NOVA IMS Information Management School

MEGI

Master Degree Program in

Statistics and Information Management

Peer-to-Peer Microgrid Electricity Market and Implementation on Blockchain

Chung-Ting Huang

Dissertation

presented as partial requirement for obtaining the Master Degree Program in Statistics and Information Management

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

PEER-TO-PEER MICROGRID ELECTRICITY MARKET AND IMPLEMENTATION ON BLOCKCHAIN

By

Chung-Ting Huang

Master Thesis presented as partial requirement for obtaining the Master's degree in Statistics and Information Management, with a specialization in risk analysis and management

Supervisor/: Ian James Scott

June 2023

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration. I further declare that I have fully acknowledge the Rules of Conduct and Code of Honor from the NOVA Information Management School.

Chung-Ting Huang

Lisbon, July 9, 2023

ACKNOWLEDGEMENTS

I would like to thank my thesis advisor, Professor Ian Scott, for providing this research opportunity. Energy market and optimization methods are areas I had little experience with, but he trusted that I would be able to conduct the research independently. Throughout the research, he was always there to provide guidance and feedback when needed.

ABSTRACT

Peer-to-peer (P2P) electricity trading will play an important role in the energy market as it can incentivize more prosumers to invest in distributed energy resources (DERs), lower costs for consumers, and reduce reliance on the traditional grid. Despite traction from the research community, there are challenges for communities to adopt and form their microgrid. First is the lack of realistic market design that considers both the inter-temporal constraint and flexibility in the scheduling of household appliances, generators, and storage devices. The second challenge is building a trading platform that is trusted, performant, and cost-effective. In this research, we proposed a mixed integer linear programming (MILP) model to optimize for overall social welfare that will address the challenge in market design. Then we developed a trading platform as a side chain of Polkadot, a blockchain of blockchains. Each side chain handles its transactions to achieve high throughput and low costs, while the aggregated state transactions are validated by the relay chain to guarantee security. Since MILP is a complicated computation not suitable to run on a blockchain, another software is developed to bridge the off-chain optimization with on-chain validation. Next, we ran a simulation using data from a published paper to show the advantages of our market design. Finally, the platform is benchmarked to demonstrate its performance.

KEYWORDS

Peer-to-peer energy market; Continuous and flexible market product; Blockchain development; Polkadot blockchain;

INDEX

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

API	Application program interface
DER	Distributed energy resource
DoS	Denial-of-service
EVM	Ethereum Virtual Machine
FiT	Feed-in-tariff
L2	Layer 2
LP	linear programming
MILP	Mixed integer linear programming
P2P	Peer-to-peer
PBFT	Practical byzantine fault tolerance
PoS	Proof of stake
PoW	Proof of work
PV	Photovoltaic
RPC	Remote procedure call
TPS	Transaction per second
Trie	Base-16 Modified Patricia Merkle tree
WASM	WebAssembly
WS	WebSocket

1. INTRODUCTION

According to the International Renewable Energy Agency (IRENA), electricity will become the main source of energy in 2050 (IRENA, 2019). Peer-to-peer (P2P) microgrid electricity trading will play an important role in this transition. It has the potential to incentivize investment into distributed energy resources (DERs), reduce costs, turn consumers into prosumers, and improve grid reliability. In a study from the first real-world P2P market in Switzerland, Ableitner et al. (2020) found participants became more incentivized to invest in DERs such as photovoltaic (PV) systems. One reason is P2P market provides more economic incentives compared to the feed-in-tariff (FiT) from selling back to the grid. Therefore, more consumers are willing to become prosumers who can both buy and sell energy. In a community in London, United Kingdom Lüth et al. (2018) showed that a P2P market can lower the overall cost for the community and recover the investment cost of storage systems. Furthermore, Espe et al. (2018) found prosumers play an important role in balancing the microgrid by generating electricity locally, reducing demand from the main grid, and storing excess energy. Besides the economic benefits, P2P markets also increase energy security. Mengelkamp et al. (2018) did a study on the Brooklyn Microgrid experiment and found that using locally produced energy reduces reliance on the main grid and improves the stability of energy supplies.

Despite recent progress, there are still many challenges in P2P markets to be addressed (Tushar et al., 2021). This research will focus on market design first. Most literatures have simplified models that do not consider the inter-temporal constraints of appliances (Esmat et al., 2021). For example, when consumers turn on their dishwashers, they usually want it to run until completion which can take multiple hours. Thus, market designs should allow consumers to specify this requirement of purchasing electricity for multiple consecutive periods. Another gap in current research is considering flexible schedules. Some consumers want to buy electricity during off-peak hours for a cheaper price. For instance, electric vehicles can be charged anytime at night as long as it finishes before the next morning. Hence, market designs should also allow consumers to express such constraints and determine the optimal schedule automatically. A review of current market designs will be provided in section 2.1. Our proposed design will be presented in section 3.2.

The second challenge is in the implementation. In a P2P market, there is no centralized authority such as a utility company to match supply with demand, determine price, or enforce delivery of energy. To incentivize participation in a P2P market, it should be trustworthy, reliable, and low-cost. Blockchain is often used to provide trust in a decentralized system and many researchers and companies have proposed using it in the energy market (Wu & Tran, 2018). Trust is provided by the consensus mechanism. No single party has the power to add data on the blockchain without agreement from the majority of the participants. Once committed, data are secured by cryptographic hashes and stored in many decentralized nodes, making them almost tamper-proof (Siano et al., 2019). However, scalability and high transaction costs have been a concern (Soto et al., 2021). Ethereum is a popular choice of blockchain that has these problems. This research adopts a different blockchain, Polkadot, a blockchain of blockchains that supports many specialized applications running in parallel. It is designed to be scalable and economically feasible for more projects. A review of blockchain technology will be provided in sections 2.2 and 2.3. Our proposed implementation will be presented in section 3.3.

The goal of this project is to design and implement a P2P market that supports complex products on a blockchain to provide trust and low operating costs. First, we model the market problem as a mixed integer linear programming (MILP) problem to optimize overall social welfare (Khorasany et al., 2018). Next, we build a specialized blockchain on Polkadot for prosumers and consumers to propose their supply and demand. We call it **Po**lkadot **E**nergy **T**rading **S**idechain (POETS). Finally, because solving MILP problems is computationally expensive, we borrowed the idea from Manupati et al. (2020) to solve the market problem off-chain but submits the result back to the blockchain for validation. A program will be developed to do that. The interaction between users, the trading platform, and the off-chain optimizer is presented in figure 1. In summary, the contributions to current research are:

- A market design that considers the inter-temporal constraints and flexibility in the scheduling of DERs.
- The first blockchain that specializes in the energy market that is not a smart contract.
- First energy market blockchain with public consensus but still has control over transaction costs and high transaction throughput.
- Demonstration of how off-chain computation can be validated on blockchain.

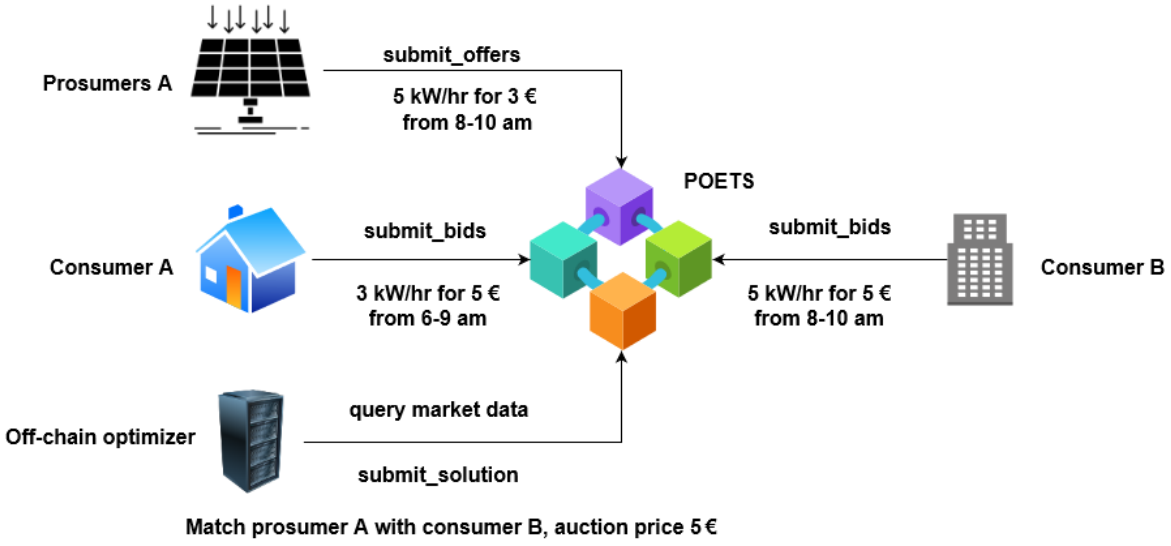

Figure 1 - System interaction

2. LITERATURE REVIEW

This research combines two important problems: a theoretical component of the market design and a practical component of the blockchain implementation of a complex market design. To justify the relevance of the research, we first provide an overview of P2P markets followed by a literature review of related market designs. Next, we examine the technical aspects of blockchain technology, followed by a review of research that applied blockchain in P2P markets. Then, we review projects that have been tested with real users. Finally, a summary of the research gap concludes this chapter.

2.1. PEER-TO-PEER MARKET DESIGN

Blockchain P2P energy markets have two dimensions. At the virtual layer, it is the trading between consumers and prosumers without an intermediary, typically formed in a local microgrid (Soto et al., 2021). Consumers submit bids on the quantity and price they want to buy energy while prosumers propose their offers. The price is above the FiT to encourage prosumers to sell to the local microgrid, and below the retail price to encourage consumers to buy locally. At the physical layer, energy delivery can be over the traditional grid or a new infrastructure (Tushar et al., 2020). In either case, the microgrid is often still connected to the main grid to sell surplus and supply shortage.

According to Khorasany et al. (2018), a P2P energy market framework should consider the market participants, market objective, and clearing method. In this section, we examine existing literature on each dimension.

2.1.1. Market Participants and Products

The market players include consumers, prosumers, and system operators. Consumers are usually households buying energy for their appliances. To design a market, we need to first understand their operating periods. Some appliances need to run for multiple consecutive hours, such as ovens. Other appliances such as water heaters only need to run for a few minutes. Esmat et al. (2021) designed a market to distinguish them by single or continuous product. The market clears multiple delivery periods at the same time using a distributed ant colony optimization to solve the inter-temporal constraints. Zhao et al. (2020) proposed a multi-settlement system combining a forward market with a real-time market. First, the forward market predicts the pricing and scheduling for the full market clearing horizon accounting for the inter-temporal constraint. Next, the real-time market uses predictions of the forward market as a guide while adjusting for the deviation from previous periods.

Another consideration of modelling appliances is their flexibility in scheduling. For instance, some consumers can be flexible with when their washer is turned on. The market should help them decide when is cheaper or when the microgrid has the lowest demand. In the model of Noor et al. (2018), each appliance has a range of operating hours and total daily energy consumption level, which is the summation of the hourly energy consumption. The hourly consumption level is between the standby power and power limit. Using a range for operating hours allows consumers to find the optimal schedule that minimizes cost. Another approach from Antunes et al. (2022) proposed to model appliances as having multiple cycles, and each cycle has its power requirement. Then they transform the constraints to operate for the whole duration of the operating cycle into the summation range for power requested and optimize by finding the best starting time.

Prosumers own appliances similar to consumers. In addition, they also own devices that can generate energy from renewable sources such as PV panels, wind turbines and hydrogen fuel cells (Tushar et al., 2021). One approach is to model them as continuous products because once turned on, they will produce for multiple periods. Another approach considers aggregating prosumers into virtual power plants (VPP) (Morstyn et al., 2018). Ren et al. (2022) further extend the concept with a dynamic tariff model to improve the financial incentive for prosumers joining VPP. Besides household appliances and generators, prosumers can have storage devices. Storage devices do not require a constant energy supply or demand. Instead, we need to consider the state of charge (SOC), initial state and the charging/discharging rate (Noor et al., 2018). Antunes et al. (2022) proposed a more advanced model that takes into account the charging/ discharging efficiency and maximum charging/discharging rate. These parameters are different for every battery. Lüth et al. (2018) conducted a study to compare the cost savings of households owning private batteries vs. a shared community battery. The result showed that the former performed better. The reason is attributed to user behaviors due to market design, not because of battery characteristics. Storage devices have been gaining attention recently because they play a vital role in balancing the stability of the microgrid by consuming surplus and releasing during a shortage. An analysis done by Li et al. (2016) concluded that integrating batteries with renewable energy sources can improve the stability of supply and reduce overall costs.

A pure P2P market will only have consumers and prosumers. However, it is unlikely to be successful due to the lack of trading infrastructure, making it hard to discover peers and define market rules. A system operator can be the mediator between market players, responsible for matching supply with demand. Khorasany et al. (2018) found that a distributor or auctioneer often assumes this role. They can be a community manager, non-profit organizations, or automated rules.

To summarize, a market design should have an impartial system operator with clearly defined rules. Then it should consider the type of products to support and define corresponding models. In this research, we propose using blockchain to run the market. Appliances are continuous products with flexible schedules. Batteries are similar to appliances, but in addition, their supply/demand quantity are also flexible to model the discharging/charging rate.

2.1.2. Market Objective

The market objective influences how it matches supply with demand and the type of market players to attract. Khorasany et al. (2018) found most researchers focus on maximizing social welfare, which is the difference between the overall utility of consumers and the overall cost of suppliers. Economics theory suggests it also maximizes the profit of each market player. Another group of researchers focuses on minimizing the cost. For example, Noor et al. (2018) proposed a model to reduce peak demand, which reduces infrastructure and therefore consumer costs. Some others want to incentivize prosumers. The work by Ren et al. (2022) would improve the earnings of prosumers joining VPPs. Note that the objective can be more than financial. The demand-side management by Noor et al. (2018) has another benefit of improving energy security.

2.1.3. Market Clearing Mechanism

The clearing mechanism is the set of rules to match supply with demand and decide on the trading price. It also determines the type of products that can be traded. Double auction, game theory and objective optimization are popular methods.

In an overview of P2P trading done by Tushar et al. (2020), double auction is found to be a common mechanism. There are two types of double auctions, continuous double auction (CDA) and periodic double auction. In both variants, the supplies are sorted by their offer prices in ascending order, while the demands are sorted by their bid prices in descending order. If there is a tie, the earlier submission has a higher priority. The intersection of the supply and demand curve determines the auction price. Supplies offered below the auction price and demands bidding above the auction price are accepted, but everyone will buy or sell at the auction price. The difference is a CDA tries to match a buyer with a seller upon submission, whereas a periodic double auction matches at the end of the trading period. Real-world projects such as the Brooklyn Microgrid (Mengelkamp et al., 2018) and Quartierstrom (Brenzikofer et al., 2019) use double auctions. Double auction is a simple mechanism to implement, however, it doesn't account for inter-temporal constraints (Esmat et al., 2021).

Another type of approach models the interaction between market participants based on game theory. Tushar et al. (2018) found that game theory can model P2P energy trading for electric vehicles, DER, storage, and services to the grid and achieve better utility or lower cost. If information can be exchanged transparently in the microgrid, and the participants are incentivized to cooperate to improve their utilities, then the market can be modelled as a cooperative game (Khorasany et al., 2018).

On the other hand, if each participant acts independently, then the market can be modelled as a non-cooperative game. Note that in non-cooperative games, the participants do not have to act against each other. Mezquita et al. (2019) designed a multi-agent system where participants in a non-cooperative game achieved lower prices for consumers and higher prices for producers. Noor et al. (2018) designed a game to optimize consumer schedules that reduce peak demand. One common design of non-cooperative games is the Stackelberg game, where there is a leader who makes decisions first, and then the followers base their strategies on the decision of the leader. Doan et al. (2021) proposed combining an auction with a Stackelberg game. In this design, the auctioneer is the leader setting the price and the buyers are the followers who update their demand to optimize average social welfare. The leader then updates the price based on the demands and buyers update based on the new price. This process iterates until neither the leader nor followers can achieve better utility.

The last method of market clearing is by solving an optimization problem. The objective is usually to maximize social welfare, reduce cost or reduce peak demand. Tushar et al. (2020) found linear programming (LP), nonlinear programming (NLP) and alternating direction method of multipliers (ADMM) have been used in the literature. For instance, Sabounchi and Wei (2017) proposed an LP problem that maximizes the payoff and satisfaction of prosumers. MILP is a special case of LP where some variables are constrained to be integers. Antunes et al. (2022) proposed multi-level optimization using modular MILP models that can be combined to achieve multiple objectives such as minimizing cost within some discomfort constraints.

In this research, we will model the market as an optimization problem to maximize social welfare. This clearing mechanism is preferred over double auction and game theory because we can model continuous products with flexible loads as a MILP problem. LP and MILP can be solved using commercial software such as CPLEX and Gurobi, or open-source solutions such as GNU Linear Programming Kit (GLPK) and COIN CBC, all of which are centralized solvers. For decentralized methods, there is ADMM. However, Esmat et al. (2021) found that current research has only used it in simple market design. This suggests optimization problems are not suitable to solve on the blockchain. Chapter 3 will provide more details on how this research addresses the challenge of integrating an off-chain optimizer with an on-chain validator.

To recap, chapter 2.1 provides an overview of the current literature on market participants, objectives and clearing methods to guide our P2P market design. The next section focuses on the implementation platform, blockchain.

2.2. BLOCKCHAIN TECHNOLOGY

2.2.1. Overview

This section covers the fundamentals of blockchain technology. Although it is often associated with cryptocurrencies, blockchain is a more generalized technology that enables cryptocurrencies. In essence, a blockchain is an append-only database maintained by a decentralized network. The key characteristics of blockchain are (Wu & Tran, 2018):

- Decentralization: Data is stored in a P2P network. There is no central authority to decide what can be stored or modified unilaterally.
- Trust: Although there is no central authority, there is trust in this system because any update must be validated and agreed upon by multiple nodes. A malicious node cannot compromise the network.
- Openness: The data is opened for access by anyone in the network. The rules that operate the system are also transparent.
- Traceability: Records are stored permanently, including the author of a transaction. If something is transferred through multiple transactions, the path can be traced back.
- Data security: Data cannot be tampered with unless the majority of the nodes are compromised.
- Anonymity: Parties exchanging data do not need to reveal their identities.

