UiT THE ARCTIC UNIVERSITY OF NORWAY

Faculty of Engineering Science and Technology

Department of Computer Science and Computational Engineering

Exploring auction based energy trade with the support of MAS and blockchain technology

Nikita Shvetsov

Thesis for Master of Science in Computer Science - June 2017

Master thesis: Exploring auction based energy trade with the support of MAS and blockchain technology

Nikita Shvetsov

June 6, 2017

Acknowledgment

I want to thank my supervisor Bernt A. Bremdal for valuable insights and experience in this field and also encouragement and opportunity to work on such interesting research area. I also want to thank my family and friends for supporting me throughout this Master's degree and Master thesis.

Abstract

This document describes a simulation of the local energy market with support of multi-agent approach and blockchain technology. The investigated points include blockchain technology and its applications, Ethereum platform and smart contracts as a tool for storing data of operations and creating assets, multi-agent approach to model the local energy market. The document explores building a solution for proposed problem with blockchain technology, agent interactions on the simulated market and auction models, that provide sustainability and profit for the local energy market overall.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This master thesis is a research project about local energy markets, blockchain technology and machine learning applications. The thesis investigates created simulation of local energy market and its data flows in time. Data includes production/consumption rates, financial balance and energy balance. Data is manipulated by the auctioneer. Blockchain technology supports the simulation with data storage.

The starting point of the study is H2020 project EMPOWER undertaken by the Norwegian Centre of Expertise of Smart Energy Markets (NCE) and paper EMPOWER: Technical specification for software development [1]. Main idea of the project is to leverage citizen energy management in smart grids. The main focus of this thesis project is to create a local grid simulation with house-agents with the support of blockchain technology and simulate transactions between consumers and prosumers using internal currency, investigate the results of the simulation and check the viability of using blockchain technology in this particular area of application.

From this we can derive major goals with the emphasis on the third one:

1. Develop and test the simulation on a one season time-line in order to get valuable results.

2. Evaluate results and adjust the parameters.

3. Make recommendations for further use of multi-agent system and blockchain technology in appliance to local energy markets.

Apart from these goals, a lot of emphasis was done on investigating blockchain technology: how it works (simple blockchain net was developed), principles of cryptoeconomics, smart contracts and oracles, decentralized applications (DApps) and different frameworks for deploying DApps. Some of these points are covered in Chapter 3.

1.1 Local energy markets

Key concepts of this thesis are smart grids and local energy markets. The overall goal of the electricity market is to provide electricity in an efficient way while meeting the demands of the consumers. The competition and regulations are the different ways to achieve this goal. The traditional electricity market is facing challenges in integrating the new sources of generation of electricity, technology, infrastructure, increasing demand and the consumer-oriented market. It has shifted the paradigm towards new market design which will be able to fit in existing market structure. One of such electricity market is local electricity market. The benefits of local power generation, storage and demand response all cooperate together and peer-to-peer model enables the participation of all participants of the market. It uses information and communication technology (ICT) for sustainable and efficiently transfer of electricity to the consumers. It integrates the distributed generation, microgrid, and smart grid into one electricity market at distribution side [2]. Local energy markets and smart grids are inextricably linked.

A smart grid is an electrical grid which includes a variety of operational and energy measures including smart meters, smart appliances, renewable energy resources, and energy efficient resources. Electronic power conditioning and control of the production and distribution of electricity are important aspects of the smart grid [3].

Due to depleting fossil fuels, increasing exponential demand of electricity and liberalization of energy market has given birth to Smart Grid concept. Smart grid technologies emerged from earlier attempts at using electronic control, metering, and monitoring. Smart meters add continuous communications so that monitoring can be done in real time. So definition of smart homes appeared. With this data to manipulate, new algorithms of balancing the grid appear. Smart grid technologies offer more control and transparency into the distribution grid. Information sharing is an immediate benefit. Continued government focus on climate issues and emission goals, along with a steady price decrease on small-scale generators and panels have created a surge for local production situated at the leaf nodes of the distribution grid. Prosumers have emerged. When dealing with solar energy, there is usually an excess of energy than can be sold back to the electricity provider or outer grid. But some people would like to help out neighbors with their excess solar energy and get some profit, and it would make sense to enable local energy trade within neighborhood market to facilitate this type of exchange between prosumer and consumer.

In some places excess surplus from these has become a reason for concern. Government programs for non-fossil heating have been introduced too, and electric vehicles have since long made their entry into many European driveways and city traffic. Because of such developments power loads concentrated in short periods are increasing and tend to threaten grid capacity. Solving problems like these as close to their source is often a good strategy [4]. This recommends that these issues should be dealt with locally. A local energy market approach is suitable for this situation. This market type can negate the problem of renewable energy and create local community that can sustain themselves.

Current, centralized energy markets seek settlements on partial equilibriums and tend disregard externalities that push costs on a third party. Local markets can better seek a general equilibrium i.e. for both energy and flexibility. The advent of Internet of Things and the influx Home Automation Systems (HAS) are creating a new platform for energy management that suggest more active users and a less centralized energy management concept [4].

All of the items listed above are arguments that provide a rationale for studying the potential of local energy markets in terms of local peer-to-peer clean energy trading.

1.2 Current projects

There are several projects, that directly relates to the investigated topic. Due to blockchain is a hot topic nowadays a lot of energy companies would like to try it out and enroll their own researches. In order to improve local energy market schemes some projects were started, including EMPOWER, Solar Coin, Brooklyn MicroGrid/Transactive Grid, Scanergy, Smart Solar.

- EMPOWER

  Main goal of EMPOWER project is to develop and verify a local market place and innovative business models including operational methods to encourage micro-generation and active participation of prosumers exploiting the flexibility created for the benefit of all connected to the local grid. In order to do it main objectives are full grid control by participants, renewable energy supply for the grid, sustainable profits for the participants. System architecture includes such entities as smart energy service provider (SESP) and software agents for each participant. SESP plays a “controller” role: it provides an arena for local exchange of energy and flexibility and provides set of utility services, like forecasting loads, grid balancing, demand management and optimization. Personal agents are needed to reduce complexity for users and increase response frequencies and for further development they could possess low intelligence in order to make decisions or choose policies. EMPOWER project facilitates all energy flows locally in order to form its own ecosystem, with houses, charging stations, Photovoltaic stations (PVs), windmills etc. The most descriptive elements of the EMPOWER model are: smart energy service provider (SESP), community principles, network-like market/ hybrid market, trades using contracts, market reinforcement (as a consequence of network effects), scalability. This project is under ongoing development and has a solid descriptive model base for implementation.

- SolarCoin

  The SolarChange project was created to financially reward producers of solar energy via a blockchain. SolarCoin is a cryptocurrency for SolarChange project, launched in January 2014 and implemented to incentivize global solar electricity generation [5]. For every megawatt of solar energy fed into the grid the producer is awarded one SolarCoin. SolarCoin can be claimed by individuals living in homes with Solar Energy panels on their roof or large solar electricity farms. Nowadays, SolarCoin removed proof of work algorithm in favor to proof-of-stake-time (PoST), which is more environmentally energy friendly. This results in slight increase of its price and mining is now only via the production of solar energy.

- Brooklyn Microgrid (Transactive Grid)

  The Brooklyn Microgrid project is currently being developed in the USA by TransactiveGrid, a joint venture between LO3 Energy and ConsenSys [6]. The aim of the project is to test how blockchain technology can be used to effect direct neighbor-to-neighbor sales of solar energy. The technology used in the project builds on the Ethereum blockchain. Since April 2016, an initial pilot project run in Brooklyn has been exploring how to integrate buildings equipped with distributed energy resource systems (solar energy) in a decentralized peer-to-peer power grid. Implementation of the project requires both smart meter technology and blockchain software with integrated smart contract functionality: smart meters are needed to record the quantity of energy produced, blockchain software is needed to effect transactions between the neighbors, and smart contracts are needed to carry out and record these transactions automatically and securely. This is an opportunity for prosumers that allows them to no longer just feed their excess energy into the grid against payment of a fixed fee, but to market it individually and be an active participant in a local community market. Currently price formation and transaction are performed manually, but later it is stated, that it will be automated with user-defined policies.

- Scanergy

  The principle that lies beyond Scanergy project is similar to previous ones. It facilitates prosumers who put energy back into the network from domestic wind turbines and solar panels. Premise for the starting this project was the fact that the European Union is aiming for an 80 percent reduction in greenhouse gases [7]. In order to achieve that they need to reform how energy distribution grids works now. More and more consumers are becoming prosumers, who produce part or all of their own energies with renewable “green” technologies such as solar energy, and are willing to trade energy with their peers. One more issue was that they need to “localize” the peer-to-peer trade, due to the fact that European energy distribution networks would become inefficient and waste significant amounts of green energy, if prosumers trade energy globally. In order to do it the European power grids need to be turned into “smart grids” that permit producing and consuming energy locally, cutting down on the transmission loss which occurs over longer distances. Scanergy is a scalable and modular system for energy trading between prosumers. It is based on an intelligent multi-agent system that can manage the electricity produced and consumed both on a lower level (neighborhoods) as well as on a higher level (cities). The Scanergy project includes a real-time automated market trading system. This exchange system checks the supply of renewables and the overall demand for electricity in a given neighborhood via its smart meters every 15 minutes, then automatically brokers trades with other neighborhoods for any excess or shortfall. It pays a Bitcoin-like digital currency NRGcoin to those, who feed energy back into the grid according to actual electricity usage rather than predicted usage.

- Smart Solar

  Also there is one interesting project, related to energy transformation. In previously seen project Scanergy, internal currency NRGcoin offers numerous advantages over fiat currency, but, unlike Bitcoin, it is generated by injecting energy into the grid, rather than spending energy on computational power. More in depth generation process relates to Smart Solar project [8]. Basically goal of Smart Solar is to design a blockchain-connected solar panel that tracks its own energy and can create and transact its own renewable energy certificates (RECs) autonomously, so anyone can prove they've produced renewable energy. This provides greater transparency and accountability in the REC market and creates incentives to generate renewable energy to trade on energy markets. This project is noteworthy in that RECs can be traded on exchanges globally. In comparison TransactiveGrid tokens by design stay local.

Chapter 2

Problem description

2.1 Challenges

There are a lot of challenges, connected with designing local energy markets. Local energy markets deals with the linking and connecting energy, economy, and environment. The electricity market and the system are interlinked. So we can distinguish 4 areas that need to be considered: technical, economic, environmental, policy [2].

- Technical

  Technical problems include the design, installation, availability of local source and infrastructure. Smart meters, PVs, batteries, service provider and connections are the necessary minimum for market functioning. Monitoring is necessary for appropriate market data flow, checking and controlling the data needed for taking decisions in the global operations of local energy markets, which later can be automated. The outer grid connection for distribution network should be considered as well as possible connections to neighbor local markets. Also the whole system is dynamic, so we need relatively low time frames for updating information. In real-world system this timeframe should be around 0.5-5 minutes. Data security is also an issue, that has high priority to consider in real-world application.

- Economic

  Energy and economy are interlinked. Technology and structure can meet energy demand which is driven by the level of revenue and finance. The financial mechanism is required to reduce the initial cost of the market integration. Long run power purchase may cause unexpected results during the market evolution. Various business and financial models are required for the creation and participation of cooperative, local community. Trust or loyalty mechanisms of members should be considered to support the economics of the local energy markets.

- Environmental

  Although local energy market operates with clean energy from PVs or windmills, there are some issues with environment we should take into consideration. The easily accessible storage option for prosumer is the battery. The toxic properties of the battery during dumping and decomposition make a hazardous effect on the ecosystem. Geographical dependency and land requirement for PV station are concern issue during the establishment of local energy market. For some seasonal time periods, production can go to extremely low numbers, so the need of mechanism of smoothing down consequences is needed.

- Policy

  Planning of local energy market includes participants, objectives, policies, tools, procedures, and strategies needed during implementation of the local energy market in the proposed scenario. Prediction error can also appear in some situations. Consequences may include wrong load scheduling based on weather and electricity prices. Prediction errors may lead to errors in balancing the grid or needed energy for the next time period.

2.2 Project goal

The overall project goal is to check if multi-agent system and blockchain technology is applicable to local energy trading. Studying of MAS and local energy markets were discussed in EMPOWER project specification papers [9] [10] [11] [1], article about MAS and electricity trading [12] and multi-agent model of smart home investigation [13]. These articles give a representation how local energy market should operate in a smart grid, give specifications for market and models of performing trading, storing and producing energy. Our task is to simulate our model, get similar results with our simulation, check how blockchain technology will influence the model and apply different parameters to see how the model reacts.

The second goal is to make recommendations for further blockchain technology in appliance to local energy markets and check if learning agents are of useful in this system compared to zero-intelligence agents.

Chapter 3

State-of-the-art

3.1 Analysis of problem area

Despite sustainable development of intelligent networks (smart grids) in the energy sector, the services of retail electricity market are still awaiting modernization. Key problems include:

- How to provide customers with reliable information on costs and consumption so that they can assess the new opportunities for a fully integrated energy market
- How to stimulate active participation, simplify transition to new contracts and manage demand-supply category in terms of changing prices
- How to ensure integration in the market of residential energy services, expand the scope of consumer choice, demonstrate the advantages of independent power generation and energy consumption, as well as local energy generation

In this context, distributed registries can act as catalysts for the transition to a new level of integration and development of the retail energy market. One of the valuable mechanisms here is energy generation with PV systems. Micro-generation implies the possibility of independent generation of electricity for consumers within a single home or local community. The concept of “market” indicates the possibility of selling energy generated in conditions of microgeneration to consumers by prosumers. Traditionally, this market has worked on the basis of pre-arranged bilateral agreements between prosumers and retail energy suppliers. Until now, prosumers did not have full access to the energy market, which remains a privileged platform for big energy suppliers. This significantly limits the economic benefits of microgeneration for end users. Distributed registers, in combination with systems and smart metering and new generation batteries (for local energy storage), in the future can open access for consumers to the energy market. Intelligent counters can be used to record and register self-generated energy in a distributed registry. Self-generated electricity can be used for domestic needs, accumulated in a new generation of batteries for later use, or transferred to the network. Another way is using the distributed and ubiquitous nature of the registry, to exchange the generated energy as a commodity elsewhere, for example, while charging an electric vehicle abroad. One more possibility is to sell it through the register to the most profitable buyer, using a mechanism similar to the mechanism of the exchange market.

Another mechanism is energy contracts. A consumer planning to switch to another energy supplier must close the current contract with the current supplier and enter into a contract with the new supplier and examine the terms of the contract for all additional energy services provided by third parties. Managing a variety of administrative processes in performing such operations is a real obstacle to the further development of a competitive retail energy services market and provide high costs for energy suppliers and distributors. The use of distributed registers for the online registration of energy contracts would greatly simplify these operations. Consumers could make the transition from one supplier to another in just a few clicks on a computer or mobile device. Similarly, energy suppliers and energy service providers could save significant resources for these operations. The issues of scalability, security and stability of such applications have not been solved yet. Nevertheless, the advantages of this technology are rather promising and our task is to show that this concept is viable.

This project is about implementing a system which will be decentralized, autonomous, smart and transparent at the same time. This can be achieved using blockchain technology with smart contracts and multiple software agents, which takes care about energy auctions between prosumer agents. In order to do that we have investigated scientific papers and articles. Through literature study we have covered main topics such as blockchain technology, Ethereum platform, smart contracts and oracles, current blockchain projects, multi-agent systems and simulations, learning techniques for multi-agent systems. First we will cover the basic technology for secure, decentralized and transparent transactions - blockchain.

3.2 Blockchain technology

3.2.1 Blockchain

A blockchain is a shared digital decentralized ledger (registry) that records transactions across a peer-to-peer network. Transactions are formed to blocks. Every block then connects to the next one with use of cryptographic signature. The block header includes its hash, hash of the previous block, hash of transactions and additional service information. Transactions are the key technology in blockchain. The only way to change the state of the registry is to use transactions. Records can be added to registry only with consensus of majority of the network. One important property of the registry in blockchain is immutability. This means that you cannot change transactions or blocks, cannot delete or insert in a random place. Immutability property is provided by cryptography mechanisms. Two most simple cryptography algorithms that lie in the core of blockchain are hash-functions and digital signatures, which provide transaction integrity and authorization.

Each transaction is encrypted and sent to many individual peers, each of which stores the data locally. The members of the network automatically verify the transactions stored on the nodes. Transaction models for centralized and decentralized architecture are shown on figure 3.2.

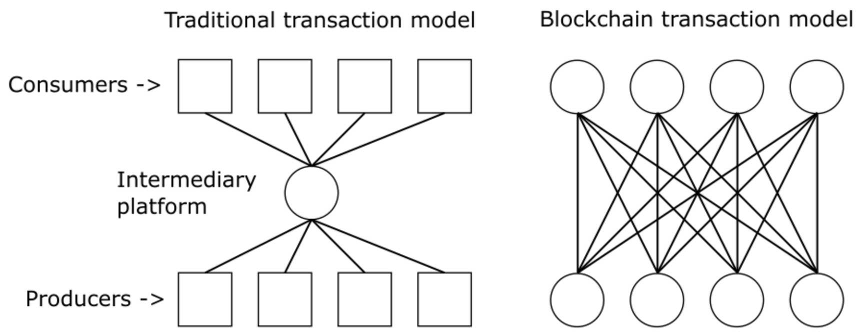

Figure 3.1: Transaction models for centralized and decentralized architecture

Globally blockchain is a network for transaction processing with protocol rules, which are used by participants in order to interpret the transaction registry to get the state of the network. With this said, blockchain is decentralized: even if some nodes will fail or be compromised, system will still work. Decentralized, open and mathematically grounded nature of the blockchain allows people and organizations to minimize the risks of interaction between themselves and conduct peer-to-peer transactions, excluding intermediaries. This also has a positive effect on safety.

Hash functions in blockchain guarantee the immutability of the entire transaction chain. In Bitcoin SHA-256 function is used to calculate hashes. One more feature connected to optimization of blockchain performance is Merkle trees concept shown on figure 3.2.

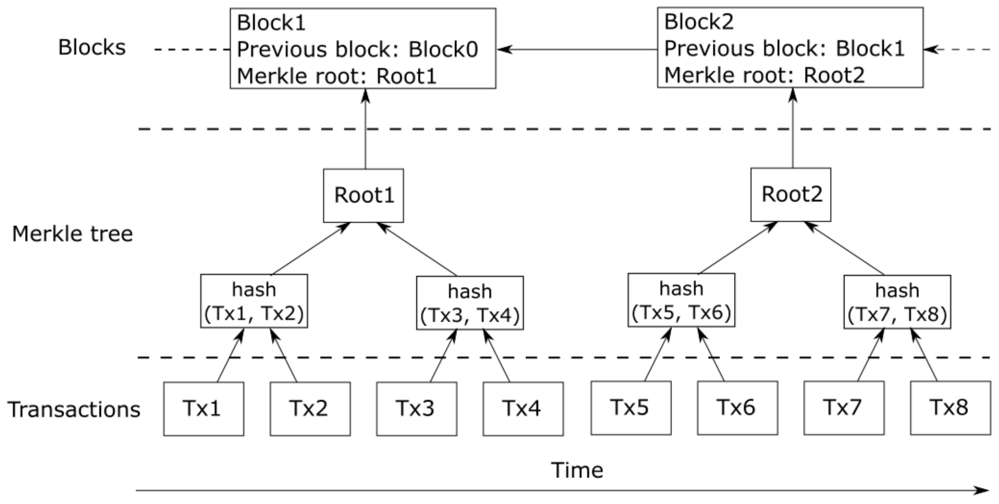

Figure 3.2: Merkle trees concept

Merkle tree is a data structure, also known as binary hash tree. A list of transaction hashes is fed to the function input. At each stage of the calculation, successive pairs of hashes are glued together using a hash function. If the hash is an odd number, then the latest is duplicated. The result is a single hash, which is the final hash value for the entire list. This algorithm enables the existence of "light client" as they only verify and synchronize headers of the blocks without transaction. This principle enables to cut the memory space of blockchain data significantly and called Simplified Payment Verification (SPV).

The purpose of the verification process is to achieve consensus on the content of the distributed ledger. Consensus-based verification is a decentralized and automated process. The following two mechanisms are most commonly used to establish consensus: proof-of-work (PoW) and proof-of stake (PoS). In PoW users (verification users or miners) are continuously verifying the hashes of transactions through the mining process in order to update the current status of the blockchain assets. It is highly demanding process in terms of computational power. PoS on the other hand is not so demanding. PoS requires users to repeatedly prove ownership of their own share (stake) in the underlying currency. This approach reduces the complexity of the decentralized verification process and can thus deliver large savings on energy and operating costs. In most blockchain systems hybrid consensus protocols are used as PoW and PoS have their own strengths and weaknesses. The first relevant blockchain application was Bitcoin, a so-called "cryptocurrency". Over recent years, Bitcoin has become the basis for other blockchain applications, most of which are currently being developed in finance. A number of businesses and initiatives have recently been launched that apply the blockchain principle to other industries, among them the energy sector. Blockchain applications are generally considered to be a very promising technology but they are still at an early stage of development. In order to check the basic blockchain concepts, simple program was developed. Block class, hashing function for blocks, function for block generation, checks for block integrity and pick longest chain rule were implemented. Basic schemes of developed chain are shown: block structure on figure 3.3, longest chain rule on figure 3.4 and system interactions on figure 3.5. Some aspects of real blockchain systems were intentionally omitted (like mining or nonce number for forming blocks).

