applied sciences

MDPI

Article

Machine Learning-Blockchain Based Autonomic Peer-to-Peer Energy Trading System

Yaçine Merrad [1,2], Mohamed Hadi Habaebi [1,2,*], Md. Rafiqul Islam [1,2], Teddy Surya Gunawan [2], Elfatih A. A. Elsheikh [3], F. M. Suliman [3] and Mokhtaria Mesri [4]

1 IoT & Wireless Communication Protocols Laboratory, Department of Electrical & Computer Engineering, International Islamic University Malaysia, Gombak, Kuala Lumpur 53100, Malaysia; yacinechoupot@yahoo.fr (Y.M.); rafiq@iium.edu.my (M.R.I.)
2 Department of Electrical & Computer Engineering, International Islamic University Malaysia, Gombak, Kuala Lumpur 53100, Malaysia; tsgunawan@iium.edu.my
3 Department of Electrical Engineering, College of Engineering, King Khalid University, Abha 61421, Saudi Arabia; eelsheikh@kku.edu.sa (E.A.A.E.); fmsuliman@kku.edu.sa (F.M.S.)
4 Department of Electronics, University Amar Télidji of Laghouat, BP37G, Laghouat 03000, Algeria; m.mesri@lagh-univ.dz
* Correspondence: habaebi@iium.edu.my

Abstract: This paper introduces a blockchain-based P2P energy trading platform, where prosumers can trade energy autonomously with no central authority interference. Multiple prosumers can collaborate in producing energy to form a single provider. Clients' power consumption is monitored using a smart meter that interfaces with an IoT node connected to a blockchain private network. The smart contracts, invoked on the blockchain, enable the autonomous trading interactions between parties and govern accounts behavior within the Ethereum state. The decentralized P2P trading platform utilizes autonomous pay-per-use billing and energy routing, monitored by a smart contract. A Gated Recurrent Unit (GRU) deep learning-based model, predicts future consumption based on past data aggregated to the blockchain. Predictions are then used to set Time of Use (ToU) ranges using the K-mean clustering. The data used to train the GRU model are shared between all parties within the network, making the predictions transparent and verifiable. Implementing the K-mean clustering in a smart contract on the blockchain allows the set of ToU to be independent and incontestable. To secure the validity of the data uploaded to the blockchain, a consensus algorithm is suggested to detect fraudulent nodes along with a Proof of Location (PoL), ensuring that the data are uploaded from the expected nodes. The paper explains the proposed platform architecture, functioning as well as implementation in vivid details. Results are presented in terms of smart contract gas consumption and transaction latency under different loads.

Keywords: blockchain; decentralization; Ethereum; K-mean clustering; GRU prediction model; peer-to-peer energy trading; proof of location; smart contract; dynamic Time of Use; transparency

check for updates

Citation: Merrad, Y.; Habaebi, M.H.; Islam, M.R.; Gunawan, T.S.; Elsheikh, E.A.A.; Suliman, F.M.; Mesri, M. Machine Learning-Blockchain Based Autonomic Peer-to-Peer Energy Trading System. *Appl. Sci.* **2022**, *12*, 3507. https://doi.org/10.3390/app12073507

Academic Editor: Agostino Forestiero

Received: 3 March 2022
Accepted: 24 March 2022
Published: 30 March 2022

1. Introduction

Since 2008, when Bitcoin has emerged as the first label of blockchain application for decentralized currency [1], it has opened the eyes to a new paradigm which is decentralization, and its potential as a new disruptor in various industries. After the emergence of Ethereum and the smart contract concept, blockchain potential for enabling decentralized applications beyond finance has been unlocked. Since then, many companies have tried to adopt this technology in different sectors varying from e-voting, supply chain tracking, health care, agriculture, and so on. Another sector where blockchain technology is presenting itself as a new revolution is the Energy market [2]. In the recent past, in traditional centralized electric power systems, customers have been at the end of the supply chain. With the advent of smart grids, we are witnessing the disclosure of distributed energy

resources (DERs) based on renewable energies that can contribute to the electricity system. This led to a paradigm shift in the energy trading architecture and the emergence of new players in the market, where consumers are becoming "prosumers" who may sell their surplus of energy to neighboring consumers. Power is produced using rooftop solar PV installations, micro wind generators, and battery energy storage systems [3,4]. This new concept of prosumers ushered in a new era of what is known as Peer-to-Peer (P2P) energy trading, where generating nodes can enter the energy market as providers. However, it is difficult for parties involved in the trading to trust each other without the supervision of a central entity or any other middle broker organization. Providers cannot guarantee being paid for the energy they supply to the client, and clients cannot guarantee being supplied the requested energy if paying in advance. In such a scenario, the basic value of coupling energy with cryptocurrencies and blockchain technology [5] is a staple that enables the forcible payments and energy supply through a distributed smart contract [6]. Being embedded in the blockchain, smart contracts are reshaping the wave of innovation in today's technology in terms of automatic agreements between two parties, while excluding the intervention of a trusted third party. They allow services costs saving and reduce the risks of potential corrupt central entities [7]. In this work, we are presenting a P2P energy trading platform where energy can be traded in a decentralized manner on an Ethereum private blockchain. The energy traded on the platform is electrical energy, thus the two terms energy and electricity as well as client and consumer will be used interchangeably in this paper. The key features which the architecture of the Proposed P2P energy trading system involve:

- Dynamic Time of Use pricing: This would maximize profit and enable consumers to get an appropriate and accurate demand response. The tariffs are proposed ahead of each trading period, based on energy consumption predictions using machine learning techniques [8]. The tariffs are then implemented in the smart contract to enable autonomous billing. A new smart contract is created ahead of each upcoming trading round, with updated tariffs based on the trading round electricity consumption predictions. Moreover, the tariffs are made known immediately after each trading round and their validity can be verified by all participants, since each one would be having his copy of the shared ledger containing the training set data. Indeed, each participant is aware of the state history representing the energy consumption data that is used in the prediction process. Based on the predictions, ToU ranges are defined using the K-mean clustering algorithm, implemented on a smart contract, and executed on the blockchain. This ensures that the ToU ranges are reliable and bias-free, since they are defined by an independent autonomous smart third party.
- Prevention against fraudulent behavior: Considering energy is traded online, it is critical to ensure that the same energy is not sold more than once. Clients must as well upload valid data of their energy consumption. To achieve such a purpose, we propose in the present paper, that the DSO plays the role of a privileged participant, which is allowed certain credentials, making our proposed solution autonomic in nature instead of fully autonomous. First, only approved nodes can join the trading platform as consumers. These nodes need to be certified by the DSO before being able to subscribe to any trading round. Any uncertified accounts have their subscription requests automatically denied. The DSO performs random checks on consumer nodes to detect any fraudulent or technically faulty ones. The DSO is the only agent given the credential to decertify accounts accordingly. Moreover, a customized consensus algorithm is proposed to spot suspicious nodes. Furthermore, a PoL protocol is implemented to confirm the aggregated data source.
- Automatic and autonomic running: Ethereum blockchain is used to implement the trading procedure through smart contracts, ensuring the forcible payment when demand is supplied, the forcible balance return in case of premature un-subscription, as well as the independent smart contract-based billing. Along with this research work; the rest of the paper is organized as follows: In Section 2, we expand the

related works. Section 3 highlights the motivation behind the present work and the raised challenges while outlining some noticed gaps from the covered literature concerning P2P energy trading and the way the proposed platform is bridging these shortcomings. The Proposed P2P energy trading system with the machine learning-based dynamic pricing mechanism is extensively explained in Section 4. This section also details how ToU ranges are defined employing GRU time series forecasting and K-mean clustering. In Section 5 we describe the implementation on a private Ethereum blockchain, on which the platform performance is assessed and the results are presented and discussed. Section 6 concludes this paper.

