smart contract function

To guarantee that transactions generated with smart contracts are legitimate, they must generally be signed by several entities before being committed to the channel ledger. Fabric's trust architecture relies heavily on multiple signatures. Multiple endorsements for a transaction prohibit one company on a channel from interfering with their peer's ledger or employing business logic that was not previously agreed upon. To sign a transaction, each organization must first invoke and execute their peer's smart contract, which then signs the transaction's output [15].

The following screenshots are showing that the smart contract is installed and signed by participants stepwise. The "Org1MSP" is signed by returning true to the network while "Org2MSP" and "Org3MSP" returned false since they did not sign the contract yet. Picture number 6.12 is showing that the "Org2MSP" is returned true once it is signed.

```

Chaincode is packaged
Installing chaincode on peer0.org1...
Using organization 1
+ peer lifecycle chaincode install basic.tar.gz
+ res=0
2021-09-08 09:09:38.372 CEST [cli.lifecycle.chaincode] submitInstallProposal -> INFO 001 Installed remotely: response:<status:200>
Chaincode is installed on peer0.org1
Installing chaincode on peer0.org2...
Using organization 2
+ peer lifecycle chaincode install basic.tar.gz
+ res=0
2021-09-08 09:09:45.207 CEST [cli.lifecycle.chaincode] submitInstallProposal -> INFO 001 Installed remotely: response:<status:200>
Chaincode is installed on peer0.org2
Using organization 1
+ peer lifecycle chaincode queryinstalled
+ res=0
Installed chaincodes on peer:
Package ID: basic.1.0:9f7859a8bcb0bc08c0ec78fb0b014a818e52376f2f6ca6939dd894fd5cc404d5, Label: basic_1.0
Query installed successful on peer0.org1 on channel
Using organization 1
+ peer lifecycle chaincode approveformyorg -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile /home/.../orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem --channelID mychannel --name basic --version 1.0 --packageID basic
2021-09-08 09:09:47.345 CEST [chaincodeCmd] ClientWait -> INFO 001 txid [631ae0653321b25fd64be72a4e5b3f009078b30ee0617b77a50454e1d]
Chaincode definition approved on peer0.org1 on channel 'mychannel'
Using organization 2
Checking the commit readiness of the chaincode definition on peer0.org1 on channel 'mychannel'...
Attempting to check the commit readiness of the chaincode definition on peer0.org1, Retry after 3 seconds.
+ peer lifecycle chaincode checkcommitreadiness --channelID mychannel --name basic --version 1.0 --sequence 1 --output json
+ res=0
{
  "approvals": {
    "Org1MSP": true,
    "Org2MSP": false,
    "Org3MSP": false
  }
}

```

Figure 5.16: Installing chaincode - Signing smart contract

```

+ peer lifecycle chaincode checkcommitreadiness --channelID mychannel --name basic --version 1.0 --sequence 1 --output json
+ res=0
{
  "approvals": {
    "Org1MSP": true,
    "Org2MSP": false,
    "Org3MSP": false
  }
}
Checking the commit readiness of the chaincode definition successful on peer0.org1 on channel 'mychannel'
Using organization 2
Checking the commit readiness of the chaincode definition on peer0.org2 on channel 'mychannel'...
Attempting to check the commit readiness of the chaincode definition on peer0.org2, Retry after 3 seconds.
+ peer lifecycle chaincode checkcommitreadiness --channelID mychannel --name basic --version 1.0 --sequence 1 --output json
+ res=0
{
  "approvals": {
    "Org1MSP": true,
    "Org2MSP": false,
    "Org3MSP": false
  }
}
Checking the commit readiness of the chaincode definition successful on peer0.org2 on channel 'mychannel'
Using organization 1
Checking the commit readiness of the chaincode definition on peer0.org1 on channel 'mychannel'...
Attempting to check the commit readiness of the chaincode definition on peer0.org1, Retry after 3 seconds.
+ peer lifecycle chaincode checkcommitreadiness --channelID mychannel --name basic --version 1.0 --sequence 1 --output json
+ res=0
{
  "approvals": {
    "Org1MSP": true,
    "Org2MSP": true,
    "Org3MSP": false
  }
}
Checking the commit readiness of the chaincode definition successful on peer0.org1 on channel 'mychannel'
Using organization 2
Checking the commit readiness of the chaincode definition on peer0.org2 on channel 'mychannel'...
Attempting to check the commit readiness of the chaincode definition on peer0.org2, Retry after 3 seconds.
+ peer lifecycle chaincode checkcommitreadiness --channelID mychannel --name basic --version 1.0 --sequence 1 --output json
+ res=0
{
  "approvals": {
    "Org1MSP": true,
    "Org2MSP": true,
    "Org3MSP": false
  }
}

```

Figure 5.17: Installing chaincode - Signing smart contract

Since the main smart contract is installed and applied to peers, all the necessary features for trading energy securely over Hyperledger Fabric are now ready.

