TALLINN UNIVERSITY OF TECHNOLOGY
Department of Software Sciences

Peeter Kokk 163123IAPM

PEER-TO-PEER ENERGY MARKET EVOLUTION BY COMBINING SOFTWARE AGENTS WITH THE BLOCKCHAIN

Master’s thesis

Supervisor: Kuldar Taveter
PhD

Tallinn 2019

TALLINNA TEHNIKAÜLIKOOL
Tarkvarateaduse instituut

Peeter Kokk 163123IAPM

ARVUTITEVAHELISE ENERGIATURU EVOLUTSIOON TARKVARAAGENTIDE JA PLOKIAHELA ÜHENDAMISE TEEL

Magistritöö

Juhendaja: Kuldar Taveter
PhD

Tallinn 2019

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Peeter Kokk

21.05.2019

Abstract

The electricity grid is going through changes related to the introduction of renewable energy sources, while energy production is becoming more decentralized. This decentralization needs new kind of approach to the energy grid that allows for local self-management of energy and measures energy usage in more detailed way, which traditional, centralized systems cannot offer. This type of smart grid also needs an efficient market, meaning it must match buyers and sellers effectively, be set up to ensure that prices will be set competitively and, finally, ensure that transactions are secure and reliable for buyers and sellers. This thesis focuses on figuring out how software agents and blockchain can help build a market of this kind. A proof of concept is also built to test the adherence to the previously mentioned attributes of an efficient market.

This thesis is written in English and is 58 pages long, including 6 chapters, 10 figures and 5 tables.

Annotatsioon

Arvutitevahelise energiaturu evolutsioon tarkvaraagentide ja plokiahela ühendamise teel

Elektrivõrk on läbimas muudatusi seoses taastuvate energiaallikate kasutuselevõtuga, samal ajal on energiatootmine muutunud detsentraliseeritumaks. See detsentraliseeritus vajab uut tüüpi lähenemist elektrivõrgule, mis võimaldab elektrienergia kohalikku isejuhtimist ja selle kasutamise üksikasjalikumat mõõtmist, mida traditsioonilised tsentraliseeritud süsteemid ei suuda pakkuda. Selline tark võrgustik vajab ka tõhusat turgu, mis tähendab, et see peab ostjaid ja müüjaid efektiivselt sobitama, tagama, et hinnad seatakse konkurentsivõimeliseks, ja lisaks tagama, et tehingud on ostjate ja müüjate jaoks turvalised ja usaldusväärsed. Antud magistritöö uurib, kuidas tarkvaraagendid ja plokiahelad aitavad sellist turgu ehitada. Ehitatakse valmis ka simulatsioon, et testida tõhusa turu eelnevalt mainitud omaduste järgimist.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 58. leheküljel, 6. peatükki, 10. joonist, 5. tabelit.

List of abbreviations and terms

P2P	Peer-to-peer
DER	Distributed energy resource
MAS	Multiagent systems
JADE	Java Agent Development Framework
ECDSA	Elliptic Curve Digital Signature Algorithm
FIPA	Foundation for Intelligent Physical Agents
JADE	Java Agent Development Framework
ACID	Atomicity, Consistency, Isolation, Durability

Table of contents

List of figures

List of tables

1 Introduction

1.1 Background

1.1.1 Smart grids

The electricity grid is going through changes related to the introduction of renewable energy sources and electric vehicles, while energy production is becoming more decentralized. "Distributed energy resources (DER) are smaller power sources that can be aggregated to provide power necessary to meet regular demand. As the electricity grid continues to modernize, DERs such as storage and advanced renewable technologies can help facilitate the transition to a smarter grid." [1]

Unlike centralized big energy sources, DERs energy supply is not constant and thus benefit greatly from models that could predict the production and consumption of energy. In order to get the consumption data, two-way communication is needed. "Advanced metering infrastructure" (AMI) or "smart meters" are "electricity meters that use two-way communication to collect electricity usage and related information from customers and to deliver information to customers" [2].

DERs also need to be integrated into a network with other DERs, due to the volatility and small scale of their energy production. For this, "smart grids" can be used. "A Smart Grid is an electricity network that can intelligently integrate the actions of all users connected to it – generators, consumers and those that do both – in order to efficiently deliver sustainable, economic and secure electricity supplies" [3].

Smart grids have a huge potential of cost savings as well and are projected to be generate more benefits than the cost of upgrading the current grids to "smart" grids.

In 2011 (EPRI Report 2011), the EPRI made a forecast that the cost to upgrade the US grid to "smart" status would be between $338 billion and $476 billion and would generate benefits of between $1,294 billion and $2,028 billion, for a projected benefit-to-cost ratio *of between 2.8 and 6.0 to 1. However, US utility smart grid business cases typically forecast benefit-to cost ratios of between 1.1 and 3.0 to 1. [4, p. 1496]*

1.1.2 Peer-to-peer markets

However, with the usage of smart grids, the issue of selling this energy needs to be solved, because decentralized sources are difficult to operate centrally. Due to the decentralized nature, "peer-to-peer markets" are often seen as a viable strategy for selling energy. "Peer-to-peer markets such as eBay, Uber, and Airbnb allow small suppliers to compete with traditional providers of goods or services." L. Einav et al. state that the "goal of peer-to-peer markets is to create trade between large numbers of fragmented buyers and sellers." [5, pp. 615,617]

1.1.3 Agents and Multiagent Systems

Peer-to-peer markets can be thought of as multiagent systems, a system consisting of agents interacting with each other. The term "agent" has been used to denote a variety of different meanings in computer science. We use it to mean, per the definition provided by Sterling and Taveter, "an entity that can act in the environment, perceive events, and reason" [6, p. 35]. According to Sterling and Taveter, this definition includes people, robots and software applications [6, pp. 8-9]. However, more importantly and more accurately, Sterling and Taveter suggest that agents have three qualities associated with them. These qualities are:

1) Have a purpose to "take an active role or produce a specified effect" and/or "act behalf on another, for example, managing business, financial, or contractual matters, or [provide] a service."

2) Have "controlled autonomy, or the ability to pursue its own goals seemingly independently."

3) Be situated – that is, be aware of the environment around it

[6, pp. 6-7,10].

Agents can be used to build "multiagent systems". A multiagent system is "a kind of a system where several, perhaps all, of the connected entities are agents" [6, p. 342].

1.1.4 Blockchain

Before we explain how blockchains can relate to smart grids, we first need to explain what they are about. A “blockchain” can be viewed as a distributed immutable database. The blockchain usually consists of large number of untrusted computers that “hold a timestamped list of blocks which record, share, and aggregate data about [all of the] transactions that have ever occurred within the blockchain network” [7, p. 330]. Cryptographic proofs make this data storage immutable and data can only be inserted, assuming most of the network is not compromised – and no one controls the majority of nodes [7, p. 330].

As noted by Weber et al., “[a] key feature of a blockchain-based system is that it does not rely on any central trusted authority, like traditional banking or payment systems. Instead, trust is achieved as an emergent property from the interactions between nodes within the network” [7, p. 332]. The way this works is that the nodes must achieve consensus on what data is inserted into the network. This, coupled with the property of allowing only data insertion, allows the participating parties to share data in a transparent way on the blockchain.

Hull et al. explain that the transactions performed on the blockchain network obey the “ACID (Atomicity, [eventual] Consistency, Isolation, Durability) transactional properties of classical database systems” [8, p. 19], although it is questioned by Tai et al. if the ACID model is even a good fit [9, p. 759]. However, blockchain platforms such as Ethereum [10] also come “with a built-in Turing-complete programming language, allowing anyone to write smart contracts…” [11, p. 13]. Smart contracts have been compared to stored procedures, which are used by relational database systems [12], but the key advantage over them is that blockchain systems force smart contracts to be immutable, as long as the blockchain security is not penetrated, the implications of which we will discuss later. Guaranteeing immutability to a much larger degree is the most important feature that makes blockchain different from traditional databases.

1.1.5 Smart Contracts

Some blockchains come with the smart contract feature, which holds some similarities to agents. In trying to explain the different interpretations of smart contracts, Stark remarks that people with technical understanding of the blockchain technology, use the term “smart contracts” to refer to code that is executed on blockchain, which he considers misleading, as “smart contracts” can act autonomously, when their functions are called, making them more like agents rather than contracts [13]. This is because smart contracts have two of the three qualities of agents we listed in section 1.1.3 – they act on behalf of someone and operate on certain information on the blockchain. They lack, however, the quality of controlled autonomy, because “smart contracts” do not reason, and thus, unlike agents, have no self-governing qualities.

1.2 Problem Statement

According to Bremer and Lehnhoff [14], in Europe, especially in Germany – where renewable energy producers are offered long term guarantees about contracts and prices –, the share of distributed energy resources (DER) is rapidly growing.” This can mainly be seen as an attempt to reduce the emission of greenhouse gasses. International Energy Agency states that “two sectors produced nearly two-thirds of global CO2 emissions from fuel combustion in 2014: electricity and heat generation, by far the largest, which accounted for 42%, while transport accounted for 23%” [15, p. 12]. However, it is noted by James et al. [16] that “decentralized generation” needs new kind of approach that allows for “local self-management” and measures usage in more detailed way, which traditional, centralized systems cannot offer. Using multiagent technology for “distributed control has proven in field studies to be an efficient way to balance supply and demand in local clusters of distributed loads and generators” [16, p. 9]. Ogston et al [17] find that using “central components, and more significantly their communications infrastructures, is undesirable in systems that contain hundreds of thousands or millions of devices” [17, p. 1]. Therefore, peer-to-peer multiagent systems seem to be a good fit for use in decentralized management of energy markets.

Multiagent systems alone are not enough for the decentralized management of energy markets. Einav et al [5] list three problems that are necessary to solve for any market, which includes P2P energy markets, to efficiently achieve its goal – to “create trade between large numbers of fragmented buyers and sellers" [5, p. 617].

1) “Search” – “match buyers and sellers effectively while keeping search frictions low.”

2) “Pricing” – “establish prices, or to organize the market so that prices will be set competitively.”

3) “Trust” – “ensure that transactions are safe and reliable for buyers and sellers.”

[5, pp. 617-618]

While software agents are capable of matching buyers and sellers (efficiency of this depending mainly on the algorithm used) and setting the price (the fairness of which depending, again, mainly on the algorithm used), they are acting on behalf of the interest of someone, so should not be expected to ensure the reliability or security of transactions. This is where blockchains with smart contracts can be of use.

Blockchains ensure the safety of transactions. Since, according to Buterin, Ethereum “transactions are all signed using the ECDSA algorithm” [18], which as of May 2016 was stated to have “no known mathematical vulnerabilities” by an ECDSA security research survey [19], the transactions can be considered secure to a very high degree.