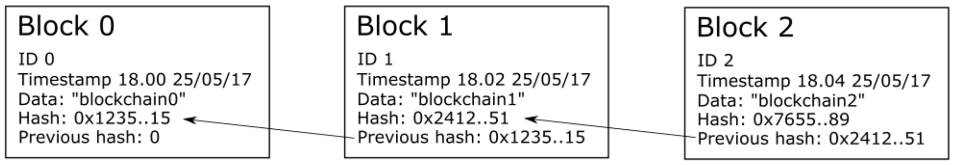

Figure 3.3: Simplified block structure in blockchain

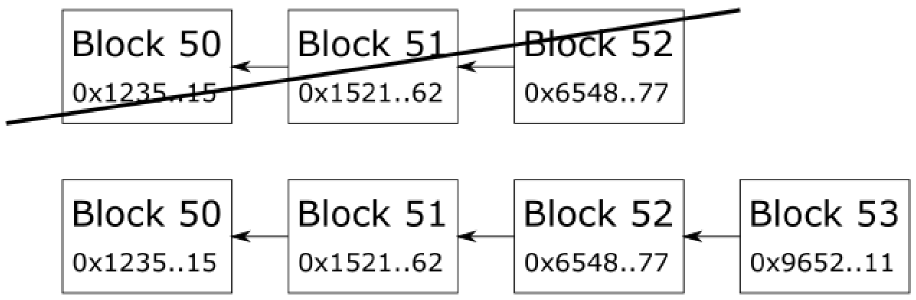

Figure 3.4: Longest chain rule principle

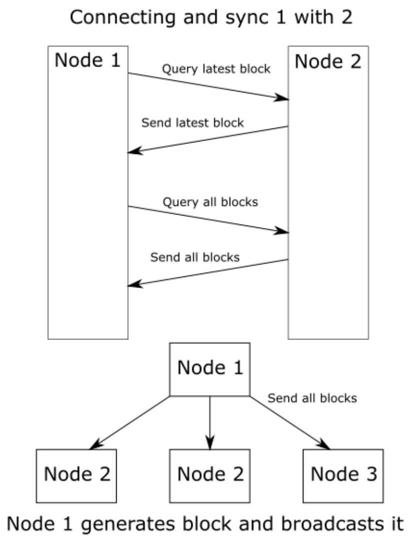

Figure 3.5: Nodes interactions

During literature study phase and state-of-the-art investigation a lot of literature, concerning blockchain was analyzed. Studied the origins of Bitcoin system, history of development, its philosophy of peer-to-peer interactions, how it works in general and solutions to overcome blockchain problems, such as light clients for portable nodes [14]. Explored the principles of blockchain technology, checked questions regarding consensus and validation techniques, noted possible security issues [15]. Got familiar with the latest blockchain projects in financial and nonfinancial sector, got the idea how blockchain transaction are verified [16]. Explored information about decentralized applications, their ecosystem, introduces peering networks for application, gives examples of successful applications like OpenBazaar (decentralized market), Lighthouse (decentralized crowdfunding), La'Zooz (online taxi) [17]. Checked different types of chain interoperability (Notaries, Relays, Hash-locking), potential use-cases it can be achieved and give comparative characteristics for these types. Also got an idea for interoperable application development and recommendations for dealing with possible issues [18].

Blockchain technology itself needs ecosystem to operate. It includes solutions for storing data, creating communications and performing calculations. Examples of such projects are Storj (distributed saving and storing files), IPFS (file service, link management, file management), Miadsafe and Ethereum (storing, communications, file management). Applying blockchain to the task at hand we can select several platforms that provide necessary blockchain functionality. They are Stellar, BigchainDB, Hyperledger and Ethereum.

The most fast developing, promising and interesting platform was chosen for further investigation - Ethereum.

3.2.2 Ethereum platform

Ethereum is a decentralized platform that runs smart contracts: applications that run exactly as programmed without any possibility of downtime, censorship, fraud or third party interference. These applications run on a custom built blockchain, an enormously powerful shared global infrastructure that can move value around and represent the ownership of property. This enables developers to create markets, store registries of debts or promises, move funds in accordance with instructions given long in the past (like a will or a futures contract) and many other things that have not been invented yet, all without a middle man or counterparty risk [19].

On traditional server architectures, every application has to set up its own servers that run their own code in isolated threads, making sharing of data hard. If a single application is compromised or goes offline, many users and other applications are affected. On a blockchain, anyone can set up a node that replicates the necessary data for all nodes to reach an agreement and be compensated by users and application developers. This allows user data to remain private and applications to be decentralized like the Internet was supposed to work.

In common, Ethereum is software running on a network of computers that ensures that data and small computer programs called smart contracts are replicated and processed on all the computers on the network, without a central coordinator. The vision is to create an unstoppable censorship-resistant self-sustaining decentralized world computer with states.

The prerequisites for the emergence of the platform stemmed from the shortcomings of Bitcoin in appliance to creating of decentralized applications. Major important limitations of Bitcoin were:

- Lack of Turing-completeness

  Though Bitcoin supports scripting language on top of it and provides some computational capabilities, it does not provide everything what was needed. Main problem were missing loops. This was done to avoid infinite loops during transaction verification. Though, in order to implement alternative elliptic curve signature algorithm would require 256 repeated multiplication rounds.

- Value-blindness

  Bitcoin scripting language has a problem with providing fine-grained control over the amount that can be withdrawn.

- Lack of state

  There are no possibilities for multi-stage contracts with Bitcoin. It only has simple data manipulation operations.

- Blockchain-blindness

  Bitcoin scripting language is blind to certain blockchain data such as the nonce and previous block hash. This severely limits some types of applications.

The point of Ethereum was to eradicate these limitations and provide more for decentralized applications. The intent of Ethereum is to create an alternative protocol for building decentralized applications, providing a different set of features that are useful for a large class of decentralized applications, with particular emphasis on situations where rapid development time and solid security. Ethereum implements it with a blockchain with a

Characteristics	Ethereum	Bitcoin
Time block	10 s	10 min
Value for block mine	5	12,5
Created blocks	>1400000	>400000
Transactions in one block	>30000	>1200
Nodes in network	>6000	7000
Currency value	∼10$	∼420$

Table 3.1: Cryprocurrencies comparison

built-in Turing-complete programming language, allowing anyone to write smart contracts and decentralized applications where they can create their own arbitrary rules for ownership, transaction formats and state transition functions.

Ethereum extends the blockchain concepts from Bitcoin which validates, stores, and replicates transaction data on numerous computers. It takes this one step further by running code on multiple machines.

Ethereum does not only play the role of distributed data storage, it also provides computations. The computer programs being run are called smart contracts (which we will cover further), and the contracts are run by participants on their machines using a virtual operating system called a "Ethereum Virtual Machine" (EVM). The design, features and inner implementation of EVM is described extensively in Ethereum Yellow Paper [20].

Basic concepts of EVM are ether, accounts, messages, transactions, gas and code execution [21]. The Ethereum network includes its own built-in currency, ether, which serves the dual purpose of providing exchange between various types of digital assets and, more importantly, of providing a mechanism for paying transaction fees. The most used denominations: 1 - wei, 10^{12} - szabo, 10^{15} - finney, 10^{18} - ether. In the near future, ether is expected to be used for ordinary transactions, finney for microtransactions and szabo and wei for technical discussions around fees and protocol implementation, the remaining denominations may become useful later. Comparison table for cryprocurrencies of Bitcoin and Ethereum is shown in table 3.1.

In Ethereum, the state is made up of objects called "accounts", with each account having a 20-byte address and state transitions being direct transfers of value and information between accounts. An Ethereum account contains four fields: nonce (for one-time possessed transaction), accounts ether balance, account's contract code and account's storage. Accounts also play a role of a public key for digital signature algorithm. There are 2 type of accounts: user and contract. In user account contract code field is empty.

Transactions are used in Ethereum transfer signed data package that stores a message to be sent from an externally owned account. Transactions contain: the recipient of the message, signature identifying the sender, amount of ether to transfer from the sender to the recipient, optional data field, $STARTGAS$ value, representing the maximum number of computational steps the transaction execution is allowed to take and $GASPRICE$ value, representing the fee the sender pays per computational step.

When you activate a smart contract, you perform the calculations within it. This costs time and energy, and gas is the mechanism to pay them for service. The payment is a small amount of ether. General formula for that is shown in equation (3.1).

$$Payment\ (in\ ether) = gas\ amount(in\ gas) * GASPRICE(in\ ether/gas) \tag{3.1}$$

The more complex the smart contract (the number and type of computational steps, memory used for storage), the more gas the contract requires to run and complete.

Whereas the $STARTGAS$ is used to run a contract is fixed for any specific contract, as determined by the complexity of the contract, the $GASPRICE$ is specified by the person who wants the contract to run. This is a competitive mechanism that allows running the contract by miners of the net. The $STARTGAS$ and $GASPRICE$ fields are crucial for Ethereum's anti-denial of service model. In order to prevent accidental or hostile infinite loops or other computational wastage in code, each transaction is required to set a limit to how many computational steps of code execution it can use. The intent of the fee system is to require an attacker to pay proportionately for every resource that they consume, including computation, bandwidth and storage. Contracts have the ability to send "messages" to other contracts. Messages are virtual objects that are never serialized and exist only in the Ethereum execution environment. Essentially, a message is like a transaction, except it is produced by a contract and not an external actor.

In terms of code execution in Ethereum contracts, code is written in a low-level, stack-based bytecode language, referred to as "Ethereum virtual machine code" or "EVM code". The code consists of a series of bytes, where each byte represents an operation. In general, code execution is an infinite loop that consists of repeatedly carrying out the operation at the current program counter (which begins at zero) and then incrementing the program counter by one, until the end of the code is reached or an error or $STOP$

or *RETURN* instruction is detected. The operations have access to stack, memory and contact's long-term storage. While EVM is running, its full computational state can be defined by the values, which defined as block state, transactions, memory, stack, messages, code, gas.

Concerning consensus algorithm in Ethereum now PoW is used. Though it works fine with Bitcoin, it has several issues like consuming tremendous amounts of energy and now Ethereum developers seek alternative, “greener” solution. The starting point is PoS. After several years of development, it became clear that alteration of PoS is needed for Ethereum, due to specifics of the platform [22].

After some development iterations an idea known as Casper appeared, which is described as “consensus by bet”. The development process involves heavy exploration of both economic and game-theoretic considerations and Byzantine-fault-tolerant protocols, trying to create a protocol that satisfies one of several constraints simultaneously. What separates Casper (and other more recent versions) from traditional PoS, is that it punishes participants who don't play by the rules. The development is still on-going and expected to be finished in late 2017 [23] [24].

Although Ethereum is still work in progress, it is already censorship-resistant self-sustaining decentralized world computer that can perform calculations, store data, and allow communications. There is a public permissionless open source version, and forks of this have been taken and adapted for private network use. The public and private versions are attempting to solve different problems.The technology is currently immature, but as more people use it, test it, develop it and build on it, it will improve and become more robust.

3.2.3 Smart contracts and oracles

Smart contracts are account holding objects on the Ethereum blockchain. They contain code functions and can interact with other contracts, make decisions, store data, and send ether to others. Contracts are defined by their creators, but their execution, and by extension the services they offer, is provided by the Ethereum network itself. They will exist and be executable as long as the whole network exists, and will only disappear if they were programmed to self-destruct.

To deploy a contract two things needed: the compiled code and the Application Binary Interface (ABI). Compiled code is needed for EVM and ABI is used for calling these contracts. Contracts are typically written in some high level language such as Solidity, Serpent or LLL, and then compiled into byte code to be uploaded on the blockchain.

Using smart contracts can simplify work in many areas of life, including logistics, management, internet of things (IoT), law and even in elections. Main advantages of smart contracts are: independence of 3rd party, safety (stored encrypted on blockchain), reliability (using blockchain principles), economy (no fees for 3rd parties), accuracy (no need in forms registration). There are also some problems associated with smart contract: contract regulations, taxes and code dependence. These problems have high impact on usage rate of smart contracts nowadays, but in time it is possible to resolve all problems connected with them.

Oracles are the extensions of the contracts. Smart contracts, by their nature, are able to run algorithmic calculations and store and retrieve data. Because every node runs every calculation, its not practical (and presently impossible) to make arbitrary network requests from an Ethereum contract. Oracles fill this void by watching the blockchain for events and responding to them by publishing the results of a query back to the contract. In this way, contracts can interact with the off-chain world. The example of an oracle can be "Proof-of-phone", which will provide information if particular address in Ethereum system is associated with specific phone number. This is rather useful technology, but has trust issues that should be resolved.

3.2.4 Ethereum applications

Generally, all Ethereum applications have the following properties:

- Cryptographically Secure

  Uses public/private signature technology. Blockchain applies this technology to create transactions that are impervious to fraud and establishes a shared truth.

- Decentralized

  There are many replicas of the blockchain database and no one participant can tamper it.

- Data and Smart Contracts

  The Ethereum blockchain can store both data and Smart Contracts in the blockchain.

- Immutable Ledger

  Blockchain is a write-once database so it records an immutable record of every transaction that occurs.

In general, there are three types of applications on top of Ethereum. The first category is financial applications, providing users with more powerful ways of managing and entering into contracts using their money. This includes sub-currencies, financial derivatives, hedging contracts, savings wallets, wills, and ultimately even some classes of full-scale employment contracts. The second category is semi-financial applications, where money is involved but there is also a heavy non-monetary side to what is being done; a perfect example is self-enforcing bounties for solutions to computational problems. Finally, there are applications such as online voting and decentralized governance.

The statistics regarding the Ethereum blockchain make it apparent that the project has attracted a lot of attention. There are already a number of decentralized applications and concepts built on top of Ethereum, some of which show great potential. The typical decentralized application architecture is shown on figure 3.6.

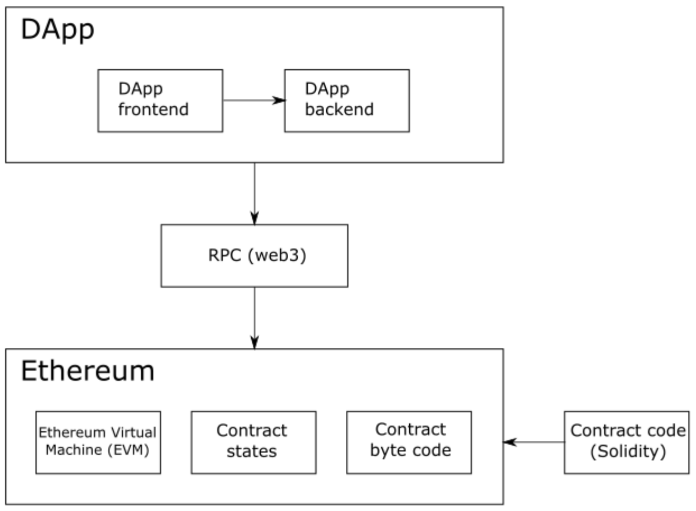

Figure 3.6: Typical DApp structure

Decentralized Applications or DApps (called in the Ethereum community) are the type of applications built using smart contracts and Ethereum platform. The goal of a DApp is to have a nice UI to smart contracts system plus any extra improvements. While DApps can be run from a central server if that server can talk to an Ethereum node, they can also be run locally on top of any Ethereum node peer. They may use the blockchain to submit transactions and retrieve data rather than a central database. Instead of a typical user login system, users may be represented by a wallet addresses and keep any user data local.

3.3 Multi-agent systems

Multi-agent system (MAS) is a system formed by several interacting intellectual or zero-intelligence agents. Multi-agent systems can be used to solve problems that are difficult or impossible to solve with a single agent or a monolithic system. Examples of such tasks are online trading, the elimination of emergencies, and the modeling of social structures.

In a multi-agent system, agents have several important characteristics:

- Autonomy: agents, at least partially, are independent
- Limited representation: none of the agents has an idea of the whole system, or the system is too complex for knowledge of it to have practical application for the agent.
- Decentralization: there are no same level agents controlling the entire system

Agents can exchange received knowledge using special methods or by third party.

Considering peer-to-peer application in a blockchain environment, we identify nodes with agents. An autonomous agent is defined as a reactive, proactive, and social entity. Autonomous agents interact with each other according to the protocol specified for the P2P network containing the nodes. Agents may enter with each other into smart contracts that are mechanisms involving digital assets and two or more parties where some or all parties put assets in and assets are automatically redistributed among those parties according to a formula based on certain data or event that is not known at the time the contract is initiated. Autonomous agents together with smart contracts regulating the relationship between them make up decentralized system. In a multi-agent system a solution to the problem to be solved is delivered through autonomous actions by and interactions between agents and environment. During investigating state of the art stage several articles were studied and analyzed: explored formal models for agent-based systems in electricity markets and got the idea of energy flows in the ecosystem of smart grids [25]; investigated how double action principle works in the market, refreshed knowledge about Nash equilibrium and explored experimental results [26]; got familiar with EMPOWER project and its general concepts, regarding multi-agent systems and zero-intelligent agents, rules of trading, process and requirements [10]; checked the real implementation of MAS with auction principle in Python, how it is programmed and organized [27]; reviewed recent implications and trends for MAS technology in controlling of the smart grid scenario[28]; explored the multi-agent system, priorities of the agents, real case studies with loads in system and its performance [13].

In our work we use zero-intelligence agents with predefined policy. In order to improve the results of agent behavior machine learning algorithms can be used, especially some modifications of reinforcement learning with dynamic programming principles.

3.4 Possible issues

Though Blockchain technology provide a lot of capabilities for effective organization of energy trading, some of the features can be a problem for developing such system. Possible issues with this project may include:

- Speed

  In order to store transactions in a blockchain, first, they should be formed into blocks and then verified. This process takes time. In public blockchain it will probably be a greater issue due to 10 seconds block formation. If time discretization of the system will be more than 10 seconds, system will handle this.

- Lack of standardization in DApps and smart contracts

  Currently DApps are in early development stage, as well as Ethereum with smart contracts. There are no standards for writing specific types of contracts or building an application. Also there is no long-term experience with this technology, which implies to develop with caution.

- Security issues

  These problems are referred to blockchain integrity and smart contracts design. These exploits already had place with one of the first Ethereum project The DAO. Smart contract error inside The DAO allowed hackers to steal ether from a crowdfunding pool, which eventually led to a community split of Ethereum to Ethereum and Ethereum

Classic, which continued the version of chain after stealing (hard-forked). Ethereum reversed changes and continue to operate. More information can be found in article [6] [29].

- Need to relocate the logic and computations

  Smart contracts and Solidity language are powerful instruments. In order to create fully distributed and decentralized application all logic and computations should be written in smart contracts. But due to Ethereum platform functionality enormous amounts of gas are needed in order to operate with these smart contracts. Also real Ethereum chains are rather slow, so hard commutations will be difficult to handle. This can be avoided in our case by choosing private chain configurations and distributing logic and computations to other program elements. This process is defined in detail in Chapter 4.

3.5 Ways of implementation

In this thesis we are developing a solution to let neighbors sell excess solar energy among each other on a local marketplace. Using blockchain is really natural and obvious solution if we consider all advantages mentioned above. The blockchain makes the whole process a lot simpler and more transparent and will allow for secure transactions that can easily be verified in real-time.

All of the trades designed platform should be executed through smart contracts, a technology which is natively supported by the Ethereum blockchain. Using a smart contract ensures no one can modify the data of the agreement once it has been recorded on the blockchain, which is the most valuable feature in using blockchain technology. There are multiple advantages to this type of solution, as buyers and sellers know exactly who they are dealing with and where the solar energy is coming from. Smart contracts are a good fit for these types of transactions, as they make the boundaries of the agreement between parties clear and tamper-proof. Wielding this technology on top of the Ethereum blockchain can be of great value to local neighborhood. In order to implement this, there are some ways to do it effectively, without scalability issues and with ability to add depth to the system. We have investigated 3 major ways:

- Python programming language with frameworks: *pyethereum, web3.py, populus, testrpc.*
- JavaScript programming language with frameworks: *ethereum-js, web3.js, Truffle, Embark*; Microsoft Azure web-server.