The network is run by distributed nodes. Who is allowed to join depends on the permission model of the network. Nodes in the network can propose and validate blocks. Each block comprises a list of transactions. The consensus protocol determines how and when the blocks are appended to the network. Blockchains usually run independently, but some of them are interoperable, meaning cross-chain communication is possible. Figure 2 shows an example blockchain network. The following subsections provide technical details of blockchain design that result in its key characteristics.

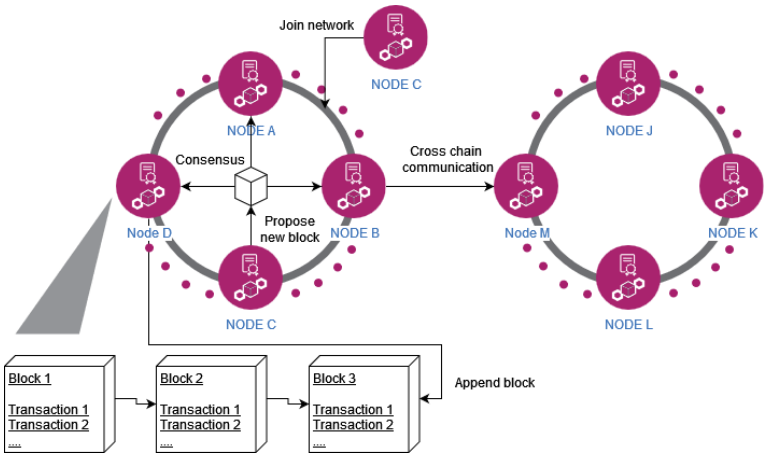

Figure 2 - Blockchain Network

2.2.2. Permission

Some blockchains allow anyone to join the P2P network. Others have restricted access. Most blockchain can be classified as public, private or consortium (Zheng et al., 2017). In a public blockchain, anyone can join the network, participate in the consensus protocol, and read transactions. More nodes in the network make it harder to tamper with data and, therefore more secure. But at the same time transaction latency increases and throughput decreases because bigger networks reach consensus and propagate transactions slower. Bitcoin and Ethereum are examples of public blockchains.

A private blockchain is controlled by a single organization that decides who can join the network. It is no longer decentralized as the organization has control over the consensus. But it can be useful if efficiency and privacy is the main concern. The middle ground is a consortium blockchain formed by multiple organizations, such as a university network. Since the organizations control who can join the network, it is partially centralized. Its security and efficiency are between public and private blockchains. Hyperledger is an example of a consortium blockchain. Another example is Ethereum deployed in a consortium network.

Besides the three common types of permission, Polkadot is a hybrid where the validation is done on the public relay chain, but application-specific side chains can require permission to join. Transaction data in Polkadot is still public because the sidechain transactions are aggregated into blocks on the relay chain.

2.2.3. Consensus Protocol

The consensus protocol determines how to append new blocks. It also influences the security and economic model. Proof of work (PoW), proof of stake (PoS), Practical byzantine fault tolerance (PBFT) and Tendermint are common classes of protocols (Zheng et al., 2017).

In a PoW protocol, the first node that finds a nonce such that the block header hash is smaller than a given value can propose the next block. Hashes are easy to validate but expensive to reverse the original value, usually through brute force. Thus, it is computationally expensive for an attacker to propose blocks consecutively. Two nodes can propose valid solutions around the same time and create temporary forks. However, it is unlikely that both forks will generate the next block

simultaneously. The protocol picks the longest fork to be the winner. The main drawback of PoW is the energy required to calculate the hashes. PoW is used by Bitcoin (Nakamoto, 2008), which is why it is criticized for its energy consumption. The first version of Ethereum (Buterin, 2014) also uses PoW, but it has been transitioning to PoS.

In a PoS protocol, nodes that want to generate new blocks put collateral in the network. The network picks a node randomly to become the next producer. To prevent malicious nodes, the collateral is confiscated if the new block cannot be validated by other nodes. Compared to PoW, blockchains using PoS are more energy efficient (Zia et al., 2020).

In a PBFT protocol, the network selects a primary node to order transactions in each round. Each round has a pre-prepared, prepared, and commit phase. The primary node can enter the next phase only if it receives more than 2/3 of the votes from all nodes. Therefore, PBFT requires each node to be known, but it can tolerate up to 1/3 of malicious nodes. Hyperledger Fabric uses this protocol.

The Tendermint protocol is similar to PBFT, but it requires the nodes to provide stakes to become validators. In each round, a node will propose a new block. It has a pre-vote, pre-commit and commit phase. The node must receive at least 2/3 of the votes from all nodes to proceed to the next phase. This protocol is used by Cosmos and a blockchain with the same name, Tendermint.

2.2.4. Transaction

The basic data structure is a block, which consists of a header and transaction data. Figure 3 provides a visualization. The header is used to store metadata which varies based on the implementation, but in general, it always has the hash of the previous block. The first block is called the genesis block and it is the only one without the hash of a previous block. The second block has the cryptographic hash of the genesis block. Further blocks are linked to their previous block though the cryptographic hash, hence the name blockchain. Blocks cannot be removed, hence the reason blockchain is append-only.

Block data typically contains a list of transactions. But for some implementations, it is possible to have blocks with no transaction. A transaction usually records activity on an asset, such as transferring between parties (Yaga et al., 2019). To transfer an asset, the sender references the event that created it or the transaction that acquired it, so the origin is traceable.

Asymmetric-key cryptography is often used to validate the authenticity of a transaction. It has a pair of public key and private key. The sender signs the transaction by encrypting it with the private key. Then the receiver validates the signature by decrypting the transaction with the sender's public key. The public key can be distributed to anyone, since in asymmetric-key cryptography, the algorithms guarantee that the private key cannot be efficiently derived from the public key. Diffie–Hellman method provides a mechanism to exchange cryptographic keys over public channels (Diffie et al., 1976).

Usually, the block header also has a field for the hash of its block data. This ensures that if data in a block is manipulated, the hash will change. The previous block hash pointer in the next block will no longer match the new hash, thus breaking the chain. To manipulate block data, an attacker will need to convince the majority of the nodes to update all the previous block hash pointers. For a large enough network, this feasibility is low.

The parties involved in a transaction do not need to reveal their real identity. Some blockchains use an address derived from each party's public key as the source and destination of the transaction (Yaga et al., 2019). Therefore, the sender doesn't need to know which person or organization it is sending to. The trust in the process is replaced by cryptography.

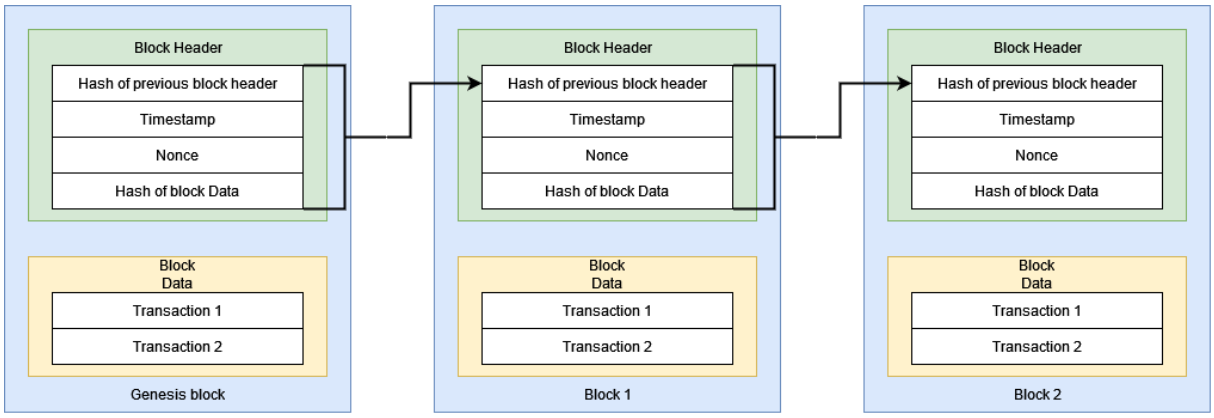

Figure 3 - Chain of Blocks

2.2.5. Interoperability

Belchior et al. (2020) defined blockchain interoperability as the ability of a blockchain to modify the state of another blockchain via transaction over some cross-chain communication protocol. For example, a person might want to exchange Bitcoin for Ethereum directly, without selling Bitcoin for fiat money then use the proceeds to buy Ethereum. Research in interoperability has been gaining traction as the ecosystem is becoming more fragmented with the creation of new blockchains. Possible solutions include sidechains, notary schemes, hashed time-lock contracts (HTLS) and blockchain of blockchains.

In the sidechain and relay solution, the mainchain considers the sidechain as an extension of itself. One way to connect the main chain and sidechain is a two-way peg with simplified payment verification (SPV). When an asset needs to be transferred from the mainchain to the sidechain, the asset is first locked on the mainchain. Then an SPV client on the sidechain verifies there is a transaction on the mainchain that locks the asset.

In a notary scheme, a centralized exchange such as Binance or Coinbase monitors multiple chains. Interledger is a popular notary mechanism (Thomas & Schwartz, 2015). The exchange triggers a transaction on the target chain when a matching event takes place in the source chain. However, this assumes the centralized exchange is a trusted anchor. There are also decentralized exchanges, such as Uniswap by Adams et al. (2021) which are smart contracts that provide a liquidity pool for a pair of assets.

In HTLC, the sender hashes a secret and then deploys a smart contract protected by the secret (*Hash Time Locked Contracts - Bitcoin Wiki*, n.d.). The smart contract is also time-locked, meaning it must be unlocked before a certain time. The receiver deploys another smart contract using the sender's hash as lock and it must be unlocked before the sender's contract expires. The sender unlocks the receiver's contract using the hash of its secret and provides the original secret as input to this contract. Now the receiver has the secret to unlock the sender's contract. Therefore, HTLC allows atomic cross-chain operations. Lightning Network for Bitcoin is an example of using HTLC.

Blockchain of blockchains, or blockchain networks, are frameworks similar to sidechains and a relay. There is typically a mainchain that is called the relay chain connecting the application-specific sidechains. The framework "provides reusable data, network, consensus, incentive, and contract layers for the creation of application-specific blockchains (customized blockchains) that interoperate with each other" (Belchior et al., 2020). Polkadot and Cosmos are examples of using this framework. In the next section, we will provide more details on Polkadot.

2.2.6. Concerns of Blockchain

Even though blockchain technology has gained traction, there are concerns about scalability, privacy (Zheng et al., 2017), malicious users, environmental impact, and governance (Yaga et al., 2019).

Scalability – Bitcoin can handle around 7 transactions per second (TPS) max (Croman et al., 2016) and Ethereum before the upgrade can handle around 15 TPS (*Ethereum Live TPS*, n.d.). For more adoption, blockchains need to be able to process more transactions. Sharding and layer 2 rollup used by Ethereum's upgrade are possible solutions to scalability.

Privacy - although blockchains use public keys to identify users, researchers have demonstrated that transactions can be linked to reveal real identity (Zheng et al., 2017). In addition, transactions on public blockchains can be queried by anyone, so the content and account balance are not private. One possible solution is using zero-knowledge (ZK) proof (Goldwasser et al., 1985). This protocol allows someone to verify a claim, without knowing the information used to generate the claim (*Zero-Knowledge Proofs | Ethereum.Org*, n.d.). It can also be used by layer 2 to generate ZK-rollup, where only the summary of batched transactions needs to be submitted to layer 1. However, generating and verifying claims involves complex computations that are best done on specialized hardware, limiting its practicality financially.

Malicious users - If a coalition of attackers has enough processing power in a PoW blockchain or stake in a PoS blockchain, they can try to create a private branch. Once a private branch is longer than the main branch, the attackers can submit it to the rest of the network. Honest nodes will accept it because of the longest chain rule and let the attackers take over the network.

Environmental impact - Bitcoin consumes more electricity than countries such as Ireland or Denmark because it uses PoW to mine new blocks (Yaga et al., 2019). Besides energy, creating a full node requires significant network bandwidth because it needs to download a large amount of data stored in the blockchain.

Governance - No entity controls the blockchain is a common misconception. Private and consortium blockchains are managed by one or more organizations. Even in public blockchains, developers of the software and publishing nodes have more influence than ordinary users. Although the software is open source, not every user has the technical knowledge to make changes. When changes are made to the software, publishing nodes can decide whether they want to adopt it. If the adoption is not unanimous, a fork might be created.

2.3. DECIDING BLOCKCHAIN FOR A P2P MARKET

The distributed nature of blockchain technology, as well as its ability to provide trust, traceability and data security, make it suitable for P2P markets. There are many blockchain implementations. First,

we provide some background on the popular ones. Then we compare them in 6 dimensions, permission, consensus protocol, application layer, scalability, cost, and currency to decide on which one is suitable for this research.

2.3.1. Popular Blockchains

Although cryptocurrencies are not the only use case of blockchain technology, it is the one that popularizes it. According to coinmarketcap.com[1], a website that tracks the market cap of cryptocurrencies, Bitcoin is the largest, followed by Ethereum.

Bitcoin was the first cryptocurrency proposed by Satoshi Nakamoto during the financial crisis of 2008 (Nakamoto, 2008). It aims to provide an electronic transaction system without relying on a trusted party and prevent double spending through PoW. Many other cryptocurrencies and exchanges were created afterwards.

In 2015 the Ethereum project was launched as "a blockchain with a built-in Turing-complete programming language" (Buterin, 2014). Developers can build decentralized applications as smart contacts executed on the Ethereum Virtual Machine (EVM) (Wood, 2014). Since EVM is Turing-complete, any conceivable computation can be carried out given enough time and memory, including infinite loops. To prevent denial-of-service attacks and encourage efficient code, each smart contract has a limited number of computational steps it can perform. In addition, each computational step consumes gas, which is the unit for fees. However, due to its low transaction throughput, high gas fees have been a concern for developers. Ethereum has been going through a series of upgrades to become more scalable, secure, and sustainable (*Ethereum Roadmap | Ethereum.Org*, n.d.).

To address the scalability concern and economic barrier of Ethereum, one of its co-founders started a new blockchain, Polkadot (Burdges et al., 2020; Wood, 2016). Although Polkadot has its token DOTs, the focus is less on creating a new cryptocurrency and more on developing a platform to foster innovation in blockchain technology. Its architecture consists of a relay chain that connects application-specific sidechains, called parachains. The relay chain is responsible for the security, consensus, and cross-chain interoperability of the network (*Architecture · Polkadot Wiki*, n.d.). The parachains run different applications in parallel to achieve scalability and isolatability. Parachains submit their state transitions to the relay chain for validation. This gives parachains the security guarantee of the relay chain.

The Polkadot network has 3 types of participants, collator, validator, and nominator (Burdges et al., 2020; Wood, 2016). Collators maintain a full node in a particular parachain to collect transactions and produce state transition proof for validators. Validators verify state transition proof from collators and participate in consensus to produce blocks on the relay chain. They stake DOT tokens to ensure they are incentivized to act in good faith (*Architecture · Polkadot Wiki*, n.d.). Someone with DOT tokens but does not want the responsibility of a validator can become a nominator. This role stakes DOT tokens to nominate one or more validators. The original whitepaper also proposed a fisherman role which acts as a bounty hunter to catch misbehaving nodes, but it has been deprecated (*Architecture · Polkadot Wiki*, n.d.).

[1] https://coinmarketcap.com/

One of the novel features of Polkadot is the interoperability between parachains through Cross-Chain Message Passing (XCMP). It guarantees the delivery and ordering of messages. The design routes messages through collator nodes of the sending and receiving parachain (*Architecture · Polkadot Wiki*, n.d.). While XCMP is still under development, Horizontal Relay-routed Message Passing (HRMP) provides a stop-gap solution. It provides the same interface, but all messages are stored in the relay chain, therefore less scalable than XCMP.

In the next few sections, we compare Ethereum with Polkadot in 6 dimensions. Bitcoin is not considered because its focus is on cryptocurrency, not decentralized applications.

2.3.2. Permission

Permission of the blockchain decides the privacy of identity, transaction confidentiality, performance, and governance. While Ethereum and Polkadot are both public blockchains, most literature reviewed opt for permission blockchains. Kirli et al. (2022)suggest this is because the identity of the market participants is usually known by at least some party in the system. For example, the grid operator usually knows their identity when energy is delivered by the main grid. Another reason is consumers often do not want their demand and bid price to be publicly known. Examples of permission blockchains include Hyperledger Fabric and Hyperledger Besu adopted by Abdella et al. (2021). Esmat et al. (2021) chose Hyperledger Burrow and identified the clearly defined authority and governance as a reason to prefer permission blockchain since it makes upgrading the network easier.

2.3.3. Consensus Protocol

The consensus protocol plays a crucial role in the security model and carbon footprint of the blockchain. PoW protocol consumes a lot more energy which defeats the purpose of P2P energy trading. PoS protocol is energy efficient compared to PoW. The block production frequency is also more predictable and therefore better transaction throughput. Ethereum after the upgrade and Polkadot both use PoS-based protocols. A preliminary study by Kapengut and Mizrach (2023) on Ethereum's transition to use PoS found energy consumption has been reduced by 99.98%. Tendermint is another kind of PoS adopted by some permission blockchains, such as the Brooklyn Microgrid (Mengelkamp et al., 2018) and Quartierstrom (Brenzikofer et al., 2019) projects.