2. Related Work

The peer-reviewed related studies may be broken mainly into two parts. The first part concerns blockchain backgrounds as a revolutionary technology that has shaped the future of banking and IoT, with the appearance of smart contracts and several distributed consensus protocols and beyond. In this regard, the advent of blockchain technology has been outlined for the first time in 1991 by Stuart Haber and W. Scott Stornetta, to implement a system that inhibits any form of tampering with document timestamps. Moreover, it has allowed digital information to be recorded, distributed, and securely exchanged. However, this would not be detailed more in our paper since a plethora of substantial works is generously provided in the literature [1,2]. The second wedge, for its part, gives a fuller insight into the most important features utilizing blockchain technology for the energy sector [3].

2.1. Incentives and Context

The threat of rising energy bills is of significant concern for businesses, worldwide. P2P-based energy trading represents a challenge that can mitigate the specific electricity consumption pricing at a given site. Meanwhile, renewable and distributed energy resources are definitely at the core of any business sustainability schedule to ensure high investment return prospects [4–6]. Peer-to-peer energy trading gives prosumers a real alternative for their electricity supply as well as the choice to decide whom they sell it to. Prosumers are required to communicate with each other for negotiating energy prices and payment transactions. Exporting back to the grid excess energy, for a small charge rate has become old-fashioned as modern and future-oriented energy systems allow clients to manage and control profitably the way their resources are distributed using microgrids and how this can affect their electricity bills. Recorded P2P energy trade goes back to the year 2016, in Brooklyn, New York, when Ethereum blockchain was used to sell extra generation of a few kilowatt-hours from a resident's solar panels energy to his neighbor. Several residential trials have followed since, whereas some companies are rushing in the P2P trade across the world. Global enterprise companies including LO3, SonnenFlat, Power Ledger, Grid+, Suncontract, and Eemnes Energie have tangible use cases for blockchain technology using platforms that have drastically changed and revolutionized the way energy is shared and distributed [7]. In addition to sustainability, suitability and profitability also play a decisive role in any new solution, which represents an active topic of research.

2.2. Energy Trading Using Blockchains

In this subsection, we will be highlighting the major role of blockchain in the use of energy trading [8–10]. P2P trading being a new trend in technology, requires overwhelming adoption by a large mass to claim success. This is achieved only if the needs and the possibilities of the individual are seriously taken into account. Using blockchain combined with smart contracts for reliable tamper-proof transactions are catching particular interest among the researchers' community and industry. As far as smarter and stronger feasible technologies with enhanced capacity and flexibility for power transmission and distribution (T & D) systems are concerned [11], the blockchain-based technology can be identified in two main classes of particular interest [12]; the first class, referred to as Electrical

energy trading, comprises principally P2P energy sales where today's energy systems are changing in ways that make old strategies obsolete [13–15]. The second class can be observed through the role of Certification of Renewable Energy emissions, which enables the prosumer in an aggregation program to be granted recognition as a participant. Additionally, Demand Response tracking, where the blockchain records the amount of green power that is injected into the grid, manages the remuneration mechanisms and ensures transparency. Moreover, this would allow the operating point of the power system to be continuously adjusted. A thorough review that elucidates what blockchain can do for power grids is found in [16]. The authors have considered an in-depth survey of the use of blockchain-based technology in the energy sector in general and the power grid especially, through a systematic categorization according to the field of activity, regardless whether this is coming from start-up companies or big technology firms. Meanwhile, a comprehensive review [3] analyzed the current trends in the energy market and how blockchain technology can contribute to this line. The authors in [17] have reviewed the demand response in smart electricity grids equipped with renewable energy sources. A critical survey was carried out in [18] that spotlights the benefits and challenges of electrical demand response. Furthermore, the work in [19] has shown how flexibility and forecasting contribute to reducing the need for grid extension, but this would require strengthened monitoring of the distribution grids. In other works, such as in [20], the authors have proposed a smart contract-based P2P energy trading system with a dynamic pricing model. In reality, several industries are investing in blockchain technology, albeit the expectations of researchers are very high, indicating the continual challenges of blockchains' applicability, in the energy sector particularly. According to the literature reviewed in the context of blockchain-enabled energy trading, it is noted that most of the p2p blockchain energy trading proposed platforms are based on energy tokenization. Energy asset-based tokens are created upon a defined energy amount generation. Created tokens can be traded, such as brikCoin [21], or redeemed as energy, such as WePower [22]. This paradigm suffers from two drawbacks that we aim to address:

- Purchased energy under consumption: When the energy is redeemed using its corresponding energy token, consumers are not refunded in case of under-consumption. This issue can be deduced from the results presented in [23], where the author laid out the purchased energy versus its consumption using his proposed energy token, namely ForDelToks. It is seen that in most cases, the energy purchased is not fully consumed. Similarly, another work presented in [24] also highlights this issue. As a matter of fact, in the result presented by the author in his proposed token-based energy trading platform, it can be noticed that the seller tokenized surplus energy is mostly under-consumed by its respective buyer. However, the author proposes that the excess energy bought be accumulated to the surplus energy generated and tokenized, when it reaches the threshold of 150 watts. This solution is not appropriate where the power is purchased by a pure consumer. In such a case, unused energy purchased is wasted.
- Token handling gas cost: Another concern in token-based energy trading systems, is the high gas cost of token manipulation to both deploy and transfer [25].
  In this direction, we propose this work to tackle the token manipulation gas cost overhead using an energy balance-based trading scheme, as well as a pay-per-use smart meter-based energy trading, as a solution to the highlighted problem under consumption.

3. Motivation and Challenges

Motivation: It is noticed through the review literature that in most of the proposed P2P energy platforms, the trading takes part between the prosumers and consumers. Prosumers' energy is injected into the power grid, which is metered and tokenized. Tokens are created using an asset-based token protocol such as the ERC20 protocol in Ethereum. A token smart contract is created with a certain amount of total supplied tokens, which are set to have an abstracted value in terms of energy. Energy aggregated to the utility power grid (by the

prosumers) is redeemed in the form of tokens, which are transferred to them. Tokens can then be traded according to their value in the market [26–28]. Accordingly, the initial smart contract defining the energy token is set with a total supply of tokens, which means, if new tokens need to be created, then, a new smart contract needs to be deployed with additional supply, which is very costly. The gas cost to deploy the EWT energy token is 1,274,198 with a cost of $171.62 [29]. Moreover, transferring tokens is costly as well. In addressing such gaps, we wish to consider in our paper that prosumers cooperate as a single provider and the energy they aggregate to the utility grid is tracked in a smart contract deployed by the DSO. This amount of energy is mapped to the issuer prosumer's account as its energy balance. Instead of a token transfer, prosumers' energy balance is updated by a mere contract call that updates a mapping, rather than a costlier financial token transfer. The energy balance of the prosumers represents the amount of energy the DSO owes them. Then, consumers can buy energy from prosumers. If the prosumer (seller) has enough energy balance to supply the demand, the payment is made from the consumer (buyer) to the seller set of prosumers. Consequently, the energy being sold is transferred from the DSO to the consumer as a debt that is paid back to the respective prosumer. His energy balance is then updated accordingly [30,31]. Mainstream token-based P2P energy trading platform bears another main drawback. In fact, in such a situation, the client does not pay per use for the energy he purchased. After the energy token is bought, it can be then redeemed as energy. However, the respective corresponding energy amount is transferred to the client and he is charged accordingly, regardless of to which extent the purchased energy is consumed. Additionally, most of the time, purchased energy is not consumed to the fullest [23]. It is as worth mentioning that some substantial research papers have been also conducted using smart contract-based electricity consumption billing system, outside the context of P2P energy trading, where household electricity bills' computation and payment are monitored by a smart contract in a traditional centralized DSO–client architecture [32–34]. In this perspective, we proposed in our research paper, a platform where consumers are billed per consumption in the same fashion the DSO bills its consumers.