- Python programming language with BigchainDB technology

These ways of creating a blockchain solution uses a lot of state-of-the-art technology starting from platforms and drivers and ending with frameworks. Geth is the official client software provided by the Ethereum Foundation. It is written in the Go programming language. Other clients are pyethapp, written in Python, eth, written in C++ and Parity, written in Rust. When Geth starts its client daemon, it connects to other clients (also called nodes) in the network and downloads a copy of the blockchain. It will constantly communicate with other nodes to keep it's copy of the blockchain up to date. It also has the ability to mine blocks and add transactions to the blockchain, validate the transactions in the block and also execute the transactions. It also acts as a server by exposing APIs you can interact with through RPC. A command line tool called geth, allows to connect to the running node and perform various actions like creating and managing accounts, query the blockchain, sign and submit transactions to the blockchain. pyethereum and ethereumjs are frameworks that provides core Ethereum functionality on Python and JavaScript respectively. They include modules for work with EVM, Blockchain, blocks and data, transactions, accounts, hashing functions. Microsoft Azure is a cloud based solution that can be used to deploy the Ethereum MultiMember Network. Network consists of a subnet for mining, a subnet for transaction processing, load balancer and regulator module. The main regulator module allows adding, deleting or changing the parameters of the participants. API application is necessary for working with a smart contract to identify customers. An application written on NodeJS can access smart contracts with *GET* and *POST* requests, which greatly facilitate the work with smart contracts. Subnets in turn consist of one or more Ethereum nodes.

Truffle and Embark are the two most popular frameworks used to develop DApps. They abstract away lot of the complexities of compiling and deploying your contract on the blockchain. It provides functions for creating, testing and deploying of the contracts easily, create customizable chains and perform diagnostics. For Python implementation this type of framework is *populus*.

Testrpc is a Python client that doesn't support real blockchain mining, so it's not a full client, but mining can be simulated for testing and development purposes, if needed.

With *populus* it is possible to run a test network using geth, or another fast way of getting a testnet running is using testrpc. Testrpc will create a bunch of pre-funded accounts for the testing when it starts up.

BigChainDB is a blockchain implementation, written in Python, which is often referred to as a solution to all problems with the data storing. BigChainDB has a very high transaction speed (1 million/sec), a huge storage capacity (due to distributed storage with partial replication). BigChainDB gets these benefits through a simplified consensus when building blocks, and by storing all the blocks and transactions in the existing NoSQL implementation of the database (RethinkDB or MongoDB).

To communicate with the Ethereum node, there is a JavaScript library called Web3.js which can be used to interact with a node. Web3 object communicates to a local node through RPC calls. Since it is a JavaScript library, you can use it to build web based DApps. Communication principle is shown on figure 3.7.

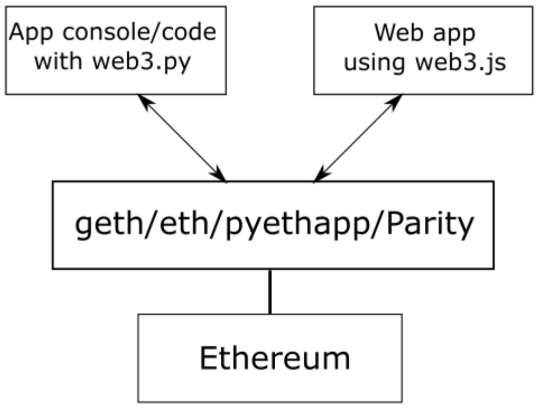

Figure 3.7: web3, clients and Ethereum interactions

As of date of writing this thesis, practically all developments, except on Go language and JavaScript, have been stopped. Due to this JavaScript based frameworks are one of the best ways to start building web-service based DApps. Still it is possible to perform development with other ways.

Chapter 4

Method

4.1 Choosing approach

In Chapter 3 we have stated 3 possible ways of developing local energy market application with multi-agent technology and blockchain support. In order to choose the most appropriate way the feature analysis was performed. All mentioned above methods were tested on simple test application and investigated for their advantages and disadvantages.

- Python and BigChainDB

  This approach have a solid background as BigchainDB provides only data storage for the records. It can be deployed with various types of nodes (test, production, development), using local server or deploy a test cluster with AWS or Microsoft Azure. It can be easily programmed in any way which good documented API for Python. Speed and data storage is rather good, compared to NoSQL databases. Developer has full control of block data and assets in the system.

  This architecture has a significant drawback - each node has full rights to write to a common data storage, which means that the system as a whole is unstable to the problem of "Byzantine generals" [30]. Such an easy approach to the fundamental problem causes community criticism of the project, because the high speed and bulk characteristics of BigChainDB in the absence of BFT (Byzantine Fault Tolerance) are not so different from those demonstrated by the noSql databases RethinkDB and MongoDB, which are used for data storage. Thus, the actual use of BigChainDB is limited to private networks. For public networks, it does not fit.

- JavaScript, *ethereum-js, web3.js, Truffle, Embark*, Microsoft Azure web-server

  This way of deploying blockchain is highly effective, but also demands funding. Microsoft Azure services are not free. Ethereum community now relies on JavaScript implementation of Ethereum platform and made a lot of decentralized application based on it. *Truffle* and *Embark* are the best instruments for creating, deploying and testing smart contracts. They have a solid code base and allow to create chain profiles for testing and production. *Truffle* and *Embark* allow to create a page template with contract functions already deployed and integrated to the page.

  This function is very convenient, especially with plugin MetaMask, which allow to test Ethereum in Internet browser. A lot of companies use this approach, for example, Russian bank “Alpha-bank” made a letter of credit system on a blockchain with Microsoft Azure, with frameworks: *NodeJS, React, Redux.*

- Python, *pyethereum, web3.py, populus, testrpc*

  Python and *pyethereum* duplicate previous method in many ways. *Populus* is analog for *Truffle* framework, with some additions to chain management. *populus* framework allows to customize chains, perform smart contracts deployment process and provides useful utility functions for development process. Earlier, Python and Serpent (smart contract programming language) were the way-to-go in developing DApps. Now there is a Solidity compiler for Python, so contracts can be still written in Solidity, rather than Serpent, which is less functional in terms of contract design. Testing with this approach is rather convenient and does not require a web server. Also it is rather easy to run a simulation and plot results with Python libraries. Main disadvantage that can be assigned is that the resulting application will not be truly DApp. A lot of functionality will be transfered to server side, which includes transaction management and value computations. Though, this approach will allow to program simulation system as it should be working in a test environment.

In order to choose best way we need to identify blockchain configuration for our application. Blockchain can be divided into 2 configurations: private and public blockchains.

Public blockchains are trustless due to consensus algorithm enabling anyone to join as a participant. It must be expensive and difficult to publish a block to prevent fraud and spam (proof of work/mining), global digital currency (e.g. Ether) are used to pay to process transactions and smart contracts. Developing in public chains is not an effective strategy, due to stated reasons.

Private blockchains or Sandboxes designed for rapid application development and instant deployment, suited for single enterprise solutions that can be configured for high throughput, does not require digital currency for transaction processing, but tokens could be useful for internal currency.

Ethereum supports both configurations and BigchainDB, as stated earlier, is good with private blockchains. So the speed of development and ease of use of frameworks is the crucial factor. So the Python with Ethereum frameworks approach was chosen.

4.2 Concepts testing

In order to test core functions of blockchain interaction simple blockchain program was developed. It is not directly included into simulation, but it helped to understand the underlying concepts. Though it ignored several features like mining, it helped to derive the basic work flow for blockchain transactions. The features of the developed program were described in Chapter 3. The work flow is shown on figure 4.1.

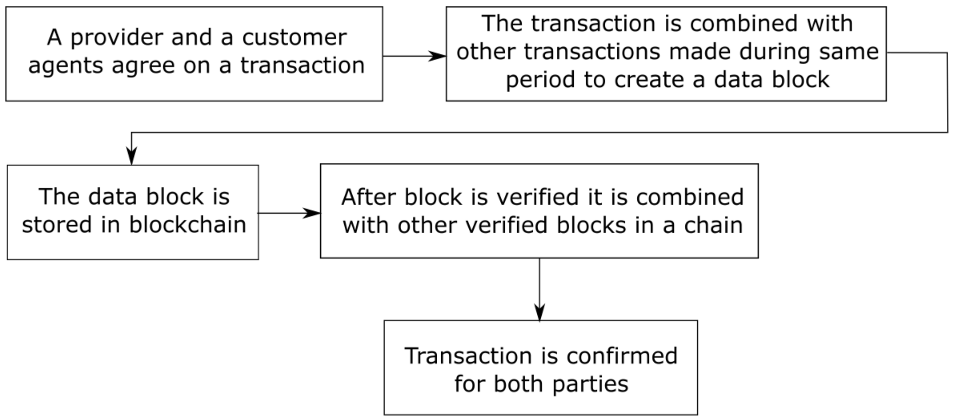
A provider and a customer agents agree on a transaction
The transaction is combined with other transactions made during same period to create a data block
The data block is stored in blockchain
After block is verified it is combined with other verified blocks in a chain
Transaction is confirmed for both parties

Figure 4.1: Transaction verification process in blockchain

With pyethereum, implementation of Ethereum platform with Python, utility functions were tested. These functions include creating a new test blockchain configuration with a custom genesis block, sending a transaction using the private key and returning value of the contract, creating a contract with given Serpent file, sending transactions with raw EVM code, mining simulation, block manipulation (getting addresses, hashes, timestamps, gas values).

With built in Ethereum tester, unit-tests were created. Scenarios include sending ether to address, simple token exchange, token buy/sell procedures and utility contract calls. Also utility functions were tested: *sha*3() - hashes a pass phrase to a private key value with SHA-3 (Secure Hash Algorithm 3) cryptographic hash function, and *privtoaddr*() - gets an address from private key.

During designing of smart contracts Serpent and Solidity smart contract programming languages were studied and analyzed. With Solidity several contracts were implemented like standard coin, safe purchase, open auction, blind auction. UML diagrams for these contracts are shown on figure 4.2.

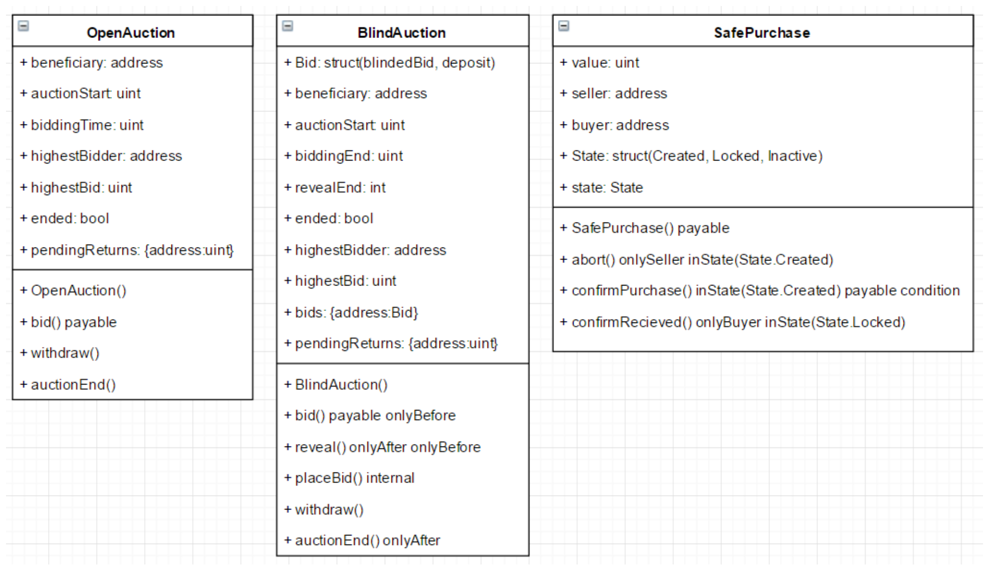

Figure 4.2: UML diagram of test smart contracts

The general idea of the simple open auction contract is that everyone can send their bids during a bidding period. Sending ether bind the bidders to their bid. If the highest bid is raised, the previously highest bidder gets money back. After the end of the bidding period, the contract has to be called manually for the beneficiary to receive his money - contracts cannot activate themselves.

Blind auction is an extension to previously described open auction. The advantage of a blind auction is that there is no time pressure towards the end of the bidding period. During the bidding period, a bidder does not actually send bid, but only a hashed version of it. This is done due to transparency of the system - one of the basic principles of public blockchains in Ethereum. Since it is currently considered practically impossible to find two (sufficiently long) values whose hash values are equal, the bidder commits to the bid by that. After the end of the bidding period, the bidders have to reveal their bids. Bidders send their values unencrypted and the contract checks that the hash value is the same as the one provided during the bidding period. Due to the fact that the auction should allow binding and be blind at the same time, the bidder should send actual ether with the encrypted bid. The contract solves this problem by accepting any value that is at least as large as the bid for preventing bidders from messing with auction mechanism.

These auction does not include solid computations, so the cost of deployment (in gas) is rather low.

Safe purchase contract allowed to test contract states for safe transition of the asset. It is done with the help of modifiers and conditions combined with states.

In order to deploy these contracts and get their ABI for the program to interact, *solc* compiler for solidity language was used. *py-solc* is a wrapper around *solc* compiler, providing to use API for compiling contracts. *py-solc* provides the high level *compile_files*() and *compile_sources*() functions for Solidity compilation from within python code. It also provides a *link_code*() function to handle library linking.

Web3.py and populus are frameworks that provide API for contract compilation, deployment, testing, test chain management and analyzing. With this frameworks a testing project environment for smart contract functionality was created. The profile for the chain was testrpc.

After testing all components, needed to build a blockchain simulation system, we need to assemble the system, using all checked technology. Blockchain is the core environment and store valuable data. As we mentioned above we moved from classical DApp architecture in favor to centralized computing. So our simulation python code will provide computations and actions to blockchain for smart contract interactions. With *web3* API we will interact with Ethereum network. DApps, written as pure HTML/JS/CSS can be completely serverless and can interact directly with Ethereum platform using web3 RPC. Typical blockchain project architecture scheme is shown on figure 4.3.

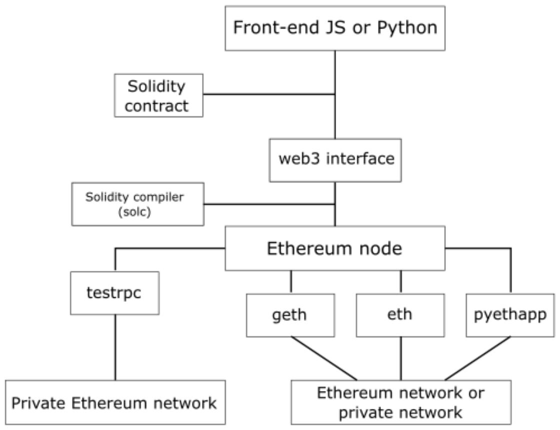

Figure 4.3: Blockchain project architecture scheme

4.3 Prototyping entities

As was stated before, creating local energy market and microgrid, which reside on top of existing infrastructure is the priority goal. The market should be transparent, effective and secure. Main feature list should include decentralization, locality, renewable energy production, sharing economy. Customers should have reliable and actual information on costs and consumption rates, market prices should be flexible and sustain the market status, local prosumers should be able to sell surplus to their neighborhood and make profit from it. All energy, generated by prosumers should be either spent on domestic needs, stored in the battery or sold on the market.

Blueprint of the entities interaction is shown on figure 4.4.

So our system should include prosumers and consumers entities with some sort of environment area, where all agent actions would take place - service provider. Service provider will be directly connected to blockchain and smart contracts. For our simulation this concept is necessary due to computations for different types of auction and transactions. Summing up, we should have houses entities that have basic information about a house-

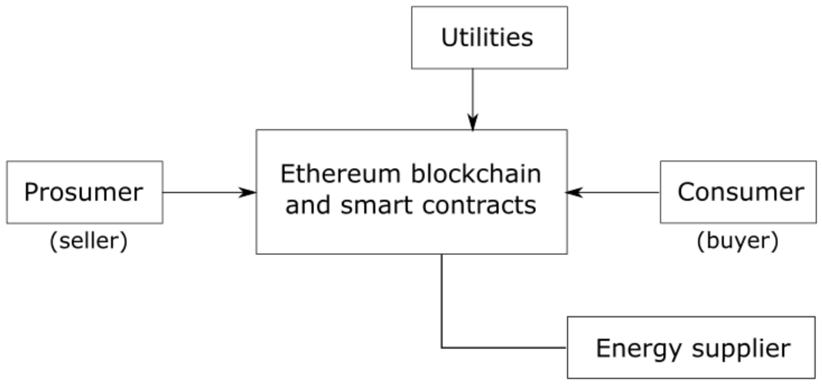

Figure 4.4: Proposed entities interaction

hold, ruler agents that will perform computations and make decisions about what should be done in the current hour. Prosumers and consumers are represented as houses (House). Each house have a ruler agent (RulerAgent) instance for data flow control and decision making. UML diagrams for House and RulerAgent classes are shown on figure 4.5.

Outer grid will also be present in a system as it should have a backup plan if our microgrid energy balance will be close to minimum. It also should be a tool, which prosumer agents will utilize for sell surplus energy, if no buyers are available at the moment. This is regulated with the battery capacity value. If it goes over some threshold value, agent will sell produced energy.

For testing the concept, simulation approach was chosen. Due to Ethereum limitations discretizing the timeline in simulation should not be high. Simulation will also be run on a rather high amount of days, so a lot of computations will be needed, so hourly simulation was tested. With hourly simulation it will run for a various ranges of days. Simulation will provided state results for each our. This state will inlcude hour, consumption and production data, monetary balance and energy balance. Also, using simulation approach we can perform learning procedure on agents or collect their experiences for future use.

Environmental parameters will be computed with various methods based on probability curves and internal parameters. Seasonal variables were taken from real data, and include summer temperature, production and consump-

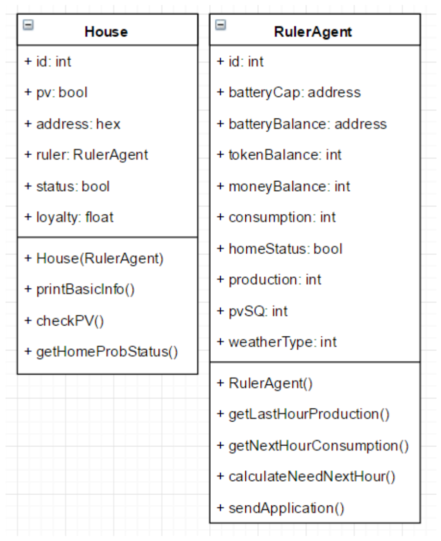

Figure 4.5: UML diagram of House and RulerAgent classes

tion profile values averaged throughout summer season. Data for production and consumption was provided by Kristoffer Tangrand. Before processing consumption data included values for a set of homes in Hvaler, Norway from 2013, and production data - values for solar panel installations in Scandinavia between 2013 and 2016.

Trading concept is the key in this system. Energy trading will be performed by prosumers and consumers, with the help of outer grid in extreme situations. Trading will be held using different types of auction, mostly with double auction mechanism. Service provider will handle auction process and send transactions to the blockchain.

UML diagrams for ServiceProvider class and Simulation are shown on figure 4.6.

Smart contracts are the interface to interact with blockchain and data. Smart contracts interactions are held after every auction instance, for fulfilling the need for energy for consumers or wish of selling for prosumers.

Thus, we should design our system and simulation with all these points in mind.

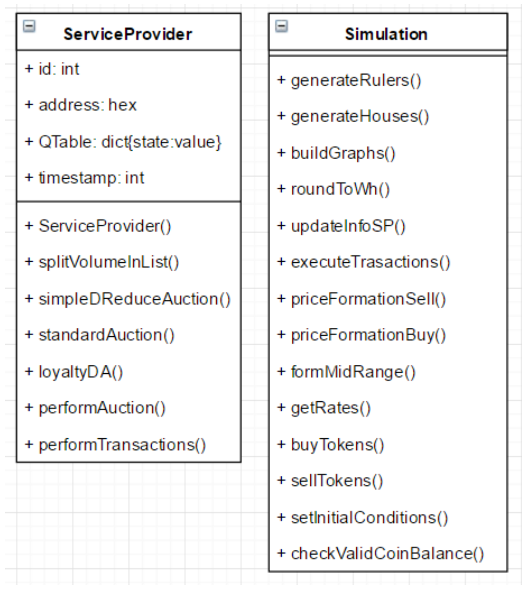

Figure 4.6: UML diagram of ServiceProvider class and simulation

4.3.1 Creating smart contracts

A contract in the sense of Solidity is a collection of code (its functions) and data (its state) that resides at a specific address on the Ethereum blockchain. Compiled solidity file consists of 2 contracts. *eToken* contract is used for internal currency for buying/selling energy. *EthEnergyMarketH* contract is used for market manipulation processes. *eToken* has data for contract owner address (which allows using administrative functions), for naming and symbol, decimal units, total supply of tokens (also known as initial distribution), sell and buy price for tokens, account balances and statuses. Functions for this contract include transferring tokens from sender to receiver, generating more tokens, buy and sell functions for the token currency and setting prices for sell/buy operations. *EthEnergyMarketH* contract data includes energy production, consumption, balance, money spend on energy and token price rates. Functions for this contract include money transferring, setting rates, sending coins, energy consuming and producing, functions to buy and sell from outer grid, initial energy balance distribution and getters for data in the contract. Designed contracts were kept as simple as possible in order to avoid issues with computations on EVM and for token ERC20 Token

Standard was used. UML diagram for contracts is shown on figure 4.7.