2.3.4. Application Layer

Blockchains by themselves are just databases. They need to be programmable to build useful applications on top of them. Ethereum supports smart contracts, which are a set of instructions to execute on an EVM when a condition is met. Smart contracts can be read by anyone in the network, and the instructions cannot be changed once deployed. Therefore, they are tamper-proof and transparent, but at the same time hard to update which makes security vulnerabilities hard to fix (Kirli et al., 2022). The blockchain itself is also hard to upgrade, requiring a hard fork. Nodes that do not accept the changes will not be able to participate in the network anymore. One example is in 2016, nodes that did not upgrade forked into a new blockchain, Ethereum Classic.

Still, a survey from the authors found smart contracts to be the most popular choice in literature. Besides Ethereum, there are EVM-compatible blockchains that can execute smart contracts, such as Hyperledger and Solana. Many of the literature referenced previously are implemented as smart

contracts. Doan et al. (2021) implemented the auction process and deployed it to Hyperledger Fabric. Esmat et al. (2021) used smart contracts to manage market enrollment, transferring of tokens and storing and validation of market clearing results.

Another approach is building a special-purpose blockchain. Polkadot has a good ecosystem for building application-specific parachains. They also support smart contracts, but one would need to find a suitable parachain to deploy to and subject to their economic design. Another strength of Polkadot is its forkless upgrade without downtime. The runtime logic is compiled into WebAssembly (WASM) and stored in the blockchain state. A new version is distributed to all nodes during the consensus process.

We have not found projects that implement a P2P energy market as an application-specific blockchain. Nevertheless, in the waste management field, Scott et al. (2021) built the BEE2WasteCrypto project as a parachain in Polkadot.

2.3.5. Scalability

The blockchain trilemma is a common belief that decentralization, security and scalability cannot be achieved simultaneously (Zhou et al., 2020). Improving scalability will either be at the cost of more centralization or less security. For example, reducing the number of nodes can improve the latency to reach consensus, but at the same time, the system becomes more centralized. Yet scalability is also important because it reduces the cost of transactions.

To address this, Ethereum has a roadmap to become more scalable and remain secure. First is layer 2 and rollup solutions. Layer 2 (L2) are standalone blockchains that aggregate multiple transactions into one and rollup to layer 1 (L1), which is Ethereum. This way L2 transaction fees are reduced while maintaining the security benefit of L1. Note that L2 and rollup do not increase the throughput of L1 Ethereum, it is still limited to 15 TPS. TPS will be improved in the next milestone, Danksharding (*Danksharding | Ethereum.Org*, n.d.). The goal is to further reduce the L2 transaction fee and scale to more than 100,000 TPS through a new sharding design. However, it is estimated to take multiple years to implement.

On the other hand, Polkadot adopts a multi-chain solution. Each parachain is a shard in the Polkadot network. They aggregate and submit their transactions to the relay chain, similar to how L2 rollup to L1. It is estimated that the TPS will scale to 100,000 to 1,000,000 (Habermeier, 2023).

2.3.6. Cost

On Ethereum, executing a smart contract consumes fees called gas. Gas consumption increases as the contract gets more complicated. The transaction cost is a real concern such that Han et al. (2020) stated saving gas as an objective of their proposed double auction algorithm. Mezquita et al. (2019) also listed using different blockchains with lower transaction fees as a possibility to improve payoffs to its participants. Finally, Kapengut and Mizrach (2023) found that gas fee has gone up since Ethereum transitioned to PoS. The same study found the median fee to transfer ETH peaked at $28 in May 2021, dropped to $0.5 before the upgrade and raised to $1.7 afterwards. Note that the drop in the dollar value of fees since the peak can be partially attributed to the decline in the exchange rate between ETH and USD. Smart contracts have more instructions to execute than transferring ETH, so

the gas price will be higher than the median fee to transfer ETH. Ethereum has a vision to reduce transaction fees to less than $0.001, but that is years away.

For Polkadot developers, there is a fixed cost to acquire a slot for their parachains to connect to the network through an auction (*Parachain Slot Auctions · Polkadot Wiki*, n.d.). The lease for the slot lasts 2 years. The winning bid for auction 44 which ended on June 8th, 2023, was $810,742[2]. Smaller applications can deploy to Kusama[3], the canary but still production-grade network of Polkadot. The winning bid ending in March ranges from $1,200 to $12,000. Each lease lasts for 48 weeks. Once connected, parachains do not pay transaction fees to have their blocks included in the relay chain. It is up to the parachain to decide if and how much transaction fee to charge from their end users. Some applications might still find parachain economically inviable. To address this, Polkadot provides a parathread model that allows multiple applications to share the same parachain slot (*Parathreads · Polkadot Wiki*, n.d.). Parathreads and parachains are similar from a technological perspective, except parathread bids for block space, on a block-by-block basis. In summary, Polkadot offers the parachain and parathread models on the Polkadot and Kusama networks. This gives application developers more control over the transaction costs.

2.3.7. Currency

P2P markets also need to consider whether the trades are in fiat currencies, such as the U.S. Dollar or Euro, cryptocurrencies, such as Bitcoin or Ethereum, or publish its token. Trading in fiat currencies benefits from the relative stability of their price. However, fiat currencies cannot be transacted natively on a blockchain. It relies on external services such as an oracle or off-chain validation with ZK proof to ensure transactions have happened. On the other hand, cryptocurrencies can be traded on blockchains, but they are more volatile than traditional assets (Woebbeking, 2021). Ethereum has ETH and Polkadot has DOT. According to the website crypto.com[4], The price of ETH fluctuated between $1,000 to $2,100 USD from June 2022 to June 2023. In the same period, Polkadot price is between $4.3 to $9.5 USD.

Some projects decide to publish their tokens, such as ERC-20 tokens on Ethereum. Polkadot also supports ERC-20 tokens. This option gives the market operator more control over the economics of the platform, but at the same time increases the operational complexity to manage the tokens. One example is Power Ledger's dual token ecosystem. A local market-level token, Sparkz, can be purchased using local fiat currency. Sparkz tokens can be exchanged for POWR tokens to gain access to the platform. POWR tokens can then be used to reward consumers for purchasing and prosumers for generating renewable energy.

2.3.8. Summary

Table 1 summarizes the 6 key dimensions of Ethereum and Polkadot. In terms of permission and consensus protocol, both are public PoS blockchains. At the application layer, Polkadot offers more flexibility to build application-specific blockchains with an easier upgrade process. As for scalability, Polkadot was designed to be scalable with high TPS. Ethereum has a roadmap, but it will take

[2] https://parachains.info/auctions/
[3] https://kusama.network/
[4] https://crypto.com/price

multiple years to fully realized. Finally, when it comes to financial feasibility, Polkadot is more viable with the Kusama canary network and parathread model.

In summary, Polkadot is more suitable for this research. It offers more flexibility to build a blockchain specialized for the energy market. In addition, it provides high TPS and control over the transaction cost without compromising security.

Table 1 - Ethereum vs. Polkadot

	Ethereum	Polkadot
Permission	Public	Public consensus on relay chain, but parachains can be private
Consensus Protocol	Started as PoW, upgraded to PoS	PoS
Application Layer	Smart contracts, hard to upgrade	Application-specific parachains and smart contracts. Parachains are designed to have forkless upgrade
Scalability	Layer 2, rollup and sharding. Up to 100,000 TPS. Currently, Layer 1 only has 15 TPS	Multi-chain sharded network. 100,000 to 1,000,000 TPS
Cost	The gas fee for executing a smart contract is proportional to the number of instructions. As a baseline, it takes about $1.7 to transfer ETH	Fixed cost to acquire a parachain slot. Kusama canary network is more affordable, as low as $1,200 for a 48-week lease. It also supports parathreads to join the network block-by-block
Currency	ETH cryptocurrency invented ERC-20 tokens	DOT cryptocurrency, supports ERC-20 tokens

2.4. REAL WORLD PROJECTS

In the last section of the literature review, we examine P2P markets using blockchain that has been tested with real users.

Mengelkamp et al. (2018) did a study on the Brooklyn Microgird, a commercial project in Brooklyn, New York, U.S.A. The project provides a double auction market with a quarter-hour time slot implemented by a smart contract. The blockchain is a private one based on the Tendermint protocol. Electricity consumption and generation are tracked by smart appliances that report back to the blockchain. In addition to the existing grid, a physical grid is built to uncouple the microgrid from the outdated infrastructure in case of emergency, such as severe weather events. The case study concluded that the project could increase locally generated renewable energy, but more testing is needed to understand the allocation efficiency of the market and pricing mechanism.

Quartierstrom is another project in Switzerland that has been tested in a local community (Brenzikofer et al., 2019). Similar to the market mechanism in the Brooklyn Microgrid project, consumers and prosumers can trade energy in a double auction market with a quarter-hour time slot on a private blockchain also based on the Tendermint protocol. The utility company and prosumers run validator nodes that can propose new blocks, while pure consumers run client nodes (Ableitner et al., 2019). One novelty of this project is the implementation of a dynamic grid usage tariff to stabilize voltage. When the voltage level is high, the tariff is lowered to incentivize consumers with

flexible loads to draw power. Ableitner et al. (2020) also conducted a follow-up user study and found that the participants became more incentivized to invest in DERs.

Besides the Brooklyn Microgrid project, Goranovic et al. (2017) provided an overview of 13 more projects that apply blockchain in microgrids. Over half of those projects use Ethereum-based blockchains.

2.5. SUMMARY AND GAP

From the literature review, we conclude that P2P energy markets are an active research area. It plays a crucial role in the future of the energy market because it can:

- Incentivize investment in DERs: a P2P market can offer better selling prices to prosumers than FiT from the grid.
- Help transition to renewable energy: consumers can buy locally produced renewable energy, such as from the PV panel of a neighbour. This in turn incentives prosumers to invest in renewable energy generators and storage.
- Improve energy security: communities can form microgrids and reduce reliance on the traditional grid. With DERs, energy does not need to be transmitted over long distances.
- Reduce peak energy demand: consumers can get lower prices if they shift usage from the peak demand of the grid, so some consumers will be incentivized to shift demand.

The literature review also examined blockchain technology. We concluded that blockchain is a suitable technology to implement the proposed p2p market as it provides trust, traceability, and data integrity in a decentralized network.

However, there are a few gaps in the current literature:

- Simple product: the model or clearing mechanism does not capture the inter-temporal constraints of real appliances. They also do not provide flexibility to shift load.
- Public blockchain with high transaction cost: most literature that adopts a public blockchain chose Ethereum. While it is very secure, the transaction cost is also high.
- Private blockchain with less security: some projects use private blockchain to avoid high transaction costs. However, they are less secure. A few malicious actors can compromise the network.

The next section will present market design and blockchain implementation that addresses these gaps.

3. METHODOLOGY

3.1. USER JOURNEY

In design science research, it is common to think about the user journey before designing a solution. We identified the four stages of energy trading for consumers in figure 4 and for prosumers in figure 5. This research will focus on building a market platform for the second and third stages. While the first and last stages are also important, they would require input from the physical layer. On our platform, consumers can submit bids and prosumers can submit offers for energy. The platform will also match supply with demand and settle trades.

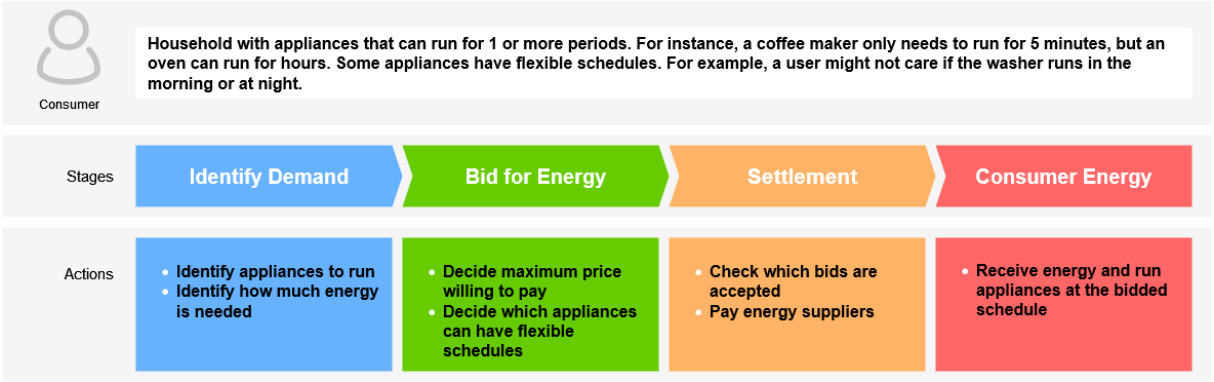

Figure 4 - Consumer user journey

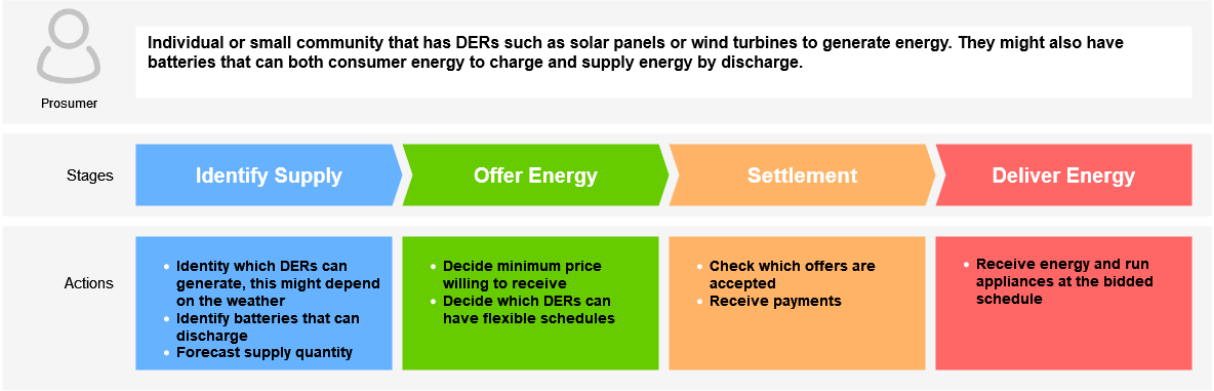

Figure 5 - Prosumer user journey

Essential to allow the consumer and prosumer to exchange energy is the design of the market. Thus, section 3.2 dives into the theoretical aspect of the market design, building on top of what we learned from the literature review in section 2.1. The outcome is a market formluation and a program to solve it. Then section 3.3 focuses on implementing a trading platform on blockchain based on the literature review in sections 2.2 and 2.3. Finally, section 3.4 covers the testing and benchmarking methodologies. The blockchain development is summarized in figure 6. The results will be presented in section 4.

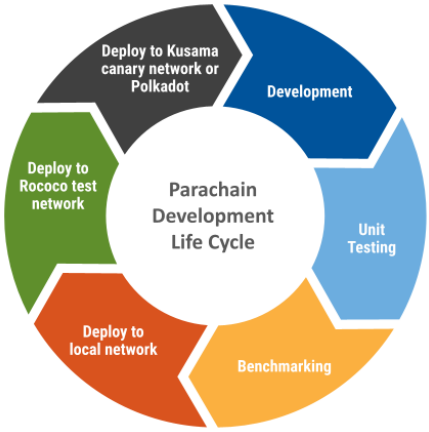

Figure 6 - Parachain development lifecycle

3.2. MARKET DESIGN

3.2.1. Model Products

To determine what products should be traded on the platform, we need to decide on the target market participants. This research focuses on consumers with household appliances and prosumers with DERs. Each appliance and DER are products that can be traded. Our definition of a product has the following properties:

- Price: For a bid, this is the maximum price which the consumer is willing to buy. For an offer, this is the minimum price the producer or prosumer is willing to sell.
- Quantity: Units of energy to buy or sell at each period.
- Starting period: The starting period to buy or sell energy.
- Ending period: The ending period to buy or sell energy.

A product that spans multiple periods is called a continuous product (Esmat et al., 2021). Otherwise, we call it a single product. To handle the inter-temporal constraint of continuous products, the market divides a day into multiple periods of equal length, but all periods are cleared together. In the rest of this research, we will call an energy demand bid and an energy supply offer.

Let t denote a period and d be the total number of periods in a day, $t \in [\![0, d]\!]$. Let B be the set of all bids and O be the set of all offers. Each bid $b \in B$ is a product with a starting period $s_{b,j} \in [\![0, d]\!]$ and ending periods $e_{b,j} \in [\![0, d]\!]$ so the operating period is $o_{b,j} = [\![s_{b,j}, e_{b,j}]\!]$. A bid has a single price $p_{b,j}$ and quantity $q_{b,j}$ throughout the operating period. An offer has a similar notation but with subscript o.

In addition, we introduce the concept of flexible products to encourage consumers to shift usage from peak hours (Tushar et al., 2021). It also incentivizes prosumers to supply during peak demand. A flexible product is expressed as a list of products where at most one of them can be accepted. The market will pick the one that maximizes social welfare. A flexible bid is indexed by $j = [\![0, k_b]\!]$ with a binary activation variable $i_{b,j} \in \{0,1\}$ such that $\sum_{j=0}^{k_b} i_{b,j} \leq 1$. A flexible offer has a similar notation but with subscript o.

Let's demonstrate the notations with an example. A prosumer might want to charge a battery either in the morning or at night. If it is charged in the morning, the prosumer is willing to pay up to 3 € from 10 am to 12 pm, each hour for 10kWh. But if it is charged at night, the prosumer is willing to pay 2 € from 8 pm to 11 pm, each hour for 5kWh. This demand can be modelled as a flexible product of 2 bids. Assuming a day is divided into 24 periods, the market has $t \in [\![0,23]\!]$. The first bid is indexed by $j = 0$ and has a price $p_{b,0} = 3$, quantity $q_{b,0} = 10$, starting period $s_{b,0} = 10$, ending period $e_{b,0} = 12$ and binary activation variable $i_{b,0}$. The second bid is indexed by $j = 1$ and has a price $p_{b,1} = 2$, quantity $q_{b,1} = 5$, starting period $s_{b,1} = 20$, ending period $e_{b,1} = 23$ and binary activation variable $i_{b,1}$. To ensure at most 1 bid is accepted, the flexible product has a constraint $0 \leq i_{b,0} + i_{b,1} \leq 1$.

