ARISTOTLE UNIVERSITY OF THESSALONIKI

Using blockchain technology for secure, transparent and traceable energy asset exchange

by

Emmanouil Kalyvas- Student ID: 93

A thesis submitted in partial fulfillment for the
Postgraduate degree

in the
Faculty of Sciences
School of Informatics
Data and Web Science

Supervising Professor: Dr. Athena Vakali

Date

Declaration of Authorship

I, Emmanouil Kalyvas, declare that this thesis titled, 'Using blockchain technology for secure, transparent and traceable energy asset exchange' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

One day or day one. You decide.

Unknown

ARISTOTLE UNIVERSITY OF THESSALONIKI

Abstract

Faculty of Sciences
School of Informatics
Data and Web Science

Postgraduate Degree

by Emmanouil Kalyvas- Student ID: 93

This thesis introduces a decentralized energy trading model implemented on a private blockchain network. It leverages blockchain technology in order to decentralize energy transactions and allow a local network to create its own energy market, controlled by the network participants. It gives the freedom to energy producers and consumers to make their own energy trading deals and define an energy price that is most convenient to both parties without any intermediates. These goals are achieved using a local Ethereum blockchain network, a decentralized application build on top of it and a web based user interface where users can interact with the decentralized application and perform energy transactions. The result of this research is a platform called PowerChain, that can be utilized to facilitated local energy transactions using blockchain technology. The source code of the platform is open source and can be found in the following GitHub repository:

`https://github.com/MnAppsNet/PowerChain`

Acknowledgements

I would like to express my sincere gratitude to those who have made this thesis possible. First and foremost, my advisors, Prof. Athina Vakali and George Vlahavas, for their invaluable guidance, encouragement, and expertise throughout my research journey. Their insightful feedback and unwavering support were instrumental in shaping this project. My thanks also extend to Aristotle University of Thessaloniki for providing me with the resources and support I needed to conduct my research. Finally, I would like to express my deepest gratitude to my family. Their love, encouragement, and understanding throughout this journey were more than words can express. I am especially grateful to my beloved wife, who is always by my side, supporting me psychologically in every step. Without the support and guidance of these individuals, this thesis would not have been possible. I am deeply grateful to all of them.

Contents

List of Figures

List of Tables

Abbreviations

Acronym	**W**hat (it) **S**tands **F**or
DER	Distributed Energy Resource
PV	Photovoltaic
P2P	Peer to Peer
P2G	Peer to Grid
VPP	Virtual power plant
CDA	Continuous Double Auction
TDDA	Time Discrete Double Auction
BMG	Brooklyn Microgrid
BFT	Byzantine fault tolerant
PBFT	Practical Byzantine fault tolerant
PoW	Proof of Work
PoS	Proof of Stake
PoA	Proof of Authority
DSO	Distribute System Operator
MCP	Market Clearing Price
TTP	Trusted Third Party
UTXO	Unspent Transaction Output
EVM	Ethereum Virtual Machine
wh	Watt hour
kWh	kiloWatt hour

Dedicated to my wife who is always by my side...

Chapter 1

Introduction

As the consumer level energy generation solutions become cheaper and more available, new opportunities arise in the energy trading sector. Consumers can produce and trade energy on a local network and distribution systems operators (DSO) are utilizing distributed energy resources (DERs) such as rooftop solar photovoltaic units (PV) and wind generators to offer affordable and eco-friendly energy solutions. In the past, electricity distribution primarily adhered to a centralized architecture where a central authority oversaw the distribution of energy generated by large-scale power plants, predominantly reliant on non-renewable sources like coal and natural gas. Additionally, the large-scale plants producing energy are usually located outside of cities and far away from consumers. This leads to a big energy waste during the transfer of energy over long distances. In recent years, ordinary consumers have begun contributing to energy generation through commercially available renewable energy generation units like rooftop solar PV systems. These consumer-producers, also referred to as ”prosumers” play a crucial role in energy community microgrids. Microgrids are decentralized systems that empower participants to engage in energy trading within local or regional markets. They can operate in both grid-connected (meaning they are also connected with the main power grid) or in island-mode (meaning they are completely autonomous). An isolated Microgrid has physical connections between the participants that are used to transfer energy and is not connected to the grid. A microgrid connected to the main energy grid can also trade energy virtualy with other migrogrids or remote participants and the actual energy exchange happens though the DSO. To facilitate regional microgrid transactions, the concept of a ”virtual power plant” (VPP) has been introduced. VPPs aggregate the capacities of various DERs, integrating diverse energy sources to ensure a reliable energy supply. [8, 17, 22]

Microgrids enable two types of energy distribution models, peer to peer (P2P) and peer to grid (P2G). In P2P model, energy is getting transacted internally in the participants of the microgrid, in a local market. In P2G model, energy is getting transacted from/to the microgrid to/from the energy grid based on the grid energy market. To construct a decentralized P2P and/or P2G

energy distribution model, it is necessary to ensure the four below propertied for the metered data:

- **Accuracy** of the logged energy assets.
- **Traceability** of the energy transactions, to ensure the origin of the generated energy and the correct flow of transactions.
- **Privacy** of each network participant. It shouldn't be possible to identify the total energy bill of a participant.
- **Security** of the energy transactions and the transactions ledger to prevent cyber attacks. [27]

Based on the above requirements, blockchain seems to be a viable option for the implementation of a decentralized energy distribution system.
Blockchain is an open-source, distributed ledger that can record transactions securely between two parties, in a verifiable and permanent way. The first running blockchain was implemented by Satoshi Nakamoto in 2008 and it was meant to be a P2P electronic cash system [21].

Chapter 2

Fundamentals

In this chapter, we will delve into the foundational aspects of blockchain technology and P2P energy trading. Initially, we will clarify the concept of a smart contract and explore its life cycle. Following that, we will introduce the most popular consensus mechanisms and various blockchain platforms, examining their distinctions. Finally, we will outline the P2P energy trading paradigm, drawing comparisons with traditional energy trading and we will explore different market clearing methods used on energy trading.

2.1 Blockchain Technology

A blockchain is a digital framework that functions as a decentralized, communal database. Within this framework, there exists an ever-expanding record of transactions organized chronologically. Essentially, this digital structure operates much like a ledger, capable of housing digital transactions, data records, and executable code. Transactions are grouped into larger units known as ”blocks,” each bearing a timestamp and cryptographic connections to prior blocks. This creates a linked sequence of records, establishing the order of events within the blockchain. Beyond defining the data structure itself, the term ”blockchain” is also widely used in academic literature to describe digital consensus systems, algorithms or application domains that are constructed upon these foundational structures.

The primary objective of blockchain technologies is to eliminate the necessity for intermediaries and replace them with a distributed network of digital participants who collaborate to authenticate transactions and preserve the ledger's integrity. In contrast to centralized systems, each member of the blockchain network maintains their own copy of the ledger or can access it in the public cloud. Consequently, anyone within the network can access the historical record of system transactions and verify their legitimacy, promoting a high level of transparency. With the removal of central management, the challenge lies in developing an

efficient method to consolidate and synchronize multiple ledger copies. The precise process of validation and ledger consolidation varies among different types of blockchains. However, in principle, network members compare their ledger versions through a process resembling distributed voting, ultimately reaching a consensus on the valid ledger state. These validation mechanisms are commonly referred to as distributed consensus algorithms.

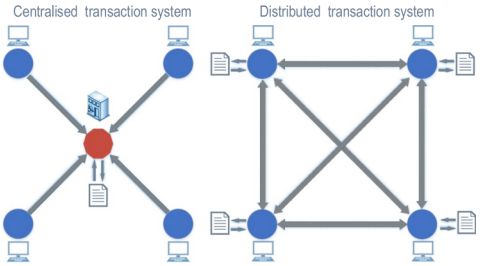

FIGURE 2.1: Centralized VS Distributed transaction systems [4]

Blockchain also uses cryptographic techniques like hash functions and public-key cryptography to secure the ledger. Cryptographic hash functions are mathematical algorithms or one-way processes that take an input and convert it into a specific-length output, such as a 256-bit string referred to as the hash output. Their effectiveness relies on the extreme difficulty of recreating the original input data from the hash output alone, ensuring collision resistance. Furthermore, blockchain employ public-key cryptography, an asymmetric cryptographic system where each user possesses two cryptographic keys, comprised of numeric or alphanumeric characters:

- A secret private key
- And a public key which can be shared with other network users

These keys are mathematically interconnected, permitting information encrypted with one key to be decrypted exclusively by its counterpart. This public-private key cryptography ensures:

- Authentication: The network can verify the sender's identity because only the sender's public key can decrypt the initial message, which is encrypted and digitally signed using the sender's private key
- Verification that a transaction originates from the claimed source. Messages processed with a recipient's public key can only be decrypted by the intended recipient possessing the corresponding private key.
- And authorization, confirming that actions are executed by users with the appropriate privileges

These security measures, along with other standard communication features such as data integrity and confidentiality, are achieved within blockchain systems through peer-to-peer communication and advanced cryptographic techniques. [4]

2.1.1 Smart Contracts

Blockchain can also be paired with smart contracts, realizing fully their potential. Smart contracts are executable programs capable of modifying a ledger automatically when specific conditions are fulfilled, often related to honoring agreements between parties. These contracts encode legal terms and constraints in a computer-readable language. They are self-enforcing and tamper-resistant, eliminating the need for intermediaries and reducing transaction-related costs. [4]
The life cycle of a smart contract, consists of 4 main phases: [34]

1. Creation of smart contract: Multiple parties negotiation on the rights, obligations and prohibitions of their contract. They reach a common agreement and a smart contract is build. The smart contract development is done using a smart contract programming language compatible with the blockchain that will be deployed. The smart contract need to be tests vigorously as logical code mistakes could result into a breach of the agreement between the parties and assets could be in danger.

2. Deployment of smart contract: The contract is deployed on a blockchain platform and can't be modified due to the immutability of the blockchain. The contract can be accessed by the parties through the blockchain platform. They can initiate the different actions coded by the contract, like for example locking an amount of digital tokens into the contract until a condition is met.

3. Execution of smart contract: The smart contract in effective, the contract conditions are evaluated and the contract statements are executed when the conditions are met.

4. Completion of smart contract: After the execution of the smart contract new states of all involved parties are updated and stored in the blockchain. At this stage the digital assets are moved between the parties based on the initial agreement.

The main challenge with the creation of a smart contract comes from the fact that blockchains are immutable. As soon as a contract is deployed on the blockchain it can't be modified. This is a big challenge for smart contract developers as they need to be sure that everything is going to work as expected from the first try. So it's very important to perform extensive functional tests on the smart contract code before its deployment. [34]

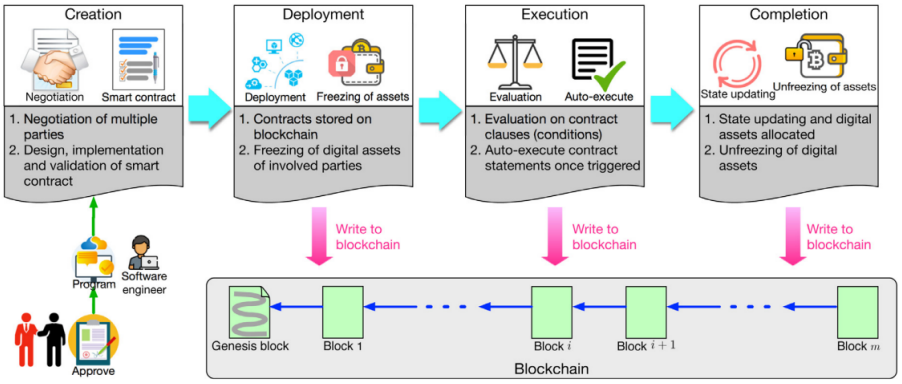

FIGURE 2.2: LifeCycle of a smart contract [34]

2.1.2 Public and private blockchain networks

A blockchain network or system can adopt various rules and system architectures based on its intended functionality and specific use cases. Typically, blockchain systems comprise of network users and validators. User nodes have the capability to initiate or receive transactions and maintain a copy of the ledger. Validators, in addition to read access privileges, are tasked with approving ledger modifications and achieving consensus across the network concerning the ledger's valid state. Depending on the system's configuration, there may be partial or universal access rights and validation rights.

Public blockchain systems are open to all internet users, while private blockchains restrict access to authorized participants only. Permissionless ledgers are fully decentralized and resistant to censorship, allowing any network member to participate in transaction validation. Conversely, permissioned ledgers grant write access rights to specific validator nodes, making the ledger modification process more controlled. In public and permissionless ledgers, users and validators have no prior knowledge of each other, relying on game-theoretic equilibria and rewards to encourage collaborative and trustworthy ledger management. This incentive structure often involves the expenditure of resources, such as computational power, electricity, or penalties, to discourage selfish behavior [4] Examples of public blockchains with a permissionless ledger are Bitcoin created by Satoshi Nakamoto [21] and Ethereum which was the first public blockchain that made use of smart contracts, proposed by Vitalik Buterin [33].

Private and permissioned ledgers, on the other hand, operate with known user identities, similar to "know-your-customer" (KYC) practices. Validator nodes are recognized and trusted to act honestly, eliminating the need for artificial incentives to ensure system operation. Consequently, private and permissioned ledgers can offer increased speed, flexibility and efficiency, although this comes at the cost of reduced immutability and censorship resistance [4]. The most popular permissioned blockchain is Hyperledger fabric proposed by IBM [5].

Aspect	Public Blockchain	Private Blockchain
Access Control	Open to anyone	Restricted access
Permission Level	Permissionless	Permissioned
Consensus	Mainly PoW & PoS	Various consensus mechanisms
Participants	Anyone can join	Approved participants
Privacy	Anonymous/Pseudonymous	Known identities (KYC)
Performance	Slower due to open access	Faster due to restricted access
Scalability	Limited	Potentially better
Example	Bitcoin, Ethereum	Hyperledger Fabric

Table 2.1: Public versus private blockchains

2.1.3 Blockchain consensus mechanisms

In this section, we will present some mechanisms used by blockchain technologies to achieve consensus.

2.1.3.1 Proof of Work (PoW)

The idea behind PoW, as used in Bitcoin, comes from 'Hashcash' which was created to prevent internet resources from being overwhelmed by denial of service attacks. In PoW, validators or miners compete to add a new block to the existing blockchain by solving a puzzle. They do this by generating a hash that begins with a certain number of zeros. To achieve this, they add a unique random number (nonce) to the block and calculate the hash of the block's information. This information includes details like the previous block's hash and a special hash for all the transactions in the block (Merkle tree). Every miner's goal is to produce a hash that's lower than a specific target value. Since miners can't predict or control the outcome, they have to try different combinations until they succeed. This process requires more and more computational power as you need more leading zeros. When a miner finally finds the hash that is less than the target, the block is added to the Bitcoin network. Other nodes check to make sure the transactions are valid and if they are, they accept the newly created block of transactions and the successful miner gets a financial reward. [4]

2.1.3.2 Proof of Stake (PoS)

Proof of Stake, differs from traditional systems by replacing computational work with a random selection method. In PoS, the likelihood of successfully mining a block is directly linked to the wealth of the validators. The chances of creating a block depend on how much cryptocurrency the nodes have invested in the system, essentially their coin ownership. This approach

potentially allows for faster blockchains with significantly lower electricity consumption and a reduced risk of a 51% attack. Unlike other systems, PoS doesn't require the constant creation of new coins to motivate validators. Instead, miners can be rewarded solely with transaction fees, making it less advantageous to invest heavily in specialized hardware like ASICs. PoS can also utilize game-theoretical mechanisms to discourage collusion and centralization, often penalizing dishonest or malicious behavior. However, PoS systems face a primary vulnerability known as the 'nothing at stake' problem. This means that validators can easily vote or claim financial rewards on multiple chains without significant cost. To address this, various solutions have been proposed, including penalties for validators who create blocks on multiple chains and automatic deductions of owned or deposited coins. Another approach is to punish validators for creating blocks on the wrong chain, similar in concept to PoW, where validators also incur the cost of electricity. The latter approach exposes validating nodes to greater risks, but it doesn't require them to be known in advance. [4]

2.1.3.3 Practical Byzantine Fault Tolerance (PBFT)

Byzantine Fault Tolerance (BFT) algorithms find their roots in the study of Byzantine faults, which was initially outlined by Lamport and his colleagues in a significant computer science paper. To put it simply, this issue revolves around a group of Byzantine generals, or in the context of blockchain, nodes, trying to come to an agreement on a coordinated course of action. Imagine these generals needing to synchronize their different army units to attack a fortress simultaneously (in the blockchain context, this means reaching a consensus on whether to validate a block or a set of transactions). The problem arises because the messages exchanged between these generals must travel through enemy territory, where they could get lost without anyone knowing (think of this as an unreliable, distributed network). Additionally, some of these generals might be traitors with a vested interest in disrupting the battle plan. They might send false or distorted messages, or simply not respond to messages at all. The main challenge is to make sure that the loyal generals can agree on the attack plan despite this, and that a small number of traitors don't lead them to adopt a flawed plan. In blockchain terms, this means that even if there are a few untrustworthy or potentially malicious nodes, they shouldn't be able to force the validation of a bad block or a set of transactions.
A practical approach for Byzantine Fault Tolerance was introduced by Castro and Liskov [9], for achieving Byzantine Fault Tolerance (BFT) with minimal computational overhead. Their approach, known as Practical Byzantine Fault Tolerance (PBFT), introduces the concept of primary and secondary replicas. In PBFT, secondary replicas are responsible for verifying the accuracy and responsiveness of the primary replica and can take over as the new primary if the original primary becomes compromised.
PBFT is better suited for use with trusted environment rather than public permissionless ledger.