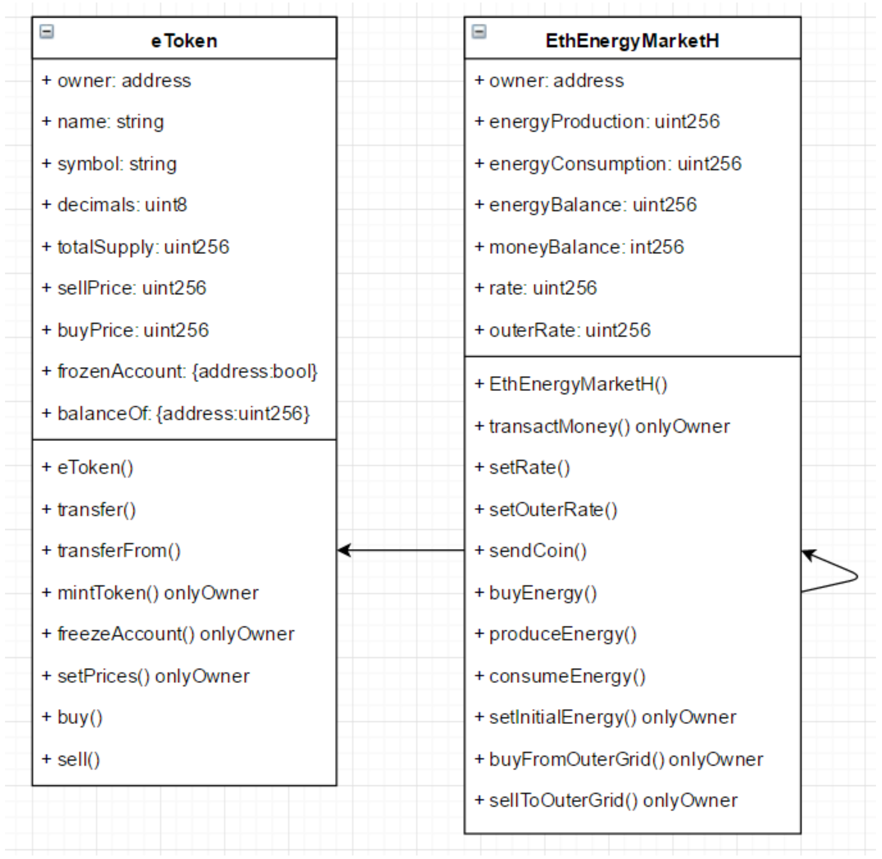

Figure 4.7: UML diagram of smart contracts

Creating final version of contracts was performed using Solidity. Usual work flow for contracts includes the following steps:

1. Start an Ethereum node (with geth/testrpc/ethersim)
2. Compile Solidity smart contract using *solc* or *py-solc*. Get back the binary.
3. Deploy compiled contract to the network. Get the contracts blockchain address and ABI - a file in JSON format that represent all of compiled contracts variables, events and methods that can be called.

4. Call the contract methods or data using *web3* API to interact with it, updating the state of the program and contract.

   After contracts were created we can move to the next step - integrate contracts and multi-agent system for our program.

4.3.2 Multi-agent system implementation

In the designed system there are 2 types of agents: ruler agents and service provider agents. Ruler agents interacts with different agents by means of service provider, which is presented as a "blackboard" for interactions and provides auction functions for ruler agents.

Ruler agents gets information form house entities and manipulate with it, forming bids or sells for service provider agent. As in this work we only investigate energy trade, only trading interaction are presented, though system can be expanded in favor of other types of interactions, like knowledge exchange (for learning algorithms), loyalty attribute or consumption/production rates. Ruler agents mostly operate with house information, regarding consumption calculation, production simulation and battery balance. They also have a copy of data for money balance, energy token balance.

Service provider agent is an information "aggregator", it possess all necessary information for trading and auctions. It tries to stabilize auction and takes no fees for this, though there is a possibility for functionality expansion for it, as it has address in Ethereum system, wallet, energy tokens and ether as any house entity in the system for market and auction operations. For this project it also possess table of successful auctions for further processing by learning algorithms. Functions for service provider will be described further in section 4.4.

In designed simulation ruler agents provide information for service provider every hour. The info contains buy/sell label and the amount of energy, which is to be bought or sold. For specific auction needs, service provider can request addition information from ruler agent, for example, loyalty value for loyalty double auction scheme. If agent will have sufficient energy for the next hour and no surplus (this value is corrected via threshold value) then the auction round will be skipped for current hour.

Consumption and production are the key functions for deciding if agent is participating in the auction round. The time frame of calculation is including previous hour, next hour and current hour timelines. For the close to real situation simulation we chose this type of time frame calculations. if we are observing hour t_i Consumption function refers to t_{i+1} hour and production function - t_{i-1}. Consumption-production principle for the current time t_i is shown on figure 4.8.

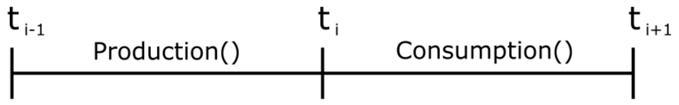

Figure 4.8: Consumption-production principle

Production function in the simulation is calculated every simulation round before production. This function simulates the amount of energy produced by house with PV, considering information about PV, size of the PV, weather type for the hour and a stochastic value. This value refers to the previous hour and is added to the total battery balance only next hour (the simulated hour). This is measured in (Wh). Multiplication by 100 is needed, concerning minimal energy unit for the system. Production calculation formula is shown in (4.1).

$$\begin{cases} 0 & if \;\; 5 < hour < 21 \\ 100 * PV_{area} * K_{weather} * K_{hour} & if \;\; 5 \geq hour \geq 21 \end{cases} \tag{4.1}$$

Consumption function is a sort of prediction function. This function is calculated after production calculation is performed. Ruler agent predict the energy need for the next hour. It uses information about home status, which is a probabilistic function. The total result is calculated with Monte-Carlo approach. It uses predefined probability graph and stochastic value. $randomHigh()$ and $randomLow()$ are random generators for high and low profile. These profiles were constructed using real data for consumption mentioned in section 4.3. Multiplication by 100 is also needed here for the same reason as for production calculation function. Consumption calculation formula is shown in (4.2).

$$\begin{cases} randomHigh * 100 & if \;\; statusHome = True \\ randomLow * 100 & if statusHome = False \end{cases} \tag{4.2}$$

Probability graph data is used for triggering Home/Away status for the ruler agent. It utilizes this info for calculating simulated consumption for the next hour. The probability graph is shown on figure 4.9.

Depending on the values of $(production{+}batteryBalance)$ and $batteryCapacity$ outer grid is involved. So we sell excess energy straight away before auction,

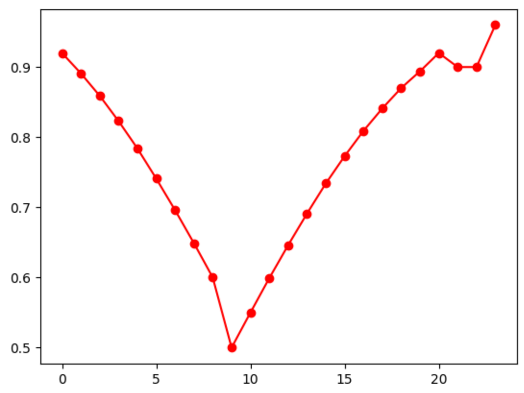

Figure 4.9: Home status probability graph

due to storage capacity. Alternatively we can use service provider storage, if we introduce battery for it.

After both values are computed, the balance after consumption is calculated, which later is processed and based on this information application for the auction is formed. After production value is added to the total energy balance and consumption for the next hour is calculated, balance after consumption value is observed. Depending on this value, the further action is taken. If this value is positive then this is trigger for "sell" action or "do nothing" action. If negative - trigger for "buy" action. The amount for selling is calculated with threshold value, which is showing the critical energy level for this house. Critical level is showing how much energy should be stored before auction process for the next day. It is usually assigned depending on average rate of consumed energy per hour and battery capacity of the observed household. If the value is under critical mark - then the household skips the auction round for the current hour and proceeds to the next hour of simulation. The amount for buying is calculated using values of consumption and battery balance. After this round of computation, we form an application as $(buy, amount)$ or $(sell, amount)$. Next step is adding a price for this application.

In order to form buy/sell transaction for auction we need a price to be formed. Function for price formation uses predefined price range as (min, max). Maximum price is close to the outer grid usual rate and minimum is chosen by market administrator. Also it depends on amount, which is being sold or bought. In order to form a price a Monte-Carlo approach and beta-function probabilistic distribution is used.

Beta distribution is a family of continuous probability distributions defined on the interval $[0, 1]$ parametrized by two positive shape parameters, denoted by α and β, that appear as exponents of the random variable and control the shape of the distribution. Beta distribution has been applied to model the behavior of random variables limited to intervals of finite length.

The Beta distribution is a special case of the Dirichlet distribution, and is related to the Gamma distribution. It has the probability distribution function (4.3):

$$f(x, \alpha, \beta) = \frac{1}{B(\alpha, \beta)} x^{\alpha-1}(1-x)^{\beta-1} \tag{4.3}$$

where the normalization, B, is the beta function (4.4):

$$B(\alpha, \beta) = \int_0^1 t^{\alpha-1}(1-t)^{\beta-1} dt \tag{4.4}$$

The probability density function for beta distribution through gamma function can be described by (4.5):

$$betaPDF(x, \alpha, \beta) = \frac{gamma(\alpha+\beta) * x^{\alpha-1} * (1-x)^{\beta-1}}{gamma(\alpha) * gamma(\beta)}, \tag{4.5}$$

for $0 < x < 1$, $\alpha > 0$, $\beta > 0$, where $gamma()$ is the gamma function, α and β are the shape parameters for beta function.

To form beta distribution we use list of prices, α and β parameters. After beta distribution is defined, we get sample from the parameterized beta distribution to compare with stochastic value. Depending on amount of energy and type of action (buy or sell), samples from list of possible prices is drawn in specific order. Beta function also redefined with new parameters α and β. These parameters were chosen in favor of usual market interactions.

- For "sell" action (4.6):

$$\begin{cases} if \;\; amount < midRange & reversed(priceList), \;\; \alpha = 2, \;\; \beta = 4 \\ if \;\; amount > midRange & priceList, \;\; \alpha = 4, \;\; \beta = 2 \\ if \;\; amount = midRange & priceList, \;\; \alpha = 8, \;\; \beta = 8 \end{cases} \tag{4.6}$$

- For "buy" action (4.7):

$$\begin{cases} if \quad amount < midRange & priceList, \quad \alpha = 2, \quad \beta = 4 \\ if \quad amount > midRange & reversed(priceList), \quad \alpha = 4, \quad \beta = 2 \\ if \quad amount = midRange & reversed(priceList), \quad \alpha = 8, \quad \beta = 8 \end{cases} \tag{4.7}$$

Probability density function (PDF) and cumulative distribution function (CDF) are shown on figure 4.10.

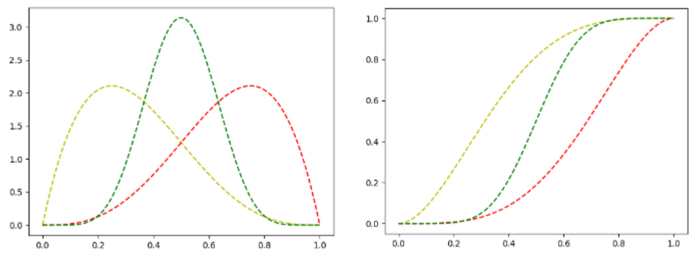

Figure 4.10: PDF and CDF for beta function

Price formation is a continuous operation. While price is not formed it iterates through prices list and generate a number with the help of Beta function. Then using Monte-Carlo approach it checks if price is acquired. When price is acquired is it assigned to previously constructed application and formed as $(action_type, amount, price)$ and sent further for service provider agent to perform auction process, which is described in section 4.4. After auction is complete and transactions are processed, info is updated, consumption process is performed and simulation moves to the next hour. The whole decision process and auction for agents is shown on figure 4.11.

Evaluation process include analyzing of the auction work. All performed transactions are noted and after evaluation have learning values in Q-table updated, according to states and action on the auction. For loyalty type of auction successful transactions get a loyalty value increase, rejected - get decrease. Returned transactions stays with starting values of loyalty. After this procedure, we need to handle rejected and returned applications.

Rejected status get an application if the price was too high for seller or too low for buyer. This process is done by auction cleaning process, which will be described in next section. If rejected application is with sell label, then it is checked if unsold energy is exceeding battery cap. If it is overflow then energy is sold to outer grid, else it stays in battery. If application is with buy label, ruler agent buys energy from outer grid.

Returned status gets an application with sell or buy tag, which was not cleared by auction, but there are not enough buyers or sellers respectively. The need is covered as with rejected status, but without loyalty decrease or negative reward signal for ruler agent.

Ruler agents interact with blockchain through service provider agent. Service provider also perform logging and gather data for analysis. Service provider interact with smart contracts using calls and transactions methods for interacting with blockchain. Interfaces were developed for each function of the contract in order to interact with contracts from simulation process. After each transaction method, ensuring that transaction is complete and data is stored, was implemented. It slows a simulation thread a bit, but provides error free flow. Call methods can only get the state or variable value from smart contracts, but it cannot change the state of the blockchain. Transact methods can change the state of the blockchain and perform computational functions in smart contracts.

Agents interactions with smart contracts are shown on on figure 4.12.

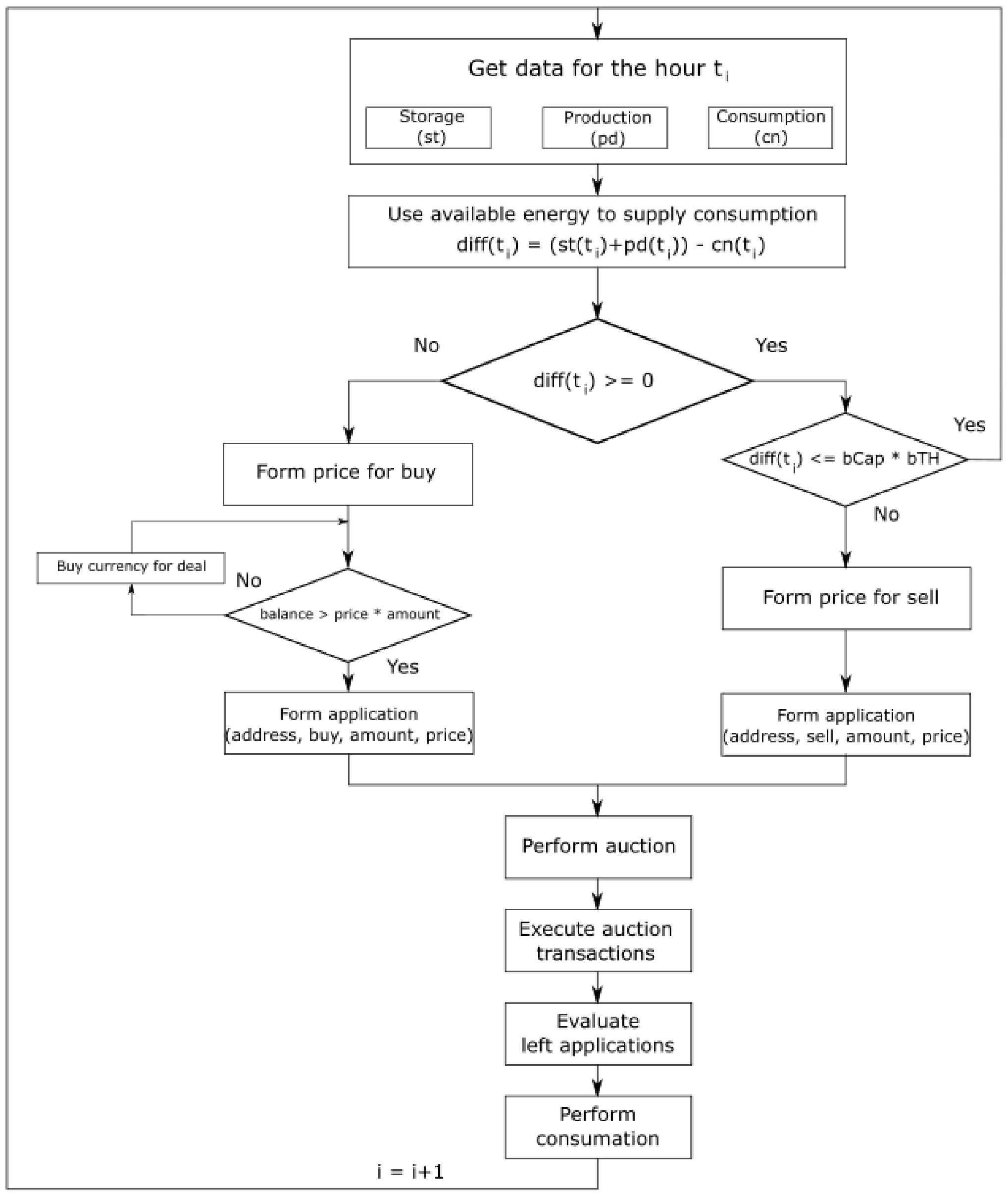
Get data for the hour t_i
Storage (st)
Production (pd)
Consumption (cn)
Use available energy to supply consumption
$diff(t_i) = (st(t_i)+pd(t_i)) - cn(t_i)$
No
Yes
$diff(t_i) >= 0$
Form price for buy
$diff(t_i) <= bCap * bTH$
Yes
No
Buy currency for deal
No
balance > price * amount
Yes
Form price for sell
Form application (address, buy, amount, price)
Form application (address, sell, amount, price)
Perform auction
Execute auction transactions
Evaluate left applications
Perform consumation
i = i+1

Figure 4.11: Agents decision process

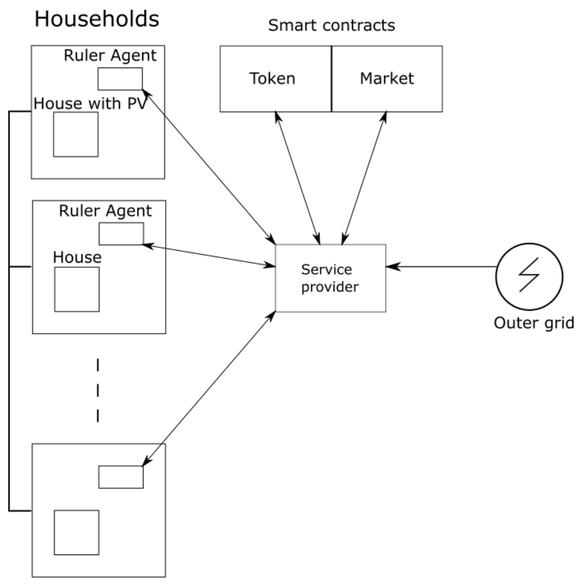

Figure 4.12: Agents and smart contract interaction

4.3.3 Utility functions

Utility functions include helpful service functions from *pyethereum* (Ethereum platform written on Python), which allows to manipulate with test chain, send transactions and manipulate with block information and address transformations. Simulation utility functions include:

- Current hour weather generation

  Due to the simulation is set for summer season, there are 3 types of weather for the simulation: sunny, cloudy, raining. Sunny weather profile implies lesser consumption profile and bigger production values. Cloudy weather profile sets to standard rates of consumption and production. Raining weather profile shifts consumption to higher and production to lower rate. The choosing of the current profile is done with Monte-Carlo approach. The probability distribution is described with the following cases (4.8) (4.9) (4.10):

$$\begin{cases} S & P('sunny') = 0.35 \\ C & P('cloudy') = 0.5 \\ R & P('rainy') = 0.15 \end{cases} \tag{4.8}$$

$$\begin{cases} S & Consumption * 0.95 \\ C & Consumption * 1.0 \\ R & Consumption * 1.3 \end{cases} \tag{4.9}$$

$$\begin{cases} S & Production * 1.3 \\ C & Production * 0.8 \\ R & Production * 0.2 \end{cases} \tag{4.10}$$

- Building graphs

  Building graph function allows to have a more convenient interface for plotting function in *matplotlib*.

- Logging

  Logging method is used for more readable output from EVM. It cuts the information about testrpc connections timestamps through ports.

4.4 Auction interactions

Auction is a market mechanism in which an object, service or set of objects, is exchanged on the basis of bids submitted by participants. An auction provides a specific set of rules that will govern the sale or purchase of an object to the submitter of the most favorable bid.

A call auction is an order driven facility which, in contrast with continuous trading, batches multiple orders together for simultaneous execution in a multilateral trade, at a single price, at a predetermined point in time, by a predetermined matching algorithm. In our work we investigated double call auction, as the most suitable for our simulation setup with fixed hourly time frame. Continuous action is not possible with this time discretization in a simulation.

Double auction is a process of buying and selling goods, when potential buyers submit their bids and potential sellers simultaneously submit their ask prices to an auctioneer and then an auctioneer chooses some price p that

Auction orders list						
Avg.price = p_avg, volume_sell = v_sell, volume_buy = v_buy						
Buy orders				Sell orders		
Buyer	Volume	Price		Seller	Volume	Price
0x1	v1	p1		0x5	v5	p5
0x2	v2	p2		0x6	v6	p6
0x3	v3	p3		0x7	v7	p7
0x4	v4	p4		0x8	v8	p8

Table 4.1: Auction orders list

clears the market: all sellers who asked less than p and all buyers who bid more than p buy at this price p.