3.2.2. Market Problem

Next, we define how the market matches supply and demand. In section 2.1.2 we surveyed the market objective used in various research and found social welfare to be commonly used. In this research, we also decided the market will optimize social welfare. Formally, social welfare is defined as the difference between the total utility of consumers and the total cost of prosumers (Khorasany et al., 2018):

$$SW = \sum_{b \in B} \sum_{j=0}^{k_b} \sum_{s_{b,j}}^{e_{b,j}} p_{b,j} q_{b,j} i_{b,j} - \sum_{o \in O} \sum_{j=0}^{k_o} \sum_{s_{o,j}}^{e_{o,j}} p_{o,j} q_{o,j} i_{o,j} \quad (1)$$

For simplicity, the market uses a uniform pricing scheme. Each period will have an auction price. The auction price is what all the buyers will pay, and sellers will receive if their products are accepted for that period. Let's denote the auction prices for period t as v_t. The market has the following constraints:

1. The bid price is the maximum price the buyer is willing to pay. Therefore, each bid price must be greater than or equal to the auction price from the starting to ending period:

$$i_{b,j} = 1 \Longrightarrow p_{b,j} > v_t \,, \forall b \in B \,, \forall j \in [\![0, k_b]\!], \forall t \in [\![s_{b,j}, e_{b,j}]\!] \quad (2)$$

2. The offer price is the minimum price the seller is willing to deliver. Therefore, each offer price must be less than or equal to the auction price from the starting to ending period:

$$i_{o,j} = 1 \Longrightarrow p_{o,j} > v_t \,, \forall o \in O, \forall j \in [\![0, k_o]\!], \forall t \in [\![s_{o,j}, e_{o,j}]\!] \quad (3)$$

3. Accepted bid quantity and offer quantity are equal for each period.

$$\sum_{b \in B} \sum_{j=0}^{k_b} q_{b,j}\, i_{b,j} = \sum_{o \in O} \sum_{j=0}^{k_o} q_{o,j}\, i_{o,j} \quad (4)$$

4. Only one product in a flexible bid can be accepted. The same applies to an offer.

$$\sum_{j=0}^{k_b} i_{b,j} \leq 1 \; and \sum_{j=0}^{k_o} i_{o,j} \leq 1$$

(5)

The third constraint of matching quantity is hard to satisfy if a product is either fully accepted or none. Since DERs such as batteries can be partially charged or discharged, the market introduces a new variable $c \in [\![0,1]\!]$ to denote the percentage of quantity that is accepted. $c \in \{0,1\}$ if the product cannot be partially accepted.

To summarize, the market objective is to maximize social welfare, subject to the constraints defined in equations 2 to 5. The market problem can be expressed as:

$$max \sum_{b\in B}\sum_{j=0}^{k_b}\sum_{s_{b,j}}^{e_{b,j}} p_{b,j}q_{b,j}i_{b,j}c_{b,j} - \sum_{o\in O}\sum_{j=0}^{k_o}\sum_{s_{o,j}}^{e_{o,j}} p_{o,j}q_{o,j}i_{o,j}c_{o,j}$$

$$s.t. \sum_{b\in B}\sum_{j=0}^{k_b}\sum_{t=s_{b,j}}^{e_{b,j}} v_t < p_{b,j}i_{b,j} + (1-i_{b,j})U$$

$$\sum_{o\in O}\sum_{j=0}^{k_o}\sum_{s_{o,j}}^{e_{o,j}} v_t > p_{o,j}i_{o,j}$$

$$\sum_{b\in B}\sum_{j=0}^{k_b}\sum_{s_{b,j}}^{e_{b,j}} q_{b,j}i_{b,j}c_{b,j} - \sum_{o\in O}\sum_{j=0}^{k_o}\sum_{s_{o,j}}^{e_{o,j}} q_{o,j}i_{o,j}c_{o,j} = 0$$

$$\sum_{b\in B}\sum_{j=0}^{k_b} i_{b,j} \leq 1$$

$$\sum_{o\in O}\sum_{j=0}^{k_o} i_{o,j} \leq 1$$

(6)

In equation (6), the first constraint satisfies equation (2). If the bid is accepted, $i_{b,j} = 1$ and therefore $v_t < p_{b,j}i_{b,j}$. If the bid is not accepted, $i_{b,j} = 0$ so $v_t < (1-i_{b,j})U$. U is an upper bound on the auction price. We pick the price to buy from the grid. The second constraint satisfies equation (3). If $i_{o,j} = 1$, then $v_t > p_{o,j}i_{o,j}$. Otherwise $v_t > 0$. The third constraint is an expansion on equation (4) to consider partially accepted products. The fourth constraint satisfies equation (5) for all bids and the last constraint satisfies equation (5) for all offers.

3.2.3. Market Clearing by MILP

The market problem defined in equation (6) has three kinds of variables to solve:

- c: the percentage of a product that will be accepted.
- i: which product in a flexible product will be accepted.

- v_t: the estimated auction price for each period. It is between the constraint of equation (2) and equation (3). In our market, the final auction price is the lowest bid price above this estimate. This gives prosumers a better price to incentivize their participation.

Equation (6) is not linear because the objective is a product of i and c, but we can use the Big-M method to linearize it (Bazaraa et al., 2011). We apply it to move the binary activation variables from the objective into two constraints. Recall that $c \in [\![0,1]\!]$ and $i \in \{0,1\}$. If $c > 0$, then $i = 1$. So, the first additional constraint is:

$$c_{b,j} <= i_{b,j}, \forall b \in B, \forall j \in [\![0, k_b]\!]$$

$$c_{o,j} <= i_{o,j}, \forall o \in O, \forall j \in [\![0, k_o]\!] \tag{7}$$

In equation (7), $c = 0$ and $i = 1$ is a valid solution. To ensure $i = 0$ when $c = 0$, the second additional constraint is:

$$i_{b,j} \leq M c_{b,j}, \forall b \in B, \forall j \in [\![0, k_b]\!]$$

$$i_{o,j} \leq M c_{o,j}, \forall o \in O, \forall j \in [\![0, k_o]\!] \tag{8}$$

M must be sufficiently large to satisfy equation (7). The market assumes that at least 1% of a product has to be accepted, so we pick $M = 100$. With equations (7) and (8), the objective function can be simplified to:

$$max \sum_{b \in B} \sum_{j=0}^{k_b} \sum_{s_{b,j}}^{e_{b,j}} p_{b,j} q_{b,j} c_{b,j} - \sum_{o \in O} \sum_{j=0}^{k_o} \sum_{s_{o,j}}^{e_{o,j}} p_{o,j} q_{o,j} c_{o,j} \tag{9}$$

The transformed market problem is a MILP optimization problem. A common technique to solve it is through the branch and bound algorithm (Clausen, 2003):

1. Relax the integer constraint, so the problem can be solved by LP such as the simplex method (Nash, 2000). This gives us an upper bound on the objective value. It found a solution if the variables are already integers. Otherwise, this is the root node of our search tree.
2. Round down each variable to the closest integer, and use it to compute the lower bound of the objective value.
3. Pick variable i with the greatest fractional part. Round it down to x.
   3.1. Solve the relaxed LP problem with an extra constraint $i \leq x$. Add this as a child node.
   3.2. Solve the relaxed LP problem but this time with an extra constraint $i \geq x$. Add this node.
   3.3. Branch from the constraint that has a higher objective value.
4. Repeat step 3 for other variables. Some branches might not yield a feasible solution.
5. Stop when there is an integer solution that has a higher objective value than all of the previous nodes.

Figure 7 provides a visualization of how the branch and bound algorithm matches 2 offers and 3 bids. For simplicity, we assume the products only have fixed loads and they cannot be partially accepted. $O = \{o_0, o_1\}, B = \{b_0, b_1, b_2\}$.

1. $o_0: p_{o_0} = 5, q_{o_0} = 3, s_{o_0} = 0, e_{o_0} = 2$
2. $o_1: p_{o_1} = 3, q_{o_1} = 3, s_{o_1} = 0, e_{o_1} = 1$
3. $b_0: p_{b_0} = 8, q_{b_0} = 2, s_{b_0} = 0, e_{b_0} = 2$
4. $b_1: p_{b_1} = 6, q_{b_1} = 3, s_{b_1} = 0, e_{b_1} = 1$
5. $b_2: p_{b_2} = 7, q_{b_2} = 2, s_{b_2} = 1, e_{b_2} = 2$

Step 1 of the branch and bound algorithm relaxes the constraints to be a continuous variable $0 <= I_x <= 1$. Then each node adds a bound to one variable until no further node yields a solution. Node 3 is the only one with all integers, so it is the final solution. The result is o_1 and b_1 are fully accepted.

For this project, we use the COIN CBC solver from Forrest et al. (2023) with an open-source license. A program is developed to read market data from a blockchain, feed it to the solver and submit the solution back to the blockchain. Section 4 provides more details on the interaction between this program and the blockchain.

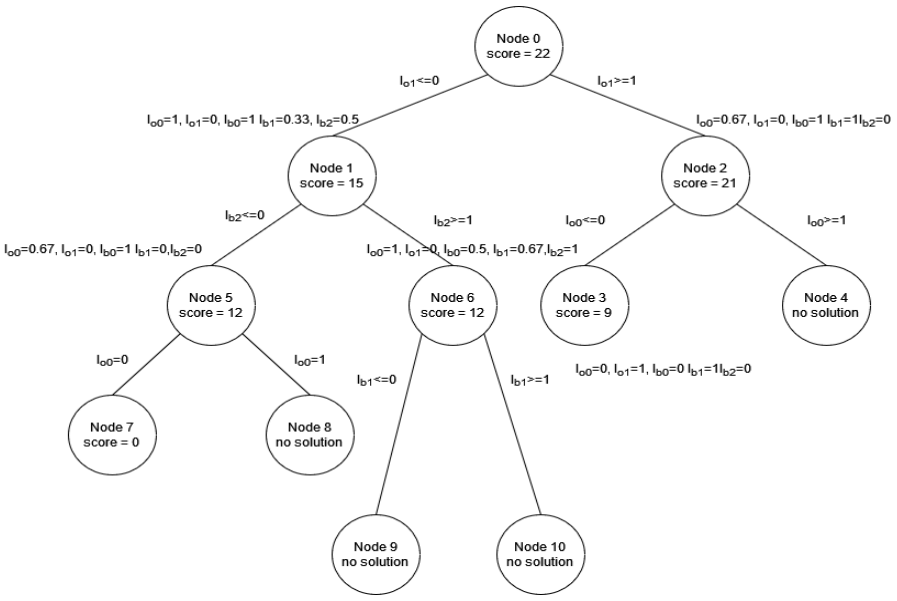

Figure 7 - Branch and bound algorithm

3.2.4. Double Auction as Baseline

The optimization problem from the previous section is not suitable for solving on the blockchain, because the result is not deterministic, and the execution time is not predictable. In section 2.1.3, we surveyed other clearing mechanisms and found double auction is another popular approach. It is simpler to implement, but it does not consider the inter-temporal constraint for continuous products. Nevertheless, it can be used to compare with solutions from a MILP solver. In the next section, we will see that it is simple enough to run on a blockchain. In essence, a double auction finds the intersection of the supply and demand curve. For our market, we assume the first product in a flexible product is the preferred one, so we do not need to worry about the mutually exclusive constraint in equation (5). Then the rest of the algorithm is:

1. Aggregate bid quantities by descending price because there are more demands when prices are lower.
2. Aggregate offer quantities by ascending price because there are more supplies as the price goes up.
3. Iterate through the aggregated bid and offer quantities until they match. This is a valid solution if the bid price is greater than or equal to the offer price. We pick the lowest bid price as the auction price.
4. Repeat for all periods.

Note that since the matching is independent between periods, a continuous bid might have matching offers in the first period, but not in the second period. In that case, we declare there is no solution. Figure 8 shows how a double auction would match the products from the example in the pervious section. In the first period, b_0, b_1 and o_1 are fully accepted while o_0 is partially accepted. In the second period, b_0 is fully accepted while o_0 is partially accepted. However, if the products cannot be partially accepted, the double auction would not find a valid solution. Whereas MILP optimization can match o_1 with b_1.

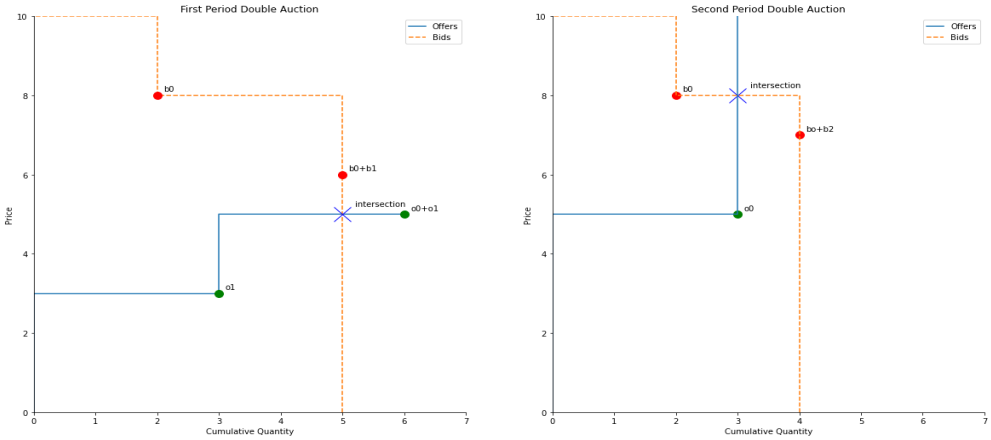

Figure 8 - Double Auction

3.3. BLOCKCHAIN DEVELOPMENT

In section 2.3, we reviewed popular blockchains and decided to implement our market on Polkadot for lower cost and better scalability. The Polkadot ecosystem supports two methods to develop applications. Developers can choose to build a parachain which is more flexible but also more complicated. On one hand, the parachain logic can be tailored to fit business goals and we have more control over the economics. On the other hand, parachain developers are responsible for finding nodes to run the network. Another option is building smart contracts on an existing parachain. Smart contract developers can focus on the business logic and not worry about the infrastructure. But they need to find a suitable parachain to host their smart contracts. For the proposed trading platform, we need more control over the direction of the blockchain, therefore parachain is a better option.

3.3.1. Substrate and Node Architecture

To build a parachain, we need to develop nodes that will form the parachain. Recall from section 2.3.1, Polkadot has validator and collator nodes. A parachain is run by collator nodes. The collators

handle transactions from the users and submit state transition proofs to validators. Once the state transitions are validated, they are included as blocks on the relay chain.

The Polkadot ecosystem provides a software development kit, Substrate[5], to simplify the development of validator and collator nodes. For this research, we focus on developing a collator node. A node consists of a runtime, off-chain workers, and a client with outer node services. The runtime has the application-specific logic to manage state transitions. It is designed to compile to WASM byte code for two advantages. First, the WASM byte code can be included in the state to support forkless upgrades. During an upgrade, the new logic will be compiled into WASM byte code and included in a block. The consensus protocol distributes the block that includes the new runtime to all nodes. The second benefit is WASM is compatible with multiple platforms (32-bit vs. 64-bit, AMD64 vs. ARM etc.), so developers do not need to worry about the hardware where the node is running (*Introduction — WebAssembly 2.0*, 2023).

Besides the runtime, a node has a separate WASM execution environment for off-chain workers. A worker is spawned during each block import. It can be used for long-running tasks that take longer than a block production time, computation with nondeterministic results or queries from off-chain data sources. This environment is not trusted, so the runtime still needs to validate results from the workers. Note that the worker is spawned by every collator, so developers must make sure the runtime logic can reconcile multiple results. The off-chain computations should not use too many resources because that will disincentivize someone from running a node.

In contrast, the client can run in native Rust since it depends on the underlying platform for functions such as reading from the file system or making network calls. It handles events outside of the runtime. The important outer node services are:

- Storage: A key-value database where the blockchain state is persisted. Substrate builds a Base-16 Modified Patricia Merkle Tree (trie)[6] abstraction on top of it. Trie is a tree data structure where the key is not stored in a node but defined by its path in the tree. This structure is space-optimized to store keys with a shared prefix. But reading and writing are expensive operations with $O(log\ N)$ complexity, where N is the number of entries in the trie. The advantage of this structure is that two tries can be checked if they are equal efficiently by comparing the root of the tree. Therefore, states can be compared for equality efficiently. For developers, Substrate provides a higher-level application program interface (API) to create single-value storage or map-based data structures.
- Remote procedure call (RPC) API: External programs can call RPC methods to query storage or submit transactions. A node has an HTTP and WebSocket (WS) port to serve RPCs. The ports can be secured by placing a proxy in front of it to upgrade connections to encrypted HTTPS or Secure WebSocket (WSS), and rate limiting to protect the node from denial-of-service (DoS) attacks.
- Consensus: The consensus mechanism consists of a block production and finalization phase. As of May 2023, Polkadot's relay chain uses BABE (Blind Assignment for Blockchain Extension)(Alper, n.d.) to produce new blocks and GRANDPA (GHOST-based Recursive Ancestor Deriving Prefix Agreement) (Stewart & Kokoris-Kogia, 2020) to finalize

[5] https://substrate.io/
[6] https://github.com/paritytech/trie

blocks. BABE is a slot-based PoS protocol. The validator nodes run the Block-Production-Lottery algorithm[7] to find out in which slots it should produce blocks. If multiple validators are selected for the same slot, the block that reaches most of the network wins. The lottery incorporates randomness, so an adversary will not know who the next block author will be to attack. To resolve forks, GRANDPA is used to finalize blocks. It is a Byzantine fault-tolerant algorithm that can handle up to 1/3 of nodes being malicious. Validators vote on the chain that they consider the best, and the votes are applied to all previous blocks in the chain. This algorithm can finalize multiple blocks that have enough votes in one round, which makes reaching consensus faster.

- Networking: Each node has a private and public key. The private key is used for encryption and the public is used to uniquely identify a node. The node connects to a bootstrap node that has a static IP address. Then the node can start discovering other peers in the network[8].