Transactions are signed and verified by known validator nodes and they are considered valid when a sufficient amount of signatures is reached ans thus also consensus is reached. [4]

2.1.3.4 Proof Of Authority (PoA)

Proof of authority consensus algorithm relays on one or a group of peers to generate new blocks for the network. Network participants holding a special key are responsible to for generating all network blocks. If the majority of all peer with the special key sign a block, then it's getting accepted into the network. So the network is putting their trust to a group of authorized peers to generate and validate new blocks. The validators could be elected by a network vote and could be also possible to add new validators with the same mechanism. [4]

2.1.3.5 Comparison of popular consensus mechanisms

In the table below we compare the different properties of the consensus mechanisms we analyzed in the section above. Please note that for PoW, 51% of hash power is regarded as the threshold for one to gain control of the network but selfish mining strategy in PoW systems could help miners to gain more revenue by only 25% of the hashing power and thus the tolerated power of adversary is considered to be 25%.

Aspect	PoW	PoS	PBFT & variants	PoA
Node identity management	Permissionless	Both cases	Permissioned	Permissioned
Energy consumption	High	Low	Moderate	Low
Node scalability	> 1000	> 1000	< 100	< 100
Transactions per second	7-30	100-200	up to 110k	High
Tolerated power of the adversary	$< 25\%$* comput. power	$< 51\%$ stake	$< 33.3\%$ replicas	$< 51\%$ of validators
Security	High	Depended on participants stake	High within known participants	Relied on approved authorities
Example	Bitcoin	Ethereum	Hyperledger Fabric	VeChain

Table 2.2: Comparison of popular consensus algorithms [7]

2.1.4 Blockchain Platforms

In this section we analyze different available blockchain platforms. Each of the blockchain platforms have different architectures but all of them try to achieve three main goals:

- **Decentralization**: How censorship resistant is the network, not being able to be controlled or influenced significantly by a small group of network participants.
- **Security**: How secure and robust is the network against attacks from bad actors.
- and **Scalability**: The ability to process transactions, store data and reach consensus as more users are added to the network.

It is very difficult to have an architecture that fully achieves these three goals simultaneously. Different approaches improve on two of the goals in expense of the third. For example, Bitcoin achieves great security and decentralization but it lacks in terms of scalability.

2.1.4.1 Bitcoin

Bitcoin is a public and permissionless blockchain platform that uses the PoW consensus mechanism. Bitcoin protocol makes use of the unspent transaction output (UTXO) model. In the UTXO model, each transaction has one or more inputs and outputs. The inputs specifies the assets involved in the transaction which must come from a previous transaction output that has not been utilized in any prior transactions. Meanwhile, the output component is responsible for transferring these assets to a new address. The UTXO model serves a crucial role within the blockchain ecosystem by allowing network nodes to track the origin of assets involved in any given transaction. This enables them to verify whether the party initiating the transaction possesses the legitimate right to spend those assets. The movement of assets is recorded in a directed acyclic graph (DAG) between addresses.
UTXOs can be owned not only by public keys but also by scripts expressed in simple stack-based programming language. The script requests the data that satisfies the script from a transaction that tries to spend the UTXO owned by the script. The bitcoin scripting language is not Turing-complete and thus it has limited capabilities not allowing the execution of complex smart-contracts. [7, 28]
Bitcoin network participants can choose to be either clients or miners. Clients are able to send and receive transactions while miners are in charge of creating new blocks through the PoW consensus mechanism. All network participants keep a copy of the blockchain containing all the recorded transactions and can verify at any time the validity of a transaction. Transactions are transmitted to the neighboring network peers and are propagated through the network. If the transaction is found to be invalid, it is not broadcasted further to the network and the

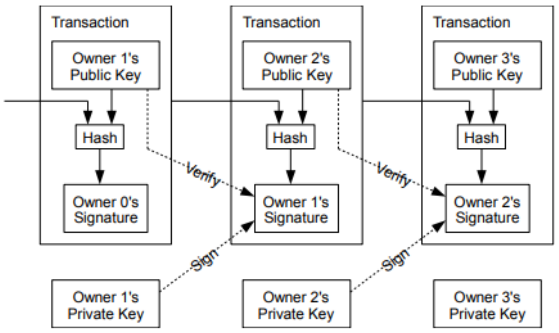

FIGURE 2.3: Bitcoin transactions schematic [21]

propagation stops. Miners store the incoming transaction in a transaction pool if they are valid. A block is constructed out of the transactions in the pool up to the upper block size limit which is about 1 MB. In a UTXO transactions the input amount can be greater that the output one leaving the rest of the amount as a fee to the miners incentivizing them to prioritize the inclusion of the transaction in the next block. [1, 7, 21, 28]

2.1.4.2 Ethereum

Ethereum is also a public and permissionless platform designed to build and use decentralized applications that run smart contracts. Ethereum smart contracts are written in a fully-fledged Turing-complete programming language (named Solidity). An Ethereum virtual machine (EVM) is hosted on each network node which is used for the execution of smart contracts. Ethereum was initially relying on the PoW consensus mechanism for the creation of new blocks but it was later transitioned in PoS.
Ethereum uses the account based transaction model where a global ledger is stored and maintained by all network parties based on the PoS consensus. Each block of the blockchain represents a different state of the ledger, this allows everyone to go through the states of the ledger and validate the origin an legitimacy of a transaction.
There are three different account types in Ethereum:

- **Contract Accounts**: A contract account is controlled by code executed by the EVM. They can create transactions with addresses stored in the contract or other contract accounts.
- **Externally Owned Accounts**: These accounts can initiate transactions to transfer Ether to other externally owned accounts, create new contracts or execute functions of existing contract accounts.
- **Miners/Stakers**: They collect unverified transactions, compute a valid state of the ledger, validate transactions, verify signatures and execute code. They collect transactions from

the unverified transactions pool and compute a new state for the ledger for the next blockchain block choosing transactions based on the fee included in them. They higher the fee the higher the incentive to choose a transaction.

When we refer to Ethereum we refer to the open and public Ethereum blockchain network although the Ethereum software is open source and can be also used to create a private blockchain network. [1, 7, 32]

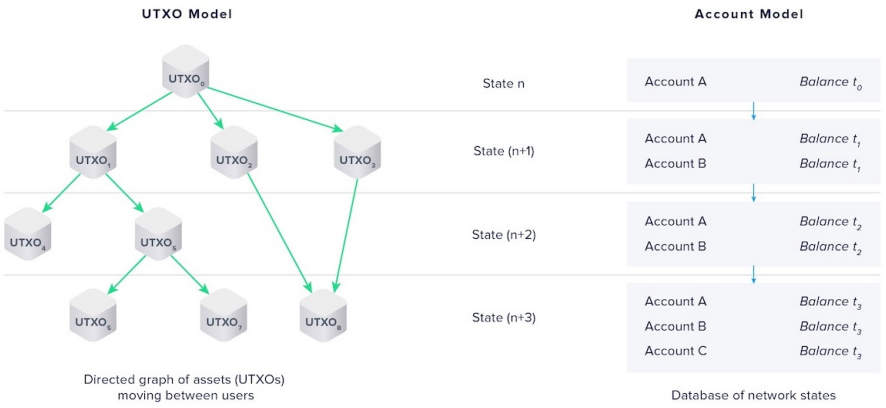

FIGURE 2.4: UTXO VS Account Model [1]

2.1.4.3 Hyperledger

Hyperledger is an open-source, global ecosystem of enterprise-grade blockchain technologies hosted by The Linux Foundation. Hyperledger frameworks are designed for permissioned blockchain applications and parties need to be authorized to join the network. In the hyperledger architecture, the following components are defined:

- Consensus Layer: Verify blocks of transactions and agree on the order.
- Smart Contract Layer: Processes transactions
- Communication Layer: Handles the P2P transport
- Data Store Abstraction: Handles different data-stores
- Crypto Abstraction: Handles cryptographic algorithms
- Identity Service: Registers and authenticates identities of the network
- Policy Service: Responsible for policy management
- APIs for interaction with other applications
- Inter-Operation service: Handles the inter-operation with other blockchain networks

Different Hyperledger frameworks have been constructed based on the Hyperledger architecture. One of the most popular Hyperledger platforms is Hyperledger Fabric.
Fabric functions as a distributed operating system designed for permissioned blockchains, running distributed applications that are coded in widely used programming languages like Go, Java, and Node.js. It maintains a secure record of its execution history using an immutable replicated ledger data structure, all while lacking any integrated cryptocurrency. In Fabric, smart contracts are called as Chaincodes and reading or writing the ledger is an operation referred to as a ”proposal”. Distributed application within Fabric consists of chaincodes. A chaincode contains the program code responsible for implementing the application's logic and operates during the execution phase. Chaincode represents the central element of a distributed application in Fabric and may be authored by a developer whose trustworthiness is not guaranteed. There are specific chaincodes designed for managing the blockchain system and overseeing parameters, collectively referred to as system chaincodes. Fabric nodes can have one of the three roles below:

- Client: Submit transaction proposals for execution.
- Peers : Execute transaction proposals and validate transactions. All peers has a copy of the blockchain ledger. Not all peers can execute all transaction proposals, only endorsing peers can. Execution of a chaincode depends on it's endorsement policy, which defines what peers and how many of them can validate it.
- Ordering Service Nodes (OSN): The ordering service receives transactions from all channels in the network, orders them chronologically on a per-channel basis, and packages them into blocks. The blocks will be delivered to peers on the channel for final validation and commitment. This design choice renders consensus in Fabric as modular as possible and simplifies the replacement of ordering service nodes.

In a private Fabric network, participants usually reach consensus through the PBFT mechanism. [6, 7]

2.1.4.4 Comparison between blockchain platforms

In this section we collected all the main differences between the three popular blockchain platforms analyzed above. In the following table the main characteristics of a blockchain platform are presented along with how each characteristic compares between the three popular platforms.

Aspect	**Bitcoin**	**Ethereum**	**Hyperledger Fabric**
Purpose	Digital Currency	Decentralized Apps	Enterprise blockchain
Target Audience	Individuals	Developers, Individuals	Enterprises, consortiums
Permission Restrictions	Permissionless	Permissionless	Permissioned
Access to data	Public	Public & Private	Private
Consensus	PoW	PoS	PBFT
Transaction Speed	Slow (7/sec)	Faster (15-30/sec)	Moderate speed
Native Currency	Bitcoin	Ether	None
Scripting	Not Turing-Complete, Limited Stack-based scripting	Turing-Complete, High-Level language Solidity	Turing-Complete, High-Level languages Go, NodeJS, Java

Table 2.3: Comparison of popular blockchain platforms

2.1.5 Trusted Authorities

Blockchain strives for a trustless and decentralized system but there are some aspects that still require some kind of a trusted authority to operate. Such example is the bridging of off-chain and on-chain data. A blockchain network has only access to what already exists in the blockchain and is completely isolated. In order to bridge off-chain data into the blockchain network, some kind of a trusted authority in needed. Such trusted authorities are also called oracles. Their responsibility is to retrieve, verify and feed off-chain data into the blockchain network which can then be used by decentralized applications. Another example of trusted authority in the blockchain network, is a group responsible of bridging off-chain assets into the blockchain network in the form of a token. Such assets could for example be a real world currency. The simplest way to achieve this is to store and lock the real world currency (effectively, moving it out of circulation) and minting the relevant amount of a corresponding token into the blockchain network. With the opposite procedure they can then lock the corresponding blockchain network token and unlock the real world asset. In general a trusted authority is a middle man that bridges real world data or assets into the blockchain network. The role of a trusted authority can sometimes be replaced by algorithms but with a cost on complexity and possibly security of the solution. [2][13]

2.2 P2P Energy Trading

Energy generation by prosumers can be unpredictable due to different variables which makes it difficult to predict. Such variables could be the amount of sun for PV units or the wind speed for wind turbines. In case of energy surplus, a prosumers has three possible options:

- To store the energy in batteries considering there is capacity left
- Export the excess energy to the electricity grid in market prices
- Or sell it to another energy consumer

I case of the P2P energy trading model, consumers and prosumers are buying and selling energy directly among them, on a local energy market, formed by the ask and offered energy prices. In such a local energy trading market, the network participants first share their generated energy locally and then trade with the retailers. The local market prices are typically set between the export energy price, which is the prices the prosumers are selling to the retailers and the retail price, which is the price to buy energy from retailers. In this way, both prosumers and consumers are benefiting as the former sell energy in a better price while the latter buy cheaper energy. [25]

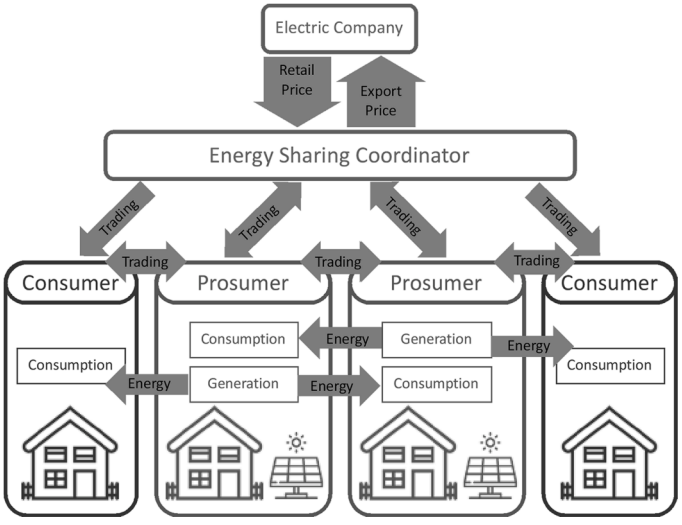

FIGURE 2.5: P2P energy trading model [25]

2.2.1 Conventional versus P2P energy trade

In this section, we will discuss what are the benefits of P2P energy trade, what its limitations and how it compares against the conventional energy trading model. Conventional energy trading is usually one-way, energy is transmitted from large-scale generators to consumers

while cash flow is the other way around, from consumers to energy providers. The market price can not be influenced directly from the consumers and it is set by energy providers based on market conditions. In contrast, P2P energy trade is a multi-directional procedures where both the energy and cash, flow between prosumers and prosumers. The energy price is set by the microgrid based on the participants' bids. P2P energy trading will also have some impact on the community. It creates new job opportunities for the maintenance and operation of the P2P systems, it increases social trust as the community works together for their energy needs and generally it creates a great attachment to the community as the participants have a direct connection with each other. [25]

Aspect	**P2P Energy Trade**	**Conventional Energy Trade**
Transparency	High transparency in pricing, sources and transactions	Less transparent, prices may be influenced by not easily traceable sources
Innovation and Technology	Relies on blockchain, smart contracts and IoT devices for trading	More traditional infrastructure, slower to adapt new technologies
Environmental Impact	Emphasizes renewable energy usage, sustainability and local energy production	May or may not prioritize renewable energy sources and sustainability
Regulatory Challenges	Can face hurdles dut to evolving energy market regulations	Complience with energy policies and regulations is more established
Scalability and Efficiency	Can be more scalable, efficient and adaptable to localized energy needs	May face challenges in adapting to rapid changes or catering to localized demands

Table 2.4: P2P versus Traditional energy trading

2.2.2 Microgrid Markets

Apart from the physical connections of a microgrid, there is also a market layer where all the participants' transactions happen. A microgrid market is usually classified by three main processes :

- Identity verification: Verifies the identity of the participant and confirms that he/she can participate in the microgrid market.
- Market opening: At market opening, sellers and buyers can submit their bids and the market automatically completes the transactions and does the price matching based on the trading rules.
- Market closing: The market closes and the participants can't submit any bids until it opens again [28]

In the following subsections, different market clearing methods will be presented.