Taking about double auction, we should mention three important points: market demand curve, market supply curve and equilibrium. Every time frame we have certain number of sellers and buyers that participate in auction. They constitute a auction order list which is shown in table 4.1.

Each record in this table is of form (Buy/Sell, Volume, Price). These applications are formed by agents based on environment and household information. No agent know the internal information about the others. After agents formed them, they send their applications to service provider, which perform auctioneer role.

If we plot this in a 2-D graph with Price and Volume as dimensions we get the supply and demand curves for the current time frame. Acquired graph is shown on figure 4.13.

The point of intersection between these 2 curves is the equilibrium point - a situation in which the supply equals the demand. In this point the perfectly liquid, frictionless market solution exists. This point is also the price P that clears the market. In article [31] similar type of price clearing was presented and was called The Quote-Accepting Policy.

In order to find this point and clear price we need to do natural ordering process with auction order list, which includes 3 steps:

- Order the buyers in decreasing order of their bid: $b_1 \geq b_2 \geq ... \geq b_n$
- Order the sellers in increasing order of their bid: $s_1 \leq s_2 \leq ... \leq s_n$
- Let k be the largest index such that $b_k \geq s_k$ (the "breakeven index")

Every price in the range $[max(s_k, b_{k+1}), min(b_k, s_{k+1})]$ is an equilibrium price, since both demand and supply are k. It is easier to see this by con-

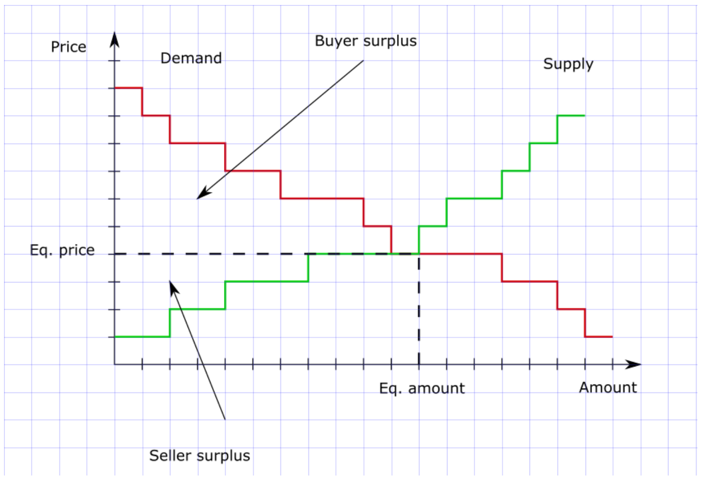

Figure 4.13: Supply-demand curves

	$s_{k+1} > b_k$	$s_{k+1} \leq b_k$
$b_{k+1} < s_k$	$[s_k, b_k]$	$[s_k, s_{k+1}]$
$b_{k+1} \geq s_k$	$[b_{k+1}, b_k]$	$[b_{k+1}, s_{k+1}]$

Table 4.2: Range of equilibrium prices

sidering the range of equilibrium prices in each of the 4 possible cases (note that by definition of k, $b_{k+1} < s_{k+1}$) shown in table 4.2 [32].

For the price formation for deals averaging mechanism was chosen as it provide such properties as Individual Rationality (IR) - no person should lose from joining the auction and Economic efficiency (EE) - the total social welfare (the sum of the values of all players) should be the best possible. This mechanism include finding k and set the price at the average of the k^{th} values: $p = (b_k + s_k)/2$. The first k sellers sell the good to the first k buyers. This is a standard mechanism, which was implemented first. Different mechanisms were studied with article about double auction schemes and their properties [33].

Other price formation mechanism includes household parameter, to which ruler agent have access to - loyalty. This parameter plays the role of reputation system within the community. It provides more profitable auction prices within equilibrium range for loyal participants and less profitable for those who have low rate of loyalty value. Loyalty - is the price formation and sorting parameter. Price formation for "loyal" type of auction include formulas 4.11, 4.12, 4.13.

$$w_{buy} = \frac{P_{buy}}{l_b + l_s} \tag{4.11}$$

$$w_{sell} = \frac{P_{sell}}{l_b + l_s} \tag{4.12}$$

$$P_{formed} = p_{min,sell} * w_{buy} + p_{max,buy} * w_{sell} \tag{4.13}$$

With these formulas we define what weight in price formation have seller and buyer. If seller have more loyalty value, the price will incline towards higher end in equilibrium price range. Else, price will incline to the lower end of range, making beneficial price depending on loyalty parameter.

Also order amount splitting to the least divisible value (100 Wh) take place, in order to exclude amount variable for transactions and provide maximal buyer-seller matching. After transactions are collected list is summed up. After transactions are executed, participated buyers and sellers get boost for loyalty parameter. Those, who does not get a supplier or buyer, but had appropriate price, get slight loyalty parameter increase. Those, who was cut from the auction by clearing price get decrease in loyalty parameter.

Marginal scenarios that should be mentioned are:

- No buyers, approved sellers

  Approved sellers sell their surplus, over battery capacity to outer grid at stable fixed rate. They do not get penalized in loyal type of auction.

- No sellers, approved buyers

  Approved buyers buy from outer grid at stable fixed rate. They do not get penalized in loyal type of auction.

Evaluation of the auctions can be performed in many ways. We use 2 types: evaluating the utility of the buyers and seller in equilibrium range and evaluating rejected and approved auctions. The evaluation results will be shown in Chapter 5.

In order to improve the utility and improve the effectiveness of the market we use machine learning technique called reinforcement learning and in particular simplified Q-learning algorithm.

4.5 Q-learning

Q-learning is a model-free reinforcement learning technique. Specifically, Q-learning can be used to find an optimal action-selection policy for any given (finite) Markov decision process (MDP). It works by learning an action-value function that ultimately gives the expected utility of taking a given action in a given state and following the optimal policy thereafter. A policy is a rule that the agent follows in selecting actions, given the state it is in. When such an action-value function is learned, the optimal policy can be constructed by simply selecting the action with the highest value in each state. One of the strengths of Q-learning is that it is able to compare the expected utility of the available actions without requiring a model of the environment. Reinforcement learning scheme is shown on figure 4.14. Additionally, Q-learning can handle problems with stochastic transitions and rewards, without requiring any adaptations. It has been proven that for any finite MDP, Q-learning eventually finds an optimal policy, in the sense that the expected value of the total reward return over all successive steps, starting from the current state, is the maximum achievable [34].

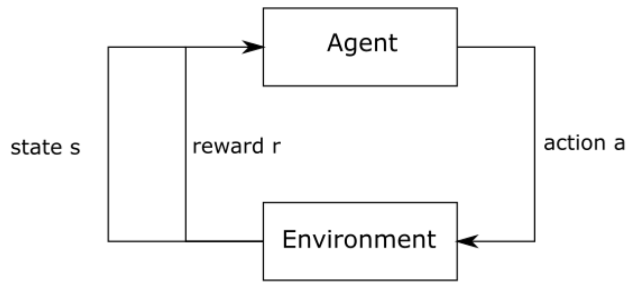

Figure 4.14: Reinforcement learning process

Basic entities of Q-learning algorithm are R-matrix and Q-matrix. R-matrix provide reward values for calculating Q-matrix. Q-matrix is a state-action matrix, which is used to record experience. Calculation of Q-matrix is performed using the formula 4.14.

$$Q(s,a) = R(s,a) + \gamma * max(Q(next\ \ state, all\ \ actions)) \tag{4.14}$$

Learning with this algorithm implies that in the model there is an agent, which explores environment. As it goes, it learns the value of different state changes in different conditions. These values uniform subsequent behavior of the agent. Once environment has been explored algorithm provides fast performance.

The general learning process in Q-learning algorithm implies going through the following steps:

1. Initialize Q-matrix
2. Choose action from Q-matrix
3. Perform action
4. Measure reward
5. Update Q
6. Go to step 2

In our work we use simplified modification of Q-learning algorithm, due to the states construction. Also one of the major problems - exploration vs exploitation - is not an issue in our case.

After investigating our environment and dependencies, we decided to use Q-table, with state-action records. Our environment does not posses multiple state sequences, so we have multiple one-state records. In order to describe it we used such parameters as current hour, battery level percentage, average consumption with predefined calculation window, price transferred from agent to auctioneer and buy/sell (0 or 1) action. The record scheme is shown in formula 4.15.

$$Q(state, action) = Q((hour, battery\%, avgCons), ([0,1], price)) \tag{4.15}$$

Due to the states construction Q-values are calculated with only reward value. Reward value is determined by evaluation of the results of auction. If auction was successful Q-value have a significant increase for the current state-action pair. If there is not enough energy provided for the auction but it's price was approved - Q-value have a slight boost. Else, if auctioneer returned price offer as not appropriate, Q-value receive a decrease. Also "loyal" auction have a multiplier for the reward calculation and have an impact on Q-value.

After predefined number of simulation iteration, discount factor is applied to all Q-values in the table in order to provide smother experience records.

This process take place for a certain period of time. For our simulation this takes 100 days. This ensures that there is enough experience records for the price evaluation decision process.

When learning process is finished, price formation process of an agent is using the acquired Q-table, instead of Bayesian probability principle. Current state of agent is analyzed and weighted euclidean distance 4.17 on high dimensional space is calculated. Weights for the distance were chosen empirically. Weight shows the "importance" of the value in a state 4.16.

$$d_{x,y} = \sqrt{\sum_{i=1}^{n} w_i(x_i - y_i)^2}, \ \ 0 < w_i < 1 \tag{4.16}$$

With this value we are looking for the record in a table. In ideal algorithm there should be a trade-off between euclidean distance and Q-value: Q-value should be maximized and euclidean distance minimized. In our implementation Q-value should be positive and euclidean distance minimized. Then the price of the previous most relevant record is chosen for the offered auction price for the auctioneer.

$$\begin{cases} hour & w_1 = 0.5 \\ battery & w_2 = 0.5 \\ avgCons & w_3 = 0.4 \\ action & w_4 = 1.0 \end{cases} \tag{4.17}$$

On the early stages of development principle component analysis (PCA) and unsupervised learning model was used. PCA was used to lower the feature vector (states) and then applied to Density-Based Spatial Clustering of Applications with Noise clustering algorithm (DBSCAN). Acquired model was used to analyze new feature vectors and assign them to appropriate price value. DBSCAN was chosen due to the fact that we do not know the number of clusters and it performs well on large number of samples. Though this approach good approximations, it was not as fast and efficient as weighted euclidean distance.

Chapter 5

Results

5.1 Tools and testing

5.1.1 Preparation phase

In order to create suitable environment for testing a virtual Linux machine was created in Oracle VM Virtual Box. The code was written in Python 2.7 and used built-in interpreter in Ubuntu 16.04 LTS to provide running program. *pyethereum* and *testrpc* are used to create Ethereum node and a private blockchain. *pyethereum* testing tools were tried in the first program developing iteration in order to check the viability. *testrpc* library provides fast and safe environment for testing contracts and account interactions in Ethereum environment. For the first versions some test contracts were written on Serpent language, and then moved to Solidity contracts. *py-solc* library provides compilation to binary code of Solidity contracts to Python version of Ethereum. *populus* is a python based framework focused on contract development and testing. *populus's* command line interface provides tools for compiling, testing, and deploying contracts. web3 add missing functionality to *populus* and allow to interact with Ethereum blockchain directly.

5.1.2 Simulation parameters

In order to test our entities and their dependences simulation process was created. The initial simulation depends on several parameters, which need to be specified. Parameters, that were used for the simulation are listed in table 5.1.

divNum - assigns the minimum energy value for the system. In order

Parameter	Default value
divNum	100
priceRange	[2,...,8]
batteryTH	0.2
auctionType	loyal
rulerCap	15000
numOfHouseholds	1
initEnergy	8000
initCoins	2000
daysToSimulate	1

Table 5.1: Simulation variables

to avoid rounding issues or use floating point calculation if financial areas this parameter should be defined. By default this parameter is equal to 100 (Wh). *priceRange* is a range in which the agents can pick prices for bids and sells in energy coins. The lower side is bounded by 2 as minimal price, maximum can be defined as lesser that price of energy in outer grid. This makes deals on the market profitable and buying or selling from outer grid is performed only in case of last resort. By default it is 9. *batteryTH* or battery threshold is a parameter involved in initial policy of agent. This parameter defines if it is appropriate to sell energy or to skip the trades. By default it is 0.1 or 10% of the total battery cap. *rulerCap* is a parameter which defines the battery capacity. In initialize functions it is used to assign maximum of the energy that can be stored for use or trading purposes. By default it is 15000 (Wh). *auctionType* allows to choose what auction can be run for the simulation. Available types are simple' for simple averaging double auction, loyal' for maximum fit double auction with loyalty parameter. By default loyal' is chosen. *numOfHouseholds* is a parameter for providing necessary number of houses into local grid. Ethereum uses it for providing necessary amount of accounts to testrpc chain. This parameter does not include service provider. By default it is equal to 1. *initEnergy* and *initCoins* are the values for initial energy and coins distribution. Service provider allocates given values to all houses in local grid, using smart contract functions. *daysToSim* is the parameter that defines number of days simulation will run. By default number of days to be simulated is equal to 1.

Other utility parameters are influencing performance of the system: *averageConsumtionWindow* is used for calculating average consumption for past hours and *decayFactor* is used for Q-learning procedure.

For providing simulation results we use data and plot it on graphs. For graph representation several methods were developed, such as *buildGraph()*. Libraries for building graphs include *matplotlib* and *seaborn*. *matplotlib* is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms and pyplot module provides a *MATLAB*-like interface for building graphs. *seaborn* is a Python visualization library based on *matplotlib*. It provides a high-level interface for drawing attractive statistical graphics. For easier usage of Ethereum functions, interfaces for every interaction were designed.

Adjusted simulation parameters, which we will take for all experiments by default:

- $initialCoin = [C_1, ..., C_n],\ \ C_i = 2000,\ \ \forall\ i$
- $initEnergy = [E_1, ..., E_n],\ \ E_i = randomize(500 - 80000)\ \ \forall\ i$
- $averageConsumtionWindow = 5$
- $decayFactor = 0.99$

Due to number of simulation parameters, stochastic nature of rates calculation and time needed for simulation to run, we decided to pick 4 experiments for our results. Based on these results discussions and conclusions will be carried in Chapter 6.

5.2 Experiment 1

First we are will take a look at base entities and their interactions within three days. We consider two households. One is prosumer - it has have solar panel available and other is consumer - without any sources of generation. During this experiment we will check production and consumption rates within three days, to check if generation and consumption patterns are right and behave as intended.

Changed parameters for the simulation cycle include:

- $daysToSimulate = 3$
- $numOfHouseholds = 2$

The acquired results is shown on figures 5.1 5.2.

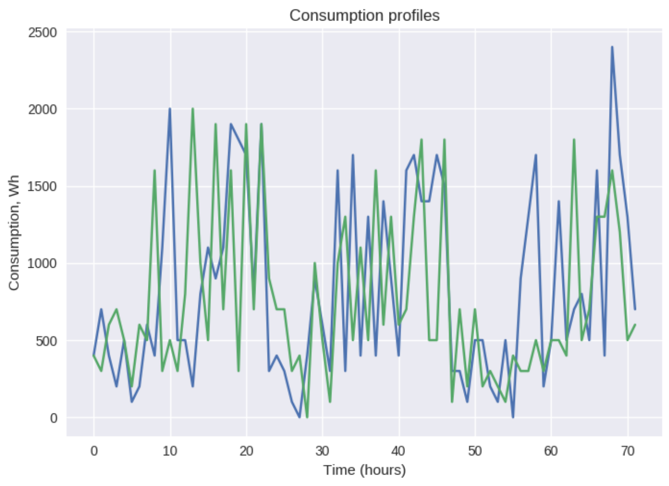

Figure 5.1: Experiment 1 - Consumption profiles

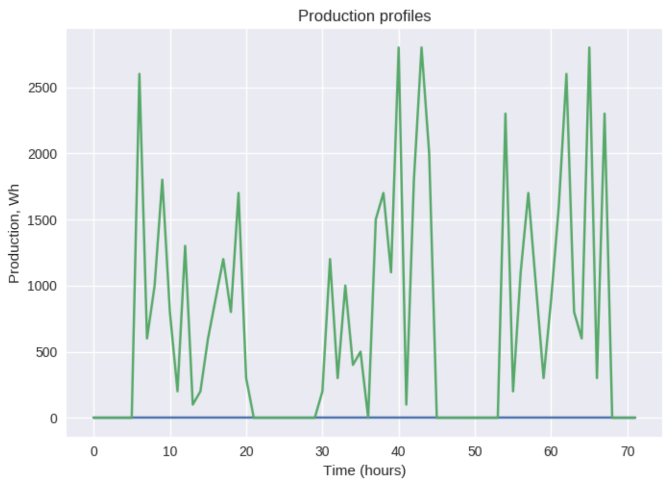

Figure 5.2: Experiment 1 - Production profiles

5.3 Experiment 2

The second experiment (and the next two) is directed towards overall market interactions, transactions held by auctions, balances (energy, coins, money) of households. In this experiment we are investigating simple averaging auction schemes and price formation. Auction is evaluated by calculating total utility for seller and buyer within equilibrium range. Also we get an idea of the carbon/solar energy print for the time period. Time period for this experiment is 1 month.

Changed parameters for the simulation cycle include:

- $daysToSimulate = 30$
- $auctionType = simple$
- $numOfHouseholds = 20$

The acquired results for battery balance dynamics 5.3, auction evaluation 5.4, coin balance 5.5, money balance 5.6 and price values distribution 5.7 are shown on figures below.

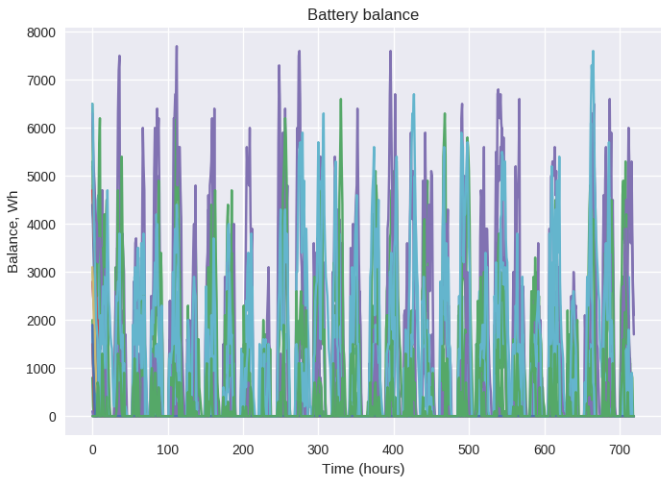

Figure 5.3: Experiment 2 - Battery balance dynamics

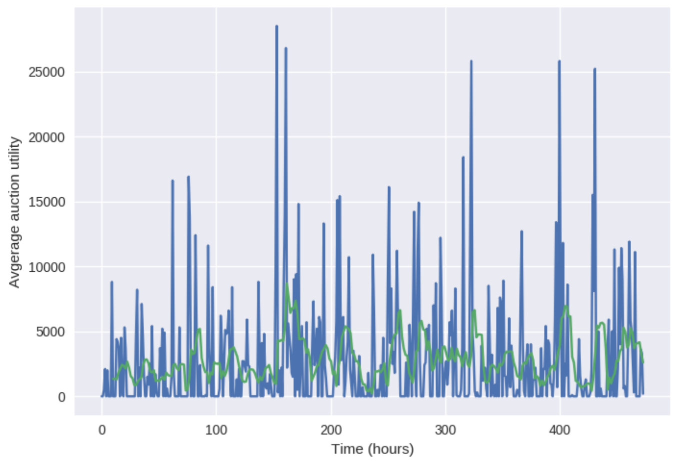

Figure 5.4: Experiment 2 - Average auction utility

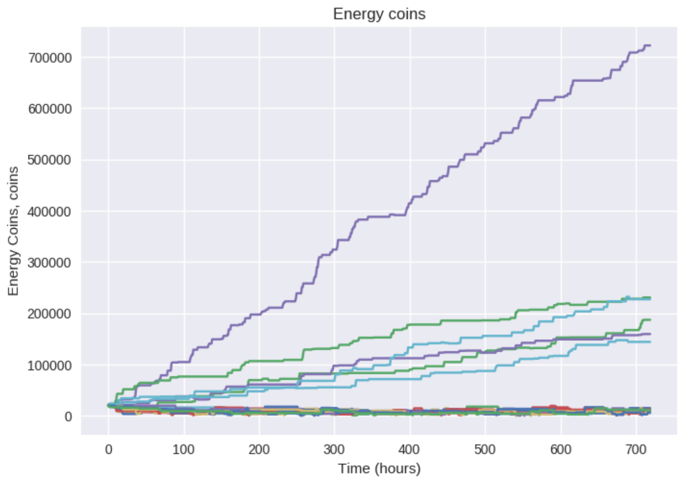

Figure 5.5: Experiment 2 - Coin balance

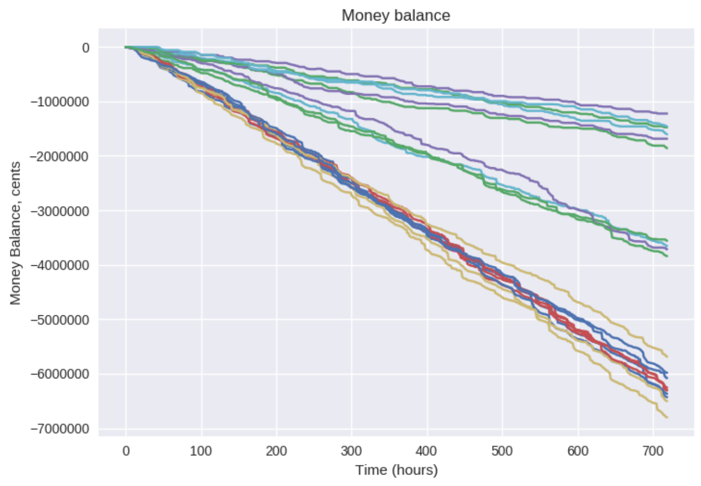

Figure 5.6: Experiment 2 - Money balance

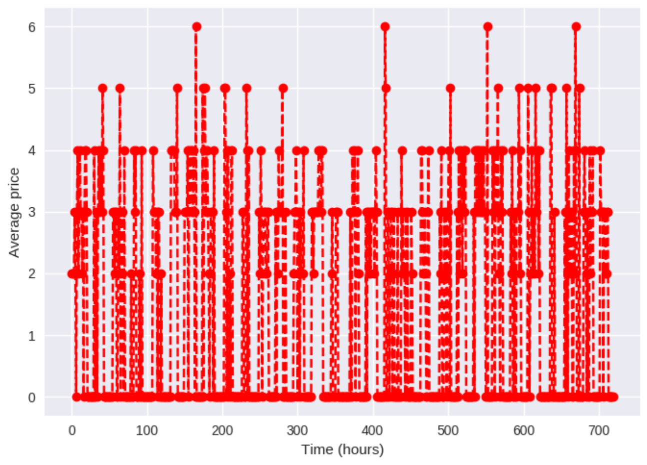

Figure 5.7: Experiment 2 - Price distribution

Total energy print for this experiment is shown in table 5.2.