The following API is implemented to check the feasibility, scalability, and capacity of the network. With the help of this API, we can run multiple transactions utilizing the network participants. The figures below show the function in the API of how transactions are generated.

```

async function main() {
  try {
    const ccp = buildCCPOrg1();
    const caclient = buildCAclient(FabricCAServices, ccp, 'ca.org1.example.com');
    const wallet = await buildWallet(wallets, walletPath);
    await enrollAdmin(caclient, wallet, mspOrg1);
    const gateway = new Gateway();

    try {
      await gateway.connect(ccp, {
        wallet,
        identity: org1UserId,
        discovery: { enabled: true, asLocalhost: true }
      });

      const network = await gateway.getNetwork(channelName);
      const contract = network.getContract(chaincodeName);
      const N = 20;
      const owners = ['prosumer', 'consumer', 'utility-grid'];
      const ttypes = ['sell', 'buy'];
      console.log('creating', N, 'transactions');
      for (let i = 0; i < N; i++) {
        console.log('--> Evaluate Transaction: Create Assets, function returns transaction id on the ledger');
        let id = await contract.submitTransaction('CreateAsset', owners[i % 3], 'Electricity', Math.Floor(Math.random() * 50) + 50, ttypes[i % 2]);
        let result = await contract.evaluateTransaction('ReadAsset', id);
        console.log('*** Result: ${prettyJSONString(result.toString())}');
      }

      console.log('\n--> Evaluate Transaction: GetAllAssets, function returns all the current assets on the ledger');
      let result = await contract.evaluateTransaction('GetAllAssets');
      console.log('*** Result: ${prettyJSONString(result.toString())}');
    } finally {
      gateway.disconnect();
    }
  } catch (error) {
    console.error('***** FAILED to run the application: ${error}');
  }
}

```

Figure 5.18: API main function

The following screenshot is showing that the network is up, authentication is completed for the participants and prosumers-consumers are trading energy according to the policies mentioned before.

```

moab19@ubuntu-20:~/HyperledgerFabric_v1/HyperledgerFabric/fabric-samples/test-network/api$ node api.js
Loaded the network configuration located at /home/moab19/HyperledgerFabric_v1/HyperledgerFabric/fabric-samples/test-network/org1.json
Built a CA Client named ca-org1
Built a file system wallet at /home/moab19/HyperledgerFabric_v1/HyperledgerFabric/fabric-samples/test-network/api/wallet
Successfully enrolled admin user and imported it into the wallet
Successfully registered and enrolled user appUser and imported it into the wallet
creating 20 transactions
--> Evaluate Transaction: Create Assets, function returns transaction id on the ledger
*** Result: {
  "id": "2",
  "owner": "prosumer",
  "energy": "Electricity",
  "price": 91
}
--> Evaluate Transaction: Create Assets, function returns transaction id on the ledger
*** Result: {
  "id": "3",
  "owner": "consumer",
  "energy": "Electricity",
  "price": 52
}
--> Evaluate Transaction: Create Assets, function returns transaction id on the ledger
*** Result: {
  "id": "4",
  "owner": "utility-grid",
  "energy": "Electricity",
  "price": 91
}
--> Evaluate Transaction: Create Assets, function returns transaction id on the ledger
*** Result: {
  "id": "5",
  "owner": "prosumer",
  "energy": "Electricity",
  "price": 70
}
--> Evaluate Transaction: Create Assets, function returns transaction id on the ledger
*** Result: {
  "id": "6",
  "owner": "consumer",
  "energy": "Electricity",
  "price": 85
}

```

Figure 5.19: Transaction creation

Finally, the below picture is showing that the smart contract is invoked successfully by returning the result status 200. The energy trading transaction is completed. The picture also shows the content of the business transactions, in this scenario, energy, amount, and price. In this case, specific amount of energy is transferred with the help of smart contract.

```

organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n
${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt --peerAddresses localhost:9051
ns/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt -c '{"function":"InitLedger","Args":[]}'
2021-09-08 09:36:00.151 CEST [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:200
moab19@ubuntu-20:~/HyperledgerFabric_v1/HyperledgerFabric/fabric-samples/test-network$ peer chaincode query -c mychannel -n basic -e '{"Args":["Get",
{"id":"asset1","energy":"Electricity","requiredenergy":40,"owner":"prosumer","price":10}, {"id":"asset2","energy":"Electricity",
{"id":"asset3","energy":"Electricity","requiredenergy":100,"owner":"utility-grid","price":25}, {"id":"asset4","energy":"Electricity",
{"id":"asset5","energy":"Electricity","requiredenergy":60,"owner":"consumer","price":16}, {"id":"asset6","energy":"Electricity","req
moab19@ubuntu-20:~/HyperledgerFabric_v1/HyperledgerFabric/fabric-samples/test-network$

```

Figure 5.20: Succeed transaction

Chapter 6

Validation and Results

In this chapter, the validity and results of the entire study are discussed. In the validation part, we have shown step-by-step implementation of our system model and validated it with the scientific theoretical resources and output of the implementation. Subsequently, we answered the thesis question in the result section. Finally, in the section of the lesson learned, we have mentioned the knowledge we learned after conducting such interesting research work.

6.1 Validation

Validation is the process of ensuring that the current program of study is practically operated, what we discussed in the theoretical studies in sections 5.2.1, 5.2.2, 5.2.3 of this study. In this part, we are trying to summarize all the steps of the implementation and also avoid confusion for the users, we provided all the necessary commands in a tabular format. In addition, the suitability of theoretical concepts with the practical concepts are shown, for example, the relative between Hyperledger Fabric life cycle and algorithm related to the implementation phase. Table 5.3 shows the necessity commands respectively and the expected outcomes addressed with their figure numbers in this thesis. Additionally, table 5.4 shows the public git repositories where you can find all the implementation codes provided by the authors of this thesis.