Challenges: One can identify a few challenges in developing such a platform. The first challenge is to be able to accurately bill the consumers for the energy they consumed. To achieve this, prosumers need to be the only supplier from the start to the end of the trading round. Trading rounds have a certain duration, each one is administrated by a smart contract that sets the electricity tariffs. For this purpose, prosumers and microgrids' owners collaborate to form a single provider, which would be a full-time supplier to the subscribed customers for an entire trading period. The second challenge is concerned with the prosumer's power generation, which is based on renewable energy and low voltage energy sources. It is a concern that it can be not sufficient to supply for consumers on a continuous period. Yet, this could be seen as a fake problem. Actually, in a research report, published in May 2017, by the European Commission [35], they provide the total potential capacity of residential solar PVs, based on the proportion of available free space that can be used for the PVs installation. A graphical representation of residential PV electrical generation potential compared to the total national electricity generation capacity in the EU is illustrated in Figure 1 [35–44]. It can be seen that, if exploited to its fullest, energy generated by households using only PV solar panels can be tremendously significant. Hence, residential electricity generation has great potential and in some countries such as the UK, and Belgium, it can reach 50% of the total energy being generated by the respective national provider.

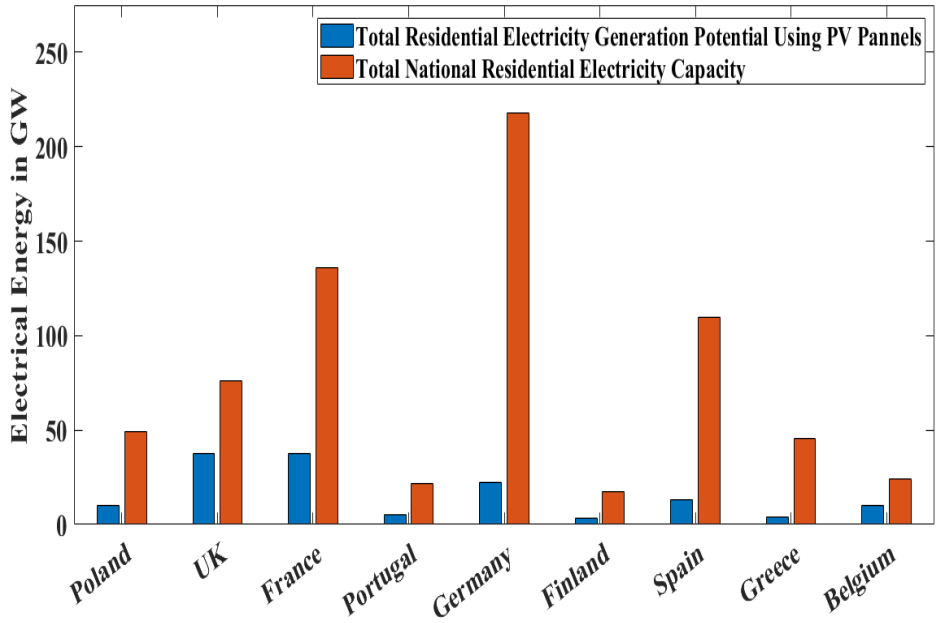

Figure 1. Residential PV electrical generation potential versus the total national electricity generation capacity in the EU.

Further studies also showed that, if prosumers formed renewable energy cooperatives, the amount of energy generation capacity would strongly resonate. In this vision, extensive research that has been conducted in [45] revealed data in favor of our claim. It is stated, that 15,000 households own shares in wind electricity generation cooperatives, which consist of 40% of the electricity generated by wind, in Denmark. Germany as well played an essential role in the development of the RE market within Europe. In this regard, community ownership is estimated to have a share of almost 20% of the total onshore wind power. Another aspect of developing a smart contract for energy trade is local legislation and the legal framework of the country where it aspires to be implemented. How each participant is considered from a legal point of view is inferred according to his role and his interactions with the rest of the participants. This implies a set of rules and regulations that they have to abide by. For example, in Germany, if a prosumer is considered a business, he has to contribute to the balancing of the grid (the German Energy Industry Act) [46] and add a withdrawal policy, according to the Consumer Rights Act in the EU [47]. These regulations must be implemented and enforced by the smart contract to ensure that each participant complies with the laws to which they are bound. A typical example is the Enerchain project [48], which is an energy trading blockchain for peer-to-peer transactions, specially designed to be compliant with the EU's Regulation on Wholesale Energy Market Integrity and Transparency (REMIT). On the other hand, some countries' legal systems, obstruct the decentralized energy P2P market. For example, according to the EU Renewable Energy Directive [49], a "renewables self-consumer" consumes local energy that is generated behind the meter. Similarly, the Dutch law requires a provider certification for the sale of power to the electricity grid [50]. In such cases, the legislative framework should be amended accordingly. In this direction, one can cite the BEST (Blockchain-based decentralized energy market design and management structures) project in Germany, which is working to establish a bidding system for the open-source electricity market supported by the German Federal Ministry for Economic Affairs and Energy [46,51,52].

On the other hand, reliability is also given priority as the third challenge. Ensuring the electricity consumption data committed by the consumers be secure, remains of considerable importance. In the following subsection, the design of the proposed platform and its architecture is vividly explained, so that the approach that is adopted in our research to solve this challenge is made clearer.

Lastly, we believe it would no doubt be useful to conclude this section with another challenge pertinent to blockchain and its applications as well as the technologies relying on current cryptographic algorithms in general, to outline future trends. Indeed, it is constructive to mention quantum computation as one of the promising directions for future development. With the expected emergence of quantum computers, current Blockchain networks would become vulnerable and need to transition to methods using quantum-resistant cryptography and quantum networks [53–55]. Current blockchain relies on cryptographic algorithms for Key-pair generation such as RSA and Elliptic-curve cryptography. The Key pair consists of a private key used by its owner to sign the transactions he issues, and the public serves to verify the digital signature by the rest of the network. Cryptographic algorithms are one-way functions, which means that current computers cannot be used to derive the private key from its matched public key. However, using a quantum computer, any current cryptographic algorithms can be broken. The generation of RSA cryptographic key pairs can be broken using Shor's quantum computing algorithm [56]. Quantum computers are still at an early stage of research, a technology that is expected to be operational over the next decade [57]. In the meantime, researchers are working to develop quantum cryptography, since the current one would not be relevant in the presence of quantum computers. This cryptography which is immuned against quantum computing is called Post-Quantum Cryptography PQC. In the context of blockchain, PQC involves the development of a quantum-resistant cryptographic signature. PQC is intended to be safe against quantum computers while being implementable on a conventional computer. Many proposals have been submitted to the National Institute of Standards and Technology NIST. However, after 3 rounds of preselection, only three finalist post quantum cryptographic digital signature algorithms remain, with one only to be selected as the new standard [58]. The three finalist are CRYSTALS-Dilithium [59], Falcon [60] and Rainbow [61].