Total energy	11271.4 (kWh)
Solar energy print	34.31%
Carbon energy print	65.69%

Table 5.2: Experiment 2 - Energy print

5.4 Experiment 3

The third experiment investigates the loyal auction scheme and the impact of loyal parameter. Though the importance can be clearly seen with longer simulation times and Q-learning enabled (see chapter 6) still this form of auction scheme provides solid results. In order to do it we check the same parameters as in experiment 5.3: market interactions, transactions held by auctions, balances (energy, coins, money) of households. Loyalty parameter serves as reputation indicator for the local energy market community. It is applied to price formation mechanism in double auction scheme. The higher amount of loyalty compared with other side of the deal you have, the more profitable the deal would be. Time period for this experiment is 1 month.

Changed parameters for the simulation cycle include:

- $daysToSimulate = 30$
- $auctionType = loyal$
- $numOfHouseholds = 20$
- $startLoyalty = 1.0$

The acquired results for battery balance dynamics 5.8, auction evaluation 5.9, coin balance 5.10, money balance 5.11, price values distribution 5.12 and loyalty values 5.13 are shown on figures below.

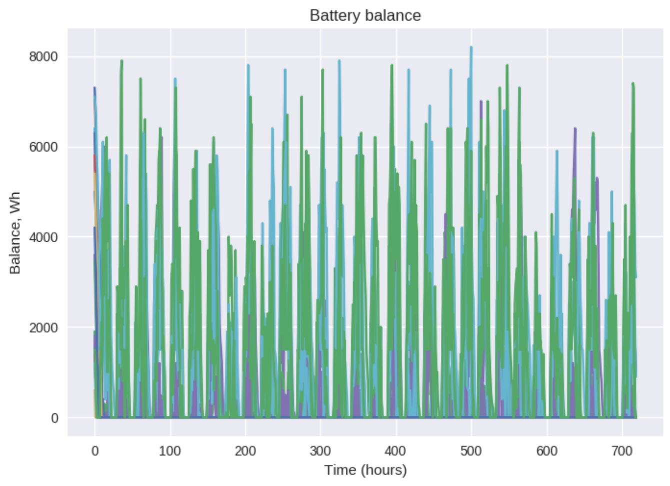

Figure 5.8: Experiment 3 - Battery balance dynamics

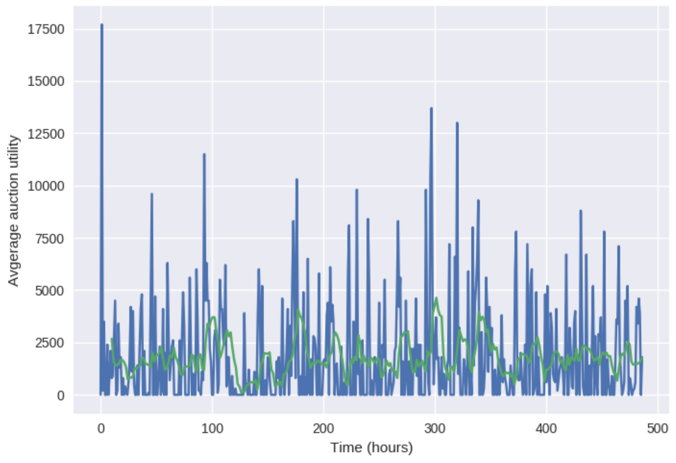

Figure 5.9: Experiment 3 - Average auction utility

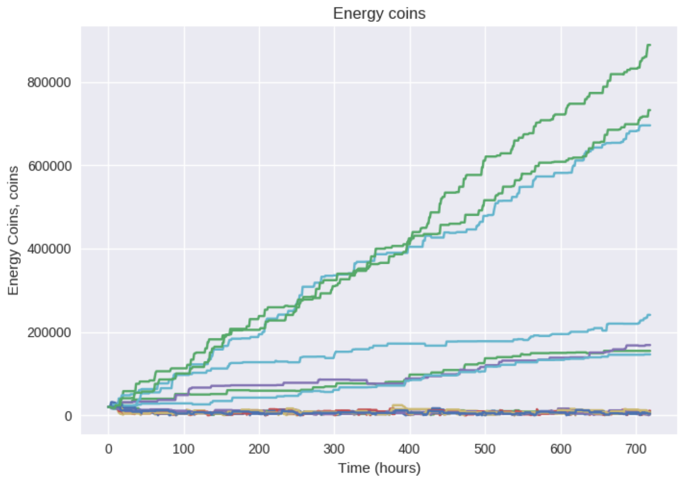

Figure 5.10: Experiment 3 - Coin balance

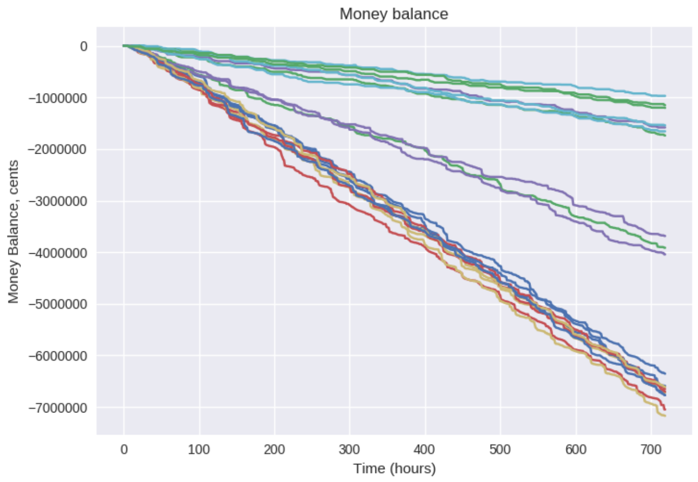

Figure 5.11: Experiment 3 - Money balance

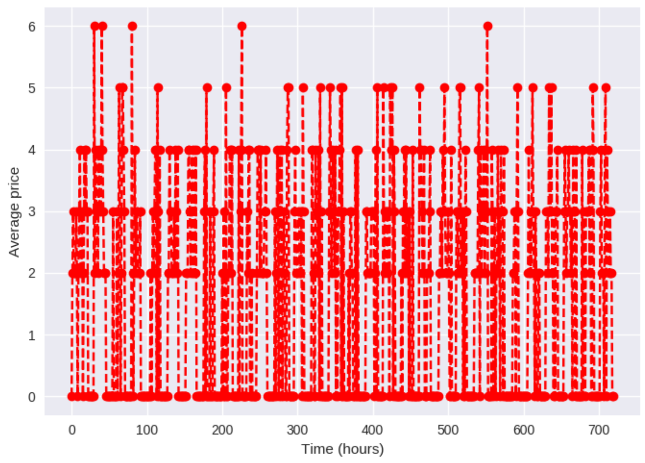

Figure 5.12: Experiment 3 - Price distribution

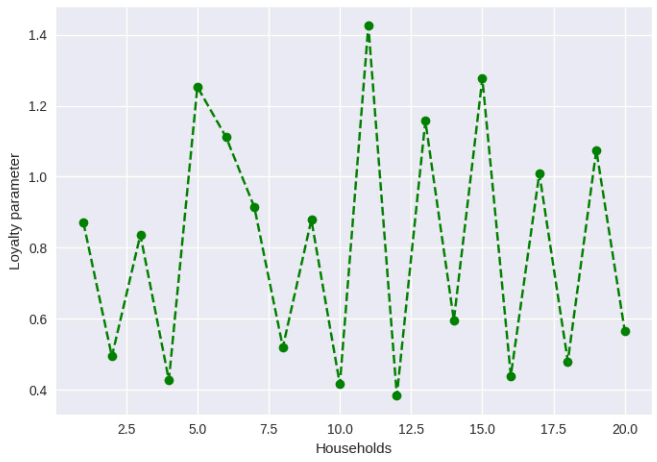

Figure 5.13: Experiment 3 - Loyalty values

Total energy print for this experiment is shown in table 5.3.

Total energy	11166.0 (kWh)
Solar energy print	57.98%
Carbon energy print	42.02%

Table 5.3: Experiment 3 - Energy print

5.5 Experiment 4

The last experiment investigates the loyal auction scheme in the long run and Q-learning impact on price formation. Total time period for this experiment is 170 days. Due to loyalty value has an impact on Q-learning process we check double auction with loyalty parameters.

Changed parameters for the simulation cycle include:

- $daysToSimulate = 170$
- $auctionType = loyal$
- $numOfHouseholds = 30$
- $startLoyalty = 1.0$

The acquired results for auction evaluation 5.14, coin balance 5.15, money balance 5.16, price values distribution 5.17 and loyalty values 5.18 are shown on figures below.

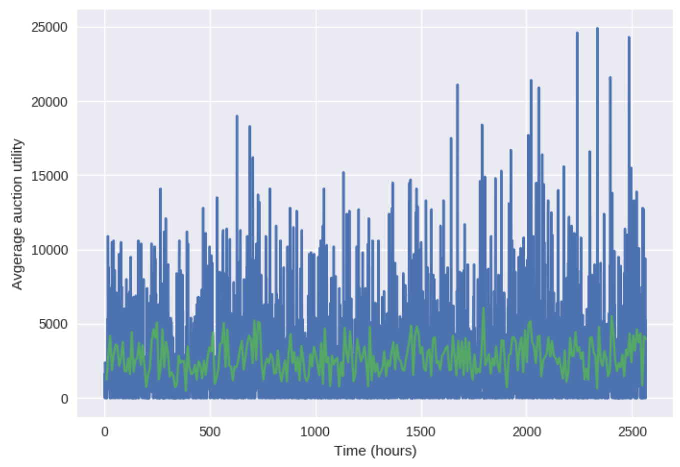

Figure 5.14: Experiment 4 - Average auction utility

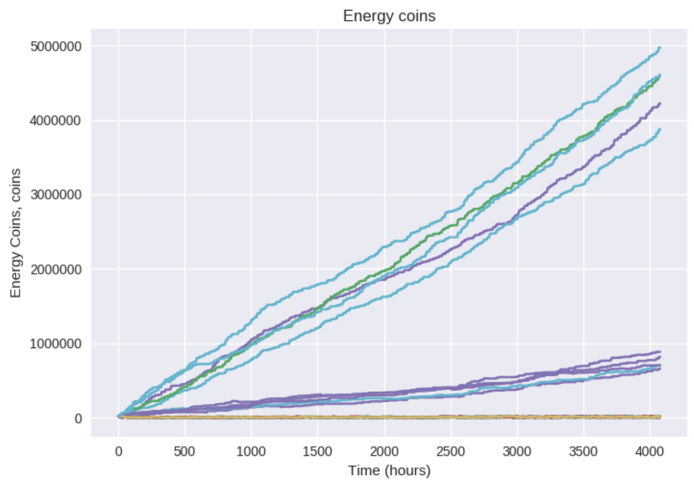

Figure 5.15: Experiment 4 - Coin balance

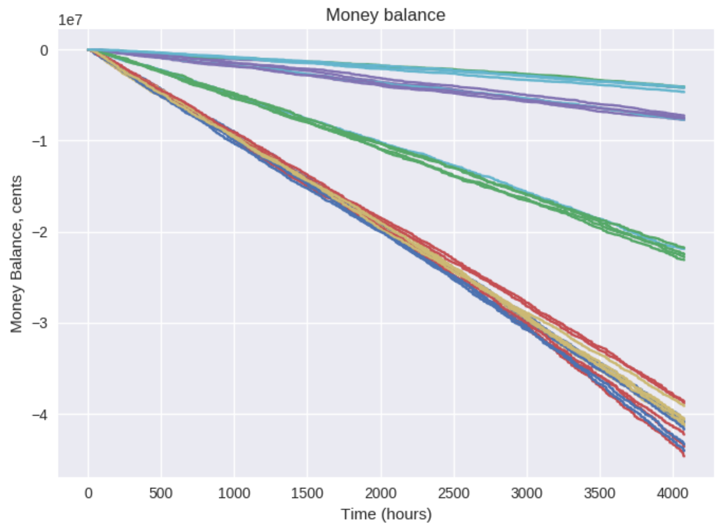

Figure 5.16: Experiment 4 - Money balance

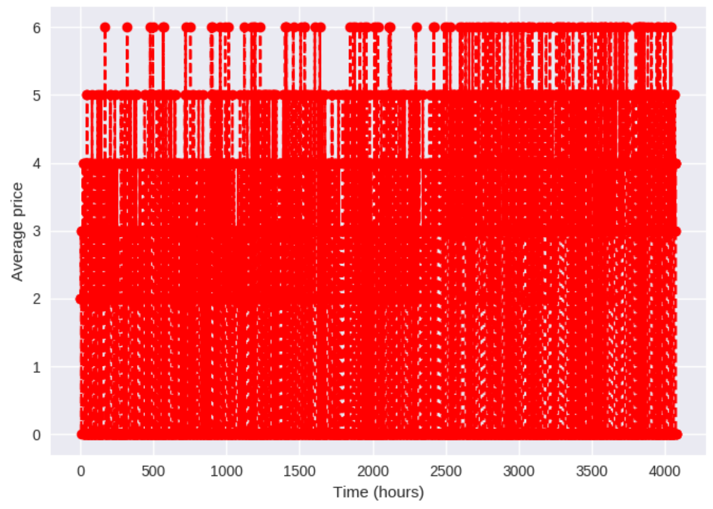

Figure 5.17: Experiment 4 - Price distribution

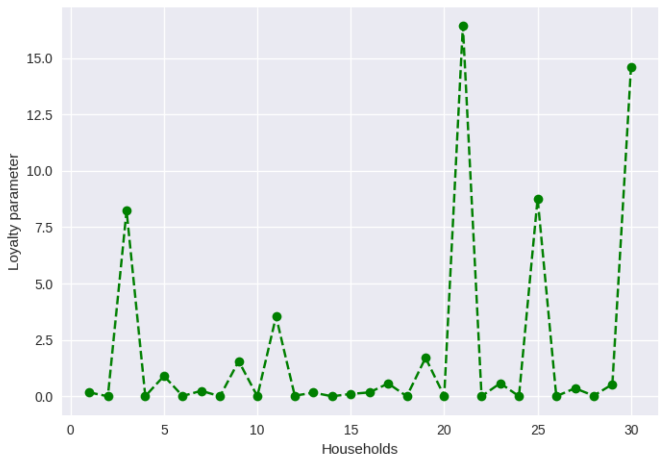

Figure 5.18: Experiment 4 - Loyalty values

Due to large number of participants, simulated days and Q-learning, experiment with proposed variables is highly demanding in terms of computations. Full simulation run took about 3 hours to complete.

Total energy print for this experiment is shown in table 5.4.

Total energy	56201.1 (kWh)
Solar energy print	41.78%
Carbon energy print	58.22%

Table 5.4: Experiment 4 - Energy print

Chapter 6

Discussion

6.1 Results interpretation

In the introductory chapters of the thesis we took a look of local energy market concepts, blockchain technology and multi-agent systems, investigated state-of-the-art of the current blockchain solutions and DApps for energy industry, and provided ways for implementing our simulation of local energy market with support of multiple agents and blockchain platform Ethereum.

The primary objective was to develop this kind of simulation, check if the concept is viable and investigate possible issues with it.

With chosen way of implementation we managed to design and build a simulation with benchmark for results evaluation. Such concepts as blockchain, smart contracts and DApps were tested and applied to the design of simulation. Ethereum platform provided support for our simulation via testrpc network and smart contracts were used to interact with blockchain. Solidity smart contract language was used to design and program smart contracts for Ethereum platform. Python framework *populus* and *web3py* provided necessary environment and functions for interacting, testing and deploying contracts to the Ethereum blockchain.

Simulation was created for testing of local energy market. Home agents and service provider entities were designed and programed for the simulation. After we set our simulation benchmark up with blockchain interfaces, utility methods and multi-agent system interaction, initial parameters were defined for soft policy of zero-intelligence agents. Mostly they were acquired empirically with various tests; some were designed as a derivative from the real data, provided by Kristoffer Tangrand 4.3.

In Chapter 5 we explored methods system implementation and simulation flow with various parameters.

The first experiment, presented in section 5.2, aimed at verification of the simulated data for consumption and production rates. Two households were used to show the difference: prosumer and consumer. The simulation ran for 3 days and have patterns similar to the real data, mentioned in section 4.3. As expected, we observed that consumption and production rate data is correlated with day/night cycles. The stochastic nature of the rate per hour is supported with weather changes and home status.

The second experiment, presented in section 5.3, had the goal to check the validity of the market and the auction process with simple price formation, based on Nash equilibrium and utility values of buyer and seller. This type of double auction has simple average price formation. As we observe the experiment results, we can point out that battery balance is as expected: third of the total households have enough production rate to have a surplus. They sell as surplus based on current battery percentage, regulated by threshold value in simulation. Due to relatively low surplus on the market and high consumption, even producers buy carbon energy from outer grid. This fact comes from negative balance of all participants. Though prosumers spent nearly 6 times less money, than consumers. Also they earn energy coins, depending on production rates and participation in the market. Auction utility was rather stable, moving average changes in a certain range. The most used price for transactions is 3-4 (energy coins per 100 Wh). One third of all used (consumed) energy in the system is solar energy.

The third experiment has the purpose of checking the viability of another scheme of price formation in double auction - so called "loyalty" double auction. In this type of auction we introduce a household parameter, accessible by ruler agent of each household - loyalty rate. The results of this type are shown in section 5.4. All parameters were duplicated from previous experiment, except for "loyal" type of auction. Average auction utility is slightly higher, than simple auction type. This results in more variance in auction and market - more participants are benefiting from it. As price is more liquid in this type of auction, more transactions were performed and the most used price stays as in previous experiment - 3-4 (energy coins per 100 Wh). Relatively, more that a half of households have high level of loyalty, which entail higher energy coins profit and lesser money spent on carbon energy. Solar energy rate used for consumption is more than a half in this experiment and is equal to 57%. This experiment possess only zero-intelligence agents with predefined policy and Bayesian probability applied on price formation, due to the time for the run of the simulation. Next experiment will utilize learning process and will use data, acquired from it.

Last provided experiment is shown in section 5.5. The main distinction from previous experiment is the number of simulated days - 170. This range of days is divided into 2 parts - learning period and exploitation period. Also there is an increased number of participants for higher load on a local grid. Loyalty variable is affecting the Q-learning table construction, not just a straight reward value, so we have more precision in calculations. Q-table is updated every 24 hours and multiplied by a decay factor. We need to do it as we do not have the decay mechanism in direct computation of Q-value. State for Q-table includes such parameters as hour, battery energy level, average consumption for past 5 hours. Action include buy/sell indicator and a price. We use Q-values and distance metrics to get the appropriate price. Average auction utility have a slight boost on exploitation time period, as agents possess information about all history records. Coin balance and money balance models saved the tendency on a exploitation time period and distribution of price have it's common range increased to 3-5 (energy coins per 100 Wh). With loyalty rates results we can observe that 6 households possess very high amounts of loyalty, compared to others. This happened because of initial trades, which boosted this value further. Solar energy print is at lower rate for this experiment compared to the previous one, but still is rather high - 41.78%. Also, increased number of participants (prosumers and consumers), as suspected, increase the overall stability of the market and increase the auction utility.