Once the installation of dependencies are completed, we should change the directory to the path where the implementation code available. In this case, we considered a Bash file locate in the directory "HyperledgerFabric_v1/HyperledgerFabric/fabric-samples/test-network".

The first command is trying to bring up the network. For setting up a secure network, private and public key should be exchanged between the peers [1]. For example, it is generating crypto materials and applies some firewall rules for two peers by calling a function shown in figure 5.5 and its result in figure 5.9. The second command is establishing a secure channel between the created peers by configuring some network details of the peers [54] which shown in figures 5.6 and 5.10. The third command is installing a state database on each peer separately using a Yaml file shown in figure 5.7 and 5.14. In this case, CouchDB is installed. In command four, we are trying to add a new peer to the

network visible in figures 5.12 and 5.13 according to the policy mentioned in section 4.2.1 in this thesis. In Hyperledger Fabric, channel shares confidential information frequently hence the distributed database CouchDB supports huge range of queries and transaction informations [26]. The fourth command is the merged form of all previous commands to set up, connect to the available channel and install the CouchDB. It is very difficult to insert fake data, as the blockchain uses the consensus of nodes that consist of fifty-one percent of the network [14]. The final step of establishing a secure p2p network is to enable all the participants to have fully secure communication by implementing the smart contract. Using the fifth command, we are applying the main smart contract (discussed in chapter three, section 3.3) to authenticate, register, and enable energy trading. Finally, to take the advantage of such a network, we designed an API to use all the steps of Hyperledger Fabric components and generate transaction for energy trading shown in figure 5.19.

When commands one, two, and three in table 6.1 below are successfully executed, we should change the directory to the path where the implementation code for adding a new peer to the network is located. In this case, it is in "HyperledgerFabric_v1/HyperledgerFabric/fabric-samples/test-network/addOrg3". Therefore, command number four should be executed. Once we added the third peer successfully, we supposed to return to the previous directory where the "./network.sh" file is located. Using command number five, we are trying to sign the main smart contract we described in detail in chapter three of this thesis. After signing the main smart contract between the peers, it is time to start energy trading. For this scenario, we implemented an API to generate energy trading using all the Hyperledger Fabric components and show the feasibility of such a scenario. To run the API, we should change the directory to the path where the code is located, for example, "HyperledgerFabric_v1/HyperledgerFabric/fabric-samples/test-network/api" and should execute command number six to see the results.

To avoid any human error for the command execution, the authors of this thesis provided a fully automated Python code to run all the steps respectively and also apply the energy trading algorithm mentioned in chapter five, figure 5.3 and 5.4. The Python code is publicly available in a git repository mentioned in the table 6.2 below.

For validating our implementation we provided the below table with output reference as well as Hyperledger Fabric work diagram. Furthermore, in the figure 6.1, two main phases is shown where the green parts of the pictures shows how endorsement phases, committing and validating phases are happening, and the blue part shows the transaction delivery phase in which the energy trading transaction authenticated successfully and energy is ready to be delivered. In addition to that, all the phases of Hyperledger Fabric life cycle [8] have been achieved through our implementation which is addressed in the table and figure 6.1 below. Therefore anyone can go to the public Git repositories addressed in table 6.2 to download the files and follow the instructions to see the exact output what we mentioned in the chapter five of this thesis.

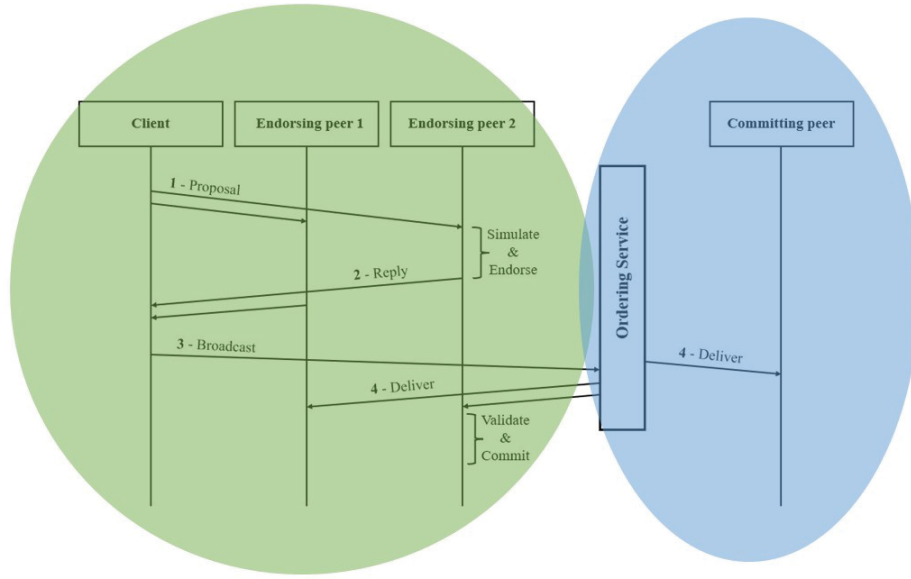


Figure 6.1: Hyperledger Fabric transaction life cycle [8]

No.	Commands	Output
1	<code>./network.sh up</code>	Figure 5.9
2	<code>./network.sh createChannel</code>	Figures 5.10 & 5.11
3	<code>./network.sh up createChannel -s couchdb</code>	Figures 5.14
4	<code>./addOrg3.sh up -c mychannel -s couchdb</code>	Figures 5.12 - 5.13
5	<code>./network.sh deployCC -ccn basic -ccp ../../mychaincode/ -ccl go</code>	Figures 5.15 - 5.17
6	<code>node api.js</code>	Figure 5.18 - 5.20

Table 6.1: Implementation commands

In table 5.4 below, all the required git repositories are provided. All the repositories are available publicly and it is open for further developments.