2.2.2.1 Continuous Double Auction (CDA)

Most P2P trading platforms are making use of the continuous double auction design for their market layer. In a CDA market, buyers and sellers can submit their bids at any time of the trading period and market clears continuously. Buyers' bids are ordered in a descending order, from highest to lowest, while sellers' bids from lowest to highest. Then a "price first and time first" principle is applied and the orders are cleared based on the given bids. A CDA market order book example can be seen on the table below. [12, 28]

Order Book Before Clearing					
Buy Orders			**Sell Orders**		
Buyer	Requested Price units per energy unit	Provided energy energy units	Seller	Requested Price units per energy unit	Requested energy units
B1	10,5	5	S1	10,2	3
B2	10,0	3	S2	10,5	1
B3	9,9	4	S3	10,8	4
B4	9,8	5	S4	11,1	3
Order Book After Clearing					
Buy Orders			**Sell Orders**		
Buyer	Requested Price units per energy unit	Provided energy energy units	Seller	Requested Price units per energy unit	Requested energy units
B1	10,5	1	S3	10,8	3
B2	10,0	3	S4	11,1	1
B3	9,9	4			
B4	9,8	5			

Table 2.5: Order book in CDA market, before and after clearing

The higher bid of a buyer is called the outstanding bid while the higher bid of a seller is called outstanding ask. When the outstanding bid is less or equal to the outstanding ask, a transaction is happening with the transaction price being equal to the average of the two prices. This process continues until there the outstanding bid is more than the outstanding ask. This process is called one round of transactions and at least one transaction occurs during that process. In the above table we can see that one round of transactions is performed which includes two transactions, the one between B1 and S1 for 3 energy units and the one from B1 to S2 for 1 energy unit. [28]

2.2.2.2 Time Descrete Double Auction (TDDA)

In a TDDA market, transaction happens at discrete time intervals rather than continuously. Specific time intervals are specified when the participants can make their buying or selling bids. At the end of this time interval, the system matches buy and sell order based on the outstanding bids and the outstanding asks, like the CDA market. TTDA is more or less and extension of CDA, it has the same mechanics with the difference that the order book is getting cleared at the end of a discrete, predefined time period. [17]

CDA & TDDA Limitations

One of the key drawbacks of the CDA system is its inability to effectively handle energy products with inter-temporal commitments. This complexity arises from market bids that rely on various time periods within the market's timeframe. In P2P energy markets, the inter-temporal impact of bids can be attributed to the unique characteristics of activities like charging and discharging batteries or running appliances like washing machines and dryers. Essentially, the CDA system can only handle bids that specify a single quantity and price within a single time step. Consequently, this restricts prosumers to just one specific time slot for buying and selling energy. These limitations also impact the participation of flexible local devices, such as battery storage systems, which often require specific charging patterns or continuous operation to maintain their efficiency and prolong their lifespan. [12]

Chapter 3

Literature Review

In this chapter we will present different energy trading models proposed in literature and their limitations. On each of the presented models, we will focus on two different aspects:

- **The market layer**: It comprises the market's allocation and payment rules and defines a clear biding mechanism. Its main purpose is to allocate the available traded energy based on the the bids of energy buyers and sellers. It defines how the market is getting cleared.
- **The blockchain/information layer**: It connects all market participant and provide the market platform with equal access to everyone in the microgrid, to avoid discrimination. It's integrated with the market layer to ensure data security and privacy. In the following chapters different blockchain technologies, used on P2P energy trading, will be presented. [12, 17]

3.1 A Hyperledger Fabric Implementation

In this section we will describe a P2P energy trading platform proposed by M. Pipattanasomporn et al [22]. This study focuses mainly on the blockchain layer implementation and uses a simplistic market clearing method. It focuses mainly on photovoltaic (PV) energy trading and makes use of the open-source Hyperledger Fabric blockchain framework to implement a private blockchain.

3.1.1 Market Layer

The proposed energy market is split in time periods of an hour. Each hour, network peers submit their buy and sell bids for energy that will be traded the next hour. The market clearing

is done based on the market clearing price (MCP). The MCP used for this study was the average bid price of the buyers:

$$MCP_t = \frac{\sum_{i=1}^{N_t} \beta_{n,t} \times kWh_{n,t}}{\sum_{i=1}^{N_t} kWh_{n,t}}$$

Where,

MCP_t : Market clearing price

$\beta_{n,t}$: Bid of buyer n at hour t (token/kWh)

$kWh_{n,t}$: Number of kWhs to be purchased from buyer n at hour t

N : Total number of buyers

The matching of buying and selling bids are done with the first-in, first-out approach (FIFO) and priority is given to buying offers with the highest value ($\beta_{n,t} \times kWh_{n,t}$). The market is cleared every hour t using the MCP_t, calculated with the formula above. [22]

3.1.2 Blockchain Layer

For the blockchain layer, a private blockchain network approach is used. Each participant need to register and be authorized to use the energy trading network. For the registration, the participant need to provide their first and last name, their email which acts also as a unique identifier of the participant and their initial balance in the token/currency used for energy exchange. A certificate authority is responsible to register new identities and issue enrolment certificates for participants.

In this design, two type of assets and four type of transactions are defined.

Assets:

- The **PV** asset: Each participant that has a PV and wants to sell their excess energy generation, need to register their PV unit as an asset in the blockchain. This asset contains a unique identifier for the PV unit and the identifier of the participant owning the unit.
- The **kWhlisting** asset: It is automatically generated by the system every hour and contains a listing id, a state, a list of all the selling offers and another one for the buying offers. The listing id consists of the date and time when the asset created and the state defines the state of the listing which can either be at the state of accepting offers from buyers and sellers or at auction closed state where the auction stops. In simple terms, this asset gathers all the buying and selling bids for a specific hour.
- Energy generation and consumption needs should be known ahead of time. The energy trading negotiations are done one hour before the actual energy generation and consumption. This could lead in surplus energy that was not consumed or required energy that was not produced.

Transactions:

- The ”**AcceptOfferBroadcast**” transaction: It is automatically send by the system to all network participant at the start of each hour, to inform them that they can start bidding.
- The ”**PVOffer**” transaction: After receiving the ’AcceptOfferBroadcast’ transaction, each participant that owns a PV unit can send this transaction to define how much excess energy they want to trade (kWh) and what is their reserve price (token).
- The ”**BUYOffer**” transaction: Once receiving the ’AcceptOfferBroadcast’ message, all participants can send the ”BUYOffer” transaction, to define the amount of energy they need (kWh) along with the price (token/kWh) they want to pay.
- The ”**CloseBidding**” transaction: It is automatically send by the system at the end of each hour to inform the participants that the bidding is closed and the market clearing process initiated.

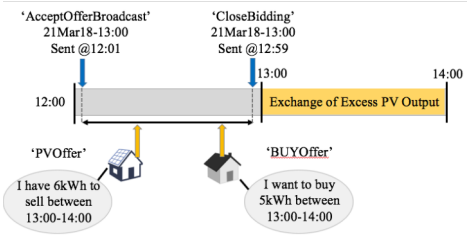

FIGURE 3.1: Transaction timeline proposed by M. Pipattanasomporn et al [22]

The market clearing process is encoded in a smart contract code that is embedded in the ”Close-Bidding” transaction. When this transaction is executed the smart contract code is executed and the market is cleared as described in the market layer section above. [22]

3.1.3 Limitations

This design has some limitations that need to be considered:

- The way the market is designed in hourly intervals, doesn’t allow for inter-temporal commitments. This means that only energy needs of the next our can be covered and no energy trading planning for different time periods could be made.
- The market clearing price algorithm is very simplistic, alternative approaches need to be explored

- Prosumers might not be truthful about their hour ahead energy production or the forecasting might have some errors. This can lead to violations to the energy trading commitments. Such a system couldn't be self sustained, it needs to be connected to the grid to cover any extra energy needs.

3.2 DeTrade and DeMarket

DeTrade is a decentralized P2P energy trading platform proposed by Ayman Esmat et al [12]. It allows the energy trading participants to trade energy among them, while guarantying privacy and security of their identities and market bids. It makes use of novel P2P energy trading market called DeMarket which is described in more detail in the **Market Layer** subsection below.
In the proposed design, a, AI smart agent is assigned to each network peer. This AI agent has the following responsibilities:

- Monitors the energy consumption and production of the peer by having access directly to the smart meters.
- Makes bids in the DeMarket
- Clears the market
- Makes Contact with the blockchain layer

The market results are stored safely and immutably on the blockchain layer after it is cleared. A smart contract is also implemented on the blockchain layer that makes sure the money are transferred only if the energy was really traded. The smart contract is the contact point between the market and the blockchain layers. [12]

3.2.1 Market Layer - DeMarket

DeMarket is a short-term parallel pool-based market that allows market participants to trade various market products. It has an uniform pricing mechanism and is divided in multiple periods of market clearing and energy delivery. The time aspect of the market, denoted as T, involves two specific timeframes $T = \{T_r, T_x\}$:

- The '**delivery horizon**' T_x: The delivery horizon is further divided into X equal consecutive time intervals, with each interval lasting anywhere from minutes to hours. These consecutive time intervals enable the trading of market products that depend on time. In each interval, sellers provide energy, while buyers acquire energy.

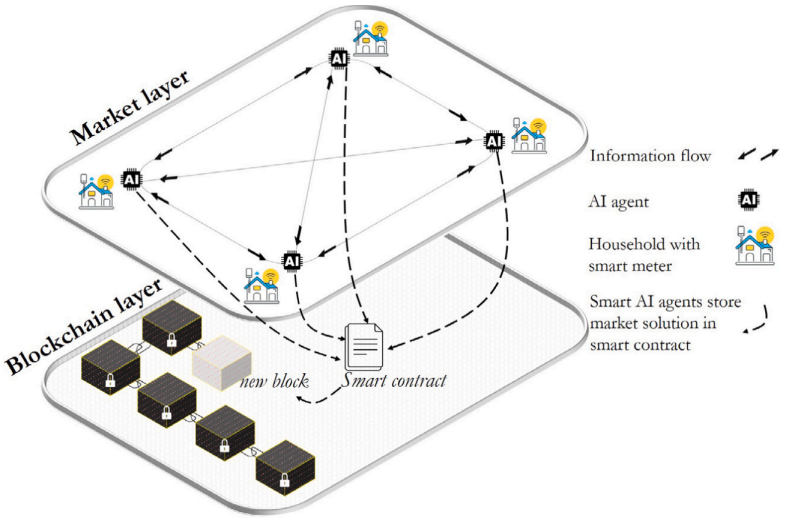

FIGURE 3.2: DeTrade Architecture [12]

- And the '**clearing horizon**' T_r : In the clearing horizon, the DeMarket operates through R consecutive stages. During each stage, all delivery intervals are simultaneously settled, which is why it's referred to as a 'parallel auction'. This process accounts for all time-related dependencies. Because the DeMarket is decentralized, achieving the optimal clearing of all delivery intervals requires several stages of information exchange among prosumers. In a centralized peer-to-peer market, it's possible to clear all delivery intervals optimally in one go.

The DeMarket's chronological sequence begins at time step $-R$ with the clearing horizon, which concludes at time step -1. Subsequently, the energy delivery horizon begins at time step 1, and the market concludes at time step X. The clearing horizon stages are defined as $T_r = \{-R, \ldots, -1\}$, and the delivery horizon intervals as $T_x = \{1, \ldots, X\}$. The final market clearing results are realized and stored in a smart contract on the blockchain between the clearing horizon at -1 and the transition to the delivery horizon at time step 0.
By splitting the delivery period in multiple time periods, we can define multiple different energy products, the authors of this paper, propose the following two:

- **Single Product**: It reflects bids that are valid only for a single delivery period. This product don't allow inter-temporal dependencies as it relates only to a single delivery period. This products allows also for multiple block of prices and quantities to be traded, so a different price can be set based on the consumed quantity for a single delivery period.
- **Continues Product**: This is a bid that is valid for the whole delivery horizon and is valid for all the periods. This one is for inter-temporal energy trading. This is a "take it all or leave it" type of bid, meaning that the whole available energy is traded over the whole delivery horizon with no partial bids allowed. [12]

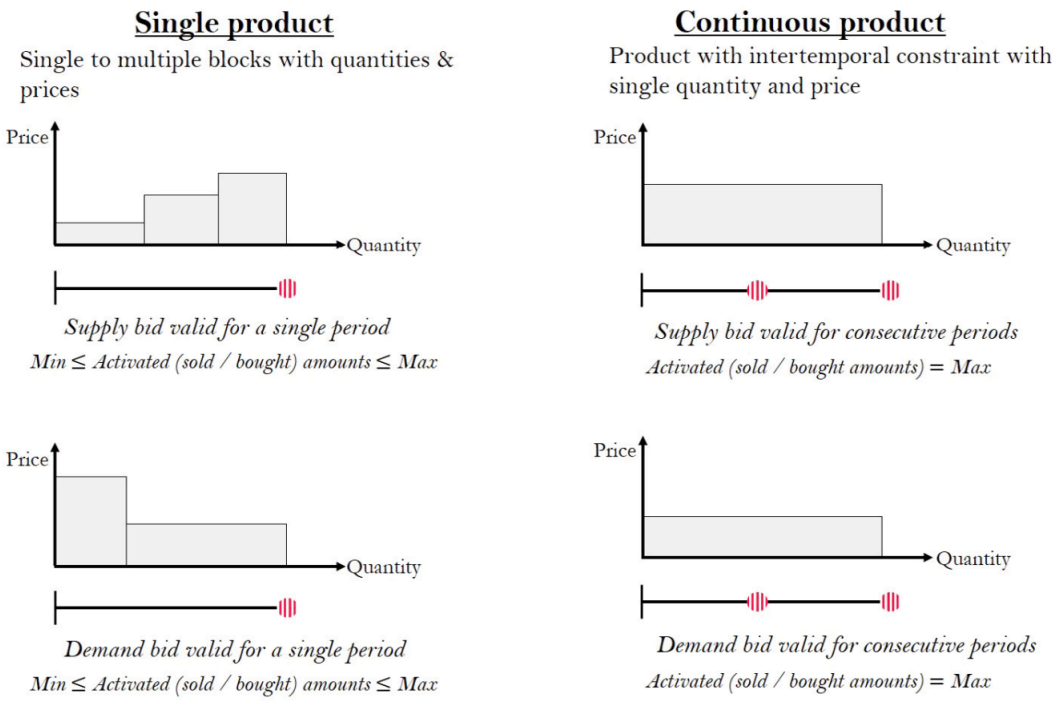

FIGURE 3.3: DeMarket Market Products [12]

This market approach allows for many different type of bids and gives the opportunity to the peers to make more efficient trades. However, it becomes more difficult to match the different bids efficiently. For this reason, M marginal prices are calculated P^x, one for each delivery period x. The market model uses a uniform pricing mechanism so on each delivery period x the cost of energy is P^x uniformly among the peers. These M marginal prices are calculated based on a social welfare maximization problem, defined by the summation of utility of all buyers minus the the summation of cost of all sellers, across all M delivery periods. The authors of DeMarket, formulate the optimization equation and uses the Ant-Colony optimization algorithm to calculate the mentioned prices P^x, in the clearing horizon steps $T_r = \{-R, \ldots, -1\}$. [12]

3.2.2 Blockchain Layer

DeTrade uses a private and permissioned blockchain layer to achieve three main goals:

- To store security the market clearing results
- To represent monetary value in the form of tokens (tokenization)
- And to automatically transfer digital tokens when the energy traded is actually delivered or consumed.

As it uses a private blockchain, a trusted third party (TTP) is needed which will be able to register new participants to the network and mint new tokens backed by real world currencies. To achieve these goals, it defines the following smart contract methods:

Method	Scope	Allowed Callers	Description
registerHousehold	public	TTP	Add a new network peer
mintEuroToken	public	TTP	Add tokens on blockchain backed by Euro
balanceOf	public	Smart Agents + TTP	Get the balance of a peer
initializeRoles	public	Smart Agents	Define the role (buyer/seller) of a peer on each clearing horizon
storeClearingResults	public	Smart Agents	Store the clearing results after the clearing horizon
getTotalPrice	public	Smart Agents	Returns the total cost of energy
receivedEnergy	public	Smart Agents	Buyers invoke this method to confirm energy is received and initiate the token exchange
isRegisteredHousehold	private	-	Checks if a peer is already registered
resetClearingResults	private	-	Resets the clearing results
validateAllClearingResults	private	-	Check if clearing results are valid
selectBestClearingResult	private	-	Selects the best clearing result
redistributePoolFunds	private	-	Moves the tokens between peers based on delivery results

Private methods cannot be invoked directly by a transaction

Table 3.1: Smart Contract methods used by DeTrade.

The blockchain technology used is Hyperledger Burrow which uses the BFT consensus algorithm and can tolerate up to $1/3$ of the participants being bad actors. [12]

3.2.3 Limitations

The two main limitation of this design are the following:

- The assumption that prosumers report their real energy generation and that their forecast is error free. In reality, due to the irregularity of decentralized energy resources, the energy production forecasting is prone to errors. This leads to violations of energy trading commitments and the migrogrid is required to be connected to the grid in order to cover any extra energy needs at market cost.
- The strategic misreport of quantity and pricing bids. A peer could potentially try to manipulate the clearing price by placing strategic bids. Game theory could potentially help on creating a strategic-proof bidding process that is close to the optimal solution of the social welfare maximization problem.