Next, we explain how the client, runtime, and off-chain worker function together, and how the nodes form a parachain. Figure 9 provides a visualization of the architecture. Every blockchain starts with the genesis block, including parachains. To start a parachain, we need to register the genesis state and WASM runtime validation logic with the relay chain. Then we can start our node as a collator. The networking service in the client connects to both the parachain network and relay chain network, configured by the chain specification file.

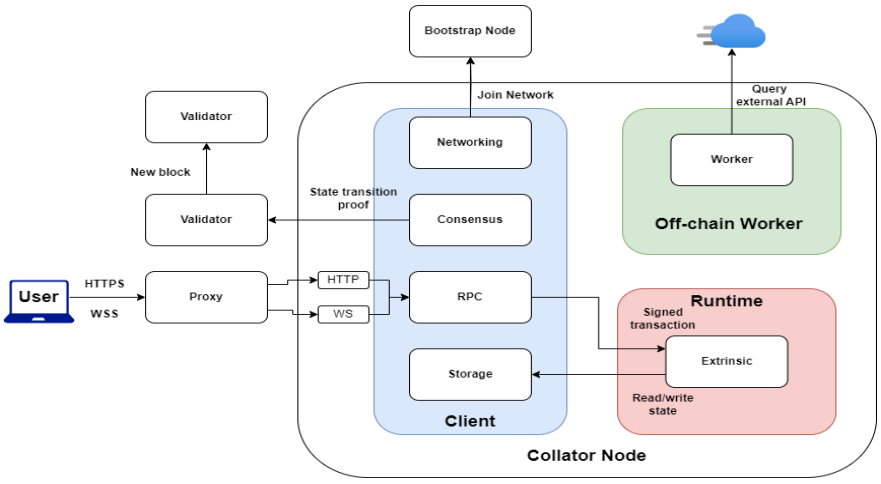

Figure 9 - Architecture of a Node

Then the consensus component comes into play. In the parachain network, it gossips transactions and blocks that are likely to be included in the relay chain to other nodes in the parachain. In the relay chain network, it sends parachain blocks and state transition proofs to the validators (Petrowski, 2020). The Merkle tree data structure allows validators to verify state transitions without reading the full state. Collator only needs to provide the list of transactions, values modified by the transactions and the hashes of nodes in the Merkle tree that remain unchanged. Polkadot reassigns different validators to parachains regularly to avoid collusion or a validator from discriminating against certain parachains.

[7] https://spec.polkadot.network/sect-block-production#sect-block-production-lottery
[8] https://crates.parity.io/sc_network/index.html#discovery-mechanisms

The outside world can interact with the node through RPCs. It can be used to submit extrinsic to the runtime to modify the state. An extrinsic can be a signed transaction using the private key of the sender plus a transaction fee. But it can also be an unsigned transaction that requires custom validation by the runtime. Once the runtime validates the transaction, it can be bundled with other transactions to be included in a block. Then the block goes through the consensus process to be finalized. There are also RPCs for querying state from storage without the caller providing an identity or paying a transaction fee. Therefore, node operators should enforce size and rate limits to protect themselves from DoS attacks.

The node can also interact with the outside world through off-chain workers. For example, it might need to query an external API to get the latest energy price. The worker can submit the API response on-chain via an extrinsic. The runtime will validate the response, possibly through some heuristics.

3.3.2. Pallet Development

Substrate provides a template to build nodes. It includes the client with outer node services and an execution environment for the runtime and off-chain worker. Our custom runtime and off-chain worker logic can be developed as a module called pallet. Then it can be added to the node through FRAME (Framework for Runtime Aggregation of Modularized Entities). The community already open-sourced many pallets for common functionality, such as handling account balance or reaching consensus. Our proposed P2P market will also be implemented as a pallet. Based on section 3.2, we identified the market should provide the following functionalities to market participants:

1. Each participant has an account which is identified anonymously by the public key.
2. An API to submit and update bids. Each bid is a flexible product.
3. An API to submit and update offers. Each offer is a flexible product.
4. An API to propose a solution to the market problem. It should include the auction prices for each period, bids accepted and offers accepted.
5. A basic solution if no market player submits one.
6. An API to query proposed bids and offers.
7. An API to query market clearing results, auction prices, accepted bids and offers.

These functionalities will be implemented as runtime logic and an off-chain worker in a pallet. Pallets are written in Rust, a compiled and memory-safe programming language. Rust has many features, such as compiler optimizations to make the code run faster. It can also be compiled into WASM for the runtime. However, many types and functions are not available in WASM because they depend on the underlying platform to support them. For example, networking requires interfacing with the platform's network stack. To make development easier, Substrate provides its implementations for many standard library features, but it still influences our implementation.

A pallet consists of storage, extrinsic, events, hooks, and configuration. Users interact with the pallet through extrinsics. The extrinsics might query or change states which are persisted in storage. Extrinsics can also generate events to notify users that something happened. The hooks are for parachain developers to implement custom logic during different lifecycles of a block. The configuration is for node operators to customize the pallet. In the next few sections, we will explain their design for our P2P market. But first, we define the data structures to model flexible products and continuous products defined in section 3.2.

```
pub struct Product {
    pub price: u64,
    pub quantity: u64,
    // A single product has end_period = start_period + 1
    pub start_period: u32,
    pub end_period: u32,
    pub can_partially_accept: bool,
}
pub type FlexibleProduct = BoundedVec<Product, MaxFlexibleLoadsPerProduct>;
```

The *Product* struct models both continuous and single bid/offer. Quantity is assumed to be in units of kilowatt-hour (kWh) and price is cents of Euro per kWh. They are both modelled as an integer type because computations in the runtime must be deterministic, but floating point arithmetic can have different results on different machines. We can work around this limitation by changing the units. For example, the price is expressed in cents/kWh, so instead of 1.99€/kWh, we have 199 c€/kWh. Next, we have *FlexibleProduct* which is a vector of *Product*. The vector is bounded in size. A bid or offer that has a fixed schedule is modelled as a *FlexibleProduct* with only 1 *Product* in the vector. Each bid or offer belongs to an account. Each account is identified by the public key in byte array format.

Next, we define how to store the bids and offers. The pallet needs to efficiently read and write bids and offers, so a map is a suitable storage. We will use Substrate's *StorageMap*[9] data structure. Each flexible product will be identified by an autoincrement integer ID, called product ID. Data stored in the map are encoded by the SCALE (simple concatenated aggregate little-endian) codec [10]. It can serialize and deserialize efficiently without copying memory. Besides *StorageMap*, data that have known size and are considered as a single unit can be stored in *StorageValue*[11] type. The pallet stores the optimal solution, accepted bids, accepted offers and current market stage as *StorageValue*. The market cycles between two stages, open and clearing. During the open staging, market participants can submit their bids and offers, but not solutions. The clearing stage works the opposite way. In Substrate time can be approximated by block numbers, so we can say the stage alternates every 50 blocks for example. In pseudo-code, we have the following data structure for storage:

```
pub type ProductId = u32;
// First type is for keys, second type is for values
pub type Bids = StorageMap<ProductId, FlexibleProduct>;
pub type AccountBids = StorageMap<AccountId, BoundedVec<ProductId, MaxProductPerPlayer>>;
pub type Offers = StorageMap<ProductId, FlexibleProduct>;
pub type AccountOffers = StorageMap<AccountId, BoundedVec<ProductId, MaxProductPerPlayer>>;

pub struct AcceptedProduct {
    pub id: ProductId,
    pub selected_index: usize,
    // Percentage of quantity accepted. [0, 100].
    // We use an integer here because the runtime cannot perform floating point arithmetic
    pub percentage: u8,
}
// The account that proposed the solution and auction price for each period.
pub type BestSolution = StorageValue<AccountId, BoundedVec<AuctionPrice, Periods>>;
pub type AcceptedBids = StorageValue<BoundedVec<AcceptedProduct, T::MaxProducts>>;
pub type AcceptedOffers = StorageValue<BoundedVec<AcceptedProduct, T::MaxProducts>>;

pub type MarketStage = u64;
pub type Stage = StorageValue<MarketStage>;
pub const MARKET_STAGE_OPEN: MarketStage = 0;
pub const MARKET_STAGE_CLEARING: MarketStage = 1;
```

[9] https://docs.substrate.io/build/runtime-storage/#single-key-storage-maps
[10] https://docs.substrate.io/reference/scale-codec/
[11] https://docs.substrate.io/build/runtime-storage/#simple-storage-values

The pallet must protect itself from large user input that could exhaust storage space or take longer than 1 block time to process. Hence the pallet has an upper bound of 5 products per flexible product, 1,000 flexible products per account and a total of 5,000 flexible products.

Most of the runtime logic is defined as extrinsic, each function executes a transaction. The pallet provides transactions for users to submit bids, offers or solutions. The first one is for submitting bids. Consumers and prosumers will use it to propose new or update existing energy demands. The logic is in the following pseudo-code:

```
pub fn submit_bids(
    origin: Origin,
    // It's a new bid if the product ID is not specified. Otherwise it's an update
    bids: BoundedVec<(Option<ProductId>, FlexibleProduct), MaxProductPerPlayer>,
) -> DispatchResultWithPostInfo {
    // 1. Validate the transaction is signed.
    // ? is an operator to return early if the function returns an error
    let sender = ensure_signed(origin)?;
    // 2. Check the current market stage can accept bids.
    if Stage::get() != MARKET_STAGE_OPEN {
        return Err(Error::WrongMarketStage.into())
    }
    // 3. Fetch the list of bids previously submitted by this account
    let current_products = AccountBids::get(&sender);
    for (bid_id, flexible_bid) in bids.iter() {
        // 4. validate the price is above FiT.
        // This is an update if bid_id is presented. Make sure this account is the owner
        // Otherwise, prosumers have more incentives to sell back to the grid.
        validate_product(&current_products, bid_id,  &flexible_bid, ProductType::Bid)?;
    }
    // 5. Get the last assigned bid ID. The next product will be assigned this ID + 1.
    let current_last_id = LastBidId::get();
    let mut new_last_id = current_last_id;
    let mut account_bids: BoundedVec<ProductId, MaxProductPerPlayer> = Default::default();
    for (id, flexible_bid) in bids.iter() {
        // 6. Insert a new bid or update existing one
        let id = match id {
            Some(id) => {
                Bids::insert(id, flexible_bid.clone());
                *id
            },
            None => {
                new_last_id += 1;
                Bids::insert(new_last_id, flexible_bid.clone());
                new_last_id
            },
        };
        account_bids.try_push(id).map_err(|_| Error::TooManyProducts)?;
    }
    LastBidId::set(new_last_id);
    // 7. Update the list of bids owned by this account.
    AccountBids::insert(sender.clone(), account_bids);
    // 8. Generate an event for the submitted bids.
    // Caller can find IDs of the new products from the event
    Self::deposit_event(Event::NewBids { account: sender, bids });
    Ok(().into())
}
```

Note that Substrate provides a transactional storage layer[12] similar to a transaction in a traditional database, so writes will not be committed if the transaction fails. If the transaction is going to succeed, we generate an event to notify external entities. For instance, a parachain explorer might subscribe to runtime events to show new bids to users.

Each extrinsic has a weight measured by the time it takes to execute. The caller will pay a transaction fee proportional to the weight. The number of read and write to the database is taken into

[12] https://docs.substrate.io/build/runtime-storage/#transactional-storage

consideration. For the submit bid extrinsic, it reads 3 times (steps 2, 3 and 5) and writes 2 + N times (steps 6, 7 and 8) to the database, where N is the number of bids submitted.

The second extrinsic is to submit offers. The logic is similar except the data are saved in different *StorageMap* and the price must be less than the grid price. The pallet does not solve the optimal solution, as it will require significant resources with unpredictable execution times. Instead, it provides a third extrinsic for market participants to submit their solutions. The parameters are the list of auction prices for each period, accepted bids, and accepted offers. The validation pseudo-code is:

```
pub fn submit_solution(
    origin: Origin,
    auction_prices: BoundedVec<AuctionPrice, ContinuousPeriods>,
    // For each account, provide a vector of which bids are accepted.
    // Each bid is represented by the product ID and flexible product index
    accepted_bids: BoundedVec<AcceptedProduct, MaxProducts>,
    accepted_offers: BoundedVec<AcceptedProduct, MaxProducts>,
) -> DispatchResultWithPostInfo {
    // 1. Validate the transaction is signed.
    let sender = ensure_signed(origin)?;
    // 2, Make sure the current market stage can accept solutions.
    if Stage::get() != MARKET_STAGE_CLEARING {
        return Err(Error::WrongMarketStage.into())
    }
    // 3. Validate the price of accepted bids for each period is higher than auction price.
    // Returns the total utility and bid quantities for each period.
    let (utility, bid_quantities) = validate_bids(&auction_prices, &accepted_bids)?;
    // 4. Validate the price of accepted offers for each period is lower than auction price.
    // Returns the total utility and bid quantities for each period.
    let (cost, offer_quantities) = validate_offers(&auction_prices, &accepted_offers)?;
    // 5. Make sure bid and offer quantity matches for each period.
    for (period, (bid_quantity, offer_quantity)) in
        bid_quantities.iter().zip(&offer_quantities).enumerate() {
        if bid_quantity != offer_quantity {
            return Err(Error::InvalidSolution)
        }
    }
    let social_welfare = utility - cost;
    // 6. Get the current solution, check if the proposed one is better
    let is_optimal = match BestSolution::get() {
        Some(current_solution) => social_welfare > current_solution.2,
        None => true,
    };
    // 7. If the social welfare score is greater than the previous score, store it.
    if is_optimal {
        BestSolution::set(Some((sender, auction_prices.clone(), social_welfare)));
        AcceptedBids::set(accepted_bids.clone());
        AcceptedOffers::set(accepted_offers.clone());
        // 8. Generate an event with the auction prices, accepted bids and offers.
        Self::deposit_event(Event::Solution {
            auction_prices,
            accepted_bids,
            accepted_offers,
            social_welfare,
        });
    }
    Ok(().into())
}
```

Pallets can implement hooks to execute some logic at different lifecycles for a block. Our pallet implements the hook that is called on the initialization of a block. If the current stage is open and 50 blocks have been added since the beginning of this stage, the market enters the clearing phase by flipping the *StorageValue* for the stage. Later when the stage transitions from clearing to open, the stage value is flipped again and the bids, offers and optimal solution have to be deleted from storage. There is no operation to clear the storage immediately. Instead, the pallet iterates through the keys and deletes them one by one. Thus, for a production parachain, there should also be a stage

between clearing and open to reset the state. Note that data on a blockchain cannot be deleted once the blocks are finalized, so historical data can still be retrieved by traversing the blocks.

Another usage of a hook is for defining the off-chain worker logic. Our pallet uses it to provide a baseline solution. When the stage transitions from open to clearing, a worker is spawned to solve a double auction for each period using the algorithm outlined in section 3.2.4. It is not solved on-chain as the computation time is proportional to the number of submitted products which can take longer than a block production time. If the worker finds a solution, it calls the submit solution extrinsic to validate it.

The pallet is functional with storage, extrinsic, events and hooks. However, some settings need to adapt to different use cases. For example, for testing, we might want the stages to alternate sooner than 50 blocks. This is where the configuration comes in. For our pallet, the maximum number of products per market player, maximum number of total products, gird price and FiT are also configurable.

Finally, to use our pallet on a node, we use FRAME to compose it with other pallets. Together they form the runtime of the node. Figure 10 shows what happens when a user submits bids to see how the various components in a pallet fit together.

In summary, we built an energy trading market as a pallet in Rust. During the open stage, market players can call extrinsics to submit their bids and offers. Then during the clearing stage, they can also participate by submitting a solution with the auction prices, accepted bids and offers. To get the list of bids and offers to match, one can send RPC to query storage. This RPC is provided by the Substrate template. When a solution is submitted, the extrinsic validates it satisfies the price and quantity constraint. It is accepted if the solution has a better social welfare score than the previous one. The pallet has a fallback mechanism to match by running a double auction in the off-chain worker in case no one submits a solution. The result is submitted on-chain for validation. The code is open source and configurable for other parachains to use. In the next section, we will explain different methods to test our implementation.

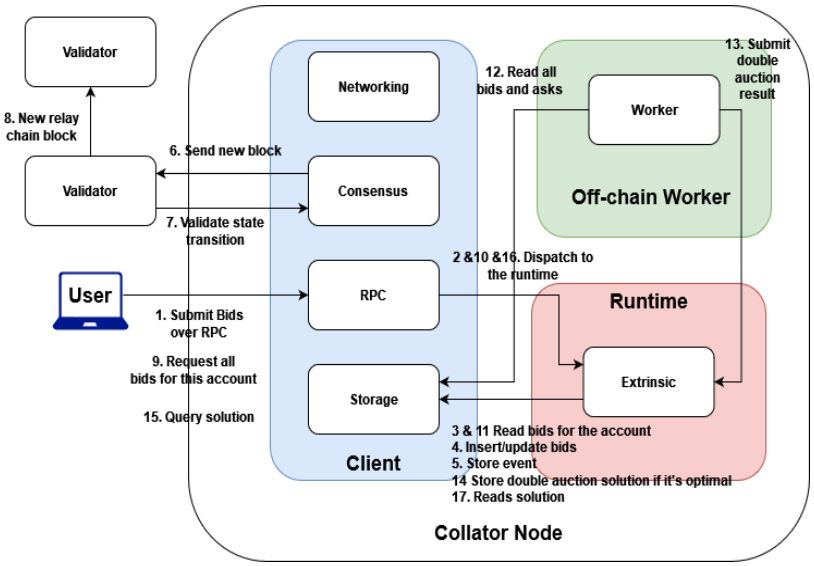

Figure 10 - Steps to Submit Bids

3.4. TEST AND BENCHMARK

Rust is a compiled language, so during compilation, it checks if functions are called with the correct type of arguments and whether errors are handled by the caller. This eliminates a class of common errors in interpreted languages such as Python. In addition, it provides a linter to catch common error patterns such as infinite loops, race conditions, unreachable logic, or gaps in pattern matching. Nevertheless, there can still be errors during runtime. We implemented unit tests and used integration tests with the local network to validate our logic. In addition, we ran benchmarks to understand the performance of the node.