4. Platform Design

In this section, the design of the proposed P2P energy trading platform is delineated. It first details the proposed model defining the ToU ranges using GRU time-series predictions and K-mean clustering. Then, the key entities that are involved are described, their roles underlined, and how they possibly interact with each other, as well as the set of rules governing these interactions. This section also presents the entities (Smart Contracts) that ensure that all interactions are bound by the agreed-upon rules and that each interaction results in the appropriate consensual outcome.

4.1. ToU Ranges Using GRU Model Predictions and K-Mean Clustering

The aim is to dynamically set the ToU peak and off-peak ranges. Based on past consumers' electricity consumption data available on the shared ledger blockchain, future electricity consumption predictions are then made, using the GRU model, subsequently, the ToU ranges for the next trading round are set by performing K-mean clustering on the blockchain as illustrated in Figure 2. The purpose behind it is to have a transparent, decentralized dynamic demand response.

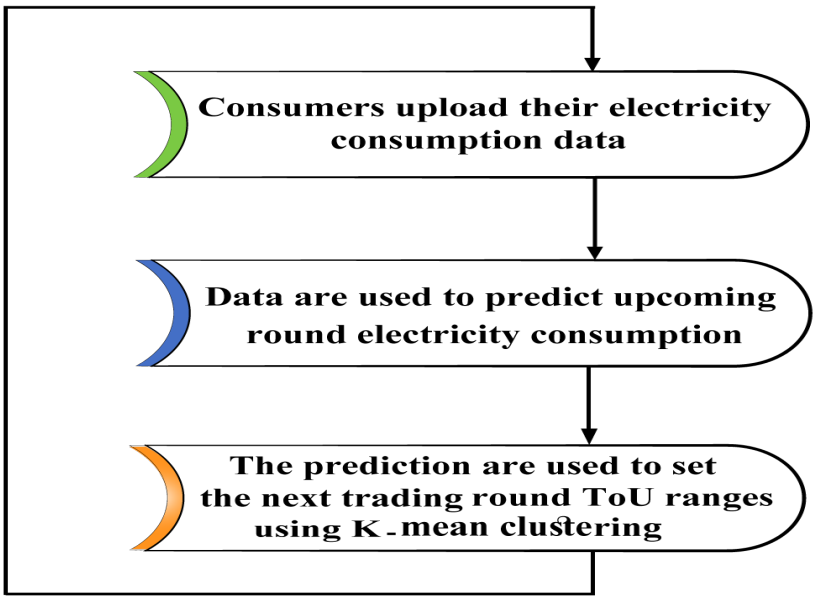

Figure 2. The proposed model defining TOU peak and off-peak ranges.

4.1.1. Energy Consumption Time Series Forecasting Using GRU

Our forecasting model is using an open-source dataset from PJM, in Megawatts [62]. The data consist of hourly records of electricity consumption in the USA for a period of time ranging from 31 December 2004 to 1 February 2018. They are structured into double columns of 121,274 values with no empty values. The energy consumption histogram provided in Figure 3 shows a normal distribution and hence, normalization of the dataset is not required.

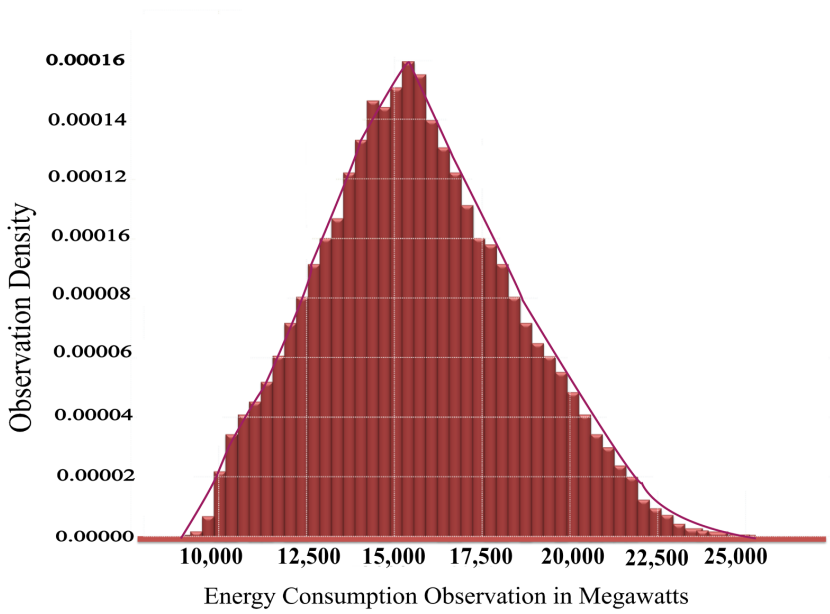

Figure 3. Distribution of the observations in the dataset.

The model is trained by utilizing the GRU network. Numerous research works including [63,64] have successfully applied the GRU network for predicting electricity consumption. The aforementioned studies have investigated the performance of several machine learning predictive models to forecast electricity consumption. Results have shown that the GRU-based method achieves more accurate predictions, making it the reason behind our choice. The implementation of the standard GRU prediction model is conducted

under the Keras framework with Tensorflow2 as the back end, using Python language. The prediction of the electricity consumption in the next defined time steps should only use the historical electricity consumption data and the date-time as predicting features. Despite the availability of other datasets containing other metrics such as consumers' household temperature and humidity, we stick only to date and time as extracted features for prediction. The reason behind it is that the prediction is meant to be performed by the electricity provider who is not supposed to have access to such private data of his clients. The predictions should solely be based on the electricity consumption data uploaded to the blockchain by the respective clients. During each new trading round, the GRU model adds the previously collected electricity consumption data to its training set. The model is evaluated by its accuracy. As a consequence, the more rounds go by, the more data available to train the model and thus the more precise the prediction should be. To assess that, we defined the trading round to be 100 days, i.e., we would be using 2400 rows to forecast the next 2400 rows. Then the number of rows $R(n)$ that is used to predict the upcoming 2400 rows in the n-th round is defined according to the expression in Equation (1):

$$R(n) = 2400 \cdot n, n > 0 \tag{1}$$

The performance of the model is assessed in terms of the Mean Average Percentage Error ($MAPE$) given in Equation (2), and the results are depicted in Figure 4.

$$MAPE = \frac{100}{n} \sum_{i=1}^{n} \left| \frac{(Y_i - P_i)}{Y_i} \right| \tag{2}$$

where Y_i is the i-th real electricity consumption reading and P_i is the i-th reading of the predicted electricity consumption, in a test set of size n. Figure 5 also provides the real energy consumption with respect to predicted energy consumption for the 50th trading round As shown in Figure 4, there are two jumps in the performance of the model. The first one occurs after nine trading rounds, which is equivalent to 9000 days and 90,000 observations, and the $MAPE$ drops from 3% to 2.42%. Then, after the 16th round, the $MAPE$ drops from 2% to less than 1%. Past the 20th round, there is no more steady decrease in the $MAPE$, however, it fluctuates approximately in the range [0.39%, 0.61%]. The advantage of this kind of prediction is that it uses on-chain data to perform the prediction. Since each node has a copy of the ledger containing the dataset that is used to train the model, this ensures the transparency of the prediction that can be verified by any node in the network.