3.3 NRGCoin

NRGCoin is a virtual currency backed by renewable energy production. It is a mechanism that facilitates the integration of renewable energy in the local grid by making it more profitable for producers and utilities and cheaper for consumers and governments. It allows local renewable energy producers to earn NRGcoins directly from their energy output, without depending on the market price or extra battery storage. The NRGcoins can then be traded on an open market for their monetary equivalent at any time.
Information about the local energy production and consumption is sent every 15 minutes from the smart meters of the peers to the street level low-voltage energy substation of the DSO. This information is used to determine the rates at which prosumers are rewarded with NRGCoins and consumers are billed for energy withdrawal. These rates are determined buy the street-level substations that follows the NRGCoin protocol. [19]

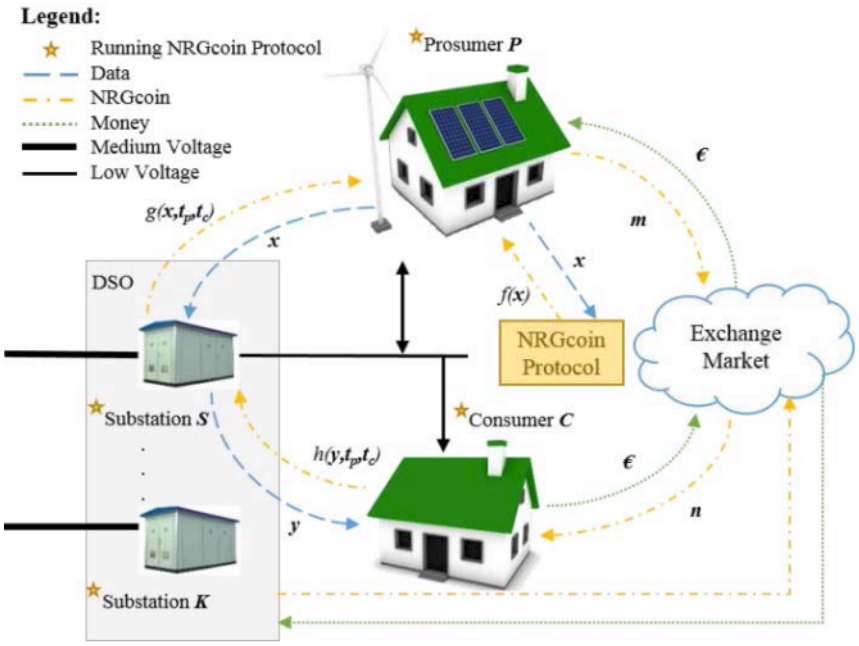

FIGURE 3.4: NRGCoin Schematic Setup [19]

3.3.1 Market Layer

NRGCoin is traded on an open currency exchange market. It can be traded openly between peers on the price they agree on. The price of the coin is influenced by its value which is influenced by the rate at which coins are rewarded to prosumers for the energy they inject to the grid. The NRGCoin network protocol is comprised by the following phases:

- A prosumer P generates x amount of energy and broadcasts this information to the network

- The network peers are then updating the the public record of P with $f(x)$ NRGCoins. $f()$ is the NRGCoin generation function that produces new NRGCoins that enters the circulation.
- The local substation S which is connected to P measures what is the total production t_p and total consumption t_c in that 15 minute slot.
- DSO transfers $g(x, t_p, t_c)$ NRGCoins to the balance of P. $g()$ is the production price function defined by DSO
- P receives $f(x)$ NRGCoins from the NRGCoin protocol to ensure new money enters the system and $g(x, t_p, t_c)$ NRGCoins to motivate agents to match consumption with production.
- P trades its acquired NRGCoins with a consumer C based on their bids. The exchange can happen on any exchange platform with the NRGCoin listed each of which can potentially use different market clearing techniques.
- C pays $h(y, t_p, t_c)$ NRGCoins to local substation S' it's connected to, to consume y amount of energy (S and S' could potentially be the same but it's not necessary).

The function $g(x, t_p, t_c)$ should be defined in a way to motivate the prosumers to produce energy based on the demand. Too much or too little energy production should result in lower rewards in order to much demand. A bell curve reward function could motivate the prosumers to produce just enough energy:

$$g(x, t_p, t_c) = \frac{x \times q}{e^{\frac{(t_p - t_c)^2}{a}}}$$

Where q is the maximum reward a prosumer can receive for their input energy x when the total energy supply t_p mather total demand t_c and is defined by the DSO. The parameter a is used for scaling.
Function $h(y, t_p, t_c)$ is defined in a way to motivate the consumers to shift their consumption on times of high energy production. It is defined as:

$$h(x, t_p, t_c) = \frac{y \times r \times t_c}{t_c + t_p}$$

Where r is the maximum cost of energy defined by DSO, when energy supply is low ($t_p \to 0$). When $t_p >> t_c$, $h(x, t_p, t_c) \to 0$ which means that in times of high production, energy is getting cheaper, incentivising consumers to consume at that times.

3.3.2 Blockchain Layer

NRGCoin is implemented as a network of hardware gateway devices interacting with the existing residential energy installation. They measure through sensors the energy input and output and send these information regularly to the NRGCoin smart contract which implements the NRGCoin protocol. The smart contract can be either running on an existing smart contract-capable public blockchain like Ethereum or on a network of the gateway devices supported by blockchain. NRGCoin is deployed in reality on the Ethereum blockchain. [20]

3.3.3 Limitations

In this design we identify two main limitation:

- The NRGCoin model is heavily depended on the current energy transfer installation. This creates a dependency to DSO and reduces the decentralization of the energy trading process
- Energy price can be manipulated by DSO

3.4 Energy trading on CDA market

This study proposes a model, leveraging blockchain and continuous double auctions. Buyers and sellers dynamically adapt prices within the auction to meet market fluctuations, with secure and transparent transactions recorded on the blockchain. In the following chapter, we describe in more details the market and blockchain layer of the proposed model and mention some of its possible limitations.

3.4.1 Market Layer

In this model, the CDA mechanism is used to clear the market bids. On each transaction period, prosumers and consumers submit their offering and asking bids to the network. Based on the CDA mechanism the bids are matched and settled through the blockchain system. Unmatched peers adjust their bids according to a pricing strategy in order to find a match. During the matching process, the market continuously discloses transaction information to the network peers, including transaction price, outstanding bid and outstanding ask. These information can then be used to formulate a biding strategy.
When the final matching are done and settled through the blockchain, the energy producer

receives the agreed payment through the blockchain and the consumer receives a digital certificate of electricity transaction. The digital certificate of electricity transaction, gives the consumer the authorization to use the corresponding electricity. [28]

3.4.2 Blockchain Layer

For this energy trading approach, the bitcoin protocol is used to settle the microgrid transaction. Consumers transfer bitcoin to the energy generators as a payment and the energy generators transfer a special energy token as a digital certificate for the sale of electricity. The transaction settlement process consists of the following steps:

- The energy seller, issues a special energy token which is proportional to the transaction volume in the blockchain network.
- The seller sends a multi-signature redeemScript to consumer which contains the public key of the seller, the buyer and a trusted third party.
- The buyer creates a UTXO which has as an input the UTXO of the previous transaction and as an output the transfer of bitcoins to the given redeemScript along with a digital signature for verification.
- The seller creates a UTXO which has as an input the UTXO of the previous transaction and as an output the transfer of the special energy token to the public address of the buyer along with a digital signature for verification.

The multi-signature redeemScript is used to make sure that all both parties abide by their energy trade agreement. A trusted third party is also included in the multi-signature in order to resolve any disputes between the parties. In case the seller doesn't transfer the special token to the buyer, the buyer doesn't sign the multi-signature transaction and the deal is rejected. The trusted third party always signed the transaction except when there is a dispute between the parties at which case the transaction need to be investigated before resolving the transaction. The special energy token issues by the sellers, prove the ownership of a certain amount of energy which can be transferred to an energy buyer to give them the permission to spend it. In the bitcoin protocol, bitcoins are the only transaction object so in order to be able to have these special tokens in the network, special labels can be added to bitcoins to distinguish them from common bitcoins. These specially labeled coins are called colored coins [31]. [28]

3.4.3 Limitations

The main limitations we notices with this model are the following:

- The CDA mechanism is not compatible with transactions that have inter-temporal commitments. This means that only energy needs of the next trading period can be covered and no energy trading planning for different time periods could be made.
- Prosumers might not be truthful about their hour ahead energy production or the forecasting might have some errors. This can lead to violations to the energy trading commitments. Such a system couldn't be self sustained, it needs to be connected to the grid to cover any extra energy needs.
- Bitcoin might not be the best suit for this use case as it has a low block finality. of course the protocol could be adjusted on a private network to meet our needs where also a different consensus mechanism could be considered.

3.5 Comparison

Based on the reviewed literature, we can distinguish two different approaches to the energy trading problem. One is the direct energy trading and the other is the pegging of a token to the energy value. In the first approach, production and consumption forecasting is needed in order to match energy trading needs of the peers while on the other an energy storage solution is needed.

Aspect	**A Frabric Implementation**	**DeTrade**	**NRGCoin**	**Energy trading on CDA market**
Blockchain	Private Hyperledger Fabric	Private Hyperledger Burrow	Public Ethereum	Public Bitcoin
Consensus	BFT	BFT	PoS	PoW
Market	MCP (average buy offer)	DeMarket pool-based	Open currency exchange	CDA
Energy Trading	Direct	Direct	Indirect Backed by MBRCoin	Direct

Table 3.2: Comparison of mentioned approaches

3.6 Summarization

The investigated four distinctive models for decentralized energy trading, leveraging blockchain technology. We discussed about a private Hyperledger Fabric blockchain implementation that employs Byzantine Fault Tolerance (BFT) consensus for hourly direct energy trading, with limitations in inter-temporal commitments and potential forecasting inaccuracies. A private

Hyperledger Burrow blockchain implementation called ”DeTrade”, focusing on direct energy trading with an emphasis on inter-temporal commitments. ”NRGCoin” which follows a different energy trading paradigm, pegging a virtual currency to renewable energy production, though it faces limitations in reliance on existing energy transfer infrastructure. Finally, we also discussed about an energy trading implementation employing the CDA market mechanism. It uses Bitcoin protocol with the concept of colored coins to achieve direct energy trading but it is facing challenges with Bitcoin's low block finality. Each model offers distinct approaches to address decentralized energy trading complexities, with the choice contingent on factors like decentralization preference and specific energy market requirements.

Chapter 4

Implementation Framework for an energy trading platform using blockchain technology

In this chapter we will go through our proposed energy trading platform architecture. We will analyze the market and blockchain layers of our implementation and will discuss about the different roles in the energy trading network.

4.1 Overview

After reviewing the current literature in regards to energy trading using blockchain technology, we introduce a novel energy trading platform that tries to improve upon the existing implementation. The main limitation we identified with the existing energy trading platforms is the need for an efficient energy generation and consumption forecast. Most prosumers make use of renewable energy generation units like PV panels which are very sensitive to weather conditions. This makes it very challenging to produce good energy generation predictions which can cause issues with a direct energy trading approach. The actual energy generated during the energy consumption period might be less than the energy trading agreement made with a consumer during the trading period. This causes the need for the microgrid to be connected to the main grid to buy any extra energy need or even sell any excess energy that was not actually consumed. The model we will introduce in the next sections replies on an indirect energy trading paradigm and introduces the role of storage provider in the microgrid. This new role will be responsible to store the energy produced by the network.
The main challenges we are trying to address with our proposed model are the following:

- **Decentralization**: We want the network to be autonomous and capable of operating by its own with no intermediates. No single authority should should be able to control the network's energy market prices which should only be influence by the network's energy trading needs.
- **Autonomy**: We want to create a model which can be as autonomous as possible with minimal energy trading transactions with the main energy grid.
- **Forecasting Errors**: We want to minimize any energy generation and consumption forecasting errors as much as possible to avoid any breach in energy trading agreements.
- **Fault Tolerance**: We need to create an architecture which is tolerant in energy storing and delivering failures.

We call our proposed energy trading platform "PowerChain" and we will refer to it as such in the following sections.

4.2 PowerChain Components

PowerChain consists of two main layers:

- The **Blockchain Layer**: It is responsible for the digitalization and transferring of energy produced by the network.
- And the **Market Layer**: It is responsible for the trading of digital energy assets with legal currencies.

These two layers are completely separate from each other and could potentially operate separately by themselves. The blockchain layer is used to digitize the available energy and makes it possible to transfer the energy assets while the market layer is used to trade the digitized energy assets for legal currencies with the only link between them being the digitized energy assets. In the following chapters more details will be presented for the different aspects of PowerChain.

4.2.1 PowerChain Layers

PowerChain defines two different type of layers, the physical layers and the information layers. The information layer works on top of the physical layer and its responsibility is to orchestrate the physical layer interactions. We recognize three physical layers and two information layers which are analyzed below.

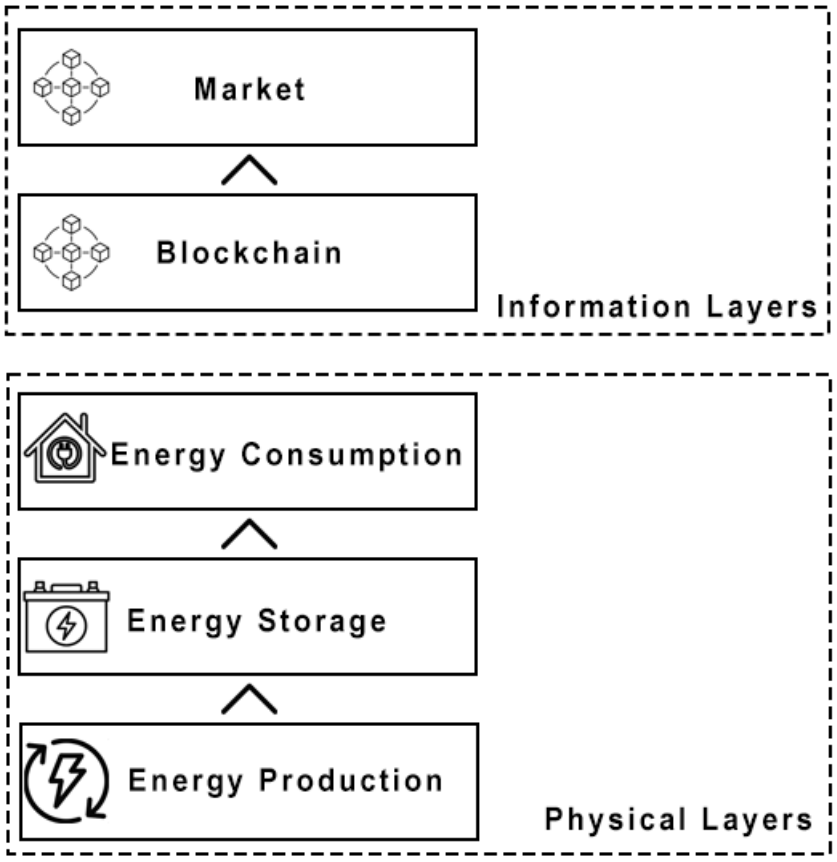

Figure 4.1: PowerChain Layers

Physical Layers:

- **Production Layer**: This layer consists of all the energy production units that exist in the microgrid. It's a very crucial layer of the architecture as it produces the network's energy.
- **Storage Layer**: This layer is comprised by all the energy storage units of the microgrid. The energy produces by the production layer is stored in the storage layer for later consumption.
- **Consumption Layer**: It is made of all the peers that consume energy from the storage layer.

Information Layers:

- **Blockchain Layer**: The blockchain layer is the heart of the PowerChain protocol. It facilitates all the necessary functionality to digitize the produced energy and and make it tradable through smart contracts.
- **Market Layer**: In the market layer, the digitized energy is getting traded between the peers. This layer could be completely separated from the blockchain layer. It consists of all the possible ways the digitized energy could be traded. PowerChain provides also a market platform that can be used initiate trades.

4.2.2 Network Roles

In literature, two roles are usually defined in an energy trading network, the one of consumer and the one of both producer and consumer called prosumer. Except from the roles of energy producer and consumer we also define the role of energy storage provider. In more detail:

- **The consumer**, is buying and consuming the available energy of the network. The available energy of the network is stored by the storage providers.
- **The producer**, is producing network's energy using energy generation units like PVs. The produced energy is send to the storage providers for storage.
- **The storage providers**, are receiving the energy from producers and they are storing it in batteries.

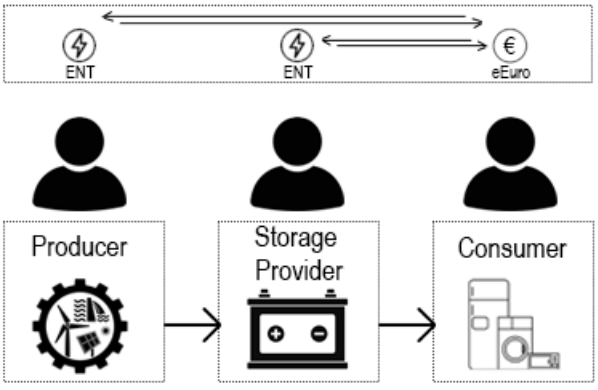

Figure 4.2: PowerChain energy exchange diagram

Each network participant can have one or multiple of these three roles. Every time a producer is storing energy in the batteries of a storage provider, the storage provider is getting a percentage of the gains. The gains are in the form of digital energy tokens, called ENT tokens,

described in more detail in the following chapter.

4.2.3 Network Tokens

PowerChain defines two different type of tokens in the network:

- **Energy Token**: The energy token is a token backed by the Network's stored energy. A single energy token corresponds to specific kilowatt-hours (kWh) of energy. It digitizes the energy of the network and thus making it exchangeable and transactable. It is the heart of the whole network as it represents the actual energy that is produced, consumed and transacted. To link the energy token value with the actual energy of the network, we deploy a token minting and burning mechanism, described in more details in the following section.
- **Currency Token**: The currency token is a taken backed by a real work currency which could be for example Euro, Dollar, Pound or any other. One currency token corresponds to exactly one unit of the backed currency. The existence of a currency token, is important for the network in order to enable token exchanges. Energy producers that want to sell their energy, they need to sell their energy tokens while energy consumers need to buy energy tokens to consume the corresponding energy. The currency token is what brings real world currencies into the network to make this exchange possible. In the following chapter we describe how to bridge a real world currency into the network.

For the purpose of this project, we will use Euro as the backed currency of currency token. For simplicity reasons we will call the Euro currency token as eEuro and the energy token as ENT. In the following sections, more details are provided for the minting and burning of the mentioned tokens.