No.	Github URL
1	https://github.com/msnband/Thesis-prerequisite-Hyperledger-Fabric
2	https://github.com/ragadeep-maker/MasterThesis-P2P
3	https://github.com/msnband/ThesisEnergyTradingSimulation

Table 6.2: Public Git repositories

6.2 Results

The implementation of an energy trading network on the Hyperledger Fabric blockchain required several researches, configurations, and programming. Several technologies are utilized to perform blockchain tests, from hardware facilities to software components. Thanks to IBM and the Linux Foundation for making an open-source Hyperledger Fabric test-case environment available. We attempted to demonstrate all the processes involved in setting up a peer-to-peer Hyperledger Fabric blockchain network in this chapter. For instance, how a network is set up, how peers are added to it, how to create a channel between peers, and how to apply chaincode (smart contract) between peers. Finally, the transactions are added to a distributed stateDB, in this instance CouchDB.

In the following sections, we will show the step-to-step of showing the Hyperledger Fabric p2p network and the results of all the phases we described in the previous chapter.

RQ-1: What are the current proposals for p2p energy sharing using blockchains and how they compare each other?

Answer: For answering the first research question, we performed literature review on p2p energy sharing using blockchain technology and the articles related to it. The accumulated information from the literature review regarding the current proposal for the p2p energy sharing using blockchain is given below in tabular format in table 6.3 below. The second column of table 6.3 shows that, the implementation area where blockchain technology has been implemented, and we found that blockchain technology is implemented mostly in the energy sector, IT sector, and cloud environment. If we see the third column, it depicts what kinds of blockchain technology have been implemented. Hyperledger Fabric and Ethereum are the most implemented technology in the p2p energy trading use cases. For increasing security, the smart contracts have been implemented on both the technology in p2p energy trading uses cases.

Furthermore, from table 6.3, we can compare the blockchain technology based on their features such as security, efficiency, transparency, scalability, etc implemented in the p2p energy sharing use case and also the current proposal for the same scenario. Finally, both the technologies Ethereum and Hyperledger Fabric can be implemented in p2p energy trading based on the system model, feature, and service management functionalities.

Reff.	Implementation area	Blockchain Technologies	Smart contract	Service management functionalities
[17]	Local energy market	Ethereum	Yes	Security
[23]	Energy sector	Ethereum	Yes	Security
[25]	Virtual power plant	Ethereum	Yes	Security, scalability, adaptability
[24]	Smart grid	Hyperledger	Yes	Security
[26]	Energy sector	Hyperledger fabric	Yes	Security, encryption, decryption, efficiency
[28]	Cloud environment	Hyperledger fabric	Yes	Security
[14]	Energy sector	Consortium	Yes	Security, authentication, scalability
[27]	Energy sector	Hyperledger	Yes	Security
[22]	Energy sector	-	-	-
[29]	Microgrids	-	-	-
[21]	Energy sector	Ethereum	Yes	
[31]	Residential communities	Hyperledger fabric	Yes	Efficiency
[30]	Microgrid		Proof of Energy	Scalability
[55]		Ethereum	Yes	
[20]	Smart grids	Ethereum	Yes	Trust management, Security, Transparency
[32]	Fairness	Ethereum	Yes	Authentication and use privacy
[33]	Smart grids	-	-	Secured algorithm
[19]	IT sector	Ethereum	Yes	Security
[18]	Smart grids	Ethereum	Yes	Privacy, security, energy data storage

Table 6.3: Comparison of different blockchains

RQ-2: Given that, there are multiple blockchain implementation technologies available. How can one select the blockchain technique for implementation of the proposed p2p energy management use case of this thesis?

Answer: After going through a lot of research papers we found that there are many blockchain technology exists in recent times such as Hyperledger Fabric, Ethereum, BIRD, Blockcloud, CORDA, and many more. Every blockchain technology has its own features which are implemented in different use cases based on suitability. In section 2.5 we have gone through different blockchain technologies and their use case in different scenarios. Considering our system model, we found that Hyperledger Fabric is the most suitable blockchain technology for our implementation. In our thesis we considered many important functionalities to select blockchain technology. Security, open-source, availability of Hyperledger, scalability are the key reasons for selecting Hyperledger Fabric in our thesis. Section 2.5 of this study describes in detail the reasons for selecting Hyperledger Fabric for the energy trading scenario.