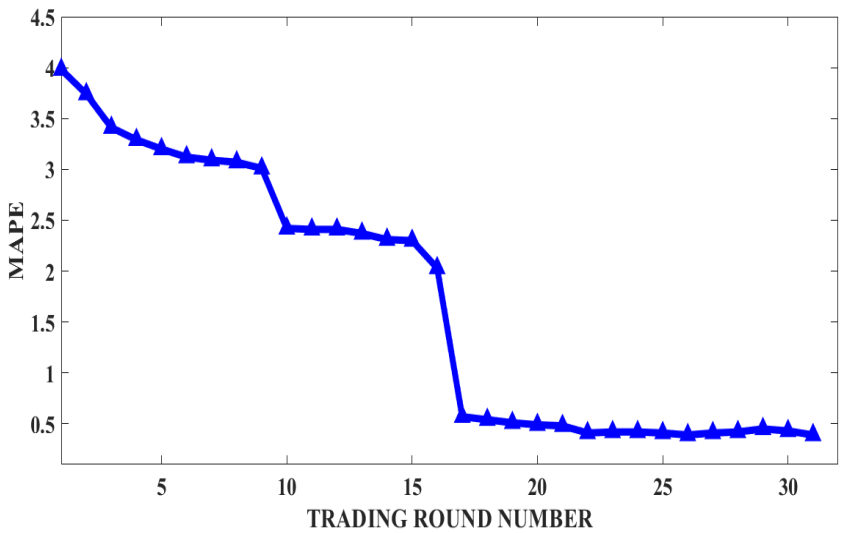

Figure 4. GRU prediction model performance evolution as the trading rounds go by.

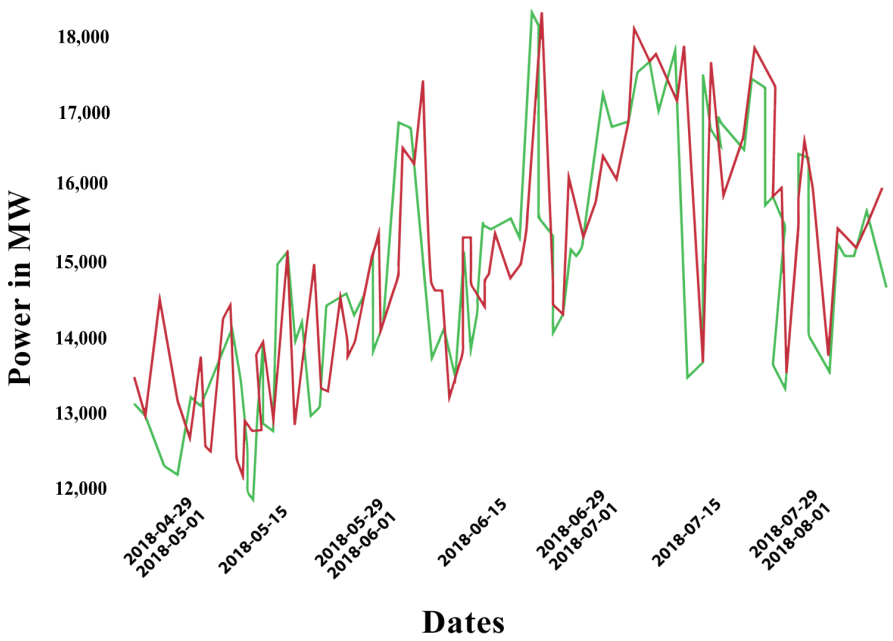

Figure 5. Real energy consumption versus predicted energy consumption for the 50th trading round.

4.1.2. Definition of ToU Ranges Using On-Chain Clustering

What is proposed, is that the energy ToU ranges and tariffs are defined ahead of each trading round; the tariffs are then defined accordingly in the smart contract arbitrating the interactions between the provider and the set of consumer clients. The novelty is that the clustering is implemented and performed on a smart contract. Interestingly, smart contracts are seen not to be suitable for implementing heavy computational algorithms due to code execution cost, which is abstracted in terms of gas. However, implementing a clustering algorithm to define ToU ranges on a smart contract is interesting in the sense that this would permit further decentralization, where ToU ranges are defined by a smart independent entity, ensuring their integrity and transparency. For this purpose, the K-mean clustering algorithm was implemented with K = 2 for two clusters, which are the electricity demand peak and off-peak, on a Solidity smart contract. It was compiled and tested on a Remix IDE platform to define ToU ranges on both predicted and real electricity consumption, for each round. The smart contract execution cost was also assessed and benchmarked to the execution cost of other decentralized applications' smart contracts presented in other published works. The aim is to assess if performing K-mean clustering on a smart contract is realistic in terms of gas consumption. The K-mean algorithm flowchart is illustrated in Figure 6.

So, the clustering is performed on a set of points within coordinates (x,y), where x designates the time of the day, ranging from 00:00 h to 23:00 h and y is the mean average of the electricity consumed on a particular hour of the day, during 100 days. For each hour of the day, the corresponding y value is computed from both predicted and real consumption data for each trading round according to the pseudocode in Algorithm 1.

Once the arrays X and Y are both defined, the clustering is performed according to pseudocodes in Algorithms 2 and 3.

Algorithm 1 Hourly mean average electricity consumption for one trading round

```
Sum ← 0
size ← 0
for i ← 0 to 24 do
    for j ← i to 100 do
        Sum ← dataset[j]
        j ← j + 24
        size ← size + 1
    end for
    y[i] ← sum/size
end for
```

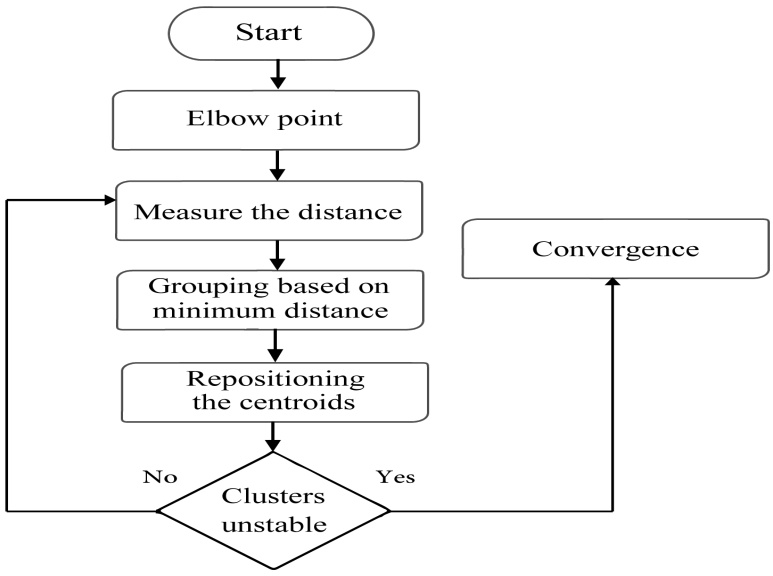

Figure 6. The K-mean algorithm flowchart.