As we see learning agents produce slight boost compared to zero-intelligence agents with predefined policy, though computational and time expenses increased. Loyalty mechanism showed a lot of promise for this type of local market. It can be developed further in two separate ways: profits maximized or community stability prioritized.

Although we get rather stable and solid results, that shows us that the concept is working successfully, some weaknesses of the simulation were detected.

The overall weakness of simulation is that it depends on several initial parameters and stochastic values of defined entities. Also battery capacity indirectly influence money balance - only if surplus is overfilling the battery, the energy is sold to the outer grid. But, due to frequent auctions, high consumption profiles and low production rates this is rarely encountered. This fact actually is not disturbing the simulation as energy coins are the currency for internal asset - solar energy and outer grid is only a last resort tool to smooth extremal conditions. Though we can convert coin balance to money balance to see overall profit if needed.

As was mentioned in section 5.5, experiment with learning agents is highly demanding in terms of computations and time. More on other problems, we have encountered, concerning the design and development of the simulation and system is described in the section 6.2. However, we address these issues as a potential for improvement in Chapter 7.

6.2 Encountered problems

6.2.1 Blockchain issues

Many problems connected with blockchain technology outflow from a fact that blockchain technology offer more control and efficiency at the cost of higher personal responsibility. The most noticeable problems during development process were:

- Block formation time and transaction verification

  In our simulation every transaction was handled by 1 computation thread, no parallel threading or alternative computational nodes were introduced. This implies that time of handling transaction is increased significantly and if number of participating nodes is quite high, the transaction processing time will increase even more, in our case linearly. In order to improve this multi-threading technique can be used. Other way is to transfer computations and transaction handling to client nodes, which is the DApp design by default. In real world project, this is the best way to handle this problem. For the simulation it is not critical as we discretized time by hours.

  Also transaction verification adds value to total time spent, as we want to check every transaction. In real world scenario, this wont be a problem, due to parallel architecture or decentralized nodes computations.

- Usage of testrpc net

  Testrpc network is virtual network, created by a framework. It is faster and allows to ignore some fundamental principles like mining and gas. Though it is perfect for development, for real project geth node should be used. If we use private network, we will still be ignoring some concepts, but we will encounter issues like slower network processing and security. For public configuration we will use real money or ether in order to run our contracts and transactions. Also gas values should be reconsidered, as this is fundamental for Ethereum. This can be done after total checks and tests of smart contracts and web code on testnet configuration, as any fails will cost significantly. The example of typical failure this is The DAO, mentioned before. In the blockchain ecosystem, there is a constant risk of new types of attacks. This aspect of the issue has been studied much worse than any other aspects of this technology.

- Lack of standards

  Developing smart contracts is not an easy task. It requires to know all features of Ethereum platform as well as downsides of blockchain technology. Nowadays standards for smart contracts are in development phase. Standards are key to assessing the long-term benefits of open distributed systems. The lack of standards or successful application of technology in practice, indicates that it is still at an early stage of its development. Thus, in the case of blockchain, there is a risk that the implemented solution will not be effective.

  Though, it worth mentioning, that some companies are in stage of active standard development for blockchain technology and smart contracts. Such companies as Eris (now Monax) [35] are developing frameworks for developing specific type of contracts and expanding Solidity for more functionality and security.

- Customizability and scaling

  Scalability is often ignored, when we talk about blockchain, but it is rather important. As peer-to-peer technology, blockchain is high demanding structure. Given the extremely fast rate of data growth, the sheer data volumes accumulating after several years of operating a blockchain place high demands in terms of security, speed and costs. It is partially solved with the creating of intermediate nodes, which have only the part of blockchain and was described in original Bitcoin paper [14]. So we ensured that technology still have issues for solving tasks such as working with high-speed or volumetric operations, the need to record information in real time or store large amounts of data. Customizability is also a big issue, as ready to operate smart contracts, when deployed in real private or public net cannot be changed anymore. The only solution for it is to upload other contract for a different account or address.

- Usability problems

Due to lack of long-term experience with blockchain technology, some customers may not be willing to participate in this type of interactions. Classical database approach is still solid on the market, despite of its obvious disadvantages. Though, with right policies and incentives it is possible to show the effective side of blockchain technologies.

Other minor issue comes from public-private key cryptography principle associated with Ethereum and blockchain in particular. If keys are managed by users, losing private key will lead to total data loss for this account. This is not a big issue but states that this technology implies higher rate of responsibility from users.

The issue of lack of central authority is controversial. Usually in classic blockchain environment all controversies are solved by consensus, but sometimes it is preferable by solving with central authority. Though, this could be done using smart contract modifiers for some type of disputes.

- Crosschain interoperability

  This is an issue, concerning the whole architecture. If we have a structure of several private chains, we need a tool to operate between chains for the whole trade system to work. This technology is still under development for Ethereum, but already has a documented ways of operating. For now, there are 3 ways to perform cross-chain interactions: notaries, relays and hash-locking. Relays are the most functional, but are hard to implement. Hash-locking is the easiest one to implement, but has some restrictions. Notaries are the golden mean [18].

6.2.2 Learning agents

In our project we used simple variation of Q-learning algorithm. During learning period zero-intelligence agents were operating, storing the acquired knowledge. We used simple reward function, instead of R-matrix and Q-table is rather simple in terms of states. Dynamic programming can help achieve a lot of valuable and efficient results, but it requires solid learning model for learning algorithm. If enough time is given for the simulation and agents to learn, the acquired results will help to identify best policy in a given state situation, though some parameters need tuning.

Other issue include that the state definition is described with a lot of variables. Reducing the number of variable in state will allow faster and more effective learning procedure. For this, aggregating algorithm can be used, though the quality of learning procedure can be an issue.

Another way of handling the process of learning is to use artificial neural network in combination with Q-learning algorithm [36]. Neural networks can be used as approximators. For building these approximators, the loss function $L_i(\theta_i)$ (where θ_i - parameters of approximation) should be consecutively optimized [37].

6.2.3 Smart contracts issues

Concerning smart contracts developing for this project, we have encountered numerous problems. Almost all of them were avoided, but still contracts should be tested in real environment. Those problems can be divided in several categories:

- Re-Entrancy

  It is not recommended to perform external calls in contracts, due to safety and integrity issues that can be caused by these calls.

- Send can fail

  When transacting money or tokens, contract code should always have trow mechanisms in all branches if function is going to fail.

- Loops can trigger gas limit

  Looping over state variables is not a safe, due to the fact that variable can change and can grow in size, which will make gas consumption hit the limits.

- Call stack depth limit

  It is not recommended to use recursion, and all precautionary measures must be observed for any call can fail if stack depth limit is reached.

- Timestamp dependency

  Timestamps should not be used in critical parts of the code, because miners can manipulate them.

- Floating point math

  As was stated before float/double type is not present in Solidity, due to possible rounding errors with financial data. In order to avoid it decimal part can be used and stored separately.

This list does not cover everything that should be kept in mind when developing smart contract. Smart contracts actually operate with real funds in public networks, so code should be covered with unit test, code reviews should be part of the development process and code audition should always take place. Smart contracts and blockchain technology is relatively new, a work in progress technology, and need standardization of contract code, which will eventually appear later.

6.3 Real-world appliance

In this section we discuss the results of this thesis in appliance to the real world scenarios.

As we can observe from our results this type of local energy market with blockchain technology support can be used as a solution for microgrid, without removing existing infrastructure. Though, some aspects of the proposed system should be changed. Due to the fact that we checked concepts with simulation, the structure of the solution should be changed. We should reconfigure it with new additions and tweaks: smart meters are needed to record the quantity of energy produced, blockchain software is needed to store transactions between the neighbours, and smart contracts are needed to carry out and record these transactions automatically and securely, user interface for more control and managment over the whole process.

First, for smart energy generation and transforming it to the solar coins we need PV systems and smart meters for home indicators control. These would register energy production and consumption in most seamless way for interacting with other entities in the system. This can be used with Arduino platform.

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - for our case it is generated solar energy - and turn it into an output - update information in a blockchain. This is done by sending a set of instructions to the microcontroller on the board. Combined with smart meters it could be a very powerful technology. This type of interactions can be performed autonomously, without user interactions adn can be integrated in existing microgrid environment. Alternative for this is RasberryPi computer, which can be nicely integrated into the system.

Secondly, for user interface the previously described DApp concept can be used. The project can be deployed as a JavaScript web service with Ethereum platform and smart contracts support. Similar technique is used in TransactionGrid project. This enables users to take control over their expenses, see the production and consumption rates, energy balance and overall community status on-line. Smart meters would measure the amount of energy produced and consumed, while energy-trading activities and cryptocurrency payments would be controlled by smart contracts and executed through the blockchain. Also it is possible to create a mobile application, with the ability to switch between manual control and automated agent control with predefined policies and rules for it.

For blockchain technology Ethereum covers practically everything, except for consensus protocol and cross-chain interactions. Now consensus protocol is not energy-efficient and require a lot of computational power. When "Casper" protocol is introduced, this problem will be solved. Cross-chain interactions are now in testing phase for Ethereum and soon will be available. For smart grid infrastructure this is way-to-go solution: connect all microgrids in one infrastructure, but for small autonomous microgrids this technology is not necessary.

Smart contracts should also be updated. Most of logic and computations should be transferred to smart contracts mechanism. Though it would increase computational costs, it would benefit the local energy market community with the ability to explore all contracts code and logic and make system more stable and transparent. Also a framework for resolving issues should be created for preventing failures and errors.

Chapter 7

Further development

7.1 Possible improvements

Further work should be concentrating on real-world appliance and solving issues, encountered in the project and described in section 6.2. Other way is to upgrade current simulation solution with other market construction approaches and larger scale simulation. Main emphasis should be done on the following points:

1. Smart contract improvements

   Smart contracts can be improved by moving calculation and logic inside them. For our simulation they only store data and states, perform simple calculations and provide interfaces for interacting with blockchain. In a DApp all logic is stored within a contract. Though it could take a lot of computational resources, further developments of Ethereum platform would be able to handle it. This will provide transparency and control for every participant in a system. Security for smart contracts should also be updated to the latest standards. It is relatively easy to test and prevent failures on non-heavy computational and logical contracts, but with increasing complexity, the testing and security difficulties will arise exponentially.

2. Cooperative economics model and larger time frame

   Our market model and auction interaction implies growing personal profit for each agent regarding other participant by price changing policies. But it can be done other way, if we pose community profit in favor. This scheme is developed by TransactiveGrid project [38] [39].

Address	Production (kWh)	Consumption (kWh)	Community credit/debit (kWh)	After settlement (kWh)	Calculation	Balance after round (eCoins)
0x1	0	50	-50	0	-50	-50
0x2	75	20	55	0	50+5	+55
0x3	0	20	-20	0	-5-15	-20
0x4	100	60	40	5	15+20	+35
0x5	10	30	-20	0	-20	-20

Table 7.1: Community market model round

Community members produce their own renewable energy, and incentive each other to purchase any excess, creating a local marketplace. The community can exist in a state of credits and debits to one another, without needing the instantaneous transactions - larger time frame for settling. A community comes together and agrees that their solar production and consumption will constitute a bank. The smart contracts on the blockchain record the transactions, and credit or debit member accounts. At that quarterly time period there will be a netting of accounts. The necessary payments will be issued at that time, and the community members can come up with their own protocol of payments. Several rules can be applied to this scheme:

- Prosumer production rate defines selling order
  The more prosumer produce, the higher is his priority in settling transactions.
- Common selling/buying rate
  For example 1kWh is equal 1 energy coin. Price rate changing can be done through consensus.
- Reputation system
  Each successful transaction increase reputation value. If no penalties are in the system decay factor may be introduced.

The example of the market regulation round is shown in table 7.1.

This process can be improved, but the trusted and transparent ledger is the foundation for cooperative economics.

Since no money is required to be exchanged at the moment of a transaction, the system incentives a greater degree of trust that builds over time. This mutual credit system has been tried and tested in communities worldwide over time without blockchain [40]. If trust is present in the community, there is no need for an additional reputation system to penalize people who break rules. However, for larger communities there will be a need for a reputation system that will limit or block the participation of members who do not pay their amounts or break rules.

Also, we can apply and test different auction schemes. In our work, we have explored call auctions, as we have hour time frame. Different forms of continuous auctions can be also viable. Different auction types can be investigated, depending on prevailing factor: social effectiveness or revenue.

3. Code improvements

   During developing our simulation we used Python environment, as it provides fast and efficient prototyping of the system. In order to boost performance of the simulation, C++ should be used. If programed right it will utilize memory handling features, fast-handling data structures and algorithms for efficient resource allocation and improved performance overall. Threading should also be used, as our simulation was single-threaded and all operations were handled successively. Some optimization techniques can be applied to our program such as code and compiler optimization.

4. DApp, client interface

   A rough draft of code and system improvments were mentioned in 6.3. Main idea here is to change the format of the system to DApp. A blueprint of an application are proposed in section 6.3 and subsection 3.2.4. This is decentralized application that allow to check the state of the contracts and interact with them by transaction. This is done as a web service with various information about current status, like consumption/production rates, energy balance and coins. Also manual and auto profiles should be added for the user convenience. Auto profile will choose the most appropriate price formation strategy in terms of market stability and production/consumption difference handling.

5. Integrate and test system

   Testing system in real microgrid is an important matter. In this thesis we showed that in theory that the simulation works, but we lack experience with this technology and real grid. We need smart meter technology, local grid and our system integration, which was described in section 6.3. Once we have the system set up, real time and real load tests should be held.

This thesis provides an insight and example of what is the general direction of improvement and what issues should be solved. Solving mentioned problems will make a solid and robust system in terms of local market economy, transaction handling transparency and stable community economical model.

7.2 Future of DApps and blockchain solutions in energy sector

7.2.1 DApps and Ethereum

There are a lot of different types of DApps already. Area of application is really impressive: social networks, decentralized super computers, prediction markets, distributed decision-making, games, insurance, renting, transportation. Smart contracts when deployed can provide many different use cases. Evidently, Ethereum has the vast potential to reach many parts of society. Further developments by the Ethereum Foundation like Casper [24] and Sharding concept [41] allows for the scaling of the platform, ensuring sufficient capacity to manage future growth in users and transactions. DApps is a new concept which seems to have lots of applications so Ethereum will likely to grow with time. The initial targeted areas will be academia, startups, venture capital firms, tech companies, companies already experimenting with Ethereum and blockchain, followed by the wider community. This has the potential to dramatically increase the adoption rate of Ethereum platform.

7.2.2 Other areas of application

There are a lot of areas, in which blockchain is perfectly fit. For example: digital securities trading, foreign exchange, data storage, delivery of digital content, digital identity, proof-of-ownership. Other newly developed areas that worth mentioning are automobile leasing, insurance and IoT.

Automobile companies ensure their clients that automobile leasing is an easy and clear procedure, but on practice it can turn a complicated matter. The main problem which is encountered by leasing chain participants is that with a single supply chain, the system for sharing the accompanying information remains fragmented. However, blocking technology allows you to monitor, access and share coherent and up-to-date information throughout the life cycle of the vehicle, regardless of its location. With blockchain automobiles can have a personal registry with concerted and comprehensive information. This will improve service, warranty and recall costs. While going through supply chain information will be constantly updated in most efficient and transparent way. With smart contracts and consensus mechanisms security, integrity and validity of the whole system is preserved. Also transaction costs and time cost of operations, connected with intermediary agents, will be significantly reduced.

Other perspective industry of appliance, that is developing nowadays is blockchain insurance and IoT.

In insurance context the application of smart contracts, in conjunction with the blockchain, they offer a number of advantages: they automate the satisfaction of claims on insurance payments, offer a reliable and transparent payment mechanism for all parties and can be used for individual terms tuning of each individual contract. For example, in the case of an automobile accident, a smart contract can guarantee that the payment will be made only if the car is repaired in a preselected technical service. It also guarantees increased transparency and trust from customers through decentralization, automation of reconciliation and data validation and reduced costs on all stages.

IoT application of decentralized technologies will allow machines and electronic devices to obtain their own insurance policies, registered and managed through smart contracts in the blockchain network. This will automatically determine the amount of damage caused to them and initiate the repair process and insurance payments.

Chapter 8

Conclusion

The objective of this thesis was to explore and solve issues with local energy markets, multi-agent systems and blockchain technology. We managed to combine these concepts and create a simulation of local energy market with agents and blockchain support, based on the main results and discussion. With this simulation we were able to analyze different market situations, prosumer and consumer trade interactions, double auction mechanisms and justify the incentive for prosumers and consumers to participate in local energy market. Blockchain technology and Ethereum platform was deeply analyzed, several solutions for our problem were tested and described.

Through the thesis we constantly ensure that blockchain technology has all premises to be adopted for local energy trading: lower transaction costs, market transparency, rise of the prosumer role. Potential and disrupting effect are enormous. Though, when it comes to blockchain applications for the energy sector, it's early days yet. Computational, scalability and security issues are still an issue for blockchain in this area. In order to apply blockchain technology to local energy markets and microgrid systems, several points should be covered, including: developing user-friendly applications for manual and automated monitoring and trading, moving to lesser-computational verification processes, applying hybrid model of blockchain and centralized authority model. With all these additions and with further developments of Ethereum platform and blockchain technology, blockchain and local energy markets can overcome all minor issues and become dominant technology in this field.

Bibliography

[1] SmartIO, Bernt Bremdal, Pol Olivella-Rosell, and Jayaprakash Rajasekharan. Empower - d6.4 - technical specifications for software development. H2020 project, September 2016.

[2] Falti Teotia and Rohit Bhakar. Local energy markets concept, design and operation. *SMIEEE Center for Energy and Environment*, 2017.

[3] Smart grid. `https://en.wikipedia.org/wiki/Smart_grid`, 2017. Online; accessed: 2017-04-30; Background, features, technology.

[4] Bernt A. Bremdal, Pol Olivella, and Jayaprakash Rajasekharan. Empower - a network market approach for local energy trade. May 2017.

[5] Solarcoin. `https://en.wikipedia.org/wiki/SolarCoin`, 2017. Online; accessed: 2017-04-30.

[6] PwC. Blockchain - an opportunity for energy producers and consumers? Technical report, PwC global power & utilities, 2016.

[7] An energy blockchain for european prosumers. `https://bitcoinmagazine.com/articles/an-energy-blockchain-for-european-prosumers-1462218142`, 2017. Online; accessed: 2017-04-30.

[8] Smart solar. `http://www.ideocolab.com/prototypes/smartsolar`, 2017. Online; accessed: 2017-04-15.

[9] SmartIO, Iliana Ilieva, Bernt Bremdal, Pol Olivella (UPC), Terje Nilsen, and Stig Odegaard Ottesen (eSmart). Empower - d6.1 - market design. H2020 project, September 2015.

[10] SmartIO, Iliana Ilieva, Jayaprakash Rajasekharan, and Bernt Bremdal. Empower - d6.2 - prosumer oriented trade: Exploration of theoretical

and practical solutions for prosumer oriented trade. H2020 project, January 2016.

[11] SmartIO, Pol Olivella-Rosell, Jayaprakash Rajasekharan, Bernt Bremdal, and Iliana Ilieva. Empower - d6.3 - trading concept development. H2020 project, July 2016.

[12] Salman Kahrobaee, Rasheed A. Rajabzadeh, Leen-Kiat Soh, and Sohrab Asgarpoor. Multiagent study of smart grid customers with neighborhood electricity trading. *Electric Power Systems Research*, 111:123–132, March 2014.

[13] Salman Kahrobaee, Rasheed A. Rajabzadeh andLeen Kiat Soh, and Sohrab Asgarpoor. A multiagent modeling and investigation of smart homes with power generation, storage, and trading features. *IEEE Transactions on smart grid*, 4:659–668, June 2013.

[14] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *The Cryptography Mailing list*, 2008.

[15] Antony Lewis. A gentle introduction to blockchain technology. *BraveNewCoin*, 2016.

[16] Michael Crosby, Nachiappan, Pradhan Pattanayak, Sanjeev Verma, and Vignesh Kalyanaraman. Blockchain technology beyond bitcoin. *University of California Berkeley*, 2015.

[17] Siraj Raval. *Decentralized Applications. Harnessing Bitcoin's Blockchain Technology.* O'Reilly, 2016.

[18] Vitalik Buterin. Chain interoperability. September 2016.

[19] Ethereum project. `https://www.ethereum.org`, 2017. Online; accessed: 2017-02-23.

[20] Gavin Wood. Ethereum: A secure decentralized generalized transaction ledger. Yellow paper, March 2017.

[21] Vitalik Buterin Karthik Gollapudi. Ethereum - next-generation smart contract and decentralized application platform. `https://github.com/ethereum/wiki/wiki/White-Paper`, 2017. Online; accessed: 2017-05-01.

[22] BitFury Group. Proof of stake versus proof of work - white paper. September 2015.