The reason why smart contracts could be useful for P2P markets is that they enable parties to specify the rules and not to dishonour them. This is because “a smart contract is a set of promises, specified in digital form, including protocols within which the [agents] perform on the [promises of other agents]” [20]. Although blockchain data insertions, as was mentioned in section 1.1.4, obey the ACID transactional properties, transactions performed on the blockchain can be considered reliable only when most of the network is not compromised (i.e., no one controls the majority of the nodes) and the rules agreed upon do not change without the consent of the stakeholders. The key advantage of smart contracts is that they are immutable because they, too, are a part of the data that is uploaded onto the blockchain network. This means that once the parties agree upon rules and write it down within a smart contract, a different smart contract cannot be used instead without the knowledge by the nodes of the network. If smart contracts are not used for enforcing the rules, there would have to be some type of a human or man-made agent, who enforces the rules. Either way, however, this would require solving the issue of how to make sure the agent enforcing the rules is trustworthy. Smart contracts can be useful in P2P markets, because they help to make sure that the transactions are reliable.

However, smart contracts alone are not enough for a P2P market system. Omohundro [21] points out that "simple and easy to write contracts appear to be sufficient for many entirely digital transactions. But as these systems start to interact with the physical world, there is likely to be a need for greater intelligence and real world knowledge in making decisions" [21, p. 20]. He goes on to suggest that AI systems are needed to translate information from sensors for smart contracts, while smart contracts that result in actions in the physical world also need to forward information to both human and non-human agents. Therefore, software agents are still necessary, when designing a P2P market system on top of a blockchain, because they can mediate blockchain with other elements of the environment.

The problem we are trying to solve is to figure out how to use software agents and blockchains to enable decentralized trading system that adhere to the "search", "pricing" and "trust" principles.

1.3 Research Goal

The research goal of this M.Sc. thesis is to find out how to build a P2P energy market that is automated by means of using software agents and blockchain technology. To design and implement such a system, we will use the design science method – by leaning on most of the seven guidelines offered by Hevner et al [22]

In subsections 1.3.1-1.3.6 we will define the research question and explain how we are going to follow the guidelines for answering the research question.

1.3.1 Research Question

The research question to be answered by this MSc thesis is as follows:

- How to build a P2P energy market system that is automated by means of software agents and blockchain technology?

For answering the research question, we will follow the guidelines presented in Sections 1.3.2-1.3.6.

1.3.2 Guideline 1: Design as an Artifact

Specifically, we are interested in how to create the architecture of a P2P energy market consisting of an agent application and a blockchain application that together manage distributed energy resources. The artifact in this case consists of both the models we will be creating by using the Agent-Oriented Modelling methodology by Sterling and Taveter [6] and the application. The application is a simulation of the proposed system, which will validate the system. The application would ideally be run on every smart meter – owned by consumers, producers or prosumers – on the smart grid.

The energy market itself will be implemented using research knowledge from the field of energy markets.

1.3.3 Guideline 2: Problem Relevance

The benefits of the proposed system are derived from the creation of a simulation of a P2P energy market, which proves that it is possible to solve the trust problem mentioned in Section 1.2 with blockchain technology, therefore making the market efficient at "creating trade between large numbers of fragmented buyers and sellers" [5, pp. 617-618]. If agent technology can solve the search and price problem discussed earlier, then this simulation would prove to be a proof of concept for the kind of energy market future smart grid could use to manage DERs.

1.3.4 Guideline 3: Design Evaluation

The algorithm used for the energy market, while relevant in practice, is considered less in this thesis, because our main objective is to demonstrate the feasibility of combining software agents and blockchain technology for building P2P energy market systems. Therefore, we will find and validate the algorithm by looking at the relevant research that has been performed in this field rather than creating our own algorithm.

We will validate the proposed system with a combination of two simulation applications: an application built on the JADE agent programming environment according to the models created with the help of the Agent-Oriented Modelling methodology [6], and a blockchain application. To validate the design of such a system, we will assess the results of running the simulations by modifying different variables to see how this affects the price dynamics and matching of the buyers and sellers.

1.3.5 Guideline 4: Research Contributions

The contribution of this research is that so far, to the best of our knowledge, there have been no attempts to create systems for P2P energy markets using both software agents and smart contracts – combining automaticity with transparency. It is important to note that we do not mean agents and blockchain, which, as can be seen in related research, has already been used together.

1.3.6 Guideline 5: Research Rigor

The design of the system will rely on research that has been conducted before in energy markets and the interactions between market participants. We will also use the research work conducted on designing and developing software agents and multiagent systems.

1.4 Overview

In chapter 2, we provide an overview of similar multiagent systems using different types of blockchains to create an energy market for software agents.

In chapter 3, we provide agent-oriented models for the proof of concept system that will run the simulation we will be building, to show how the system works from the view of software agents.

In chapter 4, we discuss the building blocks used for the system that will run the simulation – the Ethereum blockchain and Java Agent Development Framework – and give insights into the implementation.

In chapter 5, we will explain how the system relates to asymmetric markets and explain a learning strategy that the software agents can use in this type of market, which we will use in the creation of the system that will run the simulation.

In chapter 6, we will discuss what kind of parameters we made customizable for this simulation and provide the results for two test cases for the simulation and analyze the results.

2 Related Research

Aitzhan and Svetinovic [23] “implemented a proof-of-concept for decentralized energy trading system” [23, p. 9] for the smart grid using Bitcoin and its multi-signatures feature for smart contracts, which only allows a transaction to be performed on the blockchain, when a certain number of participants of the transaction have signed the transaction. The system allows anonymous messages to be sent by nodes of the network which can be either encrypted, so that they can only be read by a node with a unique private key, or not, in which case everyone can read them, in exchange for “proof-of-work” [23, p. 12] (“proof-of-work” in this context is essentially a computation). The value is transferred, when agents send anonymous messages to “Distribute System Operators” (DSOs). DSO is “a natural or legal person[s] responsible for operating, ensuring the maintenance of and, if necessary, developing the distribution system in a given area…” [24]. DSO are used here in order to assure the “trust” aspect, described in the Problem Statement, which we intend to solve using smart contracts. With the help of “Schneier attack tree” they also analyzed “vulnerable assets” from which they extracted 23 “system security attacks” that serve as a basis for eliciting 20 security and privacy requirements [23, pp. 9-12].

Mihaylov et al. [25] “propose … a novel mechanism for trading of locally produced renewable energy [on the smart grid] that does not rely on an energy market or matching of orders ahead of time” [25, p. 1]. Unlike in the paper by Aitzhan and Svetinovic [23], where value is transferred through DSOs by storing information on the blockchain [23, p. 4], Mihaylov et al. use a decentralized digital currency called “NRGcoin” instead. The NRGcoins are “generated by injecting locally produced renewable energy to the grid,” and “are traded between consumers and prosumers directly” [25, pp. 1-4] (prosumers are agents, who, according to Amato et al. [26], both consume and produce). Mihaylov et al. point out that since the role of a DSO is to collect and distribute payments, they do not create NRGcoins [25, p. 4]. DSOs measure consumption of electricity and perform real time billing, where the price they set depends on both demand and supply, that is, it cannot be set by prosumers. Both Aitzhan and

Svetinovic [23] and Mihaylov et al. [25] rely on smart meters being tamper proof, which is not necessarily true in the real world.

Hosokawa and Nishino [27] propose two trading techniques for decentralized energy trading – “aggregated demand–supply” technique and “residual electricity-based” technique. The technique of “aggregated demand–supply” creates a demand curve from demand curves of all the consumers and a supply curve from supply curves of all the producers [27, p. 125]. The “residual electricity-based” technique also considers the electricity the consumer has produced and wants to keep for their own use. This system is one way of implementing the notice board system offered in the paper by Aitzhan and Svetinovic [23, p. 5]. The fairness of the trading price depends a lot on whether prosumers produce electricity at the right time, which leaves a lot of room for speculation, as prosumers can set the price, while in the system described by Mihaylov et al. [25], a third party (a DSO) determines the price.

Amato et al. [26] introduce the “CoSSMic” (Collaborating Smart Solar-powered Micro-grids) project, which aims at “fostering a higher rate for self-consumption (<50%) of decentralized renewable energy production” [26, pp. 155-156] through using multiagent systems for energy trading and offers a framework named after the project – CoSSMic framework – to achieve this goal. The framework differs from previously talked about systems in that it separates the market from the MAS. “If an agent [in the CoSSMic framework] cannot find enough energy to satisfy its needs in the neighborhood market,” it will receive the energy from the main grid (produced by non-renewable energy resources) [26, pp. 159-160] This was most likely implicitly assumed by the systems previously discussed in this section [23] [25] [27], since renewable energy resources depend on the environment and thus cannot be expected to produce the same amount of energy each day. An implementation of the CoSSMic framework created with the JAVA Agent Development Framework (JADE) [28] is described, although the description does not describe any validation of this system.

Tasquier et al. [29] “[focus] on coordinating local energy production and consumption of individual houses in a neighbourhood” within the CoSSMic framework. They claim that “The connection between devices and agents is implemented by a distributed or centralized RESTFul Gateway (RFG) that is in charge of notifying events using the Agent Communication Language (ACL) and message passing mechanisms”

[29, p. 61]. The building sends information through this gateway (using API calls). This information is then forwarded to the Event Bus, which is implemented by “a specialized agent” that forwards information to Control Agents through “a publish/subscribe paradigm.” In other words, Control Agents act based on events they receive from the Event Bus. For example, if more energy is produced than consumed, it will try to sell the energy to other buildings via another agent termed the “Protocol Handler.” The Protocol Handler allows Control Agents to buy or sell energy according to a “defined negotiation protocol,” for which they use the “FIPA Contract Net”. In the implementation, which is created on the JADE agent platform, the events are exchanged via JSON, while for negotiations, a service level agreement (SLA) is established for JSON messages. The negotiation is implemented on both JADE and Smart Python Multi-Agent Development Environment (SPADE) [30]. The implementation of the solution by Tasquier et al. [29] is tested on theoretical data, which is loosely derived from real world data.

Amato et al [31] also offer a negotiation algorithm for the CoSSMic framework. Additionally, a “P2P overlay [network]” is implemented using Retroshare [32], “a communication platform that allows file sharing and instant messaging trough [OpenSSL library] encrypted connections” [31, p. 166]. In Retroshare, “there are no central servers” [32] and it is implemented using Distributed Hash Tables (DHT). According to Fey et al, “P2P networks set up an overlay network … [for] addressing, routing and network management are handled by the application” (for which the authors mention Gnutella and BitTorrent as examples) [33, p. 4]. The P2P overlay architecture is explained in more detail by Amato et al [34], where an overview of the Retroshare architecture is also given.

Alkhawaja et al. [35] suggest using “Message Oriented Middleware” (MOM) for the communication between distributed applications, which need to interact with each other over a network. They list RabbitMQ, Data Distribution Service (DDS) and the Extensible Messaging and Presence Protocol (XMPP) as examples of MOMs and compare them in terms of “latency, bandwidth, delivery guarantees and, message priority and [message] ordering.” [35, pp. 2-3] It should not come as a surprise that XMPP is a protocol also considered suitable for the Internet of Things (IOT) by Karagiannis et al [36], since IoT and smart grids are closely related to each other (due to, besides other reasons, smart meters being used for the latter).