The functions in Algorithms 2 and 3 are internal and can only be called in the main function, which is made public. In Algorithm 2, peaks and off-peak clusters are set for each hour of the day, according to the distance of the points from the centroids, passed as parameters. The function of Algorithm 3 calculates the new centroid for the next iteration.

Algorithm 2 SetCluster(centroid C1,centroid C2)

```
for i ← 0 to size(X) do
    d1 ← √(X[i] − C1.x^2 + Y[i] − C1.y^2)
    d2 ← √(X[i] − C2.x^2 + Y[i] − C2.y^2)
    if d1 ≤ d2 then
        clusters.push(1)
    else
        clusters.push(2)
    end if
end for
return clusters
```

To compute the clusters, X and Y arrays are defined in the smart contract as well as the two initial centroids. The initial off-peak cluster centroid is set to be the point of the set with the lowest consumption, whereas the initial peak cluster centroid is set to be the point in the dataset with the highest electricity consumption. The main function defined in Algorithm 4 invokes the functions of Algorithms 2 and 3 through a loop until convergence

according to the flowchart in Figure 6. A decentralized P2P purchase and rental application based on blockchain has been presented in [65]. This has served to make a comparison with the implemented K-mean clustering smart contract, both deployment and invoking gas consumption.

Algorithm 3 SetNewCentroids

```
sumX1 ← 0
sumX2 ← 0
sumY1 ← 0
sumY2 ← 0
TempArray ← 0
Cluster1Size ← 0
Cluster2Size ← 0
Centroid NewC2
for i ← 0 to 24 do
    if clusters[i] = 1 then
        sumX1 ←X[i]
        sumY1 ←Y[i]
        Cluster1Size ← Cluster1Size + 1
    else if clusters[i] = 1 then
        sumX2 ←X[i]
        sumY2 ←Y[i]
        Cluster1Size ← Cluster1Size + 1
    end if
end for
NewC1 ← { sumX1 / Cluster1Size , sumY1 / Cluster1Size }
NewC2 ← { sumX2 / Cluster2Size , sumY2 / Cluster2Size }
tempArray.push(NewC1)
tempArray.push(NewC2)
return TempArray
```

Algorithm 4 Main

```
TempArray1 ← SetCluster(InitC1, InitC2)
TempArray2 ← []
NewC1 ← GetNewCentroids()[0]
NewC1 ← GetNewCentroids()[1]
Exit ← False
while !Exit do
    TempArray2 ← SetCluster(InitC1, InitC2)
    if TempArray2! = TempArray1 then
        Exit ← true
    else
        TempArray1 ← TempArray2
        NewC1 ← GetNewCentroids()[0]
        NewC2 ← GetNewCentroids()[1]
        TempArray2 ← SetCluster(NewC1, NewC2)
    end if
end while
```

As can be observed from Table 1, there is a slightly higher value in deploying K-mean smart contract, and a much bigger gas consumption in calling K-mean Smart Contract main function compared to calling purchase function [65]. Calling K-mean Smart Contract is 160 times costlier. However, K-mean clustering is only called once in each trading round, which makes its gas cost reasonable and its implementation feasible and practical, due to its low invoking frequency.

Table 1. Gas consumption.

	Deploy	Call Contract Function
K-mean smart contract	801,911	1,334,160
Smart contract in [65]	760,550	82,721

4.2. Trading Platform Interacting Entities

The Provider: which represents the set of prosumers cooperating in generating energy that is injected into the power grid owned by the DSO, and transferred to the requesting consumers, on demand.

The Consumer: being the entity that purchases energy by subscribing to the provider through a smart contract for a certain duration of time called a trading round, prepays for his consumption and is supplied until exhaustion of the prepaid amount, or billed according to his energy consumption in case of unsubscribing prematurely.

The DSO: also referred to as a utility owner company. The abstraction of the interacting entities is illustrated using the Entity Relationship Diagram ERD in Figure 7. However, these entities are not stored in a traditional database, but rather, they are represented by the state of the smart contract in which they are defined. The state of the smart contract is modified upon a successful call of its functions and is stored in the blockchain shared ledger. Each entity is assigned to its corresponding field through an appropriate mapping.

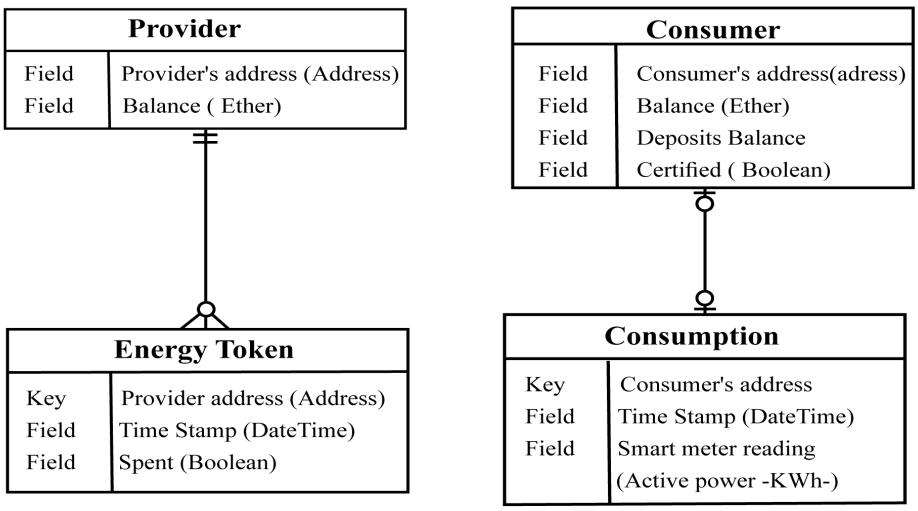

Figure 7. ERD diagram of the proposed platform.

Smart contract 1 (DSO): As its name indicates, DSO smart contract is deployed and owned by the DSO. It stores the list of certified consumer nodes as well as the energy balance of the providers. Only the DSO has the credentials to add, remove or update the certified nodes list. Providers' balance is updated by calling the appropriate smart contract function upon a new aggregation of a defined amount of energy or its consumption. The provider's energy balance is abstracted using a simple structure representing a defined amount of energy called an energy token. New instances of this data structure are mapped to prosumers accordingly upon the appropriate aggregation of energy. Consumed tokens are set to spent state. The global variables for the DSO smart contract are defined in Algorithm 5.

Algorithm 5 DSO Smart Contract: Global Variables

```
ProviderSmartContractAddresses : address[] ▷ /* Addresses of all providers' deployed smart contracts */
DsoAddress : address ▷ /*DSO account's address*/
Certified : Mapping (address: bool) ▷ /* Check if an account is certified or not. By default, accounts are not certified. Certifying an account is restricted to the DSO */
Structure
TimeStamp : uint256
Spent : Bool
EndStructure
ProviderEnergyTokens : Mapping (address: EnergyToken[]) ▷ Keep track of EnergyToken instances owned by each Provider account
```

The energy balance of the prosumer is defined by the number of unspent token instances and is publicly accessible by the function defined in Algorithm 6.

Algorithm 6 DSO Smart Contract: ProviderEnergyBalance

```
▷ /* This function returns the energy balance of a given address' account. */
Input: addr : address
Output: uint256
CountOfUnspentEnergyToken : uint256,
CountOfUnspentEnergyToken ←
for i ← 0 to ProviderEnergyTokens[addr].Length do
    if !ProviderEnergyTokens[addr][i].spent then
        CountOfUnspentEnergyToken ← CountOfUnspentEnergyToken + 1
    end if
end for
Return: CountOfUnspentEnergyToken
```

Updating the provider's balance per consumed or redeemed token is done exclusively through the provider's smart contract, by calling the DSO smart contract functions defined in Algorithms 7 and 8, respectively.