3.4.1. Unit Tests

Unit tests are the most basic level of testing in software development. It is usually used to verify a function returns the expected value under different inputs. It becomes harder to write when the function has many dependencies. Hence, writing unit tests early can remind the developer to keep the functions modularized. When a function has side effects, such as writing to a database, it is no longer a pure function. In this case, the best practice is to change the function to work with a datastore interface, so the function can write to a mock database in tests. This helps the developer create abstractions that make the function more readable and reusable. It is also common practice to test with unexpected values to make sure the function handles it gracefully, usually by returning an error. This is one of the methods in defensive programming to validate inputs before using them.

For the pallet, we implemented a unit test to verify results from double auctions and another unit test for the logic that validates solutions. Invalid solutions were provided as inputs to make sure the function rejects them. For the off-chain optimizer, a set of unit tests was designed to verify it finds the optimal solution for single products, continuous products, continuous products with overlapping operating periods and flexible products. Market conditions without a solution were also tested to make sure the optimizer returned an error.

3.4.2. Benchmarking

In section 3.3.2 we explained the concept of weight. Predicting the correct weight of a function is important because the runtime needs to know if executing a transaction will exceed the block production time. It also helps developers decide on the appropriate transaction fees to deter DoS attacks. Weight can be estimated directly based on the number of reads and writes to the database plus some constant.

Another method is benchmarking. Writing benchmarks is similar to writing unit tests. If the function can execute different code paths based on user input, the best practice is to write a benchmark for each path. Then we can use Substrate's benchmarking tool[13] to execute the benchmark functions under a range of inputs. The resulting data points are used to fit a linear model for the weight of each extrinsic. The benchmark should be run multiple times to remove outliers. Note that the result depends on the hardware, so each node should run its benchmark.

[13] https://docs.substrate.io/build/runtime-storage/#transactional-storage

3.4.3. End-to-end Test with Local Network

Not everything can be accounted for in unit tests. Eventually, we need to run our software in a production-like environment. Substrate provides a guideline to run a local relay network with 2 validators. This network is preconfigured with a few accounts that have tokens. To connect a parachain to the local network, first, we need to connect to a relay chain node to reserve a unique *ParaID* for the parachain. Next, we need to generate the WASM runtime and genesis state of the runtime. Nodes built from Substrate already include commands to export them. Then we can use a web browser to connect to the WS endpoint of a relay chain node. This will give us access to a developer portal. In the portal, we call the *Sudo* method to register the parachain with the *ParaID*, WASM runtime and genesis state. Finally, we can specify the *ParaID* in the chain specification file and run the collator node. The collator node will automatically discover the validators to connect to.

With the local network running, we can simulate user interactions with the blockchain. First, we connect to the WS endpoint of the collator node to access the developer portal. Next, we use the extrinsics of our pallet to submit a few bids and offers. The runtime will generate events to show that the transactions succeed. We also query the storage to verify the products are saved correctly. When the market transitions from the open to the clearing stage, we wait for the off-chain worker to solve the double auctions. Then we submit a better solution to verify that the pallet persists the one with the better social welfare score. Finally, we wait for the market to transition to open stage again. The storage is emptied, which proves that market data from the previous day is cleaned up. In addition to the happy path, we also verified the transactions failed in these cases:

1. Bids or offers are submitted in the clearing stage.
2. Solutions are submitted in the open stage.
3. An account tries to submit more flexible products than the per-account limit.
4. An invalid solution, such as a bid price is lower than the auction price or the offer price is higher than the auction price.

3.4.4. Deploy to Real Networks

In normal software development, the program is deployed to a staging environment after it is well-tested. In the Polkadot ecosystem, it would be deploying to the Rococo test network[14]. This would require a real account with ROC tokens which are free to acquire. Next, we need to reserve a *ParaID* like how it was done in a local network and register it as a parathread. Parathreads are only connected to the relay chain network on a block-by-block basis. The collator will offer a bid to be included in a relay chain block. To upgrade to a parachain, we can submit a request for a temporary slot. Rococo has a pool of temporary slots to assign on a round-robin rotation. At the end of the lease, our parachain will be downgraded to a parathread again. The motivation to go through all these steps is to make sure our node works in a real network. It gives assurance that our state transition can be verified by real validators.

For a production environment, we can start with the canary network, Kusama. While Polkadot focuses on stability, Kusama is designed for fast iterations. New features are deployed to Kusama first, and it usually requires fewer tokens to bid for a parachain slot, so it is more suitable for new

[14] https://substrate.io/developers/rococo-network/

projects. After Kusama, we can decide if it makes sense to deploy to the Polkadot main network. Not every project needs to be on the main network since it will increase the operating cost. In the following section, we report the results of these tests and benchmarks.

4. RESULTS

4.1. OVERALL SYSTEM

This research developed a P2P market for local communities to trade energy. Each community can run their market by deploying its parachain or parathread. The software is open-sourced [15] and configurable. Each community can decide its limit on the maximum market players and product per player, number of continuous periods to clear together, FiT and grid price. Then some members of the community can run collator nodes to form the blockchain. There is no formal hardware requirement for running a node, a laptop can do it. A portion of the transaction fee can be awarded to collators to encourage some members to run nodes. Others can connect to one of the collator nodes to make transactions or query the market state. Polkadot provides a user interface (UI)[16] to interact with nodes. However, as shown in Figure 11, it is not very intuitive, so further research should be done to build an easy-to-use app or website.

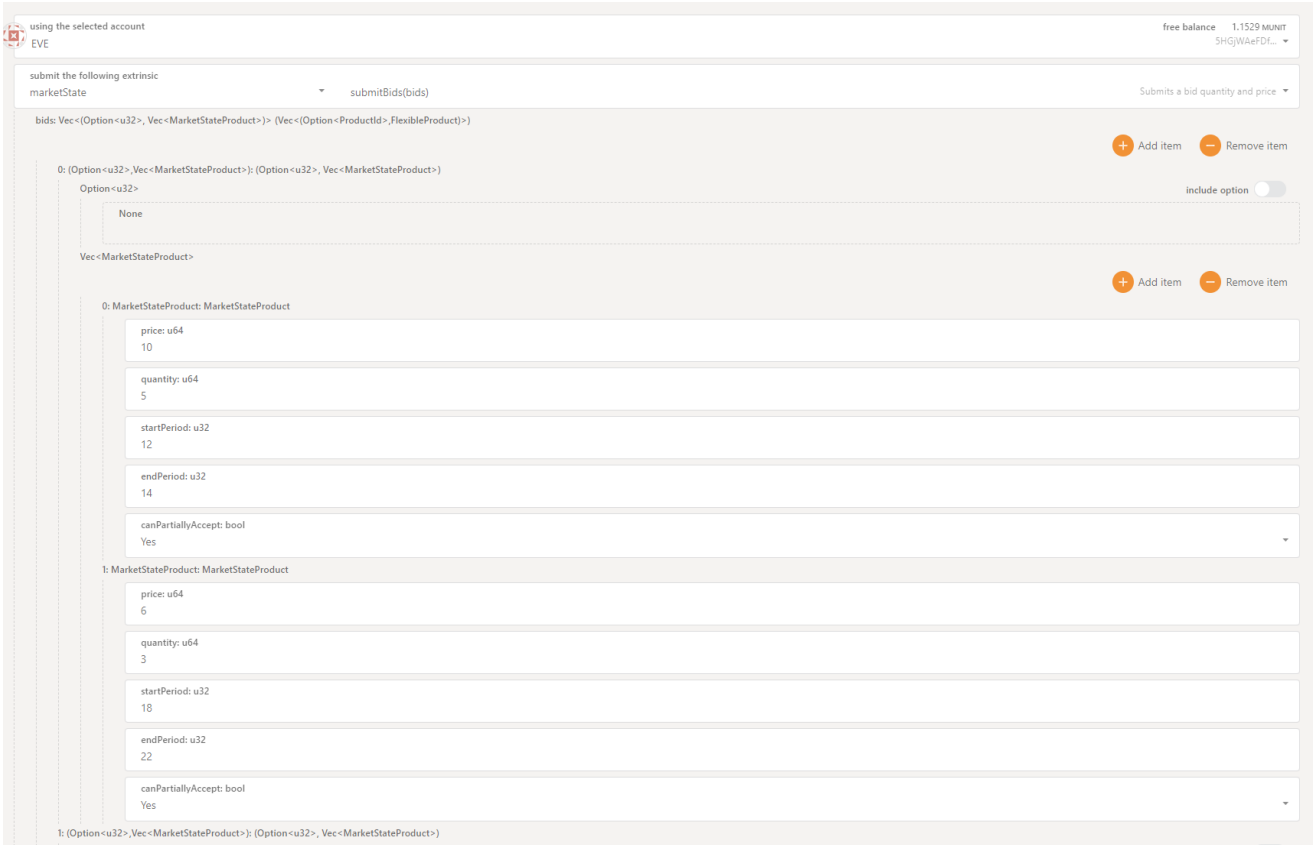

Figure 11 - Polkadot UI

The collator nodes run off-chain workers to match consumers and prosumers via double auction. However, double auction only uses the first schedule in flexible products, and it does not take into account the inter-temporal constraints of continuous products. Therefore, this research also implemented a program[17] to match the market following the MILP optimization described in section

[15] https://github.com/chungthuang/energy-trading-parachain
[16] https://polkadot.js.org/
[17] https://github.com/chungthuang/milp-solver

3.2.3. Given a collator node address, the program can query the market state from the node, solve the optimization problem and submit the solution. Anyone in the community can run this program. It is designed to be set up easily. Market players are incentivized to submit solutions because they want their bids/offers to be accepted. One can submit a solution that favours their bids/offers, but others can counter it with their solution. The one that maximizes social welfare will be the final solution. Further research can design a mechanism to award the submitter of the optimal solution and discourage biased ones.

4.2. MARKET SIMULATION

4.2.1. Single and Continuous Products

First, we compare our market with the DeMarket proposed by Esmat et al. (2021) since our design was inspired by theirs. Their simulation used data collected by Bu et al. (2019) on energy demand in the Midwest U.S. However, the prices were chosen randomly between FiT and the grid price of the Netherlands because there is no regulation on how to price energy products. Table 2 summarizes the simulation data. Agents 1-3 are producers while agents 4-6 are consumers. Agents 3 and 6 each have a continuous product. The rest of the agents only have single products.

Table 2 - Simulation data for 6 agents from Esmat et al. (2021)

Agents	Delivery periods							
	1		2		3		4	
	$\overline{q}^{r,s}_{i,j,k}$ kWh	$\lambda^{r,s}_{i,j,k}$ c€/kWh	$\overline{q}^{r,s}_{i,j,k}$ kWh	$\lambda^{r,s}_{i,j,k}$ c€/kWh	$\overline{q}^{r,s}_{i,j,k}$ kWh	$\lambda^{r,s}_{i,j,k}$ c€/kWh	$\overline{q}^{r,s}_{i,j,k}$ kWh	$\lambda^{r,s}_{i,j,k}$ c€/kWh
agent 1	1.5	10.1	2	4.2	1	7.9	0.5	5.8
	–	–	1.5	6.3	1.5	8.5	0.5	6.8
	–	–	–	–	1	9.3	–	–
agent 2	1.25	12	1.25	12	1.25	12	1.25	12
agent 4	1	7.7	1	8.7	1.25	9.9	2	7.2
	3	6.4	0.5	7.7	–	–	0.5	6.3
	–	–	1.5	5.3	–	–	–	–
agent 5	1.63	13.5	1.63	13.5	1.63	13.5	1.63	13.5
	$\overline{q}^{r,s}_{i,j}$ kWh		$\lambda^{r,s}_{i,j}$ c€/kWh		$\overline{q}^{r,s}_{i,j}$ kWh		$\lambda^{r,s}_{i,j}$ c€/kWh	
agent 3	2		9		–		–	
agent 6	–		–		6		5.98	

Note. From “A novel decentralized platform for peer-to-peer energy trading market with blockchain technology” by Esmat, A., de Vos, M., Ghiassi-Farrokhfal, Y., Palensky, P., & Epema, D, 2021, *Applied Energy*, *282*, p. 12 (https://doi.org/10.1016/J.APENERGY.2020.116123). CC BY 4.0.

The result of DeMarket is presented in table 3. It shows the allocated quantity and the cost/revenue of each agent in every delivery period. The allocated quantity from delivery periods 1 to 4 is 4.49, 2.623, 1.63 and 1.63 kWh respectively. The auction price from delivery period 1 to 4 is 6.35, 9, 12.75 and 12.75 c€/kWh respectively. The social welfare score is around 43.57. However, we believe there is at least 1 mistake in table 3. The allocated quantity for agent 1 in the first delivery period is 3.49 kWh, which is more than its offer of 1.5 kWh in table 2.

When there are no flexible products, our market design is similar to Esmat’s but with two improvements. First, our market allows continuous products to be partially accepted. Second, our auction prices are guaranteed to be lower than the prices of all accepted bids and higher than the prices of all accepted offers.

Our result from the optimizer is presented in table 4. The allocated quantity from delivery period 1 to 4 is 1.63, 3.13, 2.88 and 1.63 kWh. Besides the erroneous period 1, our market was able to match equal or more quantity in each period. The auction price from delivery period 1 to 4 is 13.5, 7.7, 9.9, 13.5 c€/kWh respectively. Our market guarantees buyers will not pay for more than their bid prices and sellers will not receive less than their offer prices. In contrast, Esmat's result will have agent 4 pay 9 c€/kWh in delivery period 2 while the agent's highest bid price is only 8.7 c€/kWh. Our social welfare score is 42.67, slightly lower than Esmat's. However, because of the error identified in table 3, we cannot guarantee their social welfare score is accurate.

Table 3 - Simulation result for 6 agents Esmat et al. (2021)

Sellers						
	agent 1		agent 2		agent 3	
T_x	$q^{r,x}_{i,j,k}$ kWh	$\Delta^{r,x}_{i,j,k}$ c€/kWh	$q^{r,x}_{i,j,k}$ kWh	$\Delta^{r,x}_{i,j,k}$ c€/kWh	$q^{r,x}_{i,j}$ kWh	$\Delta^{r,x}_{i,j}$ c€/kWh
1	3.49	22.16	0	0	1	6.35
2	1.622	14.6	0	0	1	9
3	1.49	19	0.124	1.581	0	0
4	1	12.75	0.625	7.969	0	0
Buyers						
	agent 4		agent 5		agent 6	
T_x	$q^{r,x}_{i,j,k}$ kWh	$\Delta^{r,x}_{i,j,k}$ c€/kWh	$q^{r,x}_{i,j,k}$ kWh	$\Delta^{r,x}_{i,j,k}$ c€/kWh	$q^{r,x}_{i,j}$ kWh	$\Delta^{r,x}_{i,j}$ c€/kWh
1	2.855	18.33	1.63	10.32	0	0
2	1	9	1.63	14.62	0	0
3	0	0	1.63	20.72	0	0
4	0	0	1.63	20.72	0	0

Note. From "A novel decentralized platform for peer-to-peer energy trading market with blockchain technology" by Esmat, A., de Vos, M., Ghiassi-Farrokhfal, Y., Palensky, P., & Epema, D, 2021, *Applied Energy*, *282*, p. 12 (https://doi.org/10.1016/J.APENERGY.2020.116123). CC BY 4.0.

Table 4 - Market simulation result using data from table 2

Sellers						
	agent 1		agent 2		agent 3	
T_x	$q_{o,j}$ kWh	$\Delta_{o,j}$ c€/kWh	$q_{o,j}$ kWh	$\Delta_{o,j}$ c€/kWh	$q_{o,j}$ kWh	$\Delta_{o,j}$ c€/kWh
1	1.5	20.25	0.125	1.6875	0	0
2	3.125	24.0625	0	0	0	0
3	2.88	28.512	0	0	0	0
4	1	13.5	0.625	8.4375	0	0
Buyers						
	agent 4		agent 5		agent 6	
T_x	$q_{o,j}$ kWh	$\Delta_{o,j}$ c€/kWh	$q_{o,j}$ kWh	$\Delta_{o,j}$ c€/kWh	$q_{o,j}$ kWh	$\Delta_{o,j}$ c€/kWh
1	0	0	1.63	22.005	0	0
2	1.5	11.55	1.63	12.551	0	0
3	1.25	12.375	1.63	16.137	0	0

4	0	0	1.63	22.005	0	0

4.2.2. Flexible Products

Next, we simulate the market with flexible products. The price and quantity were chosen arbitrarily because we did not find other papers with similar experiments. Table 5 shows the simulation data and results. Product 1 and product 4 are flexible products with 2 potential schedules. First, we simulate when all products have to be fully accepted. Then we test with the same dataset, but this time all products can be partially accepted.

Table 5 - Flexible product simulation

Account	Product ID	Type	Price c€/kWh	Quantity kWh	Start Period	End Period	Fully Accepted	Social Welfare	Partially Accepted	Social Welfare
Alice	Product 1	Bid	20	5	0	2	Yes	200	100%	200
			16	6	3	5	No	0	0%	0
	Product 2	Offer	14	6	1	2	Yes	-84	100%	-84
Bob	Product 3	Bid	18	5	2	4	Yes	180	100%	180
Charlie	Product 4	Offer	12	6	0	4	No	0	83%	-239.04
			14	5	0	4	Yes	-280	0%	0
Daniela	Product 5	Bid	17	4	1	2	Yes	68	100%	68
Ella	Product 6	Bid	15	2	1	2	Yes	30	100%	30
	Product 7	Offer	11	3	3	5	No	0	0%	0

The result is shown in the last 4 columns of table 5 and table 6. When all products must be fully accepted, the first schedule of product 1 and the second schedule of product 4 are accepted. The total utility is the sum of $price \times quantity \times (end\ period - start\ period)$ from product 1, 3, 5 and 6 which is 478. The total cost is the same formula but from products 2 and 4 which is 364. The overall social welfare is the total utility minus total cost $478 - 364 = 114$. When all products can be partially accepted, the first schedule of product 4 was accepted instead of the second one, because its overall cost is lower. The social welfare score increased to 154. The auction price for each period is determined from the lowest bid price. In period 1, the lowest bid price is from product 6. Both runs resulted in the same auction prices.