Capodieci et al. [37] sketch an architecture and develop a simulation for an agent-based architecture for smart grid energy markets consisting of buyers, prosumers, “Gencos” (“big energy generating companies” [37, p. 302]) and a “balancer” (“responsible for the synchronization of negotiation procedures and for the balancing aspects” [37, p. 302]). A balancer agent keeps track of how much energy is produced by the prosumers and how much buyer agents predict they will use within an interval. Based on this, it communicates to Gencos how much energy they should offer for the interval. “Each market day is divided into several time intervals and for each one every buyer has to decide in advance who is going to be its energy supplier for the next time interval” [37, pp. 301-302]. This is done by using a prediction algorithm. In reality, a hybrid of a system like this – which uses “time-of-use pricing” [38, p. 12] – and the one described in the work by Mihaylov et al. [25] – which uses “real-time pricing” [38, p. 12] – would be needed for a smart grid, since predicting energy consumption can help with keeping fluctuations to a minimum, while a real-time bargaining functionality can make surpassing a prediction less of a problem because such systems use “critical peak pricing” [38, p. 12]. In the negotiation phase, “[b]uyers do not know any bid values of the other consumers” [37, p. 301]. This is a security concern, since it introduces the possibility of rigging the market through cooperation amongst sellers or between a seller and a buyer. To create a prediction algorithm that would be fair toward all participants in such conditions where buyers do not know the bid values when placing their bids, they adopt the El Farol Bar minority game for the energy market and mix this with a stochastic game. They also provide the bidding agents with risk awareness and use fuzzy logic to adopt to the changing market conditions that result from the behaviours of other agents. Finally, they run the simulation developed on JADE, comparing agents, who use the described strategy except for the adoption algorithm and the ones who use it also with the adoption algorithm.

3 Agent-Oriented Models

3.1 Goal Models

Taveter and Sterling [6] define goal model as containers that hold system’s goals, quality goals and its roles. A goal is a functional requirement, a quality goal a non-functional requirement and a role is a position that is necessary for the system to achieve its goals.

In Figure 1 we depict a goal model, where there are five main goals.

1) Finding producers

2) Notifying the producers of the consumers interest and counteroffer

3) The producer accepting the offer from consumer

4) Finalizing the contract

Finding producers is divided into sub goals. The main goal has the quality goal of happening fast to save energy costs (that is, computational resources are not wasted). The sub goals of the main goal are finding electricity producers that have electricity to sell and selecting the producers – the quality goal is to find offers that bid the best price for a given quantity, disregarding offers that are too high for the consumer or too high compared to the rest of the offers.

When notifying the producers of the consumers interest the producers are told how much and at what price the consumer wants to buy electricity (the consumers already know at what price the producers offer to sell electricity, but here the consumer gets to haggle the price down). This should be done in a way in the system that would not let either party be treated unfairly.

Accepting the offer from consumer is divided into two sub goals. Firstly, the producer checks that they have enough energy to sell and the counteroffer’s price is not too low. Secondly, if they deem these conditions true, they accept the offer by notifying the consumer.

Once the offer is accepted, the contract is finalized by the consumer calling the smart contract functions, which pays the producer and records the information on the blockchain. This action makes the contract immutable and binding (one of the quality goals), which means it gives grounds for the consumer to prove that the contract was indeed accepted by both parties and that money was transferred. The use of smart contract also fulfills the quality goal of making the transaction secure and reliable.

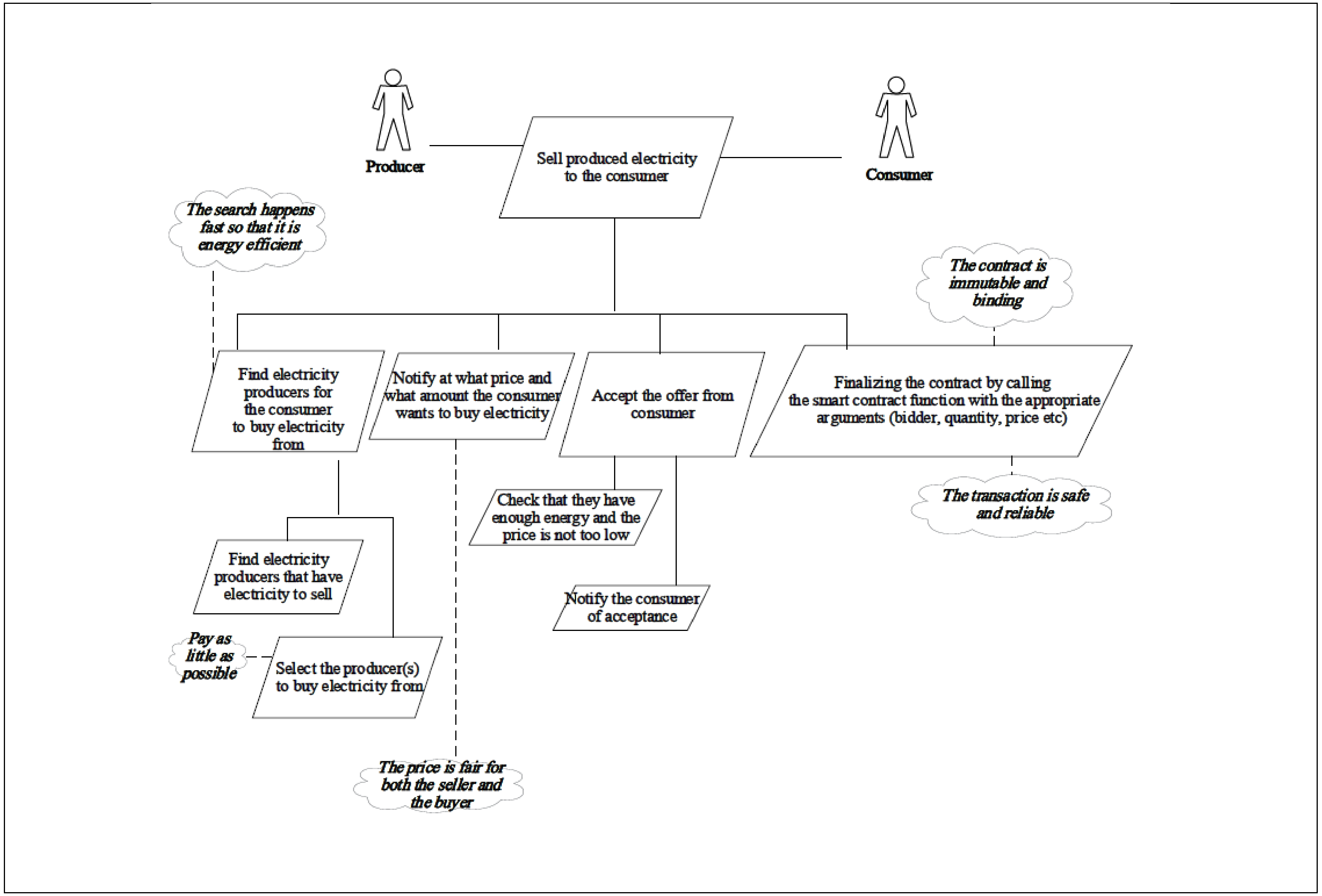
Producer
Sell produced electricity to the consumer
Consumer
The search happens fast so that it is energy efficient
The contract is immutable and binding
Find electricity producers for the consumer to buy electricity from
Notify at what price and what amount the consumer wants to buy electricity
Accept the offer from consumer
Finalizing the contract by calling the smart contract function with the appropriate arguments (bidder, quantity, price etc)
Check that they have enough energy and the price is not too low
The transaction is safe and reliable
Find electricity producers that have electricity to sell
Notify the consumer of acceptance
Pay as little as possible
Select the producer(s) to buy electricity from
The price is fair for both the seller and the buyer

Figure 1 Goal model for selling produced electricity to the consumer

3.2 Role Model

Since Goal models do not give insights into the details of the roles, Taveter and Sterling [6] offer role model concept, to show in detail what the responsibilities and constraints of the roles are on an individual level. In our case we only need to define role models for Producer and Consumer, which can be seen in Table 1 and Table 2 .

Role name	**Producer**
Description	The energy producer can either be a local homeowner – producing energy via renewable energy sources – or a company – producing it via renewable or unrenewable energy source –, who sells it to consumers
Responsibilities	Inform the customer about the amount of energy offered
	Inform the customer about the price of energy
	Accept or reject an offer made by the consumer
	Sell the energy
	Receive virtual currency using smart contracts
Constraints	A transaction cannot be completed without buyer calling the appropriate smart contract function
	Once signed, the contract is immutable and binding
	The transaction is safe and reliable
	Should receive the virtual currency that is specified in the smart contract

Table 1 Role model for producer

Role name	**Consumer**
Description	The energy consumer can be anyone, who buys energy from the producer
Responsibilities	Find energy producers that have energy to sell
	Ask energy producers about the amount and price of energy
	Select the producer(s) to buy energy from
	Ask producers if they are willing to sell a certain amount of energy for the price the producer offered
	Call a smart contract function for selling a certain amount of energy for a certain price
	Check their virtual currency balance on the blockchain
Constraints	A transaction cannot be completed without buyer calling the appropriate smart contract function
	Interact with the nearest and cheapest producers first

Once signed, the contract is immutable and binding
The transaction is safe and reliable

Table 2 Role model for consumer

3.3 Organization Model

The organization model helps us understand the relationships of the agents when defining the interactions of between them, as Taveter and Sterling [6] point out. For our case, it is a very simple relationship, where the agents are equal – that is, neither commands the other. The organization model can be seen Table 3.

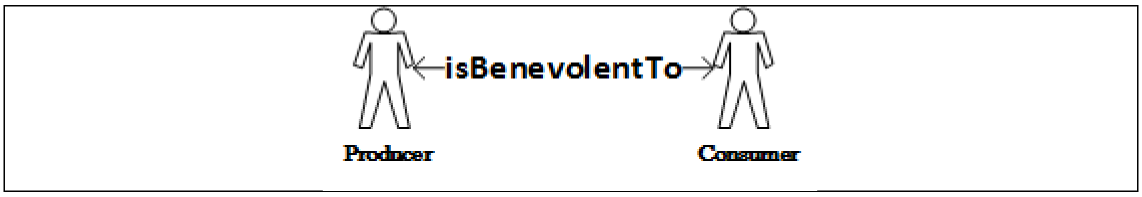

Table 3 Organization model for our system

3.4 Domain Model

Taveter and Sterling [6] use domain models to map out the knowledge about the environment the system is situated, to clarify the context of goal and role models. The domain model for our system can be seen on Figure 2. Besides the roles, the domain model contains services and resources. In the domain model, the services are denoted with a rectangular box (e.g. JADE Main Container) and underlined text and connected to resources. The resources are denoted with a rectangular box and are connected to the roles with a verb or verbs (e.g. accesses).

In our domain model, the consumer interacts with the Blockchain service and Directory Facilitator, while the producer interacts exclusively with the Directory Facilitator. This is because smart contracts are not like real contracts and do not necessarily require both parties to have access to them in order to be valid, since they can be used to send money from the producer to the consumer, which can serve as a proof of transaction on its own.