Furthermore, in chapter 4, three different system models (two technical models and one conceptual model) were discussed in which all the three are structurally and technically implementable, then the purposed model has been selected based on a scientific survey on the advantages and disadvantages of different system designs. After having a review of different studies, the authors decided to choose model-2 (Large scale energy storage system). Although model-1 has its own merits, the scalability of model-2 was the main reason for choosing this model. In the selected model, the network provider is able to develop the capacity of ESS at any time. In addition to that, hash functions are also implemented to ensure the security measures at model-2. After going through the features and functionality of our proposed system model, we have decided to implement Hyperledger Fabric to attain optimum security, availability, and scalability for p2p energy sharing use cases.

RQ-3: How can one implement the blockchain and its data structures or the service management for the considered use case of p2p energy sharing? Which management functions can be supported by a blockchain?

Answer: For implementing blockchain on energy trading we evaluated three models where the different structures of such p2p energy trading networks using blockchain technology have been discussed based on their features and functionalities in the chapter 4. Although all the mentioned models are secure enough and implementable, we selected the most suitable model in terms of scalability and performance. After studying several papers, Hyperledger Fabric is selected as the base blockchain platform for this use case. A computer-based tool as an EMS is designed in the system to have control over the behavior of the entire market in that specific region. In addition to that, each node is equipped with a smart energy meter which is responsible to keep track of data, such as consumed and produced energy and some power factors.

Based on our system model and theoretical assumption in sections 5.2.1, 5.2.2, 5.2.3 of this

study we implemented our solution and in section 5.4 all the steps have been discussed with pictorial representation.

In addition, an Application Programming Interface (API) written in Nodejs is applied to show the network's scalability and capacity which clearly proves the feasibility of the implementation. Using this API, we can perform N number of energy trading transactions in the network and save all the transactions in CouchDB using this API. In chapter five, section 5.4 of this thesis, we clearly show all the implementation steps including set-up the network, establishing a channel, adding a new peer to the network, installing the database on the peers, applying smart contract, and finally, we illustrated the API in which we enabled the energy trading over Hyperledger Fabric network. The API is proving the feasibility, scalability, and capacity of the network. Finally, section 6.1 shows that it is possible to use Hyperledger Fabric blockchain for such a p2p energy trading network.

6.3 Lesson Learned

We are excited to write what we learned, shared, and contributed from the thesis work journey. The outcomes from chapters five and six are clearly showing that we have worked with many technologies. Multiple programming languages (Python, Nodejs, Go, Bash), Linux firewalls and IP rules, generating crypto materials in Linux, working with containers such as Docker, Cloud computing (Microsoft Azure), distributed database (CouchDB), automation tools (Ansible), and getting into different blockchain technologies, after reading a large number of documentations all and all have helped us to learn and boost our technical knowledge. As a consequence, we are firmly confident about trading energy over peer-to-peer network using Hyperledger Fabric is functionally possible.

- **Lesson learnt on distributed manners:** Through doing this dissertation, we have gone through tens of scientific papers in distributed technologies. We are now confident that decentralized management and storage systems in the IT sector in which blockchain is the main sample of distributed systems, provide more security, reliability, and redundancy. Decentralized databases such as CouchDB, LevelDB, or IPFS which are widely used by blockchain technologies, are more reliable than centralized databases.
- **Lesson learnt on blockchain technologies:** After going through the literature review, we learned about different types of blockchain technologies, such as private, public, or permissioned blockchain. Furthermore, we learned about the consensus algorithms and their complexity, and their strength, weaknesses, scalability, and security of different blockchains.
- **Lesson learnt on smart contract:** The smart contract is considered a digital interaction with users and a complex distributed system like blockchain. In this thesis, we learned how to develop a smart contract for different use cases particularly

for energy trading in Hyperledger Fabric using different programming languages, such as Go or Java script.

- **Lesson learnt on implementation:** Going through practical implementation of peer-to-peer energy trading using blockchain technologies and understanding the requirements and the environment configurations are a bit complex. Because to implement such a blockchain network, we had to depend on many dependencies, for example, software and tools, programming languages, containers, and network configuration, firewalls, and securities, additionally, applying our own smart contract to enable the peers to communicate securely and how the peers could be agreed on the validity of the transaction were our achievements helped us to enhance our technical skills.

Furthermore, during the implementation we found that reaching an agreement through the smart contract and also immutability of smart contract might be the most complex process in any blockchain network and because any transaction must be validated by other participants, a huge number of loads will impose to the network. In this case, the system needs a high-performance CPU and memory to be able to handle such a number of transactions.

Chapter 7

Conclusions and Future Work

This chapter outlines the conclusion of our thesis work and the summary of answers to our research questions. Subsequently, The chapter ends with mentioning the possible direction of future work.

7.1 Conclusion

The need for security in the energy sector and its market is more complex than ever. Although decentralized systems are more complex than centralized systems, taking the advantage of a suitable decentralized network could solve several security issues and possibly make the network more scalable. In this thesis, a demonstration of a p2p energy trading using Hyperledger Fabric blockchain is implemented where the two important functionalities, security, and scalability were key concerns.

We went through several blockchain technologies in numerous aspects for the use case of peer-to-peer energy trading in chapter 2 of this paper. Taking into account several essential characteristics such as security, performance, and scalability, which is the solution to the first research question. In chapters 2 and 3, a comprehensive study of several types of blockchain technology and their use cases was conducted in response to the second research question. We selected the Hyperledger Fabric, which we described in chapter 2, section 2.5 of this study, based on the capabilities of several blockchain technologies. Following a thorough review of several system models for a p2p energy trading system utilizing blockchain in chapter 4, we selected the system model mentioned in section 4.1.1.