Algorithm 7 DSO Smart Contract: UpdateProviderBalance1

```
Comment/* This function is called to consume N tokens, it takes two inputs, an integer N and an address addr. It sets N unspent tokens from the tokens owned by addr to spent. It is a restricted function which can only be called by the providers' deployed smart contracts */
Input: addr : address, N : uint256
i ← 0
k ← 0
while i ≤ N do
    if !ProviderEnergyTokens[addr][k].spent then
        ProviderEnergyTokens[addr][k].spent ← true
        i ← i + 1
    end if
    k ← k + 1
end while
```

Energy tokens have been used to represent the providers' energy balance instead of a simple numerical abstraction, to keep track of the history of the providers' contributions to the utility grid. The relevance of keeping track of prosumers' energy contributions in a P2P energy trading platform can be found in [66].

Algorithm 8 DSO Smart Contract: UpdateProviderBalance2

```
▷ /* This function is called to redeem N tokens, it takes two inputs, an integer N
and an address addr. It sets N spent tokens from the tokens owned by addr to unspent.
It is a restricted function which can only be called by the providers' deployed smart
contracts */
Input: addr : address, N : uint256
i ← 0
k ← 0
while i ≤ N do
    if ProviderEnergyTokens[addr][k].spent then
        ProviderEnergyTokens[addr][k].spent ← false
        i ← i + 1
    end if
    k ← k + 1
end while
```

Smart contract 2 (Provider): The contract is deployed by the provider ahead of each trading round; it contains the ToU pricing for electricity consumption. Once deployed, consumers have to subscribe to the trading round during a defined period of time, by depositing money corresponding to the amount of electricity they wish to consume. The reason a deadline for subscription is defined, is that receiving a new subscription at any time can prolong the trading round indefinitely, which is not following the proposed scheme since electricity pricing is set according to prediction based on data collected during trading rounds. Each time a new subscription occurs, the smart contract checks the subscribing consumer account by calling Smart Contract 1; in case this is later certified and the provider still has enough energy balance to supply the new coming demand, the provider balance is updated accordingly, otherwise the subscription is denied. The smart contract stops accepting upcoming subscription requests if the subscription-defined time has elapsed, or if the provider has exhausted all his energy balance. The trading round then starts; it ends when all subscribing consumer deposit balance in the smart contract is zero. This time-dependent smart contract state is illustrated in Figure 8. Consumers' electricity consumption data are uploaded to the smart contract. It is used to charge consumers as well as being added to the energy consumption dataset serving to train the prediction model as discussed in Section 4.1 above.

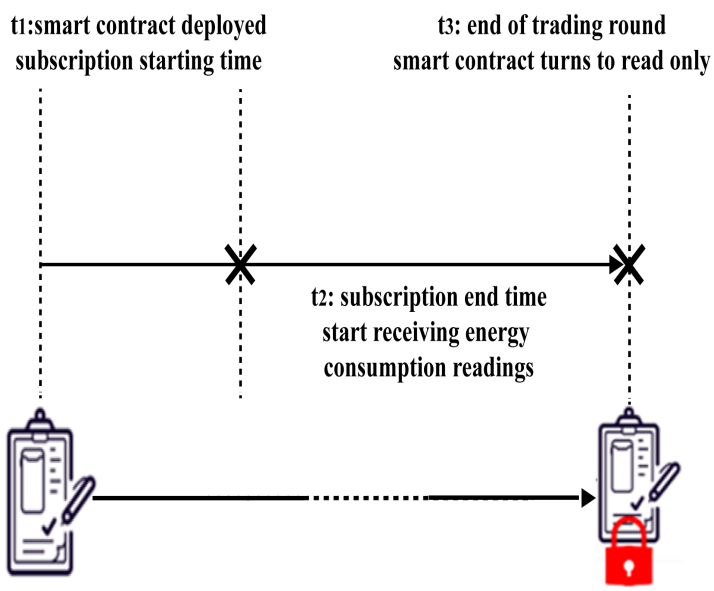

Figure 8. Time dependency of the Provider smart contract state.

To test the deployed smart contract, one Ether is set to correspond to 100 kWh. Although this is not realistic, it is simpler for the sake of testing. However, other security tokens, and even stable coins linked to the country's actual currency asset, can serve as a payment method rather than Ether cryptocurrency, which has a very fluctuating value. The global smart contract variables and events are presented below in Algorithms 9 and 10, respectively.

Algorithm 9 Provider Smart Contract: Global Variables

1: $Deposits$: $Mapping$ (address:uint256) ▷ Keep track of each consumer deposit balance
2: $Consumption$: $Mapping$ (address:uint256) ▷ Keep track of each consumer total consumption
3: $EnergyRequested$: $Mapping$ (address:uint256) ▷ Access each consumer requested energy
4: $SubscriptionTime$: uint256 ▷ Defined time span to accept new subscriptions after contract deployment
5: $DsoSmartContract$: address ▷ DSO smart contract address
6: $Subscribers$: address[] ▷ Keep track of subscribers' accounts
7: $DeployementTime$: uint256 ▷ This variable is set to contract deployment time, it is set inside the smart contract constructor called when contract is deployed
8: $EnergyTarif$: uint256

Algorithm 10 Provider Smart Contract Events

1: $event : SendEnergy(addressindexed)$ ▷ Notify the DSO to supply energy to the address passed through the event
2: $event : eventSubscriptionTimeStart()$ ▷ Notify consumers that subscription is now open after the smart contracts have been deployed
3: $event : StartTradingRound()$ ▷ Notify all parties that the trading round has started
4: $event : eventBalanceExhausted(address)$ ▷ Notify the DSO to stop supplying energy to the address passed through the event

Algorithm 11 defines the function consumers call to upload their energy consumption data through their certified nodes.

Algorithm 11 send_consumption_data

1: **Input:** $c : uint256$
2: $consumptionBalance : uint256$
3: $consumption[msg.sender] \leftarrow consumption[msg.sender] + c$
4: $consumptionBalance \leftarrow consumption[msg.sender] \times energy\_tarif$
5: $deposits[msg.sender] \leftarrow deposits[msg.sender] - consumptionbalance$
6: **if** $deposits[msg.sender] \leq 0$ **then**
7: $\quad emitBalanceExhausted(msg.sender)$
8: **end if**

Algorithm 12 defines the function called by consumers to subscribe to the trading round.

Algorithm 13 defines the function called by consumers to unsubscribe prematurely and exit the trading cycle.