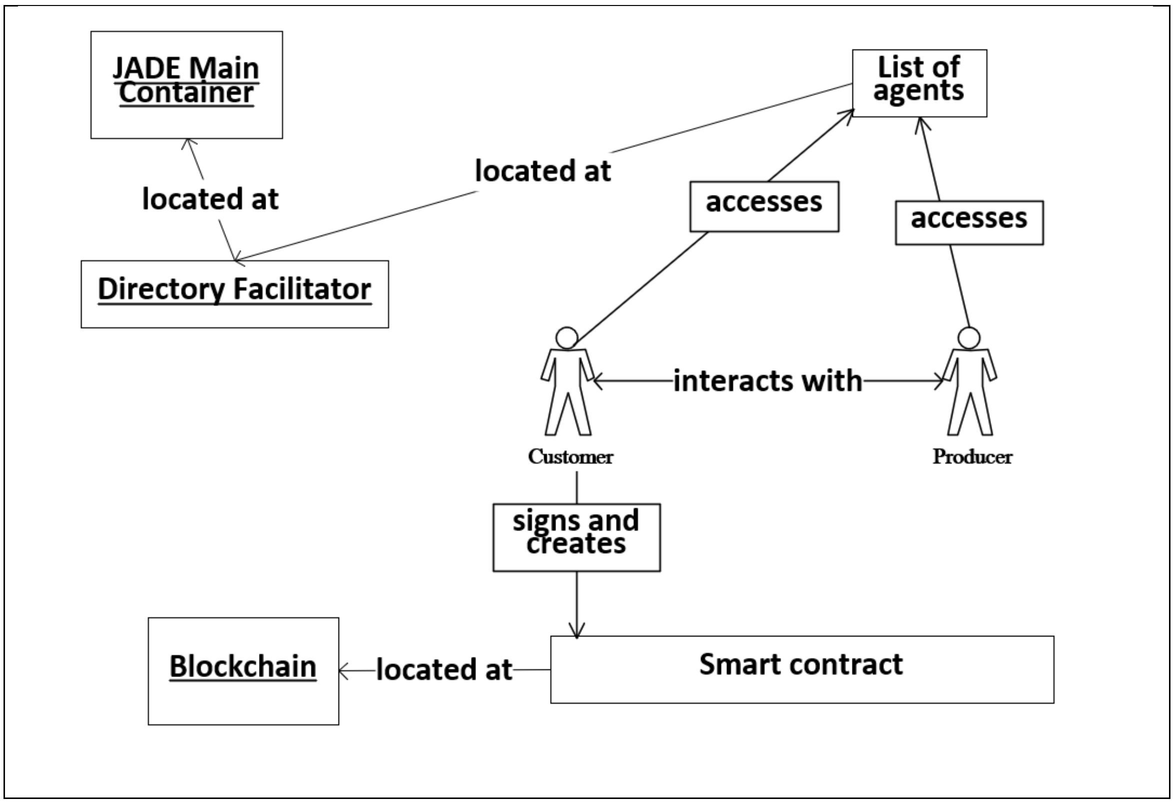

Figure 2 Domain model for our system

3.5 Agent/Acquaintance Model

The agent models represent the relationships between roles and agents. “The purpose pose of agent models is to transform the abstract constructs from the analysis stage, roles, to design constructs, agent types, which will be realized in the implementation process” [6, p. 231]. As Figure 3 shows, the Consumer and Producer are implemented as the Smart meter agent, or to put it into object-oriented language terms, they are instances of the same class. The asterisk below the Smart meter box denotes that there can be multiple instances of Smart meter that exist in our system.

Acquaintance models are introduced by Taveter and Sterling “in order to visualize the degree of coupling between agents.” [6, p. 233] The “arc represent the existence of an interaction link,” [6, p. 233] showing that two agents can communicate (in our case, this means sending messages). The asterisk implies that multiple Smart meter agents can interact with multiple other Smart meter agents

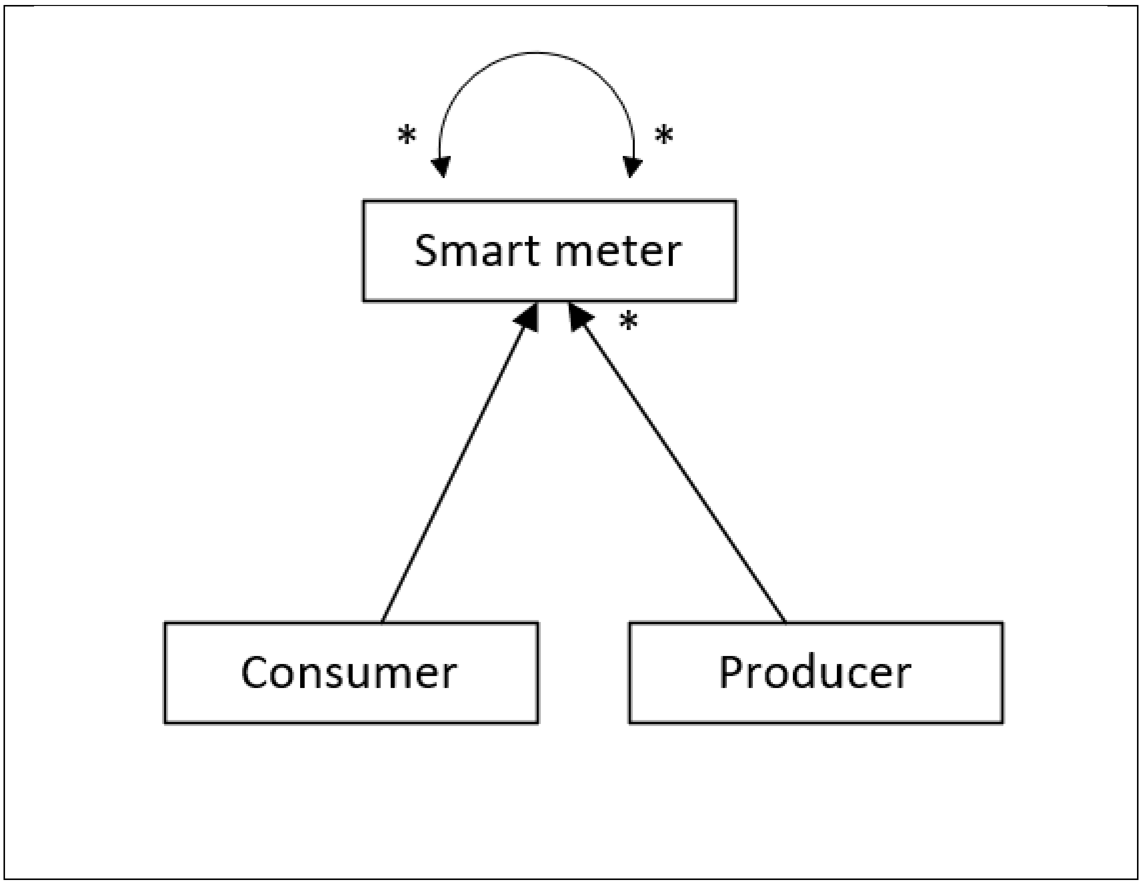

Figure 3 Merged agent and acquaintance model for our system

3.6 Scenarios

Scenarios go into specifics about the interactions of the agents, or in other words, “a scenario is a behavior model that describes how the goals set for the system can be achieved by agents of the system” [6, p. 251]. In Table 4 we define the scenario for the main goal described in chapter 3.1.

SCENARIO 1	
Scenario description	Buying energy from producer(s)
Initiator	Consumer
Trigger	A consumer wants to find energy producers that have energy to sell
Failure	The consumer is left without energy
DESCRIPTION	

Condition	Step	Activity	Agent types/roles	Resources	Quality goals
	1	Find Smart Meters that have energy to sell	Smart Meter/ Consumer	Directory Facilitator's list of agents	1) The search happens fast so that it is energy efficient. 2) Find energy producers that offer the best price
	2	Ask Producer about the amount and price of energy	Smart Meter/ Consumer, Smart Meter/ Producer		
	3	Send information about the amount of energy the Producer is willing to offer and the price of energy per a certain unit (e.g. watt)	Smart Meter/ Consumer, Smart Meter/ Producer		The price is fair for both the seller and the buyer
Repeat if necessary 3-4	3	Select the Producers to buy energy from	Smart Meter/ Consumer		
	4	Ask Producer if they are willing to sell the amount of energy for the price they previously offered	Smart Meter/ Consumer, Smart Meter/ Producer		
Options 5-6	5	Reject an offer	Smart Meter/ Consumer, Smart Meter/ Producer		
	6	Accept an offer	Smart Meter/ Consumer, Smart Meter/ Producer		
	7	Buy energy by calling the smart contract with the appropriate arguments	Smart Meter/ Consumer	Smart contract	The contract is immutable and binding

	(bidder, quantity, price etc.)			
8	Sell an amount of energy for a certain price by signing a smart contract	Smart Meter/ Producer	Smart contract	1) The contract is immutable and binding 2) The transaction is safe and reliable 3) Happens automatically once the consumer has called the function of the smart contract

Table 4 Scenario for our system

4 System Architecture

4.1 Ethereum blockchain

In designing the system, we need to store *only* that data on the blockchain, which is important from the “trust” aspect – ensuring that the transactions are not altered once they are agreed upon – as is described in the Problem Statement in chapter 1.2. This is for both speed and scalability reasons – as was stated earlier in chapter 1.1.4, data on the blockchain needs to be cryptographically verified and, by nature of decentralization, stored in multiple locations.

For data storage, Ethereum has two ways of storing data on the blockchain – either in storage variables inside smart contracts or by using logs, which smart contracts cannot access, even though logs are created when a smart contract emits a higher-level abstraction called “event” [39]. Although logs are not a part of the data stored on the blockchain, they are still verified by the blockchain, since “[l]ogs are part of the transaction receipts” and “transaction receipt hashes are stored inside the blocks” [39]. In conclusion, data can be stored via smart contracts or logs, with the latter having the disadvantage that they cannot be accessed by smart contracts afterwards.

Our goal is to store as little data as possible with storage variables inside smart contracts and inside logs, due to previously mentioned reasons. This is especially relevant because we will be using a blockchain with each node holding all the data and these “full node blockchains” would have to sacrifice either decentralization or security in return for scalability – either you store all the data on all the nodes or you store some of the data and lose on the decentralization and/or security aspect –, which would not be desired. This coupled with the fact that on Ethereum, “[l]ogs … cost 8 gas per byte, whereas contract storage costs 20,000 gas per 32 bytes” [39] (32 bytes is maximum of 32 characters in UTF-8, while 20,000 gas was on average about 0.01624 dollars at the end of October 2018 and 0.05359 dollars in the beginning of May 2019 [40]), means that storing data on logs is preferable, when thinking of data costs.[1]

According to the Solidity documentation, Events can be called within a smart contract function, upon which the arguments used to call a smart contract function are stored on the blockchain and these arguments are then in turn associated with the address of the contract [41]. This information can then be accessed if the block is accessible.