In chapter 5, we also conducted an implementation for such a system model to prove that the solution is functionally possible is related to the third research question of this thesis. Since the blockchain network is based on zero-trust, a highly secure distributed architecture is discussed. For instance, how a party can join the network and start energy trading securely. In addition, the smart contract is widely used for any transaction in the network, for example, for joining the network (main smart contract) and for energy trading (p2p smart contract). In our p2p blockchain network, the entire transaction process is hashed using the prosumer and consumer keys. The identification of the transaction's parties is hidden from other participants. A distributed database (CouchDB) is selected to

store the transactions and add the transactions to the ledger to make the entire blockchain.

In CouchDB, all the transactions are stored and fully encrypted. In this way, the sensitive information is not visible to malicious users. Once the transactions are confirmed according to the Hyperledger consensus algorithm, the transaction will store in the database and will form a block of the blockchain. Finally, according to the regulations discussed in this thesis, an API is given to initiate N number of energy trading operations. We can demonstrate the viability of the complete process for blockchain energy trading using this API. The results from chapter five depict that our objectives are successfully achieved.

7.2 Future Work

Throughout the thesis, a number of questions strike in our mind that would be relevant to anybody interested in conducting more study in the field of blockchain implementation and service management for energy trading. This research is conducted with randomly generated data using API for energy trading transactions in the cloud environment. Therefore, it can also be explored by real-time scenarios using real values or data.

Furthermore, more work can be done in the future on improving the traffic and transaction load to improve the speed and performance of the p2p blockchain network. Since the sustainability and energy consumption of such architecture is one of the concerns, enhancing the CPU and memory utilization could be interesting to do in the future. In addition to that, we also can explore the security of the nodes while energy trading on a metropolitan scale in real-time is happening. It can be interesting to conduct energy trading with another blockchain like Ethereum, therefor anyone can work more on enabling two different blockchain platforms to interact together, for example, how a Hyperledger network interacts and communicates with the Ethereum network.

Bibliography

- [1] Rabiya Khalid, Nadeem Javaid, Ahmad Almogren, Muhammad Umar Javed, Sakeena Javaid, and Mansour Zuair. A Blockchain-Based Load Balancing in Decentralized Hybrid P2P Energy Trading Market in Smart Grid. *IEEE Access*, 8:47047–47062, 2020.
- [2] Vida Ahmadi Mehri, Kurt Tutschku, Dragos Ilie, Simone Fischer-Hübner, Blekinge Tekniska Högskola, and Fakulteten för datavetenskaper. *Towards Secure Collaborative AI Service Chains*. 2019. OCLC: 1112211162.
- [3] Rune Jidrot and Gnanapalaniselvi Perumal. Workload Characterization and Performance Evaluation of a Blockchain Implementation for Managing Federated Cloud Resources - Assuming a Peer-to-peer Energy Management Use Case. page 84.
- [4] Deepak Puthal, Nisha Malik, Saraju P. Mohanty, Elias Kougianos, and Gautam Das. Everything You Wanted to Know About the Blockchain: Its Promise, Components, Processes, and Problems. *IEEE Consumer Electron. Mag.*, 7(4):6–14, July 2018.
- [5] Ericsson - can we put our trust in a decentralized marketplace?, March 2020. Accessed: 2020-08-20 - Online - <https://www.ericsson.com/en/blog/2020/3/decentralized-marketplace-cloud>.
- [6] Roman-Valentyn Tkachuk, Dragos Ilie, Kurt Tutschku, and Remi Robert. A Survey on Blockchain-based Telecommunication Services Marketplaces. *IEEE Trans. Netw. Serv. Manage.*, pages 1–1, 2021.
- [7] Adedayo Aderibole, Aamna Aljarwan, Muhammad Habib Ur Rehman, Hatem H. Zeineldin, Toufic Mezher, Khaled Salah, Ernesto Damiani, and Davor Svetinovic. Blockchain Technology for Smart Grids: Decentralized NIST Conceptual Model. *IEEE Access*, 8:43177–43190, 2020.
- [8] Parth Thakkar, Senthil Nathan, and Balaji Viswanathan. Performance Benchmarking and Optimizing Hyperledger Fabric Blockchain Platform. In *2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 264–276, Milwaukee, WI, September 2018. IEEE.
- [9] Arzoo Miglani, Neeraj Kumar, Vinay Chamola, and Sherali Zeadally. Blockchain for Internet of Energy management: Review, solutions, and challenges. *Computer Communications*, 151:395–418, February 2020.