4.2.3.1 Minting/Burning of ENT tokens

ENT tokens are minted every time energy is stored in the storage layer. For every kWh of energy stored in the storage layer, an amount of ENT tokens are minted into the wallets of the producer and the storage provider. For every kWh consumed from the storage layer, an amount of ENT is burned. The amount of ENT tokens minted or burned is determined by the PowerChain smart contract and is a function of the total ENT tokens in circulation and the

total kWhs of energy stored in the storage layer. We define the minting and burning functions as follows:

$$\text{ENT}_{mint} = \begin{cases} 1 - R & \text{, When } R < M \\ M & \text{, Otherwise} \end{cases} \tag{4.1}$$

$$\text{ENT}_{burn} = \begin{cases} 1 + R & \text{, When } R < B \\ B & \text{, Otherwise} \end{cases} \tag{4.2}$$

$$R = |\text{KWH} - \text{ENT}| \times C \tag{4.3}$$

Where,

ENT_{mint} : ENT tokens to be minted per kWh produced kWh.

ENT_{burn} : ENT tokens to be burned per kWh consumed kWh.

R : The amount of ENT to be paid as cost for minting or burning ENT.

KWH : Total kWhs stored in storage layer.

ENT : Total ENT tokens in circulation.

C : A constant where $0 < c < 1$. Defines the percentage of energy loss to be covered.

M : The maximum ENT/kWh minting cost.

B : The maximum ENT/kWh burning cost.

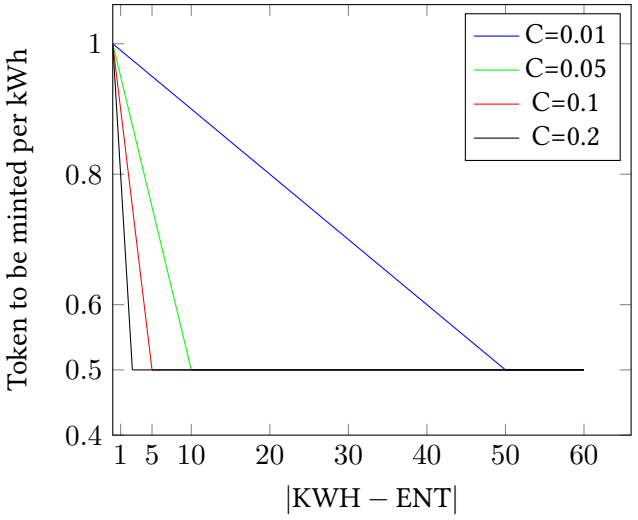

FIGURE 4.3: Minting/burning cost for 1 ENT based on difference between total kWh and total ENT in the network, for different values of C with M = B = 0.5 ENT

The reason we set a cost for minting and burning ENT tokens, is to try to keep the ratio between the total kWh and the total ENT in the network, to 1. Let's consider that for every produced kWh we mint 1 ENT token, in this situation for every X kWhs stored we will have X amount of ENT tokens. The problem arises when we lose energy due to a potential battery disfunction. In such situation, we will have $Y < X$ kWhs available while there are still X amount of ENT tokens in circulation. By applying the minting and burning functions mentioned above, more tokens will be burned per kWh spent and less tokens will be minted per kWh generated until an equilibrium is reached. This is achieved through the cost function R which adds a cost to every

minting and burning of ENT tokens in order to keep the total kWh of the storage layer equal to the total ENT in circulation. The constant C, defines the percentage of the missing energy to be covered during minting or a burning of tokens. The higher the value of the constant, the faster equilibrium will be reached but the higher the ENT cost for the individuals, meaning that in practice less network participants are going to cover the loss.
The issue with the above mentioned equations, is that we approach equilibrium asymptotically because on each step we cover a fraction of the difference and as we progress that fraction tends to zero. To reach equilibrium in a finite amount of steps and keep the energy loss cost distribution fair, the cost R is calculated only when the difference between total kWh and total ENT tokens increase. This means that for every decrease of the difference between these two values, the R value will remain the same until equilibrium is reached or the difference increases again. Having this in mind, we calculate the number of kWh n_t, that need to be produced or consumed, at a particular time t, until we reach equilibrium, if the difference between the total ENT in circulation and the total available energy in the system, at the time t, is D_t and the initial difference that set the cost R is D_0. For every kWh produced or consumed, the difference between the total kWh and the total ENT tokens in circulation, is reduced by the cost R. This means that after n kWh consumed or produced, the difference will be zero.

$$D - n_t R = 0 \Rightarrow n_t = \frac{D_t}{R} \Rightarrow$$

$$\boxed{n_t = \frac{D_t}{D_0 * C}} \tag{4.4}$$

4.2.3.2 Minting/Burning of eEuro tokens

For the minting of eEuro tokens, a trusted authority is needed. The role of this trusted authority would be to mint new eEuro tokens into the blockchain network as well as burn them. This task can only be performed by this trusted authority and it is done in a transparent way so everyone in the network can review the validity of the relevant transactions. For every newly minted eEuro token in the blockchain network, one real world Euro is locked and moved out of circulation. It only exists as its digital version in the blockchain network after the mint and can be transferred only within the network. When the trusted authority burns the eEuro, it can no longer be used in the blockchain network and real world Euro is unlocked and moved back into circulation. The network peers can give real world Euro to the trusted authority to mint them the equivalent amount in eEuro and can also burn an amount of eEuro to get real wold Euro back from the trusted authority. Such a trusted authority could be a certified bank or even a treasury account run by the community.

4.2.4 Blockchain Layer

The blockchain layer is where the PowerChain protocol is implemented and it's also where the actual energy and monetary transactions are happening. The PowerChain protocol defines the rules based on which the network tokens are minted, burned and transferred. It is implemented using a SmartContract and thus it is necessary to use a SmartContract capable blockchain platform. PowerChain uses a private blockchain due to the nature of local energy trading. The decision to use private blockchain was made because we only want to enable the trade of generated energy only locally. A private blockchain would be more secure in that setup as everyone participating in the network would need to be authorized first by the local community.
The blockchain platform we will use for PowerChain is Ethereum with a PoA consensus mechanism. We decided to implement PowerChain on Ethereum due to its active community and because our smart contracts will be directly compatible with the existing public Ethereum network. In this setup, it could be possible for our local Ethereum network to interact with the public one to enable the trading of energy tokens in a global network. More details on such integration will be discussed in the next chapters. The decision to use PoA consensus mechanism is to achieve greater stability and security in the network. PoA is also a more suitable fit for a local network with known participants. [16]
Despite the fact that we refer to a local community and to local energy trading, PowerChain is not limited to only local energy transactions. It could be possible to initiate transactions between two PowerChain instances of two different energy trading communities. Different techniques could be explored to connect two private blockchain networks and initiate transactions with one another. More details will be presented on PowerChain integration chapter.

4.2.5 Market Layer

The market layer is where the ENT token exchanges are happening. In order to use the energy stored in the storage layer, consumers need to purchase some ENT tokens from a producer or storage provider. On the other hand, producers mint ENT tokens by storing energy in the storage layer and would like to sell them for profit. To match these needs, PowerChain has a market platform where the network participant can access to trade. Consumers submit their offering bit to the market, meaning the price per ENT token they are willing to pay and producers / storage providers submit their asking bid, meaning the price they would like to be paid per ENT token. For the market clearing the Continuous Double Auction (CDA) mechanism is used. Bids from producers are ordered from highest to lowest, while bids from consumers from lowest to highest. Then a "price first and time first" principle is applied and the orders are cleared based on the given bids.

Except from the market platform provided by PowerChain, the trade of ENT tokens can happen also by any other mean. It could be possible for two network peers to meet physically and and exchange ENT tokens for cash. It could also be possible to trade the token in another trading platform, assuming that ENT token could be available at that platform. The market layer of PowerChain describes any possible way ENT token could be traded in the market and not only the PowerChain market platform.

4.2.6 PowerChain Integration

A PowerChain instance has the possibility to integrate with other PowerChain instances and also with the public Ethereum network. The integration of two remote PowerChain instances enable the communication of two remote energy trading communities and the exchange of their energy tokens. In such a setup, energy ownership can be transferred between the storage layers of the two networks and the actual energy transfer can happen through the main power grid by a DSO.
There is also the possibility to integrate with the public ethereum network in order to move assets from/to the local PowerChain network. Such assets could be tokens backed by real world currencies like USDT [26], USDC [11] or EURC [10] which could be used for the trading of ENT tokens by the network peers.
In order to facilitate such integrations we need to create an smart contract which will be responsible to lock and unlock tokens. This smart contract will be deployed on both networks we need to integrate and an integration service will be responsible to monitor both smart contracts in order to trigger the unlock of the tokens when necessary.
The integration smart contract consists of the following methods:

Method	Description	Callable By
LockTokens	Locks received tokens into the contract address. It gets as a parameter the public key of a user from the other network.	By any network user
UnlockTokens	Unlocks or generates tokens into the address of a user in the same network. Get as a parameter the public address of the user.	Only by the integration service
GetLockedTokens	Returns a list of locked tokens along with a public key of a user that belong to the other network and a unique id representing the lock transaction.	By any network user
TokensTransferred	Marks a token lock transaction as complete. Receives as parameter the token lock transaction id.	Only by the integration service

Table 4.1: Methods of integration smart contract

The token transfer between the networks happens in the following stages:

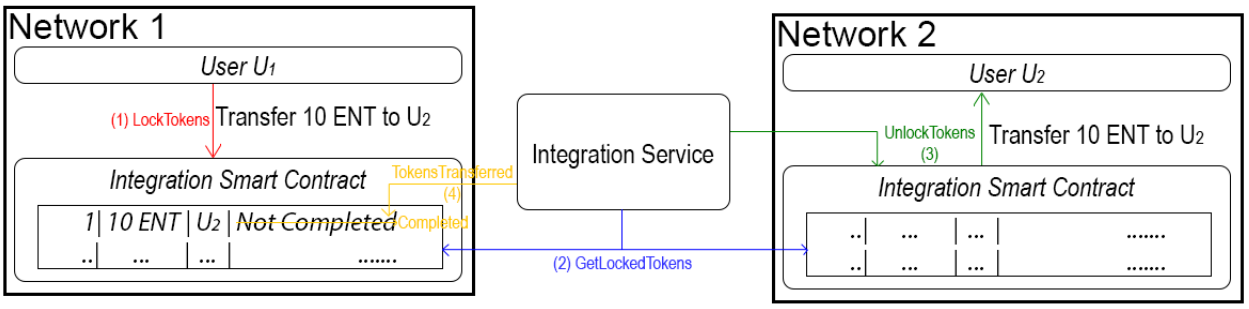

FIGURE 4.4: PowerChain Integration

1. A user U_1 in network A, wants to send some tokens to user U_2 in network B
2. U_1 calls the method **LockTokens** with the public address of U_2 as a parameter and sends X amount of tokens.
3. The integration smart contract of network A locks the tokens in its contract address and stores the public address of U_2 along with the amount X
4. The integration service calls method **GetLockedTokens** of both network A and B every T seconds
5. The integration service receives the information that X tokens was locked on network A for the public address of U_2 and initiates the **UnlockTokens** method of network B
6. Tokens are unlocked or generated by the integration smart contract of network B and transferred to the public address of U_2
7. The integration service calls the method **TokensTransferred** of network A to mark the token transfer as complete.

Chapter 5

Implementation and Core Components of PowerChain

In this chapter we will present the actual PowerChain implementation. We will describe the technical part of the implementations including what technologies we used and how we used them. The actual PowerChain implementation along with all the scripts created, can be found at the following GitHub repository:

`https://github.com/MnAppsNet/PowerChain`.

5.1 Tools Used

For the implementation of the actual PowerChain network, we used the ethereum client Geth, which is build in the programming language Go.[14] We build some Bash scripts that are leveraging geth to build a PoA network and can be used to start a new PowerChain network with any number of validators, depending on the network needs. For security reasons, for the validator nodes we used clef [15] which is a tool for signing transactions and data in a secure local environment. Clef can be configured with different rules to authorize transactions and access through RPC on the validator nodes to enhance security. For the purpose of this project, the rules are set to allow all connections, for testing purposes. To interact with the PowerChain network, we build some Python scripts using the Web3.py library [30], to deploy smart contracts to the network and to call methods of smart contract instances. We also build a PowerChain web application using React.Js [24] and the Web3.JS [29] library to interact with the blockchain network.
The following scripts have been developed to support with the implementation and management of PowerChain network:

Script	Type	Desription
pc_CreateNetwork	Bash	Initiates a guided procedure to setup and initiate a private PoA network with selected number of validators. For each validator a start script is created to start the node.
pc_CreateBootnode	Bash	Initiates a guided procedure to create a bootnode. It creates a start script that can be usd to start the node.
pc_CreateAcount	Bash	Initiates a guided procedure to create a user node. It creates a start script that can start the node.
pc_DeployContract	Python	It gets as parameters the path of a solidity and the RPC of a node in the network, compiles the contract, deploys it to the network and produces a contract json file that contains the contract address and interface.
pc_CallPowerChainMethod	Python	It gets as parameters the RPC of a node in the network, the path to the PowerChain contract json (json returned from pc_DeployContract), a method name from PowerChain, the parameters of the relevant method, calls the PowerChain contract and returns the results. It can also be imported on a python script.

Table 5.1: Scripts to create and manage PowerChain network

To simulate the production and consumption of energy in and out of a storage unit, we created a python script (storage_unit.py). This script can be used to simulate a storage unit address with capabilities to produce and consume energy. In a real life implementation this role will be handled by an IoT device that monitors the input and output of energy to the batteries of the storage unit.

5.2 PowerChain Commands

The PoweChain protocol defines a set of commands which are implemented into the blockchain layer using a Smart Contract. It determines when and how are the network tokens minted, burned and transferred. It also defines the network participant authorizations. In the netwrok, there are different types of roles with each one authorized to execute different type of commands:

- **The normal user**: This is the default role that all network participants. These users can not execute commands that require voting or commands that burn or mint tokens.

- **The voter**: The voters can execute all the commands a normal user is allowed to execute plus the commands that require voting. Such commands could be for example, the promotion of a user to voter or the registration of a new storage unit.
- **The storage unit**: This role is assigned to the smart meters managing the energy of a storage unit and they give the permission to execute commands that lead to the minting or burning of tokens. They can also execute normal user commands.
- **The banker**: This role is assigned to the address of a trusted authority which will be responsible to mint/burn eEuro tokens by moving real world euro in and out of circulation.

Below, we analyze the commands each of the mentioned roles can execute.

Voter Commands:

Register Storage Unit	Start a vote to register a storage unit, providing the unit and the owner addresses
Remove Storage Unit	Start a vote to remove storage unit, providing the unit address
Change Banker	Start a vote to register an address as banker, initially the banker address is the zero address
Set Parameter	Start a vote to set a network parameter, providing the parameter name and value

Storage Unit Commands:

Energy Produced	Reports energy production providing the producer address and the wh produced
Energy Consumed	Reports energy consumption providing the consumer address and the wh consumed
Report Actual Energy	Reports the actual energy that is available in the storage unit

Banker Commands:

Mint eEuro	Mint an amount of eEuro into the provided address.
Burn eEuro	Burn an amount of locked eEuro from the provided address and give the equivalent real world Euro to the user. Only eEuro locked by the user can be burned.
Unlock eEuro	Unlock an amount of locked eEuro, locked by the user in order to withdraw the relevant real world Euro amount.

Normal User Commands:

Get Total Energy	Returns the total energy of the network
Get Storage Units	Returns the active storage units of the network
Get Storage Unit Energy	Returns the total energy of a storage unit
Get Energy Rates	Returns the ENT minting and burning rates of the network
Start Consumption Session	Start a consumption session, providing a storage unit address and the ENT amount to spend
Get Consumption Session	Returns the consumption sessions of the caller
Get Consumption Session Energy	Returns the available energy of a consumption session, providing the address of the session counterpart
Get Storage Unit Info	Returns the state, the owner and the available wh of a storage unit based on its address
Clear Old Sessions	Clear old sessions that have been expired
Get Parameters	Returns the network parameters
Get total ENT	Returns the total amount of ENT in circulation
Get total eEuro	Returns the total amount of eEuro in circulation
Get ENT balance	Returns the amount of locked and unlocked ENT tokens
Get eEuro balance	Returns the amount of locked and unlocked eEuro tokens
Transfer ENT	Transfer an ENT amount to the provided address
Transfer eEuro	Transfer an eEuro amount to the provided address
Lock eEuro	Lock an amount of eEuro, making burnable by the banker. The banker can then burn the tokens and give the equivalent amount of real world Euro to the user.
Add Order	Add an order in the orderbook to buy ENT based on required price per unit
Remove Order	Removes an order from the orderbook
Get Orders	Returns a list of available orders in the network

5.3 Network Parameters

PowerChain smart contract contract, defines some parameters that influence the network behavior. These parameters can be modified by the voters, based on the needs of the network participants. The mentioned network parameters are the following:

- **M** : Maximum minting cost. This is the maximum ENT cost per kWh to be applied when energy is produced.
- **B** : Maximum burning cost. This is the maximum ENT cost per kWh to be applied when energy is consumed.
- **C** : Missing energy recover rate. This parameter defines how fast will the ration between total energy and total ENT in circulation will return to 1 in case of misalignment.
- **H** : Consumption session validity period. This is the amount of hours a consumption session will remain valid. After the validity period the consumption session is closed and any remaining assets are returned to the owners.
- **F** : Storage provider fee. This is the percentage of the total ENT minted by an energy producer, to be rewarded to the storage provider.