4.2 Java Agent Development Framework

For the software agent creation and management, we will use Java Agent Development Framework. The software agents will be running on JADE's Main Container and we will be using its Directory Facilitator for the agents to find and interact with other agents on the Main Container. According to JADE documentation, "[e]ach running instance of the JADE runtime environment is called a Container as it can contain several agents." A platform consists of containers, with just one Main container, with which "all other containers register … as soon as they start." The Directory Facilitator "provides a Yellow Pages service by means of which an agent can find other agents providing the services he requires in order to achieve his goals." [42]

The buying and bidding of energy will be done through the FIPA Contract Net Interaction Protocol as is shown in Figure 4. The producer will play the initiator role, while the consumer will be in the participant role. Every interaction is started by the Initiator asking for the Directory Facilitator for Consumers who are interested in buying and making a "call for proposal" (CFP) request to them. The Participant must reply in a given deadline with either a "refuse" or a "propose" response message. The "proposed" messages, in turn, must be accepted or rejected by the Initiator. Finally, the Participant must confirm with a "failure", "inform-done" or "inform-result" message (the latter two of which can be considered equal, with "inform-result" also containing some

[1] It is important to note here that if a new solution is implemented (such as sharding proposed by Vitalik Buterin [47]), which would enable a blockchain to increase the scalability without significantly sacrificing decentralization and security, the data storage will be less of an issue.

information), upon receiving which, the Initiator can consider the interaction to be done. [43]

The legend for Figure 4 is the following:

1) m – the number of CFP requests sent by the Initiator

2) n – the number of responses from the Participant in total

3) i – the number of responses that are “refused”

4) j – the number of responses that are “proposed”

5) k – the number of answers with “propose” messages rejected, sent by the Initiator

6) l – the number of answers with “propose” messages accepted, sent by the Initiator

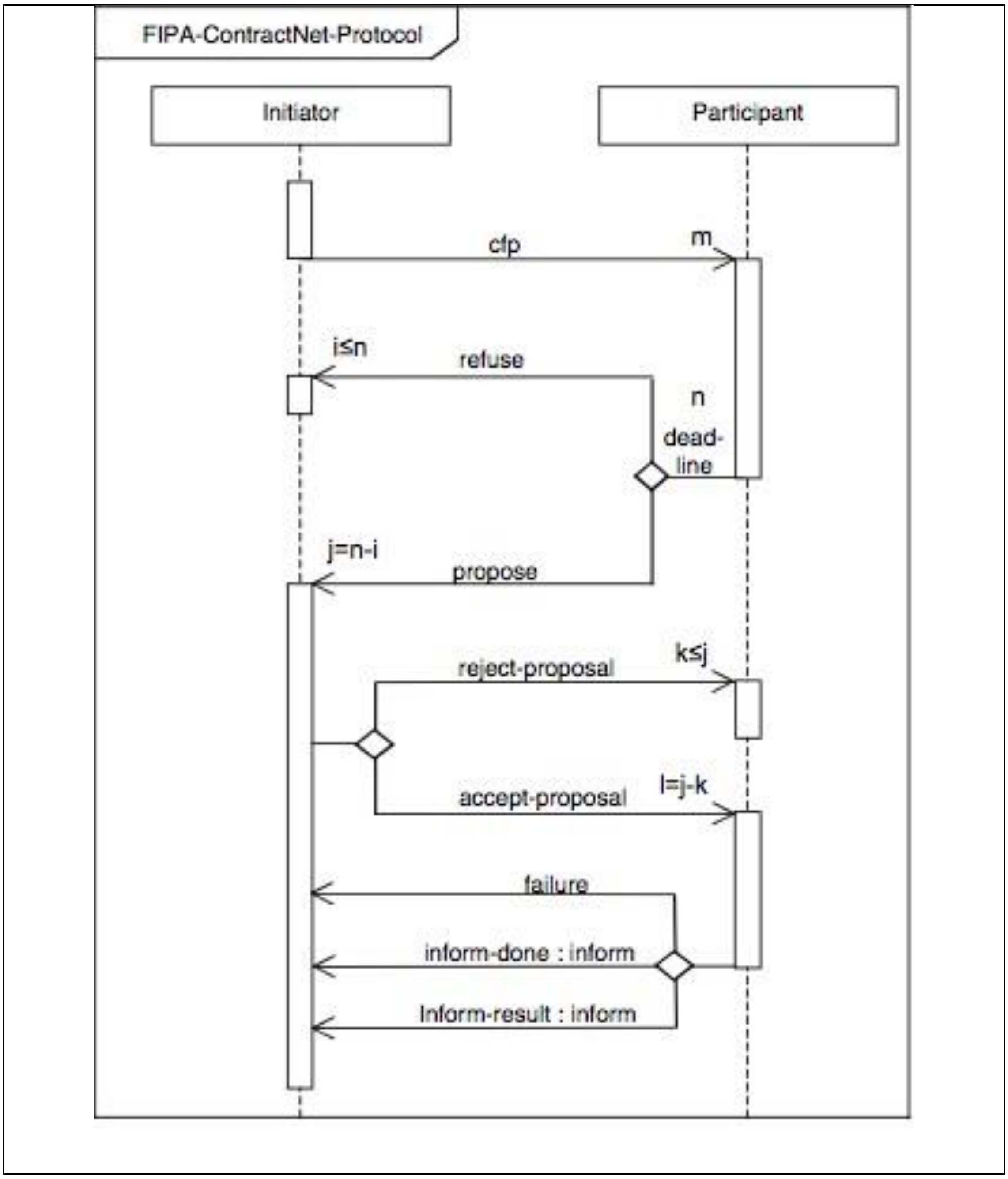

Figure 4 FIPA Contract Net Interaction Protocol [43]

5 Market Setup

5.1 Asymmetric Markets

We expect the energy market to be asymmetric, where “each bidder tries to maximize his expected profit,” and not symmetric, where “the bid functions of all bidders are identical” [44, pp. 171-172]. This is because it is unlikely all bidders have the same type of technology, which means that they cannot produce the same amount of energy or derive the bid function the same way in the real world. Because of these two reasons, the bid functions of all bidders cannot be identical.

For an asymmetric P2P energy market, neither simulation or theory can give final answers, but simulation can give some insight how the market could operate using blockchains and software agents. Asymmetric markets – i.e., a market, where each energy producers offer different amounts and/or ask a different price for it – are difficult to analyse, since the probability of a bidder being chosen by a customer is not something that can be predicted with enough accuracy for this prediction to be of practical use. This is, besides other reasons, because both bidders and buyers can adopt various types of strategies and these cannot be known about in advance. Therefore, simulation only proves that the setup works for the market, not that the strategy itself is useful.

5.1.1 The Three Types of Asymmetric Markets

To simulate the P2P energy market, we need an algorithm that would enable the producers to bid to their customers and react to changes in the market conditions. Rosen and Madlener [44] look at three types of asymmetric markets. The markets are discerned by the amount of information the bidders have access to.

In the first market, only the “total supply and aggregated price curve of accepted bids” [44, p. 173] is available. In the second, all the bids that the consumers have accepted, are known. In the third, the bidders only know whether their bids were successful or not.

The first type of market is not implementable in a peer-to-peer way for Ethereum. It requires that either there exists a central authority, who holds all the information about the winning bids – which would not be desirable for a P2P energy market solution –, or all the data is stored inside a smart contract’s storage variable so that only a smart contract knows about the bids in detail and can be programmed to give out summary information about the bids. The latter option is not very cost effective for an Ethereum full node blockchain (which is what we use to run our P2P energy market), as described in chapter 4.1. It is also possible to see all the data stored inside a smart contract in Ethereum [45], which would defeat the purpose of storing data inside the smart contract. Therefore, we will not consider trying to implement the first type of market.

The second type of market is implementable without any foreseeable issues. As stated before in chapter 4.1, it is more cost effective, if blockchain logs are used to store information related to the winning bids, not smart contracts storage variable.

The third type of market, as Rosen and Madlener note [44], can be used to encourage competition and avoid collusion. However, since all of data is publicly available, this would mean that more detailed information would not be stored on the blockchain which in turn would make it hard to verify the transaction later. Therefore, we will not consider this type of market for implementation.

5.2 Learning Strategy for Software Agents

For the learning strategy, we will need an algorithm. Rosen and Madlener [44] provide a learning strategy for bidders (energy producers) to try to maximize their profits in the asymmetric markets, which we described above. The algorithms for the learning strategy, which contains both a bidding and a price determination algorithm, can be seen on Figure 5 and on Figure 6 below. $\mu_{discount}$, $\sigma_{discount}$ and b (discount factor) are determined based on the preference of the seller.

For determining the theoretical profit, we use a simplification of Rosen and Madlener's equation [44]. Equation 1 gives the Expected profit of an individual bidder for a round. From this, we derive Equation 2, where we have removed the probability function, since it is not possible to calculate the probability of winning the bid for a quantity q with enough accuracy for it to have practical use since the probability of a bidder being chosen by a customer is not predictable in the real world. Equation 2 tells us how to calculate the theoretical profit of an individual bidder ($\pi_{theoretical,i}$) for the i-th successful bid in the last round (the use of this theoretical profit is depicted in Figure 5). We will simplify this algorithm even more and arrive at Equation 3, removing the cost, since our experiments do not rely on real word data and there would be no benefit in figuring in cost.

The legend for the variables in Equation 1, Equation 2 and Equation 3 is the following:

1) $E(\pi_i)$ – expected profit of an individual bidder for a round

2) $q_{i,l}$ – the quantity offered by the i-th bidder for the highest price

   $l = \sum_{i=1}^{y} l_i$ – the total number of bids submitted by bidders, where y is the number of bidders

   l_i – the number of total bids that the i-th bidder makes (this also includes a “mandatory” nil offer – that is, both price and quantity are zero)

   $q_i = \sum_{d=1}^{l_i} q_{i,d}$, where $q_{i,d}$ is the quantity that the i-th bidder offers for their d-th lowest price

3) $p_i(q)$ – prices offered by the i-th bidder for each of the l_i bids

4) $c_i(q)$ – (a very broadly defined) cost of offering the buyers energy for quantity q that includes technology-related costs, energy generation costs, as well as opportunity costs that rise with quantity

5) $f(q)$ – probability of winning the bid for a quantity q

6) $\pi_{theoretical,m}$ – the theoretical profit of an individual bidder for the i-th successful bid in the last round

7) q_m – the quantity offered in the m-th bid in the last round

$$\boxed{E(\pi_i) = \int_0^{q_{i,l}} f(q)(p_i(q) - c_i(q))\, q dq}$$

Equation 1 Expected profit of an individual bidder for a round offered by Rosen and Madlener [44, p. 171]

$$\boxed{\pi_{theoretical,m} = p_i(q_m) - c_i(q_m)}$$

Equation 2 Theoretical profit of an individual bidder for m-th bid, derived from Equation 1

$$\boxed{\pi_{theoretical,m} = p_i(q_m)}$$

Equation 3 Theoretical profit of an individual bidder for m-th bid, derived from Equation 2