- [10] Amanda Ahl, Masaru Yarime, Kenji Tanaka, and Daishi Sagawa. Review of blockchain-based distributed energy: Implications for institutional development. *Renewable and Sustainable Energy Reviews*, 107:200–211, June 2019.
- [11] Marcela T. Oliveira, Gabriel R. Carrara, Natalia C. Fernandes, Celio V. N. Albuquerque, Ricardo C. Carrano, Dianne S. V. Medeiros, and Diogo M. F. Mattos. Towards a Performance Evaluation of Private Blockchain Frameworks using a Realistic Workload. In *2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, pages 180–187, Paris, France, February 2019. IEEE.
- [12] Kurt Tutschku and Dragos Ilie. Symphony – supply-and-demand-based service exposure using robust distributed concepts. page 12, 2019.
- [13] Ragadeep Patha. Service management for p2p energy scenarios using blockchain-identification of computational efforts. page 88, 2022.
- [14] Juhar Ahmed Abdella and Khaled Shuaib. An Architecture for Blockchain based Peer to Peer Energy Trading. In *2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*, pages 412–419, Granada, Spain, October 2019. IEEE.
- [15] Creating a channel. https://hyperledger-fabric.readthedocs.io/en/release-2.2/test_network.html?highlight=deploycc#starting-a-chaincode-on-the-channel. Accessed: 2021-09-05.
- [16] Blockchain Infrastructure for the Decentralised Web - www.parity.io - accessed 2021-11-07.
- [17] Adamu Sani Yahaya, Nadeem Javaid, Fahad A. Alzahrani, Amjad Rehman, Ibrar Ullah, Affaf Shahid, and Muhammad Shafiq. Blockchain Based Sustainable Local Energy Trading Considering Home Energy Management and Demurrage Mechanism. *Sustainability*, 12(8):3385, April 2020.
- [18] Aparna Kumari, Arpit Shukla, Rajesh Gupta, Sudeep Tanwar, Sudhanshu Tyagi, and Neeraj Kumar. ET-DeaL: A P2P Smart Contract-based Secure Energy Trading Scheme for Smart Grid Systems. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1051–1056, Toronto, ON, Canada, July 2020. IEEE.
- [19] Zhi Li, Ali Vatankhah Barenji, and George Q. Huang. Toward a blockchain cloud manufacturing system as a peer to peer distributed network platform. *Robotics and Computer-Integrated Manufacturing*, 54:133–144, December 2018.
- [20] Aparna Kumari, Rajesh Gupta, Sudeep Tanwar, Sudhanshu Tyagi, and Neeraj Kumar. When Blockchain Meets Smart Trading in Demand *IEEE Network*, 34(5):299–305, September 2020.
- [21] Ioan Petri, Masoud Barati, Yacine Rezgui, and Omer F. Rana. Blockchain for energy sharing and trading in distributed prosumer communities. *Computers in Industry*, 123:103282, December 2020.
- [22] Vlada Brilliantova and Thomas Wolfgang Thurner. Blockchain and the future of

- energy. *Technology in Society*, 57:38–45, May 2019.
- [23] Asma Khatoon, Piyush Verma, Jo Southernwood, Beth Massey, and Peter Corcoran. Blockchain in Energy Efficiency: Potential Applications and Benefits. *Energies*, 12(17):3317, August 2019.
 - [24] Olamide Jogunola, Mohammad Hammoudeh, Bamidele Adebisi, and Kelvin Anoh. Demonstrating Blockchain-Enabled Peer-to-Peer Energy Trading and Sharing. In *2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE)*, pages 1–4, Edmonton, AB, Canada, May 2019. IEEE.
 - [25] Serkan Seven, Gang Yao, Ahmet Soran, Ahmet Onen, and S. M. Mueeen. Peer-to-Peer Energy Trading in Virtual Power Plant Based on Blockchain Smart Contracts. *IEEE Access*, 8:175713–175726, 2020.
 - [26] Ravi Kishore Kodali, Subbachary Yerroju, and Borra Yatish Krishna Yogi. Blockchain Based Energy Trading. In *TENCON 2018 - 2018 IEEE Region 10 Conference*, pages 1778–1783, Jeju, Korea (South), October 2018. IEEE.
 - [27] Ayman Esmat, Martijn de Vos, Yashar Ghiassi-Farrokhfal, Peter Palensky, and Dick Epema. A novel decentralized platform for peer-to-peer energy trading market with blockchain technology. *Applied Energy*, 282:116123, January 2021.
 - [28] Jun Duan, Alexei Karve, Vugranam Sreedhar, and Sai Zeng. Service Management of Blockchain Networks. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pages 310–317, San Francisco, CA, USA, July 2018. IEEE.
 - [29] Zhiyi Li, Shay Bahramirad, Aleks Paaso, Mingyu Yan, and Mohammad Shahidehpour. Blockchain for decentralized transactive energy management system in networked microgrids. *The Electricity Journal*, 32(4):58–72, May 2019.
 - [30] Zhitao Guan, Xin Lu, Wenti Yang, Longfei Wu, Naiyu Wang, and Zijian Zhang. Achieving efficient and Privacy-preserving energy trading based on blockchain and ABE in smart grid. *Journal of Parallel and Distributed Computing*, 147:34–45, January 2021.
 - [31] Shivam Saxena, Hany Farag, Aidan Brookson, Hjalmar Turesson, and Henry Kim. Design and Field Implementation of Blockchain Based Renewable Energy Trading in Residential Communities. In *2019 2nd International Conference on Smart Grid and Renewable Energy (SGRE)*, pages 1–6, Doha, Qatar, November 2019. IEEE.
 - [32] Meng Li, Donghui Hu, Chhagan Lal, Mauro Conti, and Zijian Zhang. Blockchain-Enabled Secure Energy Trading With Verifiable Fairness in Industrial Internet of Things. *IEEE Trans. Ind. Inf.*, 16(10):6564–6574, October 2020.
 - [33] Sudeep Tanwar, Shriya Kaneriya, Neeraj Kumar, and Sherali Zeadally. *ElectroBlocks* : A blockchain-based energy trading scheme for smart grid systems. *Int J Commun Syst*, page e4547, July 2020.
 - [34] Sung Jun Moon, In Hwan Park, Beom Suk Lee, and Jang Ju Wook. A Hyperledger-based P2P Energy Trading Scheme using Cloud Computing with Low Capability Devices. In *2019 IEEE International Conference on Smart Cloud (SmartCloud)*, pages