These network parameters should be defined carefully in order to incentivize the network peers to participate in the network and provide a fare token distribution. In case of a big misalignment between the total ENT in circulation and the total energy in the network, parameters M and B are used to make sure that producers are receiving a minimum fare amount of ENT and consumers are not spending too much ENT for their energy needs, until the network reaches again equilibrium. Parameter C controls how fast we reach a ratio of 1 between total ENT and total energy, by burning some extra ENT during consumption or minting a fewer ENT during production. A big C value, means that we will reach a balance quicker but the network participants will have to cover a bigger part of the energy loss cost for each kWh consumed or produced. On the other hand, if parameter C is too small, the cost to return network in balance is smaller per kWh produced or consumed but network will reach equilibrium slower. Parameter H was introduced to avoid having consumption session assets locked forever. Consumption session will remain valid within the time frame defined by H. Finally, parameter F controls what percentage of ENT, from the ENT minted by a producer, should a storage provider keep, for providing its batteries. These parameters have some default values and can be then adapted based on the network needs. The default parameter values are the following:

- **M** = 0.5 ENT / kWh
- **B** = 0.5 ENT / kWh
- **C** = 5 % of the loss covered each minting / burning (step), equilibrium is reached in at least 20 steps
- **H** = 2 Hours until consumption session becomes invalid
- **F** = 20 % of minted ENT tokens are kept by storage provider

5.4 PowerChain Voters

The PowerChain contract voters have a crucial role in the maintenance and smooth operation of the energy trading network. They are responsible to audit and agree on the new voters and storage units to be included in the network. They have also the responsibility to monitor the network and propose new network parameter values that could benefit the participant.
As we already mentioned on the previous sections, PowerChain has a set of voter commands. These voter commands are a special kind of commands as they are not directly executable. Every time a voter executes one of these commands, a voting procedure starts with one positive vote from the voter that executed it. The actual action of the command is not executed until the vote is passed. Whenever another voter executes the same command with the same parameters, it agrees with the vote and another positive vote is counted. If the same command with the same parameters is executed again by the same voter, the vote changes to a negative vote. For simplicity reasons and in order to keep the voting process fair, we defined that a vote is passed if it has at least as many positive votes as the number of voters in the network divided by 2.

$$\boxed{\text{Vote Passed when: } V_+/N > 0.5} \tag{5.1}$$

Where,
V_+ : Number of positive votes.
N : Total number of voters
The voter that adds the last positive vote that marks the voting as passed, is also executing the action of the command and the voting process is concluded.

5.5 PowerChain Web Application

The PowerChain web application allows the network users to interact easily with the PowerChain contract, though a user interface. The web application is a control panel that consists of four basic sections and two special ones. To connect the web application with the blockchain network, we make use of MetaMask which works as a blockchain wallet [18]. More details are presented below for each section.

- **Tokens**: This section provides an overview of the user's tokens and it gives also the possibility to transfer the tokens.
- **Energy**: In this section more information about the network's energy are provided. We can see the available storage units and the amount of energy they contain in kWh. There is also the option to start a new consumption session with a storage unit and start consuming energy.

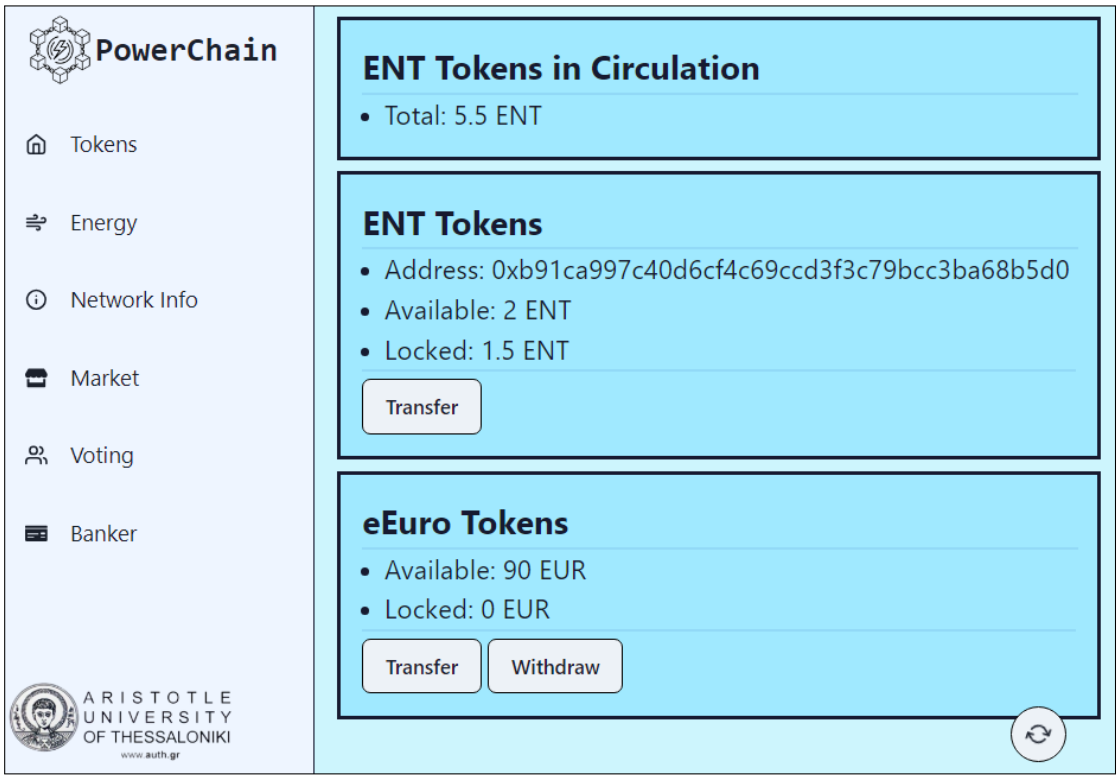

FIGURE 5.1: PowerChain Web Application

- **Network Info**: In this section more network information are provided to the user like the banker address, the total eEuro in circulation and even the values of the network parameters.
- **Voting**: This is a special section that is only available to the users with the voter role. It provides an overview of the user's votes and gives the possibility to start a new vote.
- **Banker**: This is also a special section that is available only to the banker address. It gives the option to mint, burn and unlock eEuro tokens.

Chapter 6

Experimentation & Validation

In this chapter we go through our Network setup and showcase the final implementation. This is a proof of concept implementation with its energy consumption and production functionality simulated. As future work for this project, PowerChain could be tested against real world production and consumption data or even be implemented as a small real life energy trading network.

6.1 Network Setup

For our network setup, we created a virtual linux Ubuntu environment with the help of PROXMOX [23]. We used this virtual environment to run our validator nodes. To setup the PowerChain network, we used the script pc_CreateNetwork which we created to be able to easily setup a bootnode and the required validator nodes. For our testing purposes, we created a network that submits blocks every second, with two validator nodes that run on the same virtual environment. In a real life implementation, the different validators must be controlled by different systems and by different users to improve decentralization. The more validator nodes a network has the better it will be in terms of decentralization. To visualize the block creation and transaction submissions to the network, we also setup the Ethereum Lite Explorer by Alethio.[3]

After the setup of the network, we need to deploy the PowerChain smart contract that contains all the decentralized functionality of our implementation. To deploy the contract, we make use of the pc_DeployContract script we created to deploy contract in the blockchain network. The address that deploys the contract is automatically given the voter roles and is initially the only voter with the power to pass any votes by itself. As a first step to a real world PowerChain implementation, it is important to add more voters in order to split the decision making and

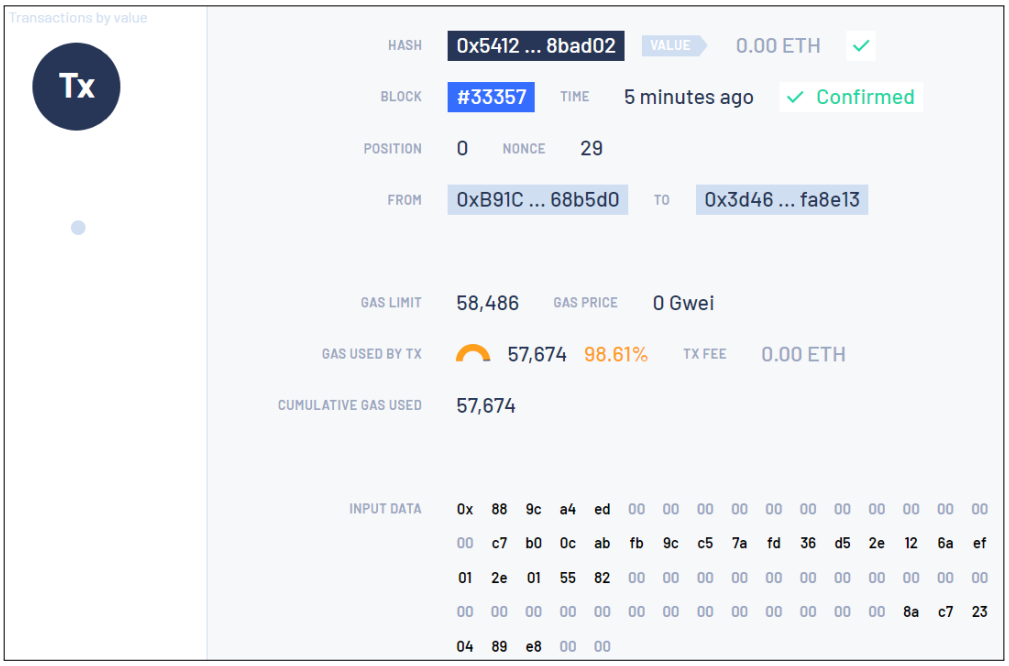

FIGURE 6.1: Example contract method execution transaction on blockchain explorer used with PowerChain

balance the interests of all parties involved.

The next step to finalize the network setup, is to deploy the PowerChain web application that will be used by the network users to interact with the smart contract. For our setup, we run the web application in the local network. We now have a working network with the PowerChain smart contract deployed on it and a web application running on the local network to interact with the contract. The network parameter defined in the deployed PowerChain contract should have their default initial value (figure 6.2).

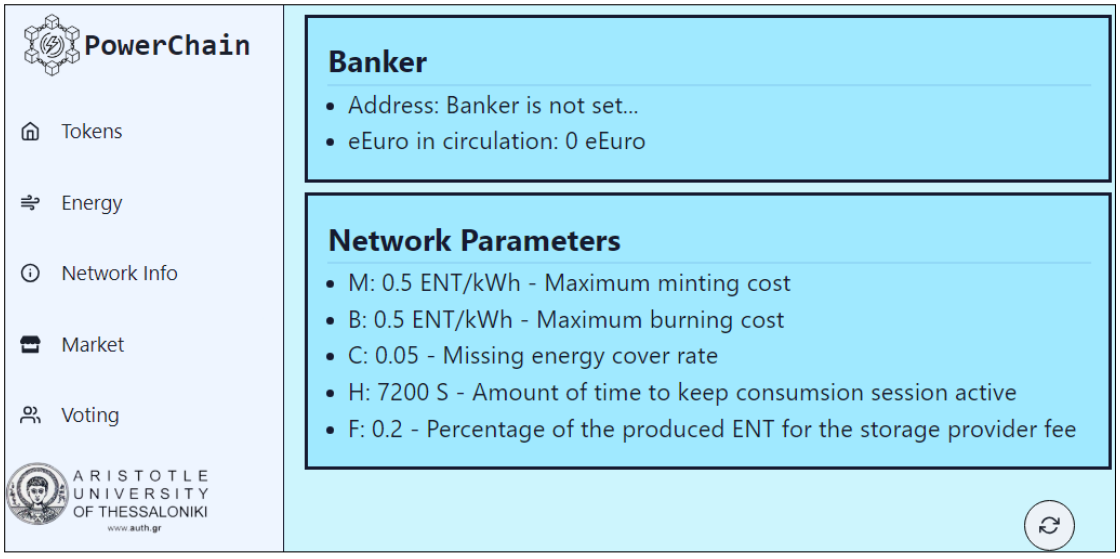

FIGURE 6.2: Initial values for the PowerChain network parameters

6.2 Testing the PowerChain Contract

To test the PowerChain contract, we constructed different energy trading scenarios and performed the relevant action using the PowerChain web application. Some of these test scenarios are presented in the following sections. Apart from the manual tests, we also created some jupiter notebooks that performs some pre-defined tests and checks the results automatically.

6.3 Testing The Voting System

For this test scenario, we go through the voting system of our PowerChain implementation and test its capabilities. When the PowerChain contract is initially deployed the only voter in the network is the address that deployed the contract. At this address had 100% of the voting power and can execute any voter command immediately.
For this test, we have three blockchain network addresses:

- A = ’0xb91ca997c40d6cf4c69ccd3f3c79bcc3ba68b5d0’
- B = ’0xc7b00cabfb9cc57afd36d52e126aef012e015582’
- C = ’0x80fc7a6634ea90774a21f43d51d2a84655ff3958’

The PowerChain contract is deployed by address A and thus A is automatically assigned the voter role. As a first action address A is executing the command ’Add Voter’ to make address B a voter. This starts a voting procedure which is immediately passed and the action to make address B a voter is executed (figure 6.3). There are now two voters in the system, this means

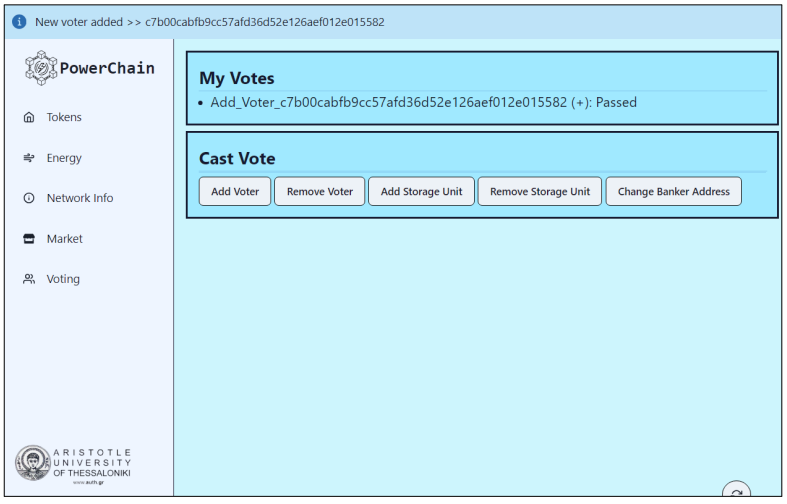

FIGURE 6.3: Address A which has the voter role, assigns successfully the voter role to address B

that in order to execute another voter command, both of them need to agree. This is because the number of voters is now 2 and in order for the equation 5.1 to be valid, we need at least 2 positive votes.

Next step is to give the voter role to address C also. Initially, address B executes the 'Add Voter' command to make address C a voter but the action is not immediately executed, a voting procedure starts with one positive vote from address B (figure 6.4). Now it's time for address A

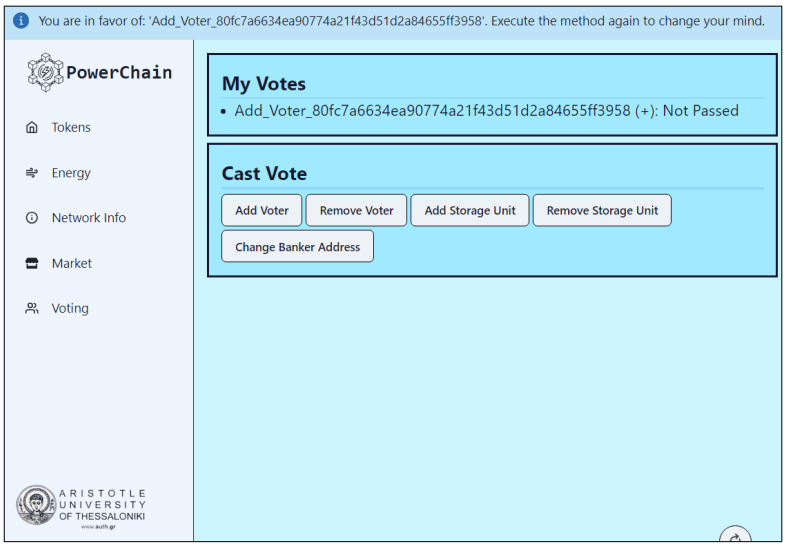

FIGURE 6.4: Address B which is a voter, starts a voting procedure to add the voter role to address C

to cast its positive vote by executing the command 'Add Vote' to give the voter role to address C. This will make the voting to pass and the action to be actually executed which means that address C will now be a voter (figure 6.5).

For the next step, we need to remove the voter role from address C. For this to happen at least

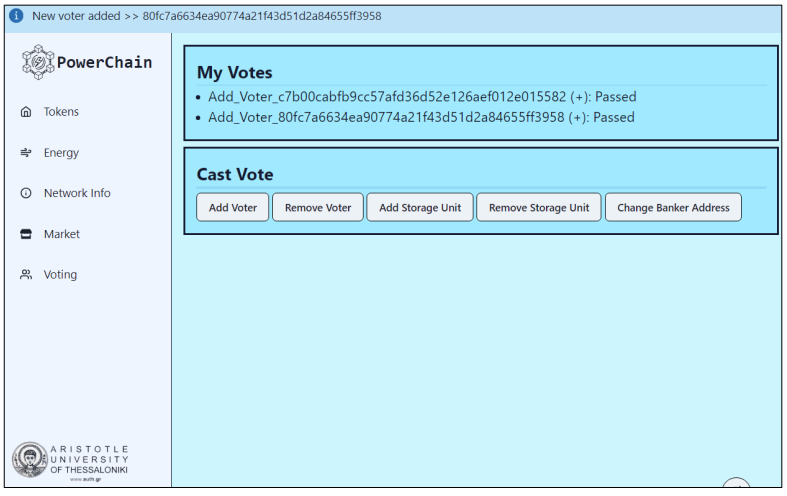

FIGURE 6.5: Address A agrees on the voting to add the voter role to address C and the action is executed

two of the voters has to agree on this action. Address A and B vote for the removal of address

C from voters, the vote is passed and the voter role is revoked from address c (figure 6.6).