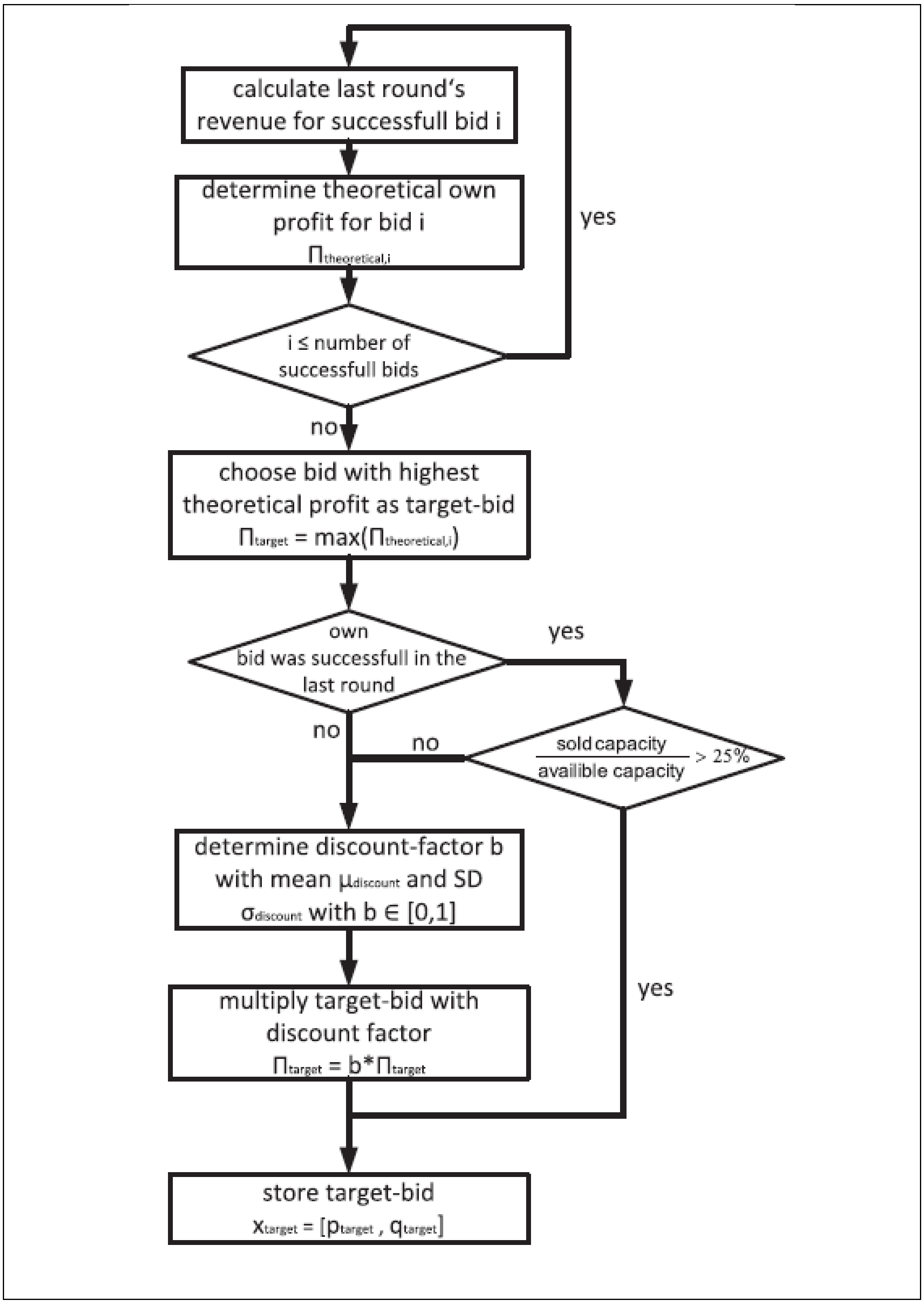
calculate last round's revenue for successfull bid i
determine theoretical own profit for bid i
$\Pi_{theoretical,i}$
i ≤ number of successfull bids
yes
no
choose bid with highest theoretical profit as target-bid
$\Pi_{target} = max(\Pi_{theoretical,i})$
own bid was successfull in the last round
yes
no
no
sold capacity / availible capacity > 25%
yes
determine discount-factor b with mean $\mu_{discount}$ and SD $\sigma_{discount}$ with $b \in [0,1]$
multiply target-bid with discount factor
$\Pi_{target} = b*\Pi_{target}$
store target-bid
$x_{target} = [p_{target}, q_{target}]$

Figure 5 The price determination algorithm for energy producers in an asymmetric market, offered by Rosen and Madlener [44, p. 174]

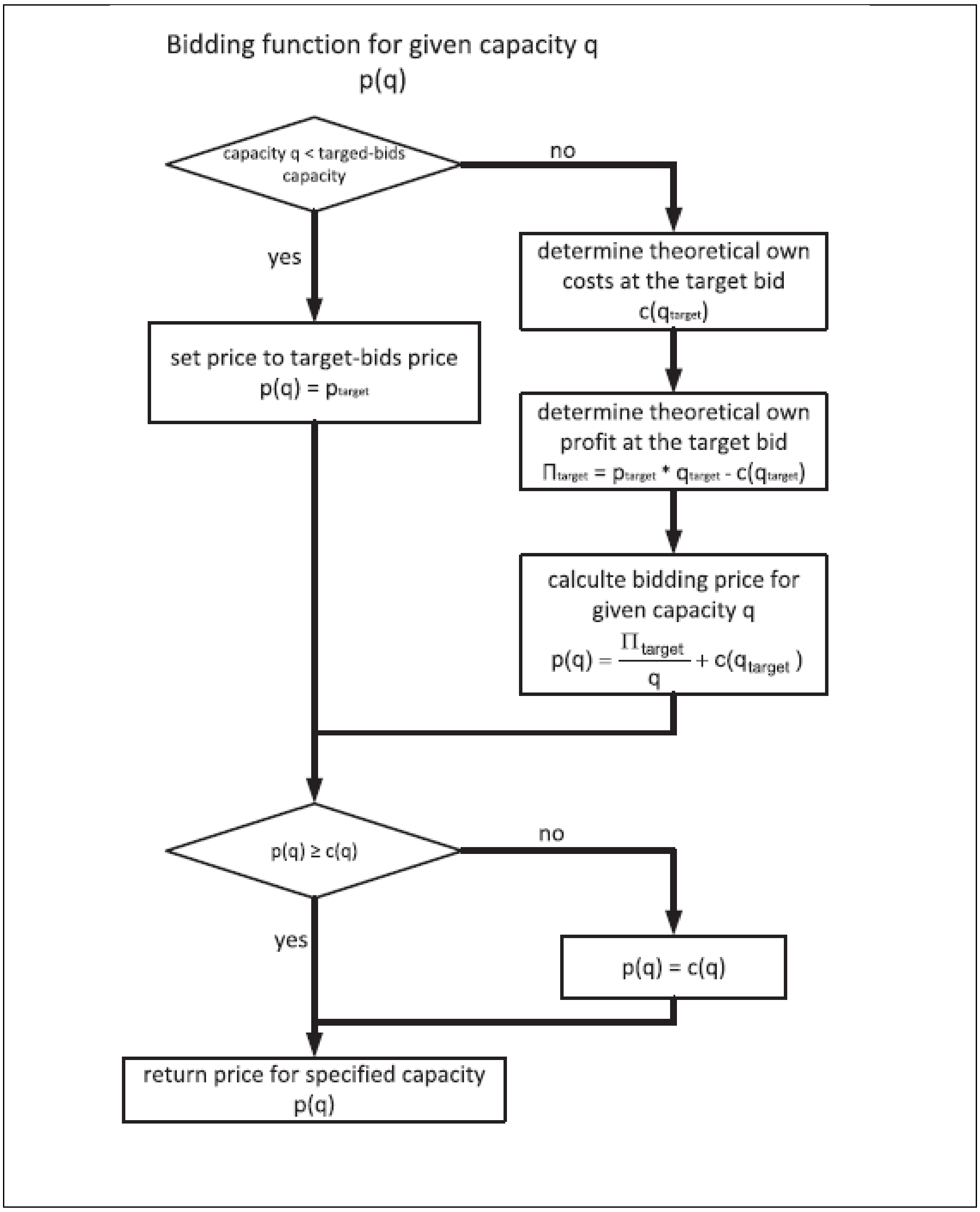
Bidding function for given capacity q
p(q)
capacity q < targed-bids capacity
no
yes
determine theoretical own costs at the target bid
$c(q_{target})$
set price to target-bids price
$p(q) = p_{target}$
determine theoretical own profit at the target bid
$\Pi_{target} = p_{target} * q_{target} - c(q_{target})$
calculte bidding price for given capacity q
$p(q) = \frac{\Pi_{target}}{q} + c(q_{target})$
$p(q) \geq c(q)$
no
yes
$p(q) = c(q)$
return price for specified capacity
p(q)

Figure 6 The bidding function algorithm for energy producers in an asymmetric market, offered by Rosen and Madlener [44, p. 174]

5.3 Demand-Side Management

Demand-side management will not be added to this algorithm. This is because it is difficult to predict to what extent people would use such a feature – it is unlikely that a person would be comfortable with, for example, switching from watching TV from 7 PM to 8 PM to watching it from 2 AM to 3 AM, just to save money. The other issue is that it would require granular data about the usage of energy by the consumer, which might be obtained if we were better funded.

6 Simulation

6.1 Modifiable Variables

For the simulation test cases, we made it possible to modify the following variables:

1) The discount factor (b) for both consumer/buyer and producer/seller/bidder agents.
2) The number of agents that would participate in a round (agents per round).
3) The total number of agents per test case (agents in total).
4) Rounds per test case

In addition,

1) For each consumer we set the highest ratio at which they were willing to buy (consumer's highest ratio).
2) For each producer, we set the lowest ratio at which they were willing to sell (producer's lowest ratio).

Price to quantity function is defined in Equation 4, where r is price to quantity function, q is the amount of energy the producer is willing to sell and p the price they are willing to pay. While theoretically the ratio should be a nonlinear equation, it will not affect our results to have a linear equation, if we set the limits in highest and lowest ration, as described above. This is because having limits serves the same purpose as a non-linear equation exponentially increasing or decreasing the output value to indicate resource limits of a producer or a consumer.

$$p = r(q), \text{where } \mathrm{r} = \frac{\mathrm{p}}{\mathrm{q}}$$

Equation 4 Price to quantity function

6.2 Test Cases

For the simulation, we will be testing two different cases. In both cases we set the following constraints for variables:

1) Discount factor for all consumer at 90%.

2) Both the consumer's highest and producer's lowest ratio were pseudo randomly generated [1] between integers of 10 and 20.

3) Both the quantity offered by the producer and bought by the consumer were pseudo randomly generated between integers of 1 and 100

We used a pseudo random integer generator, because the distribution of numbers is difficult to predict, without having real world data.

In the first case the simulation ran for 10 rounds and for each round 50 new producer agents and 50 new consumer agents were introduced with the same wallet address. The agents that were unsuccessful continued bidding until they succeeded or the simulation test case ended – this means that there were cases where two agents or more agents with the same wallet address were bidding.

In the second case we introduced 10 producer and 10 consumer agents and the simulation ran for 100 rounds.

We ran these two test case simulations, because simulation was not able to run 100 agents for 100 rounds and produce results in a reasonable amount of time. We wanted to see what happened with a lot of agents participating in the bidding process and how reliable the algorithm is in the long run. However, when we tried to run 100 agents for 100 rounds but were faced with a speed bottleneck that grew exponentially and would mean that the rounds would not be completed in a reasonable amount of time (2 minutes). The bottleneck was introduced from accessing the Ethereum logs – every unique wallet needed to create a unique connection to the Ethereum blockchain. This access needed to

[1] In all cases, for pseudo randomness we used Java's ThreadLocalRandom's nextInt method [46].

be managed by only a few semaphores, in order to avoid out of memory issues. Since we did not see a way around the issue, we settled for two test cases instead of just one.