- 190–192, Tokyo, Japan, December 2019. IEEE.
- [35] Seyed Mojtaba Hosseini Bamakan, Amirhossein Motavali, and Alireza Babaei Bondarti. A survey of blockchain consensus algorithms performance evaluation criteria. *Expert Systems with Applications*, 154:113385, September 2020.
 - [36] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. page 9.
 - [37] Sunny King and Scott Nadal. PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake. page 6.
 - [38] Fan Yang, Wei Zhou, QingQing Wu, Rui Long, Neal N. Xiong, and Meiqi Zhou. Delegated Proof of Stake With Downgrade: A Secure and Efficient Blockchain Consensus Algorithm With Downgrade Mechanism. *IEEE Access*, 7:118541–118555, 2019.
 - [39] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance and proactive recovery. *ACM Trans. Comput. Syst.*, 20(4):398–461, November 2002.
 - [40] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, Srinivasan Muralidharan, Chet Murthy, Binh Nguyen, Manish Sethi, Gari Singh, Keith Smith, Alessandro Sorniotti, Chrysoula Stathakopoulou, Marko Vukolić, Sharon Weed Cocco, and Jason Yellick. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference*, pages 1–15, Porto Portugal, April 2018. ACM.
 - [41] Umit Cali and Austin Fifield. Towards the decentralized revolution in energy systems using blockchain technology. *SGCE*, pages 245–256, 2019.
 - [42] Lin William Cong and Zhiguo He. Blockchain Disruption and Smart Contracts. *The Review of Financial Studies*, 32(5):1754–1797, May 2019.
 - [43] Jei Young Lee. A decentralized token economy: How blockchain and cryptocurrency can revolutionize business. *Business Horizons*, 62(6):773–784, November 2019.
 - [44] Daniel Bergström, Ben Smeets, Mikael Jaatinen, James Kempf, Jonas Lundberg, Nicklas Sandgren, and Gaspar Wosa. A decade after its launch, blockchain is still the only internet-age technology that is able to facilitate online trust using mathematics and collective protocolling exclusively. page 12, 2019.
 - [45] Victor Clincy and Hossain Shahriar. Blockchain Development Platform Comparison. In *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, pages 922–923, Milwaukee, WI, USA, July 2019. IEEE.
 - [46] Mohammad Dabbagh, Kim-Kwang Raymond Choo, Amin Beheshti, Mohammad Tahir, and Nader Sohrabi Safa. A survey of empirical performance evaluation of permissioned blockchain platforms: Challenges and opportunities. *Computers & Security*, 100:102078, January 2021.
 - [47] Eung Seon Kang, Seung Jae Pee, Jae Geun Song, and Ju Wook Jang. A Blockchain-Based Energy Trading Platform for Smart Homes in a Microgrid. In *2018 3rd International Conference on Computer and Communication Systems (ICCCS)*, pages

- 472–476, Nagoya, Japan, April 2018. IEEE.
- [48] Chao Long, Jianzhong Wu, Yue Zhou, and Nick Jenkins. Peer-to-peer energy sharing through a two-stage aggregated battery control in a community Microgrid. *Applied Energy*, 226:261–276, September 2018.
 - [49] Alexandra Lüth, Jan Martin Zepter, Pedro Crespo del Granado, and Ruud Egging. Local electricity market designs for peer-to-peer trading: The role of battery flexibility. *Applied Energy*, 229:1233–1243, November 2018.
 - [50] Faizan Safdar Ali, Moayad Aloqaily, Omar Alfandi, and Oznur Ozkasap. Cyber-physical Blockchain-Enabled Peer-to-Peer Energy Trading. *Computer*, 53(9):56–65, September 2020.
 - [51] Jae Geun Song, Eung seon Kang, Hyeon Woo Shin, and Ju Wook Jang. A Smart Contract-Based P2P Energy Trading System with Dynamic Pricing on Ethereum Blockchain. *Sensors*, 21(6):1985, March 2021.
 - [52] Using CouchDB — hyperledger-fabricdocs master documentation.
 - [53] Types of peers. <https://hyperledger-fabric.readthedocs.io/en/release-2.2/network/network.html>. Accessed: 2021-09-08.
 - [54] Types of peers. https://hyperledger-fabric.readthedocs.io/en/release-2.2/create_channel/create_channel_overview.html?highlight=secure%20channel. Accessed: 2022-02-04.
 - [55] Riseul Ryu and Soonja Yeom. Blockchain-based renewable energy trading system with smart contract in a small local community. page 6, 2020.