Concluding this testing scenario, we see that PowerChain smart contract protects important

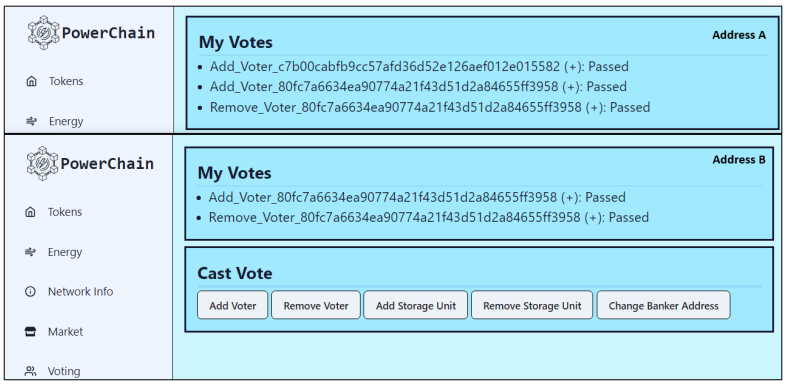

FIGURE 6.6: Address A and B which have the voter role, agrees to remove the voter role from address c and the action is executed

method executions that can impact the network significantly using a voting system. Important network changes should be voted and agreed by most of the voters of the network. The voting system of PowerChain is handled automatically by the smart contracts, giving the possibility to the voters to execute voter commands freely. The commands are only truly performed if they are executed by most of the voters.

6.4 Testing Energy Production And Consumption

For this scenario, we will add two new storage units in the network and start producing and consuming energy. For this purpose, we created the following blockchain addresses:

- Voter Address A = ’0xB91CA997C40D6CF4C69CCD3F3C79BCC3BA68B5D0’
- Voter Address B = ’0xC7B00CABFB9CC57AFD36D52E126AEF012E015582’
- User Address C = ’0x80FC7A6634EA90774A21F43D51D2A84655FF3958’
- Storage Unit Address U1 = ’0x6E383425169CD23c3eb72709AEf8f81E7847310E’
- Storage Unit Address U2 = ’0x16E36D7423AD7181B77818462fE0F5Ec5369aEb7’

As a first step, the addresses U1 and U2 need to be given the storage unit role to have the possibility to mint and burn energy tokens (ENT) in the network based on the energy produced and consumed. To do so, voter A and voter B need to start two voting sessions, one to give the address U1 the storage unit role and another one for U2. We will define as the owner of both storage units, the user address C. As we only have two voter, both have to agree for the votes to pass (figure 6.7).

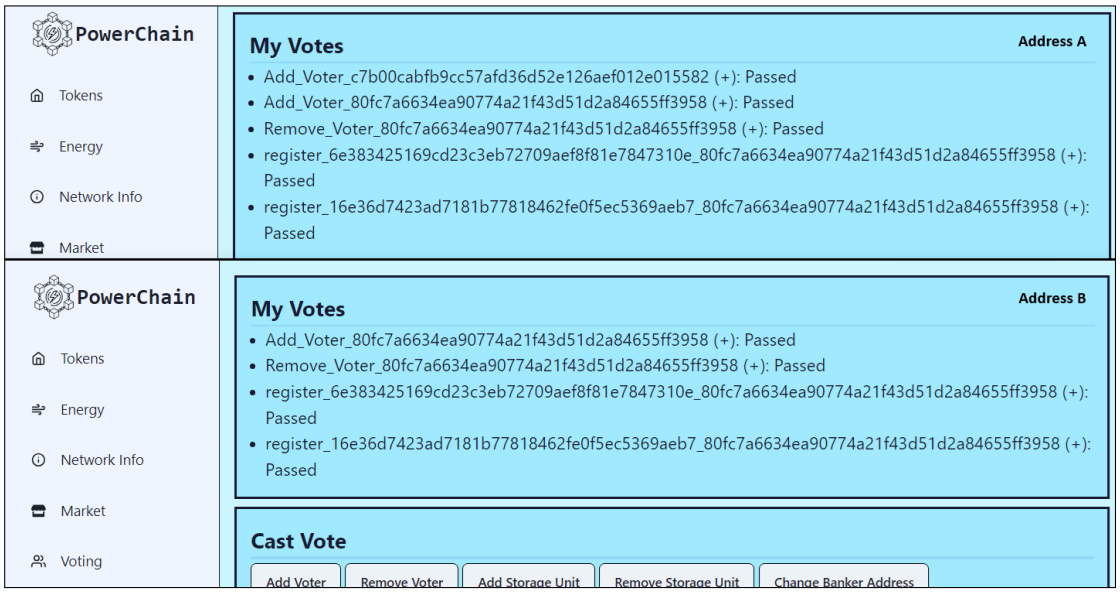

FIGURE 6.7: Address A and B which have the voter role, vote for the addition of two new storage units. The last two votes in the image represent the vote for the addition of the new storage units.

Now that we have two storage units in the network that are owned by user address C, the peer with address A, produced 10 kWh in the storage unit U1. The storage unit U1 reports the production to the blockchain and the relevant ENT tokens are minted. We simulate the energy production reporting using a python script (storage_unit.py). In a real world scenario, there should be a smart meter (and IoT device) that controls the storage unit address. This devise is responsible to monitor the inbound/outbound energy and report relevant production and consumption of energy to the blockchain network. So after the production of 10 kWh from address A in the storage unit U1, address A should be rewarded with 8 ENT and the address C with 2 ENT. That's because the storage unit U1 belongs to user with address C and the storage provider fee parameter is set to 20%. As for the storage unit U1, it should now have 10 kWh available (figure 6.8).

For the next step, user with address C wants to consume some of the energy of storage unit 1. It has 2 ENT available which were received from the energy production of address A as a storage provider fee. It starts a consumption session with unit 1 for 2 ENT which corresponds to an energy consumption of 2 kWh as there is no burn cost currently in the network. With the consumption session of 2 kWh active, user address C consumes 1 kWh and unit U1 reports it. At this stage, user address C should have 1 locked ENT (the other one was consumed) and the storage unit U1 should have 8 kWh available, 1 already consumed and another one locked in the consumption session with user address c. The total ENT in circulation should be reduced to 9 and the same should happen also to the total energy in the network (figure 6.9).

Proceeding further, we will study the case when the user address C consumes more kWh than were locked in the consumption session. This could happen due to an issue with the energy

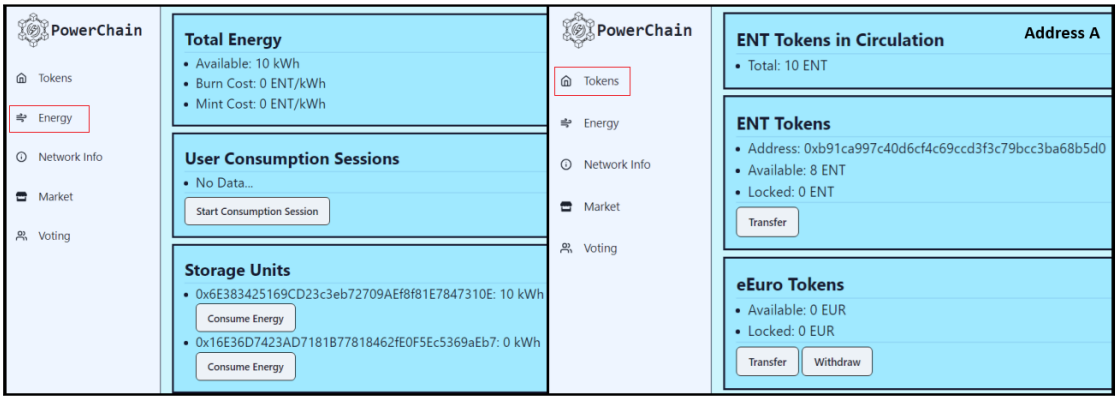

FIGURE 6.8: Overview of the web application information provided to address A after the energy production

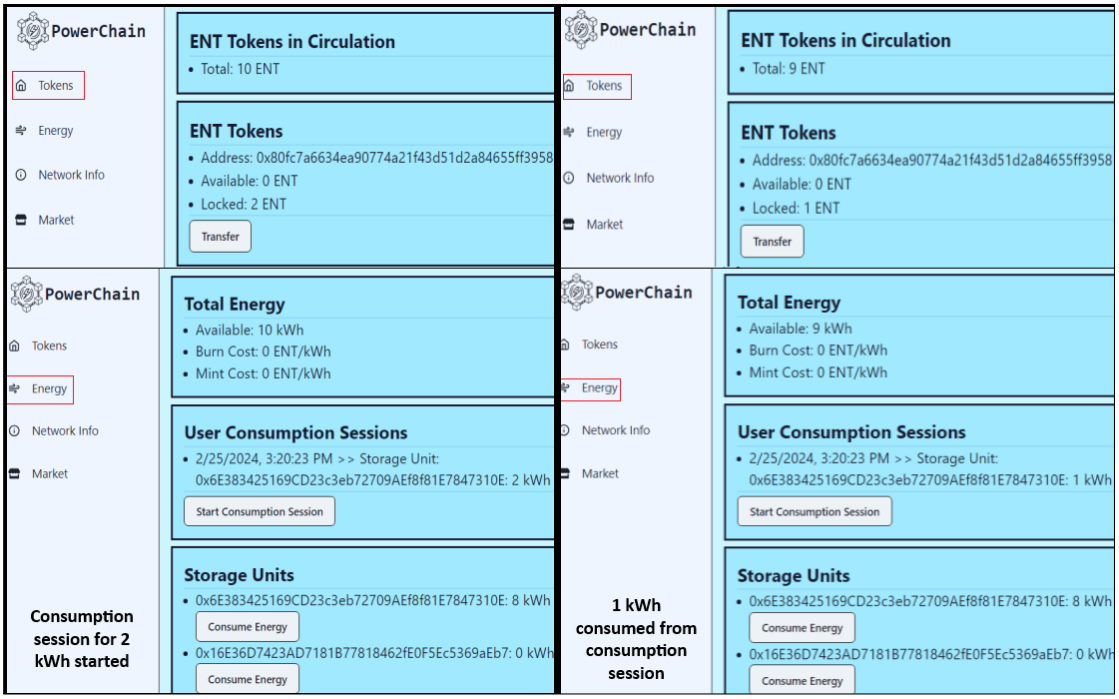

FIGURE 6.9: Consumption session of user address C before (left side) and after (right side) the consumption.

consumption monitoring of the smart meter for example, which might allow the consumer to consume more than what was agreed with in the consumption session. In this case, we will have an imbalance in the system which will lead in a minting and burning cost for the network, in order to counteract it. User address C has now 1 kWh left in the consumption session with unit U1 but due to a malfunction, 2 kWh were consumed. Unit U1 reports this consumption and the network will now have 8 ENT but only 7 kWh in total. The consumption session with user address C is terminated (figure 6.10).

The PowerChain smart contract will try to balance the ration between total ENT tokens and total kWh by applying a const to each mint and burn of ENT tokens. Using the equation 4.4, we can calculate when equilibrium will be reached: $n_0 = \frac{D_0}{D_0 * C} \Rightarrow n_0 = \frac{1}{C} \Rightarrow n_0 = 20kWh$. So

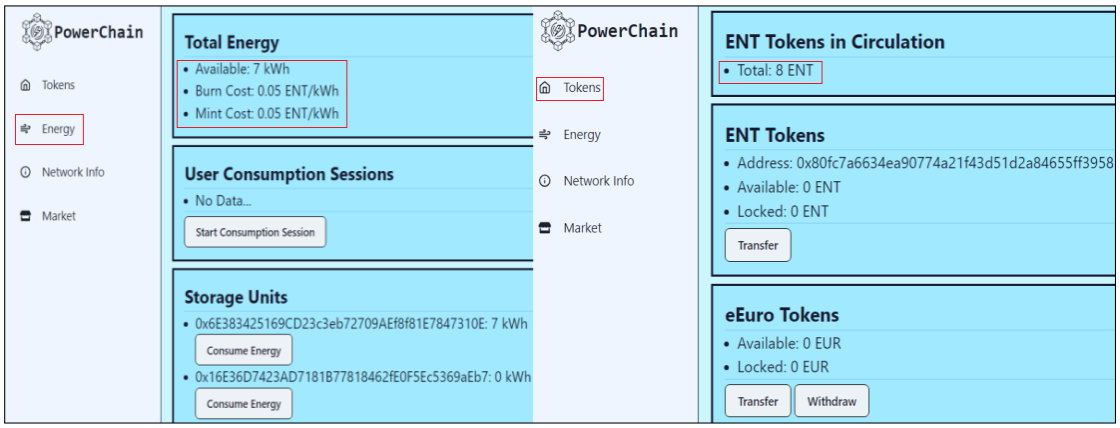

FIGURE 6.10: Due to network energy imbalance, a burning and minting cost is applied. The above image is an overview of the information provided to user address C.

in the next 20 kWh that will be produced or consumed, the network will reach equilibrium. To study this behavior, we first produce 10 kWh from address B into storage unit 2. This will reduce the difference to $D_{t1} = D_0 - 10 * R \Rightarrow D_{t_1} = D_0 - 10 * D_0 * C \Rightarrow D_{t_1} = 0.5$. From this energy production E_p, the ENT tokens minted ENT_{mint}, based on the minting cost $R = D_0 * C = 0.05$, are $\text{ENT}_{mint} = E_p * (1 - R) \Rightarrow \text{ENT}_{mint} = 10 - 10 * 0.05 = 9.5ENT$. Due to the storage provider fee F, the ENT tokens received by address B, are $\text{ENT} = \text{ENT}_{mint} - \text{ENT}_{mint} * F = 7.6ENT$ (figure 6.11).

As a final step for this scenario, we try to resolve the imbalance with an energy consumption

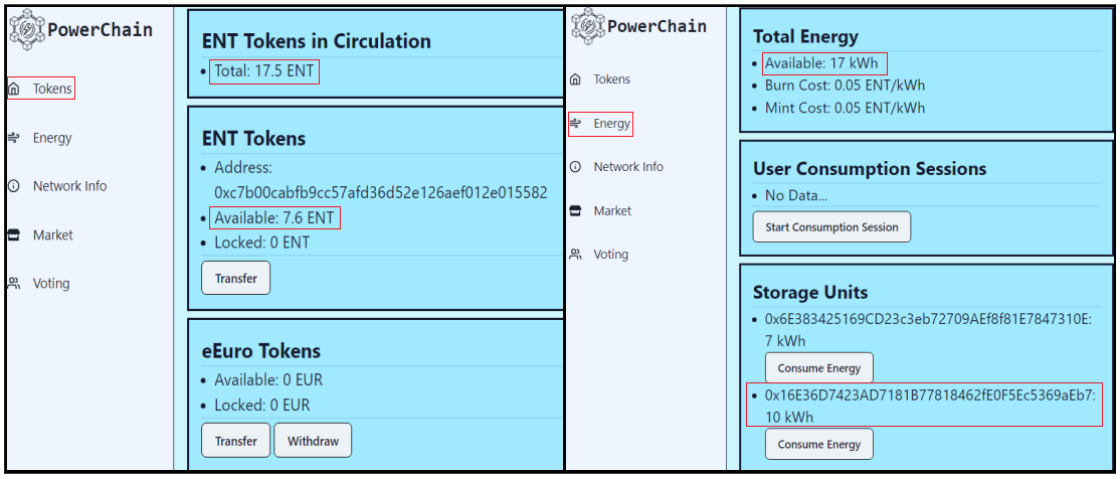

FIGURE 6.11: After the production of 10kWh, the energy imbalance is reduced. In this figure we see the information provided to address B.

worth of 15.6 ENT. To do so, address B transfers its 7.6 ENT tokens to address A in order for address A to have 15.6 ENT available. Address A starts a consumption session for 7.35 ENT with unit U1 and for 8.25 ENT with unit U2. Based on the burn cost, this is equivalent to $7.35/(1 + R) = 7kWh$ for the consumption session with unit U1 and $8.25/(1 + R) = 7.857kWh$ for the consumption session with unit U2. Address A consumes initially the 7 kWh from unit U1 and the storage unit reports the consumption to the blockchain network.