6.2.1 First test case simulation

Figure 7 and Figure 8 show how quantity and price were distributed in each of the rounds. The fluctuation of both the price and quantity stems from the number of successful bids as can be seen on Figure 9 – the sum of price in Figure 7, the sum of quantity in Figure 8 and the successful bids in Figure 9 form nearly the same shape. The first round is the only round in which the algorithm does not rely on best bid data to determine the price and quantity to offer.

The simulation with 50 producer agents and 50 consumer agents seems to have suffered from the same bottleneck issues we mentioned earlier in the beginning of chapter 6.2. The sum of price dropped by 75% in the third round, by which time there were at least 95 active agents. Concurrently handling more than about 50 agents seems to have created issues for the simulation, where results cannot be reliably assessed. The simulation would need to be tested using more than one computers, in order to guarantee that the speed bottleneck is not creating the issue and that these graphs are the results of a bad learning algorithm, for which we lack the resources.

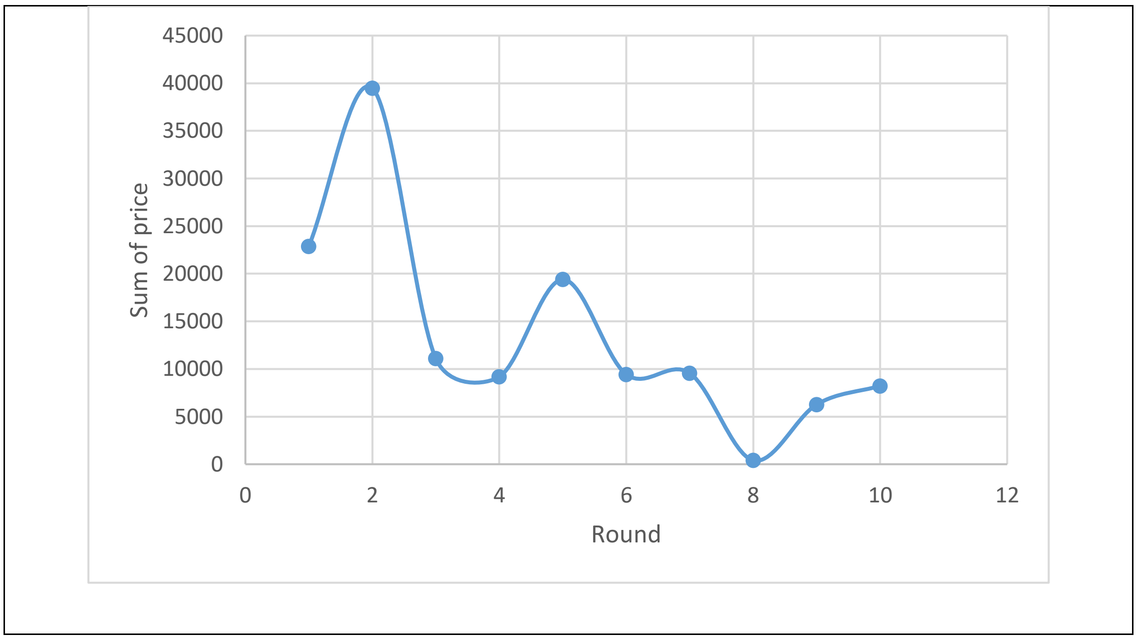

Figure 7 Sum of price in the first test case simulation

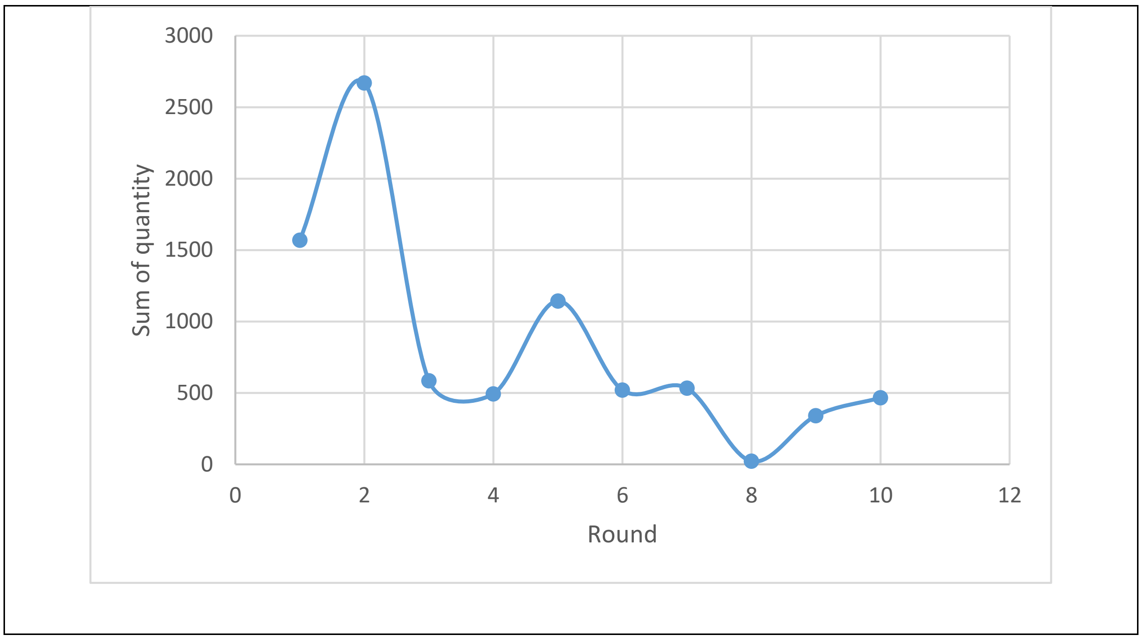

Figure 8 Sum of quantity in the first test case simulation

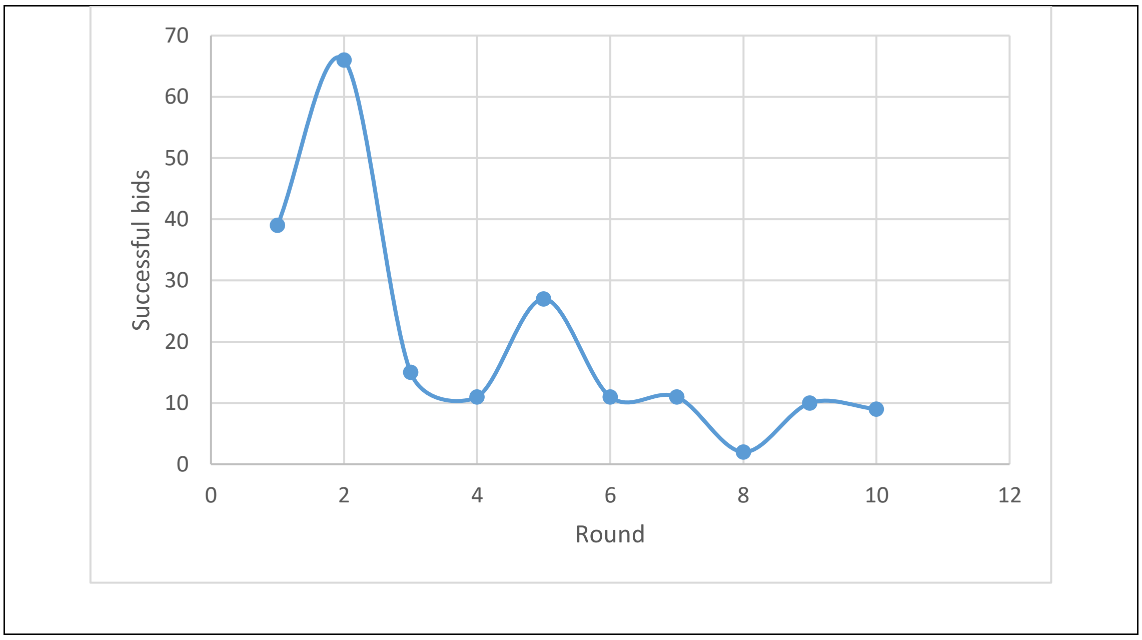

Figure 9 Successful bids in the first simulation test case

6.2.2 Second test case simulation

In the second case simulation, where we had only 10 agents per round, the bidding algorithm seems ineffective at winning bids. Figure 10 shows that the number of successful bids fluctuates from round to round, not achieving better results compared to previous rounds for more than one round. Although the quantity and price per successful bid is higher in the second half, the overall results compared between first and second half prove the first half to be more successful, as can be seen in Table 5.

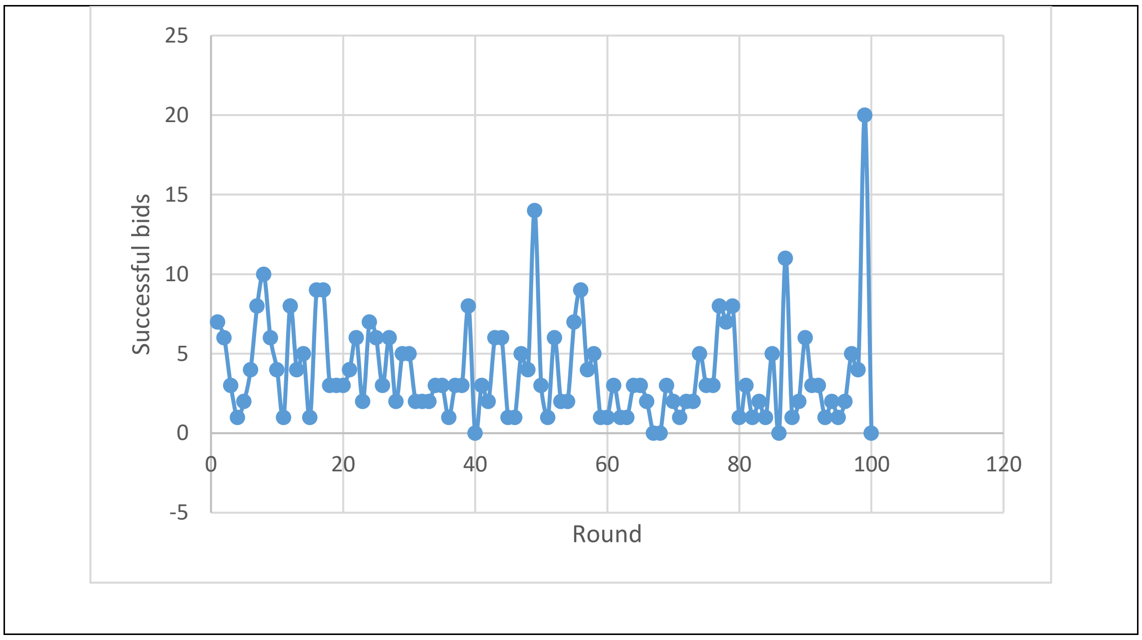

Figure 10 Successful bids in the second simulation test case

	Sum of quantity	Sum of price	Successful bids
First half	8710	140202	212
	Sum of quantity	Sum of price	Successful bids
Second half	7737	120527	172
Ratio	0.89	0.86	0.81

Table 5 Second test case simulation results for first and second half

6.3 Conclusion

The first test case simulation showed us that with more than 100 agents concurrently bidding, the system is no longer able to match buyers and sellers effectively, failing to adhere the search principle, which means that the connection to the blockchain bottleneck needs to be addressed to simulate such a system for larger number of agents.