Table 6 - Flexible products market result

Period	0	1	2	3	4	5
Auction price c€/kWh	20	15	18	18	0	0
Accepted bids	Product 1	Product 1, 5, 6	Product 3	Product 3	-	-

Bid quantity Kwh	5	5+4+2=11	5	5	0	0
Accepted offers	Product 4	Product 2, 4	Product 4	Product 4	-	-
Offer quantity Kwh	5	5+6=11	5	5	0	0

4.2.3. Comparison with Double Auction

Table 7 shows the partial result of a double auction. Because the algorithm always starts to match the bid with the highest price and the offer with the lowest price, in period 3 it matches Ella's offer (product 7) with Bob's bid (product 3) first. After Ella's offer, Bob's still has 2 KWh of outstanding demand. The algorithm proceeds to match it with Charlie's offer. The result is that 83% of Charlie's offer is matched in periods 0, 1 and 2, but only 33% is matched in period 3 which violates the inter-temporal constraint that continuous product has a single acceptance percentage across periods.

Table 7 - Double auction partial result

Period	0	1	2	3
Auction price c€/kWh	20	15	18	18
Accepted bids	Product 1	Product 1, 5, 6	Product 3	Product 3
Bid quantity Kwh	5	5+4+2=11	5	5
Accepted offers	Product 4 (83%) Schedule 1	Product 2, 4 (83%) Schedule 1	Product 4 (83%) Schedule 1	Product 4(33%) schedule 1, Product 7
Offer quantity Kwh	6(83%)=5	6(83%)+6=11	5	6(33%)+3=5

We did not simulate double auction with data from table 2 because Substrate does not support floating point arithmetic. On the contrary, our optimizer is a standalone program that handles floating-point inputs. It converts the solution to integers before submitting it to the parachain.

4.3. PERFORMANCE BENCHMARKS

This research wants to answer two questions with benchmarking. First is the appropriate weight for an extrinsic. This determines how many transactions a node can process within the block production time. The second is how long it takes to clear the market. This helps us understand the complexity of the market problem and the performance of the double auction.

4.3.1. Extrinsic Weights

The weight of an extrinsic estimates the computation time. It is determined by 3 main factors, the hardware, size of input, type of database and the number of times it reads and writes to the database.

- Hardware: The result presented in this research was measured on a laptop with 11th Gen Intel (R) Core (TM) i7-1195G7 @ 2.90GHz processor and 16 GB of RAM, repeated 10 times.

- Input size: Our extrinsic accepts a list of products as input, each product incurs some computation and database read/write to process it. Hence, the weight is proportional to the size of the input.
- Database: By default, Substrate uses RocksDB, so in this section, we present results assuming this database. Substrate assumes each read takes a constant time of 25 µs and write takes 100 µs, independent of the hardware and current storage size. However, theoretically, the time for reading and writing from the underlying Merkle Tree data structure is $O(log\ N)$, where N is the number of entries in the trie. The benchmarking result should take this into account.

The result of benchmarking is a linear model for each extrinsic. The unit is in picoseconds (10^{-12} seconds). Let r denote the constant time to read from the database and w denote the constant time per database write. Equations (10) and (11) are the weight of extrinsic to submit bids and offers respectively. They have the following variables and coefficients:

- c: input size, the number of bids/offers submitted by a user.
- b: number of bids already in the database.
- o: number of offers already in the database.
- B_0: base computation time.
- B_1, B_2 and B_3: coefficient for c, b and o.

$$sumbit\_bids = B_0 + B_1 c + B_2 b + B_3 o + 3r + (2 + c)w \tag{10}$$

$$sumbit\_offers = B_0 + B_1 c + B_2 b + B_3 o + 3r + (2 + c)w \tag{11}$$

In section 3.3.2, we showed that submit a bid/offer extrinsic requires 3 reads and 2 + c writes to the database. The result from benchmarking agrees with that assessment. The estimated coefficients tested with different input ranges are presented in table 8. Because each transaction is called by an account, the upper bound of c is determined by the maximum product per market player configuration of the pallet. The market supports multiple accounts, so the upper bound of b and o are determined by another pallet configuration for the maximum total products. The table shows that the base time (B_0) and process time per product (B_1) increase with the input range. On the other hand, B_2 and B_3 are negligible compared to B_1. The result does not support the theory that it takes longer to traverse the underlying Merkle Tree data structure as the number of entries grows. The logic between the two extrinsic are almost identical except they have different validation rules and use different storage objects. We expect the results to converge if we repeat the benchmark enough times.

Table 8 - submit_bids and submit_offers coefficients

Input Range	submit_bids				submit_offers			
	B_0	B_1	B_2	B_3	B_0	B_1	B_2	B_3
$c = [1, 100]$	74,036,000	2,455,504	18,770	0	54,637,964	1,959,849	20,563	22,661

$b = o = [1, 1{,}000]$								
$c = [1, 1{,}000]$ $b = o = [1, 10{,}000]$	46,350,846	2,669,905	0	9,055	110,376,000	2,310,420	1,147	1,887
$c = [1, 10{,}000]$ $b = o = [1, 100{,}000]$	187,883,219	3,823,100	5,363	0	1,366,789,145	3,251,415	0	5,097

Equation (10) and (11) can also be used to estimate transaction throughput. We consider 3 measures, weight per transaction (W/T), transaction per second (TPS) and transaction per block (TPB). 10^{12} unit of weight is approximately 1 second, so $TPS = 10^{12}/weight\ per\ transaction$. According to the (*Polkadot Protocol Specification v0.2.1*, 2022), the block production time is 6 seconds, so $TPB = 6 \times TPS$. The result of the calculated throughput assuming maximum input in each transaction is presented in table 9. Note that in this calculation we only consider the computation time, but not the amount of data. The protocol specification limits the block size to 5 MB. Figure 12 provides a visual comparison of equation (10) and (11) under different input ranges. Because B_2 and B_3 are negligible compared to B_1, the plot assumes $b = o = 10{,}000$ and only varies c.

Table 9 - submit_bids and submit_offers throughput

Input Range	Submit_bids			Submit_offers		
	W/T	TPS	TPB	W/T	TPS	TPB
$c = 100$ $b = o = 1{,}000$	348,631,400	2868	17210	585,831,219	1706	10241
$c = 1{,}000$ $b = o = 10{,}000$	2,907,080,846	344	2064	2,551,411,000	392	2351
$c = 10{,}000$ $b = o = 100{,}000$	39,955,458,219	25	150	35,390,914,145	28	170

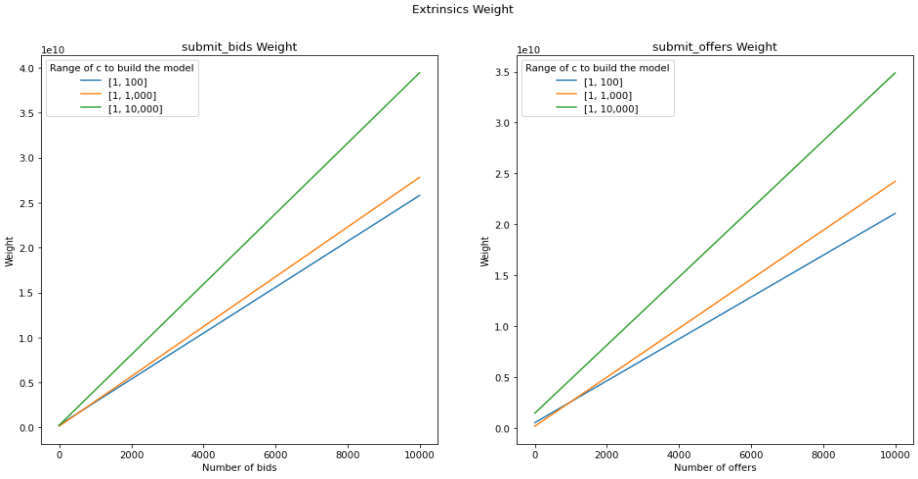

Figure 12 - submit_bids & submit_offers weight

Equation (12) is the weight of the extrinsic to submit a solution. When writing the benchmark, we follow the principle of testing the most computationally expensive case. Thus, the benchmark assumes all submitted bids and offers are accepted. The equation has the following variables and coefficients:

- b: input size, the number of accepted bids.
- o: input size, the number of accepted offers.
- B_0: base computation time.
- B_1 and B_2: coefficient for b and o.

$$submit\_solution = B_0 + B_1 b + B_2 o + (2 + b + o)r + 3w \tag{12}$$

Each accepted bid and offer requires a read from the database. In addition, each transaction needs to get the market stage and current solution, so 2 extra reads from the database. Finally, the transaction stores the auction price, accepted bids, and accepted offers, so 3 writes to the database. The estimated coefficients tested with different input ranges are presented in table 10. Similar to the findings from table 8, the base time and coefficients increase as the input range grows. The throughput is summarized in table 11. Figure 13 provides a visualization for equation (12) with input range [1, 100,000]. The models from other input ranges have similar shapes, but different magnitudes on the axes.

Table 10 - submit_solution coefficients

Input Range	B_0	B_1	B_2
[1, 1,000]	981,861,134	3,497,570	3,281,744
[1, 10,000]	15,574,078,470	2,540,763	2,606,298
[1, 100,000]	314,042,121,879	5,178,462	4,536,442

Table 11 - submit_solution throughput

Input Range	W/T	TPS	TPB
$b = o = 1{,}000$	7,811,575,134	128.01	768.9
$b = o = 10{,}000$	67,545,088,470	14.80	88.83
$b = o = 100{,}000$	1,290,532,921,879	0.77	4.65

In conclusion, the benchmarking tool from Substrate provides an easy way to estimate the weight of each extrinsic. The result depends on the hardware and pallet configuration. Hence, every node should set the maximum product per market player and maximum total product according to its market and run the benchmark.

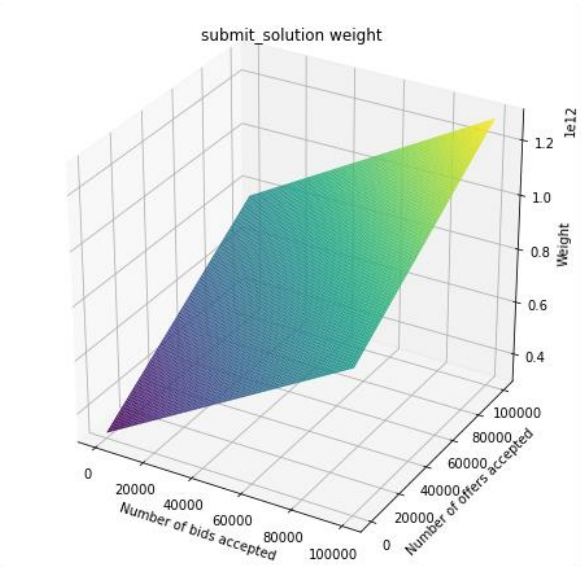

Figure 13 - submit_solution weight

4.3.2. Market Clearing Speed

The previous section focused on the performance of transactions on the blockchain. This section will focus on the performance and complexity of the optimizer and double auction. The simulation in section 4.2.1 had 26 products in total, 12 single bids, 1 continuous bid, 12 single offers and 1 continuous offer. It had 4 clearing periods. The optimizer solved it with 56 variables and 84 constraints. The variables are listed in table 12 and the constraints are listed in table 13. The continuous bid adds constraint 2 twice and the continuous offer adds constraint 3 twice. The benchmarking command from Rust ran the simulation multiple times and found the execution time was 17.58 ± 2.48 ms.

Table 12 - Variables in simulation 4.2.1

Variable	Purpose	Equation	Count
$i_{b,j}/i_{o,j}$	If a bid/offer is accepted	2/3	26
$c_{b,j}/c_{o,j}$	Percentage of a bid/offer that is accepted	6	26
v_t	Auction price for each period	6	4

Table 13 - Constraints in simulation 4.2.1

	Purpose	Equation	Count
1	Bid price > auction price when the bid is accepted	2	13+1=14
2	Offer price < auction price when the offer is accepted	3	13+1=14
3	Bid quantity = offer quantity	4	4
4	The product is accepted if the percentage accepted > 0	7	26
5	The product is not accepted if the percentage accepted = 0	8	26

Next, the experiment in section 4.2.2 had 6 periods and 7 products, 2 of them are flexible products. It was solved with 24 variables and 63 constraints. The variables and constraints are listed in table 14 and table 15. The flexible products contribute 2 $i_{b,j}/i_{o,j}$ and $c_{b,j}/c_{o,j}$. The execution time was 5.85 ± 1.31 ms.

Table 14 - Variables in simulation 4.2.2

Variable	Purpose	Equation	Count
$i_{b,j}/i_{o,j}$	If a bid/offer is accepted	2	7+2=9
$c_{b,j}/c_{o,j}$	Percentage of a bid/offer that is accepted	6	7+2=9
v_t	Auction price for each period	6	6

Finally, we present the performance of the double auction from section 4.2.3. We only consider the time it takes to read the products submitted and match them. We do not consider the time to submit the result for validation. To get the list of products, it had to read them one by one from the database. That is 7 reads in total. The matching is done by a for loop for each period. The execution time was 0.39 ± 0.18 ms. Compared to the optimizer, the double auction runs an order of magnitude faster.

Table 15 - Constraints in simulation 4.2.2

	Purpose	Equation	Count
1	Bid price > auction price when the bid is accepted	2	8
2	Offer price < auction price when the offer is accepted	3	11
3	Bid quantity = offer quantity	4	6
4	The product is accepted if the percentage accepted > 0	7	18
5	The product is not accepted if the percentage accepted = 0	8	18
6	Only one of the flexible bids/offers can be accepted	5	2

To summarize, both market clearing mechanisms by MILP optimization and double auction solve a small dataset within 20 ms. We expect they can scale to serve a community with 100s of products.

5. CONCLUSION AND FUTURE WORK

P2P energy trading is important for our energy future because it has the potential to incentivize more investments in DERs, reduce the cost of energy and improve grid reliability. Therefore, the first goal of this research was to design a more realistic P2P market that improves the payoff of prosumers, reduces the cost of consumers, and reduces peak demand. To achieve that, we first need to have models that represent real energy supply and demand. While reviewing recent literature, we noticed most markets are designed for simple products. They often assume a product will run for a single period at the same price and quantity and use double auctions to match supply with demand. However, household appliances, generators and storage systems often run for multiple hours. Thus, we adopt the concept of continuous product. Then to encourage consumers to shift energy usage from peak hours, we added the concept of flexible products. Consumers can propose a list of continuous products that represent their flexibility to shift usage to different periods at what price and quantity. Then the market will determine the optimal one. At most 1 continuous product in a flexible product will be accepted. Flexible products are also used to incentivize prosumers to supply when demand surges. For example, they might have batteries that can discharge in different periods.

To match these complex supply and demand that have inter-temporal constraints, the market must clear multiple periods together. It needs to decide which products will be accepted to maximize overall social welfare while ensuring the supply quantity matches the demand quantity for all periods. This renders naturally as a MILP problem where each product has a binary variable that decides if it is accepted, another variable to determine the percentage accepted, and linear constraints to ensure supply and demand quantities are matched. To summarize, this research designed a P2P market that supports products with continuous and flexible schedules. The market problem is MILP optimization on overall social welfare.

The second goal of this research is to address the practicality of the market. It must be trusted, and run at low cost while still highly reliable. Blockchain technology is often used to solve the trust issue in a decentralized system. In addition, blockchain provides traceability of records, tamper-proof of data and anonymity of users, all are desirable characteristics for a market. The research community has focused on deploying markets as smart contracts on Ethereum or permission blockchains like Hyperledger. Ethereum is very secure due to its network size. However, it has high transaction costs and low throughput. Although the Ethereum community has a roadmap to improve on these aspects, it can take years to achieve. On the other hand, permission blockchains are more scalable and we get to control the transaction price. But the tradeoff is it becomes easier to compromise.

To address the shortcomings in existing blockchain solutions, this research takes a novel approach to adopting the Polkadot blockchain. It is a public blockchain of blockchains. Application-specific blockchains, called parachains, are connected by a relay chain. Security and scalability are achieved by having the parachains only submit their aggregated state transitions to the relay chain for validation. Our operation cost will mostly be a fixed cost to acquire a parachain slot on the network. Using the Kusama canary but still production-grade network, the estimated auction price is around a few thousand per year. The network does not charge parachains for transactions, so parachain developers can decide what to charge their end users. By adopting Polkadot, we can build a secured application-specific blockchain with costs lower than Ethereum.

This research built a parachain specifically for the P2P market. It has an open stage and a clearing stage. In the open stage, it provides consumers with a transaction to submit bids and prosumers with a transaction to submit offers. In the clearing stage, it will try to match the supply and demand using a double auction. This is just a baseline solution. Market players can also submit their solutions. They are validated by making sure all accepted bids are higher than the auction price, all accepted offers are lower than the auction price, and supply quantity and demand quantity match for all periods.

The MILP optimization is not solved on the parachain because it is a complex computation that does not have a predictable execution time. Instead, this research implemented a separate program that can fetch market data from the parachain, formulate it as a MILP problem and use an open-source optimizer to solve it. Market players can use this program to propose their solutions. At the end of the clearing period, the market will pick the solution with the highest overall social welfare.

To conclude, this research proposed a model that captures the inter-temporal constraints and flexible schedules of energy products. Based on that, the research designed a P2P market as a MILP problem to optimize social welfare. Then, the research implemented the P2P market as a system of Polkadot parachain and off-chain optimizer. Finally, benchmarking showed that the parachain and optimizer have good performance.