This will result in reduction of the difference between the total ENT and available energy to $D_{t_2} = D_0 - 17 * R \Rightarrow D_{t_2} = D_0 - 17 * D_0 * C \Rightarrow D_{t_2} = 0.15$ (figure 6.12). To

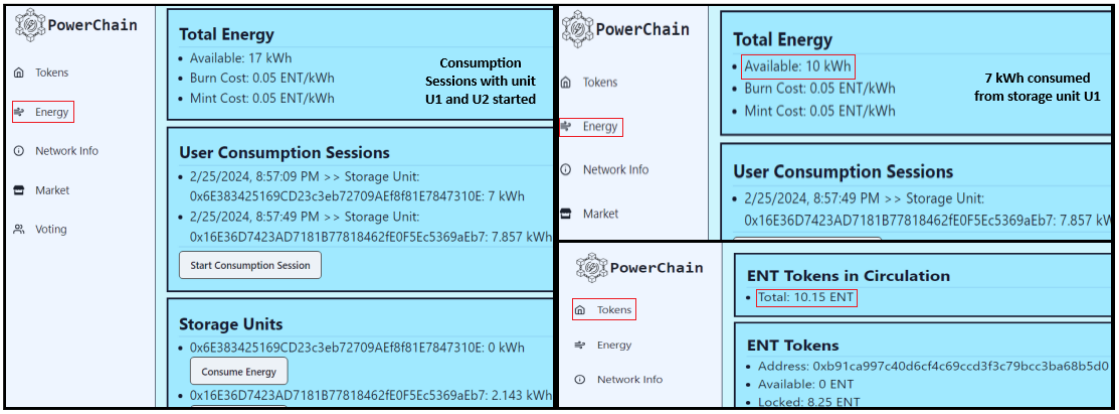

FIGURE 6.12: Address A starts a consumption session with unit U1 and U2 (left side) and then consumes 7 kWh from storage unit U1 (right side).

cover the final difference $D_{t_2} = 0.15$ and reach equilibrium, we need to consume or produce another $n_{t_3} = \frac{D_{t_2}}{D_0 * C} = 3kWh$. To do so, address A consumes the final 7.857 kWh from the consumption session with unit U2. From these 7.857 kWh, $E_e = 3$ will be consumed with ENT burning cost of 0.05 ENT/kWh and the rest with no cost because after the 3 kWh, equilibrium will be reached and the cost R will be set to 0. This means that the ENT to be burned from this energy consumption are $\text{ENT}_{burn} = 3 * (1 + R) + (E_c - 3) = 8.007ENT$. This means that $8.25 - 8.007 = 0.243ENT$ will be returned back to address A (figure 6.13).

Concluding this test scenario, we see that the PowerChain smart contract is able to handle

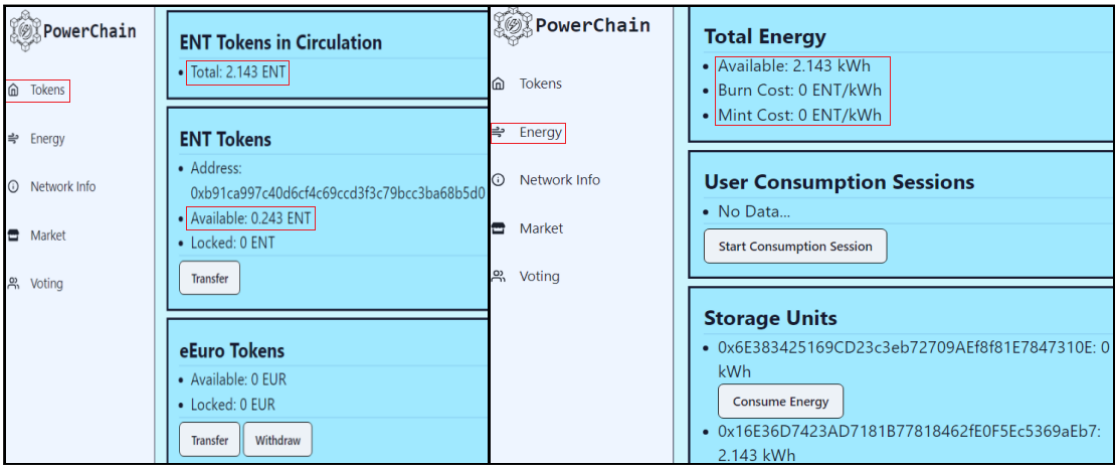

FIGURE 6.13: This this figure we see the information provided to address A after the consumption of the energy from the consumption session with unit U1. The imbalance between the total ENT in circulation and the available kWh in the network is resolved.

energy consumption deals between the network users and the storage units. It is also capable to automatically handle any imbalance between the available energy in the network and the ENT in circulation, by applying a minting and burning cost. The automatic balancing mechanism is

important to keep the ENT token tethered to the value of a particular amount of energy which is 1 kWh is our case.

6.5 Testing the market layer

In this testing scenario, we test the market layer of the PowerChain contract. For this test we use the same addresses with the previous testing scenario. In the current state of the network, address A has 0.243 ENT tokens and address C has 1.9 ENT tokens. In order to use the market layer, we need to introduce the banker role in our network. The banker role will allow us to mint eEuro tokens that can be used to buy and sell ENT tokens. For this purpose, voter address A and voter address B agree on giving the banker role to address C (figure 6.14). Address C can

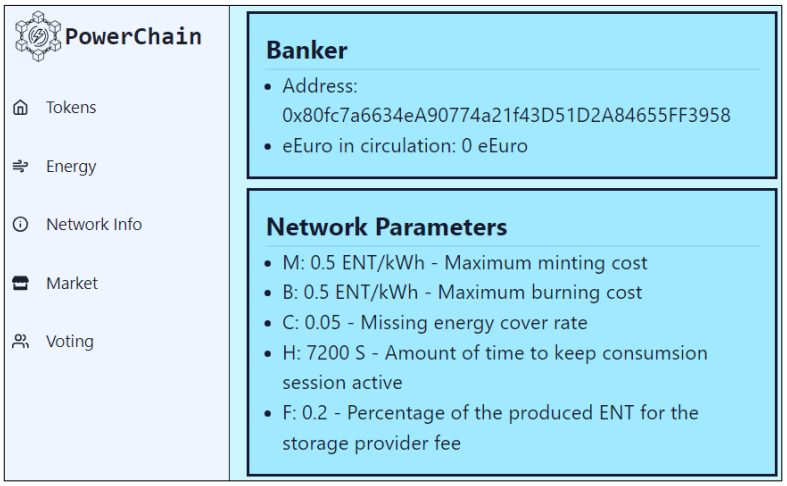

FIGURE 6.14: In this figure we see the information provided to address A. The banker address has been changed to user address C.

now move eEuro tokens in and out of circulation by minting and burning them. In a real world case, this role should be assigned to a trusted third party that will be responsible to real world EUR and mint the relevant amount of eEuro in the PowerChain network. It is also responsible to unlock real world EUR and burn the relevant amount of eEuro, so the network users can withdraw their eEuro into real world EUR. For the purposes of this test scenario, address C mints 10 eEuro into address A (figure 6.15).

Address C has 1.9 ENT available and starts a sell order with cost 0.5 eEuro/ENT. Address A wants to buy 10 ENT with cost 1 eEuro/ENT and creates the relevant order in the market. The system matches the buy order of address A with the sell order of address C and 1.9 ENT tokens are bought from address A for a total of 0.95 eEuro. This will reduce the requested amount of the buy order of address A to 8.1 ENT with requesting price of 1 eEuro/ENT (figure 6.16).

Concluding this test scenario, we see that the PowerChain smart contract can automatically match the buy and sell orders of the users, using an order book style CDA market. This market

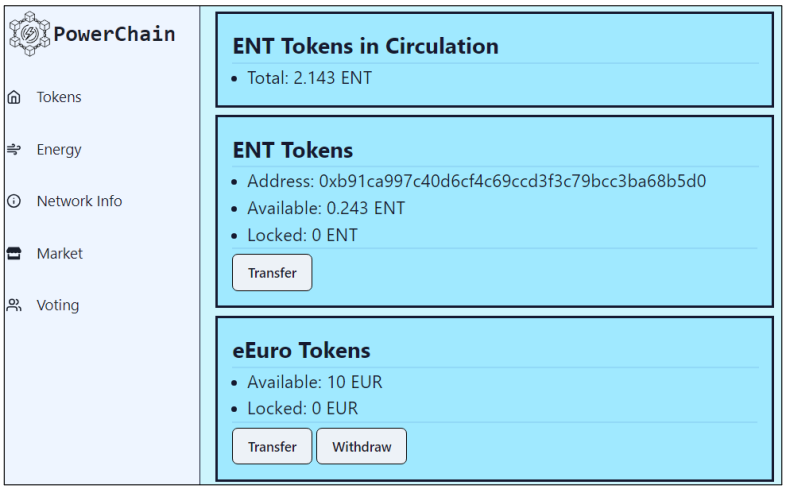

FIGURE 6.15: In this figure we see the information provided to address A. The banker address has been changed to user address C.

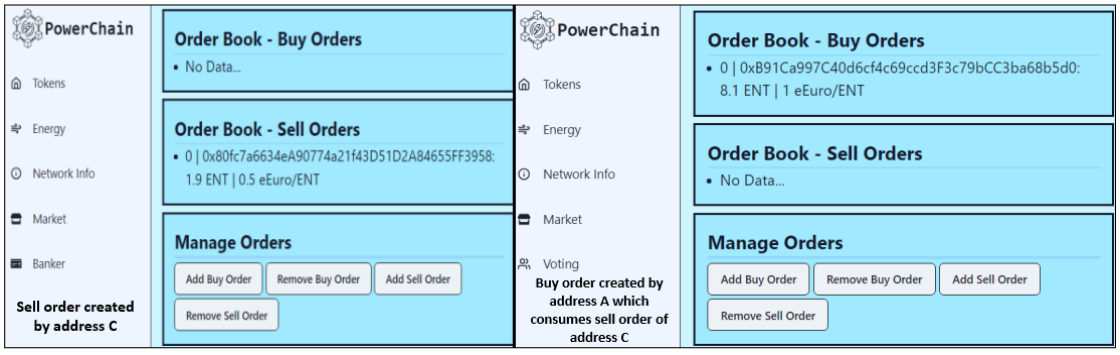

FIGURE 6.16: In this figure we can see the sell order created by address C (on the left side) and the buy order created by address A (on the right side) which automatically consumed the sell order of address C

maker mechanism is very simplistic and could be enhanced to improve the user experience and possibly provide a unified energy price.

Chapter 7

Conclusions & Future Work

This research successfully implemented a decentralized energy trading model called PowerChain, utilizing a private blockchain network. The results demonstrate the feasibility of our model in facilitating secure, transparent, and traceable energy asset exchange within a local network. While PowerChain was proven to be a feasible energy trading model theoretically, more research is needed to practically implement it in a real world scenario.

Our research was mostly focused on the blockchain layer and the handling of network assets but there is also more investigation needed on the physical devices that can implement the model in practice. One such device is the smart meter which is an IoT device, controlling a blockchain address, responsive to monitor energy production/consumption and report them to the blockchain network. It is a very important part of the network because it works as an interface between the digital assets and the actual energy production and consumption. More research is needed on how to setup such devices and how to make sure that the metrics they report can not be tampered with and will always represent the actual metrics with a sufficiently high accuracy.

Another aspect that need to be studied further, is the integration of a PowerChain instance with other remote PowerChain instances or even with other public EVM compatible blockchains. In this study we presented an initial draft of an integration solution in chapter 4.2.6 but more investigation is needed. The integration of a PowerChain instance with other instances will open the doors to energy trade deals between energy trade communities, allowing to export and import energy to each other. The option to integrate with public blockchains will allow the bridging of existing assets from the public blockchain into the local PowerChain network. Such assets could be a stable coin that can be used for the trading of ENT tokens and thus avoid putting trust in a trusted authority to bring EUR assets into the network.

A final suggestion for future work, would be to give the possibility to replace the banker role with a community fund. The banker role in the PowerChain network has a lot of power can could potentially exploit the network. Using the voter roles, it would be possible to implement

a community fund that is controlled by many people in the network instead of a central trusted authority.

Bibliography

[1] Horizen Academy. Accessed on 23/09/2023. The UTXO vs Account Model. https://www.horizen.io/academy/utxo-vs-account-model/

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

[3] Alethio. Accesed on 24/02/2024. Ethereum Lite Explorer. https://github.com/Alethio/ethereum-lite-explorer

[4] Merlinda Andoni, Valentin Robu, David Flynn, Simone Abram, Dale Geach, David Jenkins, Peter McCallum, and Andrew Peacock. 2019. Blockchain technology in the energy sector: A systematic review of challenges and opportunities. *Renewable and sustainable energy reviews* 100 (2019), 143–174.

[5] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. 2018. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the thirteenth EuroSys conference*. 1–15.

[6] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. 2018. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the thirteenth EuroSys conference*. 1–15.

[7] Marianna Belotti, Nikola Božić, Guy Pujolle, and Stefano Secci. 2019. A vademecum on blockchain technologies: When, which, and how. *IEEE Communications Surveys & Tutorials* 21, 4 (2019), 3796–3838.

[8] Umit Cali and Austin Fifield. 2019. Towards the decentralized revolution in energy systems using blockchain technology. *Int. J. Smart Grid Clean Energy* 8, 3 (2019), 245–256.

[9] Miguel Castro, Barbara Liskov, et al. 1999. Practical byzantine fault tolerance. In *OsDI*, Vol. 99. 173–186.

[10] Circle.com. Accesed on 12/11/2023. About EURC. https://www.circle.com/en/eurc

[11] Circle.com. Accesed on 12/11/2023. About USDC. https://www.circle.com/en/usdc

[12] Ayman Esmat, Martijn de Vos, Yashar Ghiassi-Farrokhfal, Peter Palensky, and Dick Epema. 2021. A novel decentralized platform for peer-to-peer energy trading market with blockchain technology. *Applied Energy* 282 (jan 2021), 116123. https://doi.org/10.1016/j.apenergy.2020.116123

[13] Ingo Fiedler and Lennart Ante. 2023. Stablecoins. In *The Emerald Handbook on Cryptoassets: Investment Opportunities and Challenges*. Emerald Publishing Limited, 93–105.

[14] Geth.Ethereum.org. Accesed on 11/02/2024. Geth documentation. https://geth.ethereum.org/docs

[15] Go-Ethereum. Accesed on 11/02/2024. Clef documentation. https://pkg.go.dev/github.com/ethereum/go-ethereum/cmd/clef

[16] Manuel Adelin Manolache, Sergiu Manolache, and Nicolae Tapus. 2022. Decision making using the blockchain proof of authority consensus. *Procedia Computer Science* 199 (2022), 580–588.

[17] Esther Mengelkamp, Johannes Gärttner, Kerstin Rock, Scott Kessler, Lawrence Orsini, and Christof Weinhardt. 2018. Designing microgrid energy markets: A case study: The Brooklyn Microgrid. *Applied energy* 210 (2018), 870–880.

[18] MetaMask. Accesed on 24/02/2024. MetaMask Documentation. https://docs.metamask.io/

[19] Mihail Mihaylov, Sergio Jurado, Narcís Avellana, Kristof Van Moffaert, Ildefons Magrans de Abril, and Ann Nowé. 2014. NRGcoin: Virtual currency for trading of renewable energy in smart grids. In *11th International conference on the European energy market (EEM14)*. IEEE, 1–6.

[20] Mihail Mihaylov, Ivan Razo-Zapata, and Ann Nowe. 2018. NRGcoin—A blockchain-based reward mechanism for both production and consumption of renewable energy. In *Transforming climate finance and green investment with blockchains*. Elsevier, 111–131.

[21] Satoshi Nakamoto and satoshin@gmx.com. [n. d.]. Bitcoin: A Peer-to-Peer Electronic Cash System. https://doi.org/ng

[22] Manisa Pipattanasomporn, Murat Kuzlu, and Saifur Rahman. 2018. A blockchain-based platform for exchange of solar energy: Laboratory-scale implementation. In *2018 international conference and utility exhibition on green energy for sustainable Development (ICUE)*. IEEE, 1–9.

[23] PROXMOX. Accesed on 24/02/2024. PROXMOX Documentation. `https://pve.proxmox.com/pve-docs/`

[24] React. Accesed on 11/02/2024. React.JS documentation. `https://react.dev/learn`

[25] Esteban A. Soto, Lisa B. Bosman, Ebisa Wollega, and Walter D. Leon-Salas. 2021. Peer-to-peer energy trading: A review of the literature. *Applied Energy* 283 (6 feb 2021), 116268. `https://doi.org/10.1016/j.apenergy.2020.116268`

[26] Tether.to. Accesed on 12/11/2023. How USDC works. `https://tether.to/en/how-it-works/`

[27] David Vangulick, Bertrand Cornélusse, and Damien Ernst. 2023. Blockchain for peer-to-peer energy exchanges: design and recommendations. IEEE.

[28] Jian Wang, Qianggang Wang, Niancheng Zhou, and Yuan Chi. 2017. A novel electricity transaction mode of microgrids based on blockchain and continuous double auction. *Energies* 10, 12 (2017), 1971.

[29] Web3.js. Accesed on 11/02/2024. Web3.js documentation. `https://web3js.readthedocs.io/en/v1.10.0/`

[30] Web3.py. Accesed on 11/02/2024. Web3.py documentation. `https://web3py.readthedocs.io/en/stable`

[31] Bitcoin Wiki. 2020. Colored Coins. `https://en.bitcoin.it/wiki/Colored Coins`

[32] Gavin Wood. 2014. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper* 151 (2014), 1–32.

[33] Gavin Wood et al. 2014. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper* 151, 2014 (2014), 1–32.

[34] Zibin Zheng, Shaoan Xie, Hong-Ning Dai, Weili Chen, Xiangping Chen, Jian Weng, and Muhammad Imran. 2020. An overview on smart contracts: Challenges, advances and platforms. *Future Generation Computer Systems* 105 (2020), 475–491. `https://doi.org/10.1016/j.future.2019.12.019`