The second case simulation showed that the algorithm proposed by Rosen and Madlener [44] is not very efficient at winning bids, at least, with a small number of agents bidding, and should also be considered to be replaced with a smarter algorithm or a reinforcement learning approach.

7 Summary

The electricity grid is going through changes related to the introduction of renewable energy sources and electric vehicles, while energy production is becoming more decentralized. This decentralization needs new kind of approach to the grid that allows for local self-management of energy and measures energy usage in more detailed way, which traditional, centralized systems cannot offer. This type of smart grid also needs an efficient market, meaning it must match buyers and sellers effectively (the search principle), be set up to ensure that prices will be set competitively (the pricing principle) and, finally, ensure that transactions are secure and reliable for buyers and sellers (the trust principle). In this thesis, we tried to figure out how to build such a market using software agents and blockchains.

Contrary to the research we gave an overview in the Related Research in chapter 2, we used smart contracts to ensure the security and reliability of transactions, by storing data on Ethereum's logs, which are verified just like data on storage variables inside smart contracts, but is cheaper to store.

To match buyers and sellers effectively and ensure that prices will be set competitively, we used software agents. The software agents ran on Java Agent Development Framework (JADE), communicating using the FIPA Contract Net Interaction Protocol.

In order to test whether such a system indeed could provide a market that adheres to the search, pricing and trust principles, we ran two test case simulations. The first test case simulation showed us that with more than 100 agents concurrently bidding, the system is no longer able to match buyers and sellers effectively, failing to adhere the search principle, which means that the connection to the blockchain bottleneck needs to be addressed to simulate such a system for larger number of agents. The second case simulation showed that the algorithm proposed by Rosen and Madlener [44] is not very efficient at winning bids, at least, with a small number of agents bidding, and should also be considered to be replaced with a smarter algorithm or a reinforcement learning approach.

References

[1] Electric Power Research Institute, "Distributed Energy Resources," [Online]. Available: http://www.epri.com/Our-Work/Pages/Distributed-Electricity-Resources.aspx. [Accessed 22 March 2017].

[2] U.S. Department of Energy, "Smart Grid Asset Descriptions," [Online]. Available: https://www.smartgrid.gov/files/description_of_assets.pdf. [Accessed 14 March 2017].

[3] Global Smart Grid Federation, "Smart Grids," [Online]. Available: http://www.globalsmartgridfederation.org/smart-grids/. [Accessed 2 March 2017].

[4] W.-Y. Chen, T. Suzuki and M. Lackner, Handbook of Climate Change Mitigation and Adaptation, Springer International Publishing Switzerland, 2016.

[5] L. Einav, C. Farronato and J. Levin, "Peer-to-Peer Markets," *Annual Review of Economics*, vol. 8, p. 615–635, 2016.

[6] L. S. Sterling and K. Taveter, The Art of Agent-Oriented Modeling, London: The MIT Press, 2009.

[7] I. Weber, X. Xu, R. Riveret and G. Governatori, "Untrusted Business Process Monitoring and Execution Using Blockchain," in *International Conference on Business Process Management*, 2016.

[8] R. Hull, V. S. Batra, Y.-M. Chen and A. Deutsch, "Towards a Shared Ledger Business Collaboration Language Based on Data-Aware Processes," *International Conference on Service-Oriented Computing*, pp. 18-36, 2016.

[9] T. S, E. J and K. M, "Not ACID, not BASE, but SALT-a transaction processing perspective on blockchains," in *Proceedings of the 7th International Conference on Cloud Computing and Services Science*, Porto, 2017.

[10] Ethereum Foundation, "Ethereum Project," [Online]. Available: https://www.ethereum.org/. [Accessed 21 February 2017].

[11] Ethereum Foundation, "Ethereum White Paper," 25 December 2016. [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper. [Accessed 21 February 2017].

[12] V. Morabito, "Smart Contracts and Licensing," in *Business Innovation Through Blockchain*, Springer International Publishing, 2017, pp. 101-124.

[13] J. Stark, "Making Sense of Blockchain Smart Contracts," 4 June 2016. [Online]. Available: http://www.coindesk.com/making-sense-smart-contracts/. [Accessed 15 January 2017].

[14] J. Bremer and S. Lehnhoff, "Decentralized Coalition Formation," in *International Conference on Practical Applications of Agents and Multi-Agent Systems,* Sevilla, 2016.

[15] International Energy Agency, "CO2 emissions from fuel combustion HIGHLIGHTS 2016," [Online]. Available: https://www.iea.org/publications/freepublications/publication/CO2EmissionsfromFuelCombustion_Highlights_2016.pdf. [Accessed 22 March 2017].

[16] G. James, W. Peng and K. Deng, "Managing Household Wind-energy Generation".

[17] E. Ogston, A. Zeman, M. Prokopenko and G. James, "Clustering Distributed Energy Resources for Large-Scale Demand Management," in *Self-Adaptive and Self-Organizing Systems*, 2007.

[18] V. Buterin, "On Abstraction," 5 July 2015. [Online]. Available: https://blog.ethereum.org/2015/07/05/on-abstraction/. [Accessed 22 March 2017].

[19] H. Mayer, "ECDSA Security in Bitcoin and Ethereum: a Research Survey," 17 May 2016. [Online]. Available: http://blog.coinfabrik.com/ecdsa-security-in-bitcoin-and-ethereum-a-research-survey/. [Accessed 22 March 2017].

[20] N. Szabo, "Smart Contracts: Building Blocks for Digital Markets," 1996. [Online]. Available: http://www.alamut.com/subj/economics/nick_szabo/smartContracts.html. [Accessed 21 February 2017].

[21] S. Omohundro, "Cryptocurrencies, Smart Contracts, and Artificial Intelligence," *AI Matters,* vol. I, no. 2, pp. 19-21, 2014.

[22] A. R. Hevner, S. T. March, J. Park and S. Ram, "Design Science in Information Systems Research," *MIS Quarterly,* vol. 28, no. 1, pp. 5-105, 2016.

[23] N. Z. Aitzhan and D. Svetinovic, "Security and Privacy in Decentralized Energy Trading through Multi-Signatures, Blockchain and Anonymous Messaging Streams," *IEEE Transactions on Dependable and Secure Computing,* vol. PP, no. 99, 2016.

[24] European Parliament, Council of the European Union, *Directive 2009/72/EC,* 2009.

[25] M. Mihaylov, S. Jurado, K. Van Moffaert, N. Avellana and A. Nowe, "NRG-X-Change: a Novel Mechanism for Trading of Renewable Energy in Smart Grids," [Online]. Available: https://cdn.hackaday.io/files/10879465447136/mihaylov_etal_smartgreens14.pdf. [Accessed 22 March 2017].

[26] A. Amato, B. Di Martino, M. Scialdone and S. Venticinque, "Multi-agent Negotiation of Decentralized Energy," *Intelligent Distributed Computing,* vol. VIII, pp. 155-160, 2015.

[27] S. Hosokawa and N. Nishino, "New mechanisms in decentralized electricity trading to stabilize".

[28] Tilab, "Jade Site," [Online]. Available: http://jade.tilab.com/. [Accessed 21 February 2017].

[29] L. Tasquier, M. Scialdon, R. Aversa and S. Venticinque, "Agent Based Negotiation of Decentralized Energy Production," *Intelligent Distributed Computing,* vol. VIII, pp. 59-67, 2015.

[30] "SPADE 2.2.1," [Online]. Available: https://pypi.python.org/pypi/SPADE. [Accessed 13 March 2017].

[31] A. Amato, B. Di Martino, M. Scialdone and S. Venticinque, "A Virtual Market for Energy Negotiation and Brokering," in *P2P, Parallel, Grid, Cloud and Internet Computing,* 2015.

[32] "Retroshare," [Online]. Available: http://retroshare.net/. [Accessed 14 March 2017].

[33] S. Fey, P. Benoit, G. Rohbogner, A. H. Christ and C. Wittwer, "Device-to-Device Communication for Smart Grid".

[34] A. Amato, B. Di Martino, M. Scialdone and S. Venticinque, "A Negotiation Solution for Smart Grid using a fully decentralized, P2P approach," in *Complex, Intelligent, and Software Intensive Systems*, 2015.

[35] A. R. Alkhawaja, L. L. Ferreira, M. Albano and R. Garibay, "QoS-enabled Middleware for Smart Grids," IPP Hurray! Research Group, 2012.

[36] V. Karagiannis, P. Chatzimisios, F. Vazquez-Gallego and J. Alonso-Zarate, "A Survey on Application Layer Protocols for the Internet of Things," [Online]. Available: https://www.researchgate.net/profile/Periklis_Chatzimisios/publication/303192188_A_survey_on_application_layer_protocols_for_the_Internet_of_Things/links/577b656608ae213761c9d91d.pdf. [Accessed 12 December 2018].

[37] N. Capodieci, G. Cabri, G. A. Pagani and M. Aiello, "Adaptive Game-based Agent Negotiation in Deregulated Energy Markets," in *Collaboration Technologies and Systems*, 2012.

[38] U.S. Department of Energy, "Benefits of Demand Response in Electricity Markets and Recommendations for Achieving Them," February 2006. [Online]. Available: https://emp.lbl.gov/sites/all/files/report-lbnl-1252d.pdf. [Accessed 11 March 2017].

[39] ConsenSys, "Technical Introduction to Events and Logs in Ethereum," [Online]. Available: https://media.consensys.net/technical-introduction-to-events-and-logs-in-ethereum-a074d65dd61e. [Accessed 20 Januar 2019].

[40] "ETH Gas Station," [Online]. Available: https://ethgasstation.info/calculatorTxV.php. [Accessed 16 May 2019].

[41] Ethereum Foundation, "Contracts," [Online]. Available: https://solidity.readthedocs.io/en/develop/contracts.html. [Accessed 2 December 2017].

[42] Tilab, [Online]. Available: http://jade.tilab.com/doc/tutorials/JADEProgramming-Tutorial-for-beginners.pdf. [Accessed 20 January 2019].

[43] The Foundation for Intelligent Physical Agents, "FIPA Contract Net Interaction Protocol Specification," [Online]. Available: http://www.fipa.org/specs/fipa00029/SC00029H.html. [Accessed 16 May 2019].

[44] C. Rosen and R. Madlener, "An auction design for local reserve energy markets," *Decision Support Systems*, vol. 56, p. 168–179, 2013.

[45] Ethereum, "Solidity Contracts," [Online]. Available: https://solidity.readthedocs.io/en/v0.4.21/contracts.html#visibility-and-getters. [Accessed 20 January 2019].

[46] Oracle, "ThreadLocalRandom," [Online]. Available: https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/concurrent/ThreadLocalRandom.html#nextInt(int,int). [Accessed 16 May 2019].

[47] V. Buterin. [Online]. Available: https://github.com/ethereum/wiki/wiki/Sharding-FAQ. [Accessed 20 January 2019].