[23] Alyssa Hertig. Where's casper? inside ethereum's race to reinvent its blockchain. `http://www.coindesk.com/ethereum-casper-proof-stake-rewrite-rules-blockchain`, January 2017. Online; accessed: 2017-04-29.

[24] Vlad Zamfir. Introducing casper the friendly ghost. `https://blog.ethereum.org/2015/08/01/introducing-casper-friendly-ghost`, August 2015. Online; accessed: 2017-04-28.

[25] Sebastian Beer and Hans-Jurgen Appelrath. A formal model for agent-based coalition formation in electricity markets. *IEEE PES Innovative Smart Grid Technologies Europe*, 4, October 2013.

[26] Steven Gjerstad and John Dickhaut. Price formation in double auctions. *GAMES AND ECONOMIC BEHAVIOR*, 1995.

[27] Mikko Berggren Ettienne, Steen Vester, and Jorgen Villadsen. Implementing a multi-agent system in python with an auction-based agreement approach. *ProMAS 2011*, 7217, 2011.

[28] Mahesh S. Narkhede, S.Chatterji, and Smarajit Ghosh. Multi-agent systems (mas) controlled smart grid a review. *International Journal of Computer Applications*, 2013.

[29] Cade Metz. The biggest crowdfunding project ever - the dao - is kind of a mess. `https://www.wired.com/2016/06/biggest-crowdfunding-project-ever-dao-mess`, June 2016. Online; accessed: 2017-02-23.

[30] Byzantine fault tolerance. `https://en.wikipedia.org/wiki/Byzantine_fault_tolerance`, 2017. Online; accessed: 2017-05-05.

[31] Perukrishnen Vytelingum, Sarvapali D. Ramchurn, Thomas D. Voice andAlex Rogers, and Nicholas R. Jennings. Trading agents for the smart electricity grid. In *Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, May 2010.

[32] Double auction. `https://en.wikipedia.org/wiki/Double_auction`, 2017. Online; accessed: 2017-03-05.

[33] Moshe Babaioff and Noam Nisan. Concurrent auctions across the supply chain. *Journal of Artificial Intelligence Research*, 21:595–629, May 2004.

[34] Q-learning. `https://en.wikipedia.org/wiki/Q-learning`, 2017. Online; accessed: 2017-05-10.

[35] Monax - the ecosystem application platform. `https://monax.io`, 2017. Online; accessed: 2017-04-09.

[36] Martin Riedmiller. Neural fitted q iteration - first experiences with a data efficient neural reinforcement learning method. *Springer-Verlag Berlin Heidelberg*, pages 317–328, 2005.

[37] Volodymyr Mnig, Koray Kavukcuoglu, David Silver, and Andrei A. Rusu. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015.

[38] Peer-to-peer energy transaction & distributed energy resource control. `http://transactivegrid.net`, 2017. Online; accessed: 2017-02-30.

[39] Blockchain community solar: the value of a renewable energy reputation. `https://medium.com/@ConsenSys/`, 2016. Online; accessed: 2017-03-02.

[40] Global community initiatives. `http://www.global-community.org`, 2017. Online; accessed: 2017-05-15.

[41] Vitalik Buterin. On sharding blockchains. `https://github.com/ethereum/wiki/wiki/Sharding-FAQ`, 2017. Online; accessed: 2017-05-12.

Appendix A

Source code

All the source code developed for the needs of this project, including the Solidity smart contracts as well as Python simulation code is in the following Bitbucket repository (`https://bitbucket.org/NikShvetsov/eth_trading_system`).

Appendix B

Thesis description

Faculty of Engineering Science and Technology

Department of Computer Science and Computational Engineering

UiT The Arctic University of Norway

Exploring auction based energy trade with the support of MAS and blockchain technology

Nikita Shvetsov

Thesis for Master of Science in Computer Science

Problem description

This project will be associated with the H2020 project EMPOWER undertaken by the Norwegian Centre of Expertise of Smart Energy Markets (NCE). It seeks to explore the use of Multi-Agent Systems (MAS) and blockchain (BC) technology and consists of three main parts:

1. Literature study and conceptualization
2. Development of MAS system for local energy trade
3. Conceptualization and implementation of a simple blockchain support for the energy trade

Part 1 is a prerequisite for the two others. It requires the study of the state-of-the-art for both MAS and BC related technologies. A study of applications within the energy domain as well as in related domains beyond the energy field is imperative. Focus on the EMPOWER H2020 concept for local energy trade is essential. The student should also acquaint himself with relevant blockchain approaches such as Ethereum from ConsenSys and digital currencies such as BitCoin and Zcash. The use of smart contracts and software agents in relation to such are important.

Main emphasizes should be placed on MAS development. This should be pinned on methods that the student are already familiar with, but associated with the EMPOWER market concept. The MAS concept should include a market arena that supports a form of local trade in energy related contracts. The market arena will be facilitated by an auctioneer, which is also an agent. This auctioneer should support a so called call auction based on price scanning and a continuous double auction. Software agents should be constructed so that they interact with the auctioneer. Agents that wish to buy make a bid, while agents that wants to sell place an ask with the auctioneer. Equilibrium is reached in accordance with the principles of settlement defined by the two types of auctions considered. Agents may trade in different types of contracts, including contracts for sale and procurement of local, renewable energy or in contracts that are more elaborate and which includes degrees of flexibility and services. Each of the trading agents represents a household that consume and/or generate energy. Surplus energy is sold. Imbalance between supply and demand must be covered by means of an agent that operates a battery and trade in the central market. The student may capitalize on work that has been produced by Kristoffer Tangrand and the student's previous semester assignments. The idea is to prove for multiple scenarios that the auction reaches Nash Equilibrium and that energy balance can be reached in the most economical way. The student must decide whether to use low-intelligence agents that operate and learn together or agents with an individual and potent form of learning i.e. reinforcement learning.

The student should highlight how the trading arena could be accommodated by a blockchain solution and explain how it could work. A simple BC demonstrator should be produced that shows how the MAS system created can operate on top of a blockchain solution.

References

CIRED 2017: CREATING A LOCAL ENERGY MARKET

Bernt A. BREMDAL
Smart Innovation Østfold & University of Tromsø (UiT) – Norway

Pol OLIVELLA-ROSELL Jayaprakash RAJASEKHARAN Iliana ILIEVA
Smart Innovation Østfold – Norway Smart Innovation Østfold – Norway Smart Innovation Østfold – Norway

Dates

Date of distributing the task:	20.01.2017
Date for submission (deadline):	06.06.2017

Contact information

Candidate	Nikita Shvetsov nsh010@post.uit.no
Advisor at UiT-IVT	Bernt Bremdal bernt.bremdal@uit.no
Co-advisor at UiT-IVT	Kristoffer Tangrand

General information

This master thesis should include:

- ❄ Preliminary work/literature study related to actual topic
  - A state-of-the-art investigation
  - An analysis of requirement specifications, definitions, design requirements, given standards or norms, guidelines and practical experience etc.
  - Description concerning limitations and size of the task/project
  - Estimated time schedule for the project/ thesis
- ❄ Selection & investigation of actual materials
- ❄ Development (creating a model or model concept)
- ❄ Experimental work (planned in the preliminary work/literature study part)
- ❄ Suggestion for future work/development

Preliminary work/literature study

After the task description has been distributed to the candidate a preliminary study should be completed within 4 weeks. It should include bullet pints 1 and 2 in “The work shall include”, and a plan of the progress. The preliminary study may be submitted as a separate report or “natural” incorporated in the main thesis report. A plan of progress and a deviation report (gap report) can be added as an appendix to the thesis.

In any case the preliminary study report/part must be accepted by the supervisor before the student can continue with the rest of the master thesis. In the evaluation of this thesis emphasis will be placed on the thorough documentation of the work performed.

Reporting requirements

The thesis should be submitted as a research report and could include the following parts; Abstract, Introduction, Material & Methods, Results & Discussion, Conclusions, Acknowledgements, Bibliography, References and Appendices. Choices should be well documented with evidence, references, or logical arguments.

The candidate should in this thesis strive to make the report survey-able, testable, accessible, well written, and documented.

Materials which are developed during the project (thesis) such as software/codes or physical equipment are considered to be a part of this paper (thesis). Documentation for correct use of such information should be added, as far as possible, to this paper (thesis).

The text for this task should be added as an appendix to the report (thesis).

General project requirements

If the tasks or the problems are performed in close cooperation with an external company, the candidate should follow the guidelines or other directives given by the management of the company.

The candidate does not have the authority to enter or access external companies' information system, production equipment or likewise. If such should be necessary for solving the task in a satisfactory way a detailed permission should be given by the management in the company before any action are made.

Any travel cost, printing and phone cost must be covered by the candidate themselves, if and only if, this is not covered by an agreement between the candidate and the management in the enterprises.

If the candidate enters some unexpected problems or challenges during the work with the tasks and these will cause changes to the work plan, it should be addressed to the supervisor at the UiT or the person which is responsible, without any delay in time.

Submission requirements

This thesis should result in a final report with an electronic copy (i.e. CD/DVD, memory stick) of the report included appendices and necessary software codes, simulations and calculations. The final report with its appendices will be the basis for the evaluation and grading of the thesis. The report with all materials should be delivered in one signed loose leaf copy, together with three bound. If there is an external company that needs a copy of the thesis, the candidate must arrange this. A standard front page, which can be found on the UiT internet site, should be used. Otherwise, refer to the "General guidelines for thesis" and the subject description for master thesis.

The final report with its appendices should be submitted no later than the decided final date. The final report should be delivered to the adviser at the office of the IVT Faculty at the UiT.

Appendix C

Intermediate report

First intermediate report on thesis project: Exploring auction based energy trade with the support of MAS and blockchain technology

Shvetsov Nikita

February 27, 2017

1 Introduction

In this report we will cover major investigations concerning our main project topics, show, what we studied to understand current state-of-the-art, show the early prototype, risks and issues, related to the proposed systemand time schedule for the project.

In order to start the literature analysis, we have stated the most crucial topics for us to study:

- Local energy market and smart grids for prosumers
- Using multi-agent systems (MAS) technology for energy trading systems
- Blockchain technology and smart contracts

Each of these topics were analyzed and for each topic the literature was searched. The analysis for the literature is presented in the next section.

2 State-of-the-art investigation

This project is about implementing a system which will be decentralized, autonomous, smart and transparent at the same time. This can be achieved using blockchain technology with smart contracts and multiple software agents, that takes care about energy auctions between prosumer agents.

In order to do that we have investigated scientific papers and articles. Through literature study we have covered main topics mentioned above:

- Blockchain
  - Bitcoin: A Peer-to-Peer Electronic Cash System [1]
    Studied the origins of Bitcoin system, history of development, its philosophy of peer-to-peer interactions, how it works in general and solutions to overcome blockchain problems.

  - A gentle introduction to blockchain technology [2]

    Explored the principles of blockchain technology, checked questions regarding consensus and validation techniques, noted possible security issues.

  - Blockchain technology beyond Bitcoin [3]

    Got familiar with the latest blockchain projects in financial and non-financial sector, get the idea how blockchain transaction are verified.

- MAS in energy trading

  - A Formal Model for Agent-Based Coalition Formation in Electricity Markets [4]

    Explored formal models for agent-based systems in electricity markets. Got the idea of energy flows in the ecosystem of smart grids.

  - Price Formation in Double Auctions [5]

    Studied how double action principle works in the market, refreshed knowledge about Nash equilibrium and explored experimental results.

  - Prosumer oriented trade: Exploration of theoretical and practical solutions for prosumer oriented trade [6]

    Got familiar with EMPOWER project and its general concepts, regarding multi-agent systems and zero-intelligent agents, rules of trading, process and requirements.

  - Implementing a Multi-Agent System in Python with an Auction-Based Agreement Approach [7]

    Checked the real implementation of MAS with auction principle in Python, how it is programmed and organized.

  - Multi-Agent Systems (MAS) controlled Smart Grid A Review [8]

    Reviewed recent implications and trends for MAS technology in controlling of the smart grid scenario.

  - A Multiagent Modeling and Investigation of Smart Homes With Power Generation, Storage, and Trading Features [9]

    Explored the multi-agent system, priorities of the agents, real case studies with loads in system. Also performance of such system is shown in the paper.

- Local energy market and smart grids

  - A Survey on Energy Trading in Smart Grid [10]

    Explored required frameworks and enabling technologies for functioning of the energy market between microgrids, evaluated various trading models and described simulation-based solution.

  - Multiagent study of smart grid customers with neighborhood electricity trading [11]

    Provided schemes for local smart grid trading with description of entities, flowchart for possible scenarios and described real-case scenario with various parameters, offered several strategies to optimize the load.

  - Agent-based Micro-Storage Management for the Smart Grid [12]

    Explored framework to analyze agent-based micro-storage management for the smart grid and showed that with certain strategy storage profile converges towards a Nash equilibrium. Also this provides real case with UK energy market.

Besides literature mentioned in list above, we have studied such relevant projects as Ethereum project concept, Brooklyn Microgrid project, TransactiveGrid, Waves platform, SolarCoin, Power Ledger. It should be noted that GridSingularity project tries to do relevant stuff, but does not open the results of the work done.

Issues that were explored are PoS(proof-of-stake) vs PoW(proof-of-work), blockchain scalability, DAO(decentralized autonomous organization), smart contracts, private and public blockchains, cross-chain interactions.

3 System prototyping and expectations

The developing system should possess the following qualities:

- Every node has equal rights

  In order for system to be decentralized every node should be treated as equal.

- Trading platform regulates auctions

  All trading operations will go through trading platform with autonomous auction agent, which will perform energy trade.

- Direct connections between nodes

  All nodes are connected to each other and have no third parties involved.

- Storage and computing modules are present in all nodes

  In order to secure the system and being able to store and produce energy each node should have appropriate modules installed.

3.1 Technologies

Used technologies include:

- Ethereum platform

  Ethereum platform provides a blockchain ecosystem with smart contracts for creating and testing our system.

- Python environment

  With Python and Ethereum implementation (pyethapp) we can create a simulation for evaluating of the system.

- Solidity

  Solidity is a JavaScript - like programming language of the smart contracts.

- EMPOWER concepts

  Our project will use some insights of EMPOWER project like SESP(smart energy service provider), BM(balancing market), DSO(distribution system operator) [6].

3.2 Prototyping

First stage of building a prototype will include simplified scheme. This will be used to create a prototype and test it relatively fast. Also testing is important due to investigation a proper learning procedure for agents in the system.

Proposed scheme is illustrated on fig 1.

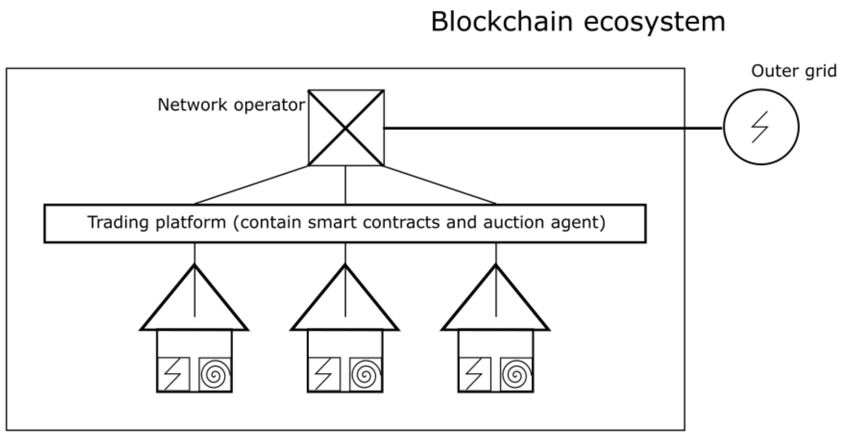
Blockchain ecosystem
Outer grid
Network operator
Trading platform (contain smart contracts and auction agent)

Figure 1: Simplified scheme prototype

3.3 Testing and evaluation

Main goal for our testing simulation is to check if the concept of our system is viable.

In order to do that the plan is to create a simulation with a small local grid with about 10 nodes and outer grid. System will be simulated for a period of 1 year. This time period should give us a clear understanding of the viability of the concept and local market should converge to equilibrium. With all information we can analyze the currency flows and energy generation. This will give us an insight how the energy market will work for prosumers.

The last phase is to compare intelligent and zero intelligent agents performance and propose the most profitable one in terms of system stability.

4 Risks and limitations

Introducing this kind of system can be accosiated with certain risks, which are connected with blockchain technology and relatively new prosumer role in local energy markets:

- Lack of standardization in this area
- Hard to customize and scale
- Lack of long-term experience with this technology
- Lack of acceptance for some parts of customers
- Possible fraud between software and hardware
- Danger of loosing accounts if requisites are lost
- No central authority in case of disputes

Many of these outflow from a fact that blockchain technology offer more control and efficiency at the cost of higher personal responsibility.

5 Possible issues

Possible issues for our proposed system include security problems, anonymity questions, scalability issues, PoW to PoS verification, cross-chain interactions and token regulations.

Security is a very important matter. The whole system can be compromised if false data will emerge. So, there should be no weak spots in logic of the system and it should be transparent for all users.

Anonymity is non-trivial matter if we talk about blockchain. Technically blockchain assumes that only address can be ties to a person, but on practice through some techniques you can associate it by correctly analyzing transactions. It is also a matter of public vs private blockchains, in which every choice comes with its pros and cons.

Scalability is often ignored, when we talk about blockchain, but it is rather important. As peer-to-peer technology, blockchain is high demanding structure. Given the extremely fast rate of data growth, the sheer data volumes accumulating after several years of operating a blockchain place high demands in terms of security, speed and costs. It is partially solved with the creating of intermediate nodes, which have only the part of blockchain and was described in original Bitcoin paper [1].

PoS and PoW issue is of great importance. Till 2017 Ethereum platform was on PoW verification algorithm and demanded a lot of computing power from nodes. Now it is transferred to PoS, which has some advantages over PoW.

Cross-chain and token regulation issues need to be solved in order to make the whole ecosystem of energy trading work. For now, there are 3 ways to perform cross-chain interactions: notaries, relays and hash-locking. Relays are the most functional, but are hard to implement. Hash-locking is the easiest one to implement, but has some restrictions. Notaries are the golden mean.

6 Development planning

- End of February

  Finishing on literature review stage

- March - April

  Programming stage:

  - Simulation environment and MAS structure
  - Integrate blockchain based on Ethereum code
  - Introduce smart contracts
  - Experiment with learning agents and auction scenarios
  - Evaluating results and make further adjustments to the program

- May - Early June

  Reporting stage:

  - Making a report of the system
  - Submitting the finished report

7 Conclusion

At the moment, we are working on Ethereum platform and programming the environment for our project. Main goal for the month - working prototype of the test environment with the support of smart contracts.

At the same time, we should check all changes for the Ethereum platform as it is rapidly developing and keep in touch with blockchain technology by participating in webinars on theme theme.

Moreover, active experience exchange and collaboration is performing with the researcher from NTNU(Norwegian University of Science and Technology) - Fredrik Blom, who is studying the same topic.

References

[1] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *The Cryptography Mailing list*, 2008.

[2] Antony Lewis. A gentle introduction to blockchain technology. *BraveNew-Coin.*

[3] Michael Crosby, Nachiappan, Pradhan Pattanayak, Sanjeev Verma, and Vignesh Kalyanaraman. Blockchain technology beyond bitcoin. *University of California Berkeley*, 2015.

[4] Sebastian Beer and Hans-Jurgen Appelrath. A formal model for agent-based coalition formation in electricity markets. *4th IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe)*, 2013.

[5] Steven Gjerstad and John Dickhaut. Price formation in double auctions. *GAMES AND ECONOMIC BEHAVIOR*, 1995.

[6] Bernt Bremdal, Pol Olivella-Rosell, and Jayaprakash Rajasekharan. Technical specifications for software development - empower project. Technical report, EMPOWER, 2016.

[7] Mikko Berggren Ettienne, Steen Vester, and Jorgen Villadsen. Implementing a multi-agent system in python with an auction-based agreement approach. 2011.

[8] Mahesh S. Narkhede, S.Chatterji, and Smarajit Ghosh. Multi-agent systems (mas) controlled smart grid a review. *International Journal of Computer Applications*, 2013.

[9] Salman Kahrobaee, Rasheed A. Rajabzadeh, Leen-Kiat Soh, and Sohrab Asgarpoor. A multiagent modeling and investigation of smart homes with power generation, storage, and trading features. *IEEE TRANSACTIONS ON SMART GRID*, 2013.

[10] Safak Bayram, Muhammad Z. Shakir, Mohamed Abdallah, and Khalid Qaraqe. A survey on energy trading in smart grid. *math.OC*, 2014.

[11] Salman Kahrobaee, Rasheed A. Rajabzadeh, Leen-Kiat Soh, and Sohrab Asgarpoor. Multiagent study of smart grid customers with neighborhood electricity trading. *Elsevier*, 2014.

[12] Perukrishnen Vytelingum, Thomas D. Voice, Sarvapali D. Ramchurn, Alex Rogers, and Nicholas R. Jennings. Agent-based micro-storage management for the smart grid. *9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, 2010.