Algorithm 12 Provider's Smart Contract: Subscribe

```
Time : uint256
TotalEnergyRequested : uint256
ProviderEnergyBlance : uint256
time ← block.timestamp
TotalEnergyRequested ← TotalEnergyRequested + msg.value / 10^18
EnergyRequested[msg.sender] ← msg.value
ProviderEnergyBlance ← CallDSOContarct1(addressDsoContract, addressProvider)
        ▷ /* Call DSO smart contract ProviderEnergyBalance function, to get Provider's energy balance */
deposits[msg.sender] ← deposits[msg.sender] − consumptionbalance
if TotalEnergyRequested > ProviderEnergyBlance ∨ TimeSubscriptionTime ≤ Time − Deployement∨!CallDsoContract2(addressDsoContract, msg.sender) then
    Revert()
    for i ← 0 to Subscribers.Length do
        emit SendEnergy(Subcribers[i])
    end for
    emit StartTradingRound()
else
    Deposits[msg.sender] ← Deposits[msg.sender] + msg.value
    subsribers.push(msg.sender)
    CallDsoContract3(addressDsoContract, addressProvider, msg.value / (10^18))   ▷ /* Call DSO smart contract UpdateProviderBalance1 function to update the provider's energy balance according to the subscribers' energy demand that needs to be supplied to them */
end if
```

Algorithm 13 Provider Smart Contract: Unsubscribe

```
BillAmount : uint256
TotalEnergyRequested : uint256
BillAmount ← consumption[msg.sender] ∗ EnergyTarif
Provider.Transfer(BillAmount)
msg.sender.Transfer(deposits[msg.sender] − billamount)
Deposits[msg.sender] ← 0
EnergyRequested[msg.sender] ← msg.value
        ▷ // Call DSO smart contract UpdateProviderBalance2 function to re-update the provider's energy balance accordingly after consumer unsubscribing:
CallDSOContarct1(addressDsoContract, addressProvider, deposits[msg.sender] − billamount/10^18)
emit BalanceExhausted(msg.sender)
```

4.3. Agents' Interactions and Workflow in the Proposed Platform

Different agents interact with the pair of smart contracts, according to the defined credentials granted. On the one hand, the provider, consumer, and DSO interact with the smart contract by calling authorized functions. On the other hand, smart contracts also interact with the rest of the platform agents through events, as shown in Table 2. The event trigger is defined within the smart contract. Agents subscribe to a particular event through their back-end application and get systematically notified whenever the event is fired during the smart contract execution, with parameters being passed by the smart contract to the subscribing application. Moreover, an event handler is set within the agents' applications to respond accordingly to the particular event and process the parameters passed along. The pseudocode of the functions in the provider's smart contract is given below. Along with these functions, calls to functions of the DSO smart contract are also defined within the provider's smart contract as procedures; they take as parameters

the DSO smart contract address beside the parameter to pass to the DSO smart contract function to call.

Moreover, the interaction and workflow between the platform agents are highlighted in Figures 9 and 10. The respective responses are given in chronological order in Figure 9. When the trading round starts, all subscribers' energy demand is supplied until prepaid amount exhaustion. In the case of a client's premature un-subscription, he is charged according to his actual consumption and the change from his prepaid deposit is returned to him. The provider's balance is updated as well, by setting back n spent tokens to unspent, where n represents the number of tokens corresponding to the change amount returned to the consumer.

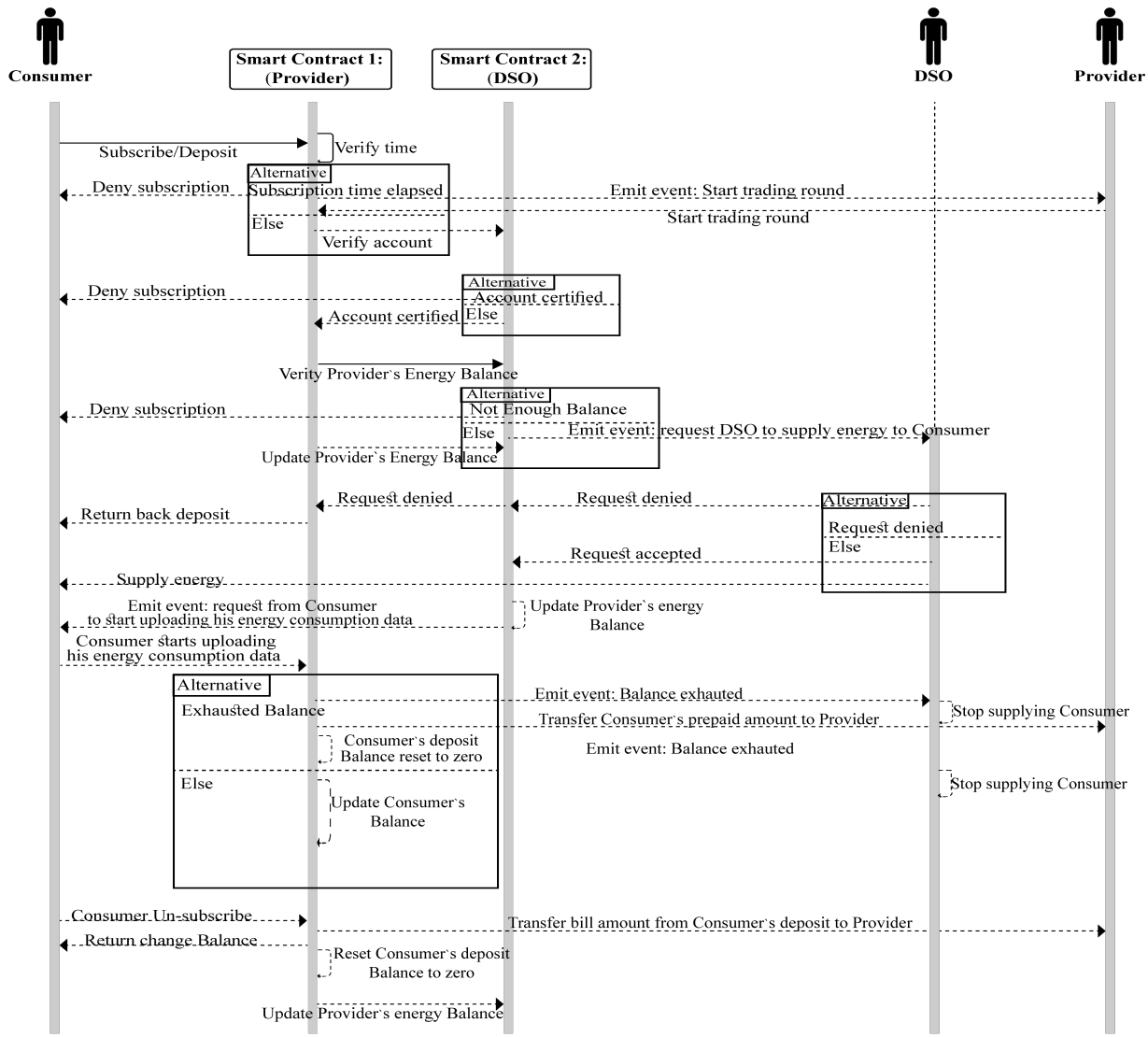

Figure 9. The proposed P2P Energy trading sequence diagram.

Table 2. Smart contracts' events in the proposed platform.

Event	Source of Emission	Event Trigger	Event Subscribers	Event Handler
Event 1:Subscription-TimeStarted ()	Smart Contract 1	Smart Contract 1 is deployed	Consumer	Get notified by subscription starting time
Event 2:Subscription-TimeEnded ()	Smart Contract 1	Subscription time elapses	Consumer	Get notified that subscription time has ended.
Event 3:TransferEnergy (Address add);	Smart Contract 2	Provider energy balance is verified to be enough	DSO	Supply energy to the consumer.
Event 4:StartTradingRound ();	Smart Contract 1	Smart Contract 2 confirms that the provider has energy balance	Consumer	The