5.1. LIMITATIONS

This research focused on the economics of a market but not the physical infrastructure. We assumed the grid could transfer all the matched supply and demand, but in reality, the power flow is a delicate balance. Too much energy flowing into the grid can cause network congestion. To address the network layer, we would need to have real-time data on the energy flow and characteristics of the grid, such as its capacity and voltage limits.

Another physical aspect that was not considered is the actual delivery of energy. First, the market only knows the public key of the participants. It does not know their actual identities to decide how to deliver. Second, the market assumed that once it is cleared, participants will act accordingly. However, some prosumers might not be able to fulfill their commitments because renewable energy sources are not always predictable. Because we assume the microgrid is still connected to the main grid, surplus or shortage of supply can be balanced by the main grid. Still, the market can have a mechanism to discourage misbehaviours.

Finally, the system is only simulated in a local network. We will discover areas to improve if we test with real users. For example, we might find out that users still cannot express their bids or offers with the proposed models. On the other hand, we might find out the complexity of the models hurts user experience. Another possible finding is that not enough users are willing to run a node to form a parachain. To address this, the market will need to provide better incentives for node operators.

5.2. FUTURE WORK

The first area that can be improved is the market design. This research assumed a product can either consume or produce energy. But recently batteries have been gaining traction because they can do both. Section 2.1.1 provided a brief overview of modelling batteries. One future research direction is to incorporate batteries into our model. Another direction is to consider the physical layer. The market can have additional objectives and constraints to minimize power surges. Furthermore, the

market should also have a mechanism that holds market players accountable. For each bid or offer submitted, the market can ask for a deposit. This deposit is released when the bid or offer is fulfilled, confiscated otherwise. Besides penalties, the market should also provide more incentives for solution providers and node operators. By rewarding the submitter with the optimal solution, we create an environment that encourages more market players to develop better algorithms. By rewarding node operators, enough market players will be willing to run nodes to form a parachain.

Another area of research is testing the system on real networks and users. We can start by deploying the parachain to the Rococo test network and invite real users to try out the market. Most users will probably not know how to quantify their supply or demand, so tools will have to be developed to help them. In addition, a website or mobile app will need to be developed to help users interact with the market. Finally, after enough testing on the Rococo network is done, we can auction for a slot on the Kusama canary network.

BIBLIOGRAPHICAL REFERENCES

Abdella, J., Tari, Z., Anwar, A., Mahmood, A., & Han, F. (2021). An Architecture and Performance Evaluation of Blockchain-Based Peer-to-Peer Energy Trading. *IEEE Transactions on Smart Grid*, *12*(4), 3364–3378. https://doi.org/10.1109/TSG.2021.3056147

Ableitner, L., Meeuw, A., Schopfer, S., Tiefenbeck, V., Wortmann, F., & Wörner, A. (2019). *Quartierstrom -- Implementation of a real world prosumer centric local energy market in Walenstadt, Switzerland*. https://doi.org/10.48550/arxiv.1905.07242

Ableitner, L., Tiefenbeck, V., Meeuw, A., Wörner, A., Fleisch, E., & Wortmann, F. (2020). User behavior in a real-world peer-to-peer electricity market. *Applied Energy*, *270*, 115061. https://doi.org/10.1016/J.APENERGY.2020.115061

Adams, H., Zinsmeister, N., Salem moody, M., River Keefer, uniswaporg, & Robinson, D. (2021). *Uniswap v3 Core*.

Alper, H. K. (n.d.). *BABE | Research at Web3 Foundation*. Retrieved May 22, 2023, from https://research.web3.foundation/Polkadot/protocols/block-production/Babe

Antunes, C. H., Alves, M. J., & Soares, I. (2022). A comprehensive and modular set of appliance operation MILP models for demand response optimization. *Applied Energy*, *320*, 119142. https://doi.org/10.1016/J.APENERGY.2022.119142

Architecture · Polkadot Wiki. (n.d.). Retrieved January 29, 2023, from https://wiki.polkadot.network/docs/learn-architecture

Bazaraa, M. S., Jarvis, J. J., & Sherali, H. D. (2011). Linear programming and network flows: Fourth edition. *Linear Programming and Network Flows: Fourth Edition*, 1–748. https://doi.org/10.1002/9780471703778

Belchior, R., Vasconcelos, A., Guerreiro, S., & Correia, M. (2020). A Survey on Blockchain Interoperability: Past, Present, and Future Trends. *ACM Computing Surveys*, *54*(8). https://doi.org/10.48550/arxiv.2005.14282

Brenzikofer, A., Meeuw, A., Schopfer, S., Wörner, A., & Dürr, C. (2019, June 2). Quartierstrom: A Decentralized Local P2P Energy Market Pilot On A Self-Governed Blockchain. *CIRED 2019 Conference*. https://www.cired-repository.org/handle/20.500.12455/181

Bu, F., Yuan, Y., Wang, Z., Dehghanpour, K., & Kimber, A. (2019). A Time-Series Distribution Test System Based on Real Utility Data. *51st North American Power Symposium, NAPS 2019*. https://doi.org/10.1109/NAPS46351.2019.8999982

Burdges, J., Cevallos, A., Czaban, P., Habermeier, R., Hosseini, S., Lama, F., Alper, H. K., Luo, X., Shirazi, F., Stewart, A., & Wood, G. (2020). *Overview of Polkadot and its Design Considerations*. https://doi.org/10.48550/arxiv.2005.13456

Buterin, V. (2014). *Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform.*

Clausen, J. (2003). *Branch and Bound Algorithms-Principles and Examples*.

Croman, K., Decker, C., Eyal, I., Gencer, A. E., Juels, A., Kosba, A., Miller, A., Saxena, P., Shi, E., Sirer, E. G., Song, D., & Wattenhofer, R. (2016). On scaling decentralized blockchains (A position paper). *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9604 LNCS*, 106–125. https://doi.org/10.1007/978-3-662-53357-4_8/TABLES/5

Danksharding | ethereum.org. (n.d.). Retrieved June 9, 2023, from https://ethereum.org/en/roadmap/danksharding/

Diffie, W., Diffie, W., & Hellman, M. E. (1976). New Directions in Cryptography. *IEEE Transactions on Information Theory*, *22*(6), 644–654. https://doi.org/10.1109/TIT.1976.1055638

Doan, H. T., Cho, J., & Kim, D. (2021). Peer-to-peer energy trading in smart grid through blockchain: A double auction-based game theoretic approach. *IEEE Access*, *9*, 49206–49218. https://doi.org/10.1109/ACCESS.2021.3068730

Esmat, A., de Vos, M., Ghiassi-Farrokhfal, Y., Palensky, P., & Epema, D. (2021). A novel decentralized platform for peer-to-peer energy trading market with blockchain technology. *Applied Energy*, *282*, 116123. https://doi.org/10.1016/J.APENERGY.2020.116123

Espe, E., Potdar, V., & Chang, E. (2018). Prosumer Communities and Relationships in Smart Grids: A Literature Review, Evolution and Future Directions. *Energies 2018, Vol. 11, Page 2528*, *11*(10), 2528. https://doi.org/10.3390/EN11102528

Ethereum Live TPS. (n.d.). Retrieved July 8, 2023, from https://ethtps.info/Network/Ethereum

Ethereum roadmap | ethereum.org. (n.d.). Retrieved July 8, 2023, from https://ethereum.org/en/roadmap/

Forrest, J., Ralphs, T., Santos, H. G., Vigerske, S., Forrest, J., Hafer, L., Kristjansson, B., jpfasano, EdwinStraver, Lubin, M., Jan-Willem, rlougee, jpgoncal1, Brito, S., h-i-gassmann, Cristina, Saltzman, M., tosttost, Pitrus, B., … to-st. (2023). *coin-or/Cbc* (2.10.10). https://doi.org/10.5281/ZENODO.7843975

Goldwasser, S., Micali, S., & Rackoff, C. (1985). KNOWLEDGE COMPLEXITY OF INTERACTIVE PROOF-SYSTEMS. *Conference Proceedings of the Annual ACM Symposium on Theory of Computing*, 291–304. https://doi.org/10.1145/22145.22178

Goranovic, A., Meisel, M., Fotiadis, L., Wilker, S., Treytl, A., & Sauter, T. (2017). Blockchain applications in microgrids: An overview of current projects and concepts. *Proceedings IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society*, *2017-January*, 6153–6158. https://doi.org/10.1109/IECON.2017.8217069

Habermeier, R. (2023). *Polkadot Roadmap Roundup*. https://polkadot.network/blog/polkadot-roadmap-roundup

Han, D., Zhang, C., Ping, J., & Yan, Z. (2020). Smart contract architecture for decentralized energy trading and management based on blockchains. *Energy*, *199*, 117417. https://doi.org/10.1016/J.ENERGY.2020.117417

Hash Time Locked Contracts - Bitcoin Wiki. (n.d.). Retrieved June 9, 2023, from https://en.bitcoin.it/wiki/Hash_Time_Locked_Contracts

Introduction — WebAssembly 2.0. (2023). https://webassembly.github.io/spec/core/intro/introduction.html#design-goals

IRENA. (2019). Global Energy Transformation: A Roadmap to 2050. In *International Renewable Energy Agency*. https://www.irena.org/publications/2019/Apr/Global-energy-transformation-A-roadmap-to-2050-2019Edition

Kapengut, E., & Mizrach, B. (2023). An Event Study of the Ethereum Transition to Proof-of-Stake. *Commodities 2023, Vol. 2, Pages 96-110*, *2*(2), 96–110. https://doi.org/10.3390/COMMODITIES2020006

Khorasany, M., Mishra, Y., & Ledwich, G. (2018). Market framework for local energy trading: a review of potential designs and market clearing approaches. *IET Generation, Transmission & Distribution*, *12*(22), 5899–5908. https://doi.org/10.1049/IET-GTD.2018.5309

Kirli, D., Couraud, B., Robu, V., Salgado-Bravo, M., Norbu, S., Andoni, M., Antonopoulos, I., Negrete-Pincetic, M., Flynn, D., & Kiprakis, A. (2022). Smart contracts in energy systems: A systematic review of fundamental approaches and implementations. *Renewable and Sustainable Energy Reviews*, *158*. https://doi.org/10.1016/J.RSER.2021.112013

Li, N., Uckun, C., Constantinescu, E. M., Birge, J. R., Hedman, K. W., & Botterud, A. (2016). Flexible Operation of Batteries in Power System Scheduling with Renewable Energy. *IEEE Transactions on Sustainable Energy*, *7*(2), 685–696. https://doi.org/10.1109/TSTE.2015.2497470

Lüth, A., Zepter, J. M., Crespo del Granado, P., & Egging, R. (2018). Local electricity market designs for peer-to-peer trading: The role of battery flexibility. *Applied Energy*, *229*, 1233–1243. https://doi.org/10.1016/J.APENERGY.2018.08.004

Manupati, V. K., Schoenherr, T., Ramkumar, M., Wagner, S. M., Pabba, S. K., & Inder Raj Singh, R. (2020). A blockchain-based approach for a multi-echelon sustainable supply chain. *International Journal of Production Research*, *58*(7), 2222–2241. https://doi.org/10.1080/00207543.2019.1683248

Mengelkamp, E., Gärttner, J., Rock, K., Kessler, S., Orsini, L., & Weinhardt, C. (2018). Designing microgrid energy markets: A case study: The Brooklyn Microgrid. *Applied Energy*, *210*, 870–880. https://doi.org/10.1016/J.APENERGY.2017.06.054

Mezquita, Y., Gazafroudi, A. S., Corchado, J. M., Shafie-Khah, M., Laaksonen, H., & Kamisalic, A. (2019). Multi-agent architecture for peer-to-peer electricity trading based on blockchain technology. *ICAT 2019 - 27th International Conference on Information, Communication and Automation Technologies, Proceedings*. https://doi.org/10.1109/ICAT47117.2019.8938926

Morstyn, T., Farrell, N., Darby, S. J., & McCulloch, M. D. (2018). Using peer-to-peer energy-trading platforms to incentivize prosumers to form federated power plants. *Nature Energy 2018 3:2*, *3*(2), 94–101. https://doi.org/10.1038/s41560-017-0075-y

Nakamoto, S. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. https://www.researchgate.net/publication/228640975_Bitcoin_A_Peer-to-Peer_Electronic_Cash_System

Nash, J. C. (2000). The (Dantzig) Simplex Method for Linear Programming. *Computing in Science & Engineering*, *2*(01), 29–31. https://doi.org/10.1109/5992.814654

Noor, S., Yang, W., Guo, M., van Dam, K. H., & Wang, X. (2018). Energy Demand Side Management within micro-grid networks enhanced by blockchain. *Applied Energy*, *228*, 1385–1398. https://doi.org/10.1016/J.APENERGY.2018.07.012

Parachain Slot Auctions · Polkadot Wiki. (n.d.). Retrieved May 20, 2023, from https://wiki.polkadot.network/docs/learn-auction

Parathreads · Polkadot Wiki. (n.d.). Retrieved May 20, 2023, from https://wiki.polkadot.network/docs/learn-parathreads

Petrowski, J. (2020). *The Path of a Parachain Block*. https://polkadot.network/blog/the-path-of-a-parachain-block

Polkadot Protocol Specification v0.2.1. (2022). https://spec.polkadot.network/id-weights#defn-polkadot-block-limits

Ren, Z., Verbič, G., & Guerrero, J. (2022). Multi-period dynamic tariffs for prosumers participating in virtual power plants. *Electric Power Systems Research*, *212*, 108478. https://doi.org/10.1016/J.EPSR.2022.108478

Sabounchi, M., & Wei, J. (2017). Towards resilient networked microgrids: Blockchain-enabled peer-to-peer electricity trading mechanism. *2017 IEEE Conference on Energy Internet and Energy System Integration, EI2 2017 - Proceedings*, *2018-January*, 1–5. https://doi.org/10.1109/EI2.2017.8245449

Scott, I., de Castro Neto, M., & Pinheiro, F. L. (2021). Bringing Trust and Transparency to the Opaque World of Waste Management with Blockchain: A Polkadot Parathread Application. *SSRN Electronic Journal*. https://doi.org/10.2139/SSRN.3825072

Siano, P., de Marco, G., Rolan, A., & Loia, V. (2019). A Survey and Evaluation of the Potentials of Distributed Ledger Technology for Peer-to-Peer Transactive Energy Exchanges in Local Energy Markets. *IEEE Systems Journal*, *13*(3), 3454–3466. https://doi.org/10.1109/JSYST.2019.2903172

Soto, E. A., Bosman, L. B., Wollega, E., & Leon-Salas, W. D. (2021). Peer-to-peer energy trading: A review of the literature. *Applied Energy*, *283*, 116268. https://doi.org/10.1016/J.APENERGY.2020.116268

Stewart, A., & Kokoris-Kogia, E. (2020). *GRANDPA: a Byzantine Finality Gadget*. https://arxiv.org/abs/2007.01560v1

Thomas, S., & Schwartz, E. (2015). *A Protocol for Interledger Payments*. https://interledger.org/interledger.pdf

Tushar, W., Saha, T. K., Yuen, C., Smith, D., & Poor, H. V. (2020). Peer-to-Peer Trading in Electricity Networks: An Overview. *IEEE Transactions on Smart Grid*, *11*(4), 3185–3200. https://doi.org/10.1109/TSG.2020.2969657

Tushar, W., Yuen, C., Mohsenian-Rad, H., Saha, T., Poor, H. V., & Wood, K. L. (2018). Transforming energy networks via peer-to-peer energy trading: The potential of game-theoretic approaches. *IEEE Signal Processing Magazine*, *35*(4), 90–111. https://doi.org/10.1109/MSP.2018.2818327

Tushar, W., Yuen, C., Saha, T. K., Morstyn, T., Chapman, A. C., Alam, M. J. E., Hanif, S., & Poor, H. V. (2021). Peer-to-peer energy systems for connected communities: A review of recent advances and emerging challenges. *Applied Energy*, *282*, 116131. https://doi.org/10.1016/J.APENERGY.2020.116131

Woebbeking, F. (2021). Cryptocurrency volatility markets. *Digital Finance 2021 3:3*, *3*(3), 273–298. https://doi.org/10.1007/S42521-021-00037-3

Wood, G. (2014). *ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER*.

Wood, G. (2016). *POLKADOT: VISION FOR A HETEROGENEOUS MULTI-CHAIN FRAMEWORK*.

Wu, J., & Tran, N. K. (2018). Application of Blockchain Technology in Sustainable Energy Systems: An Overview. *Sustainability 2018, Vol. 10, Page 3067*, *10*(9), 3067. https://doi.org/10.3390/SU10093067

Yaga, D., Mell, P., Roby, N., & Scarfone, K. (2019). *Blockchain Technology Overview*. https://doi.org/10.6028/NIST.IR.8202

Zero-knowledge proofs | ethereum.org. (n.d.). Retrieved January 28, 2023, from https://ethereum.org/en/zero-knowledge-proofs/

Zhao, J., Zheng, T., & Litvinov, E. (2020). A Multi-Period Market Design for Markets with Intertemporal Constraints. *IEEE Transactions on Power Systems*, *35*(4), 3015–3025. https://doi.org/10.1109/TPWRS.2019.2963022

Zheng, Z., Xie, S., Dai, H., Chen, X., & Wang, H. (2017). An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends. *Proceedings - 2017 IEEE 6th International Congress on Big Data, BigData Congress 2017*, 557–564. https://doi.org/10.1109/BIGDATACONGRESS.2017.85

Zhou, Q., Huang, H., Zheng, Z., & Bian, J. (2020). Solutions to Scalability of Blockchain: a Survey. *IEEE Access*, *8*, 16440–16455. https://doi.org/10.1109/ACCESS.2020.2967218

Zia, M. F., Benbouzid, M., Elbouchikhi, E., Muyeen, S. M., Techato, K., & Guerrero, J. M. (2020). Microgrid transactive energy: Review, architectures, distributed ledger technologies, and market analysis. *IEEE Access*, *8*, 19410–19432. https://doi.org/10.1109/ACCESS.2020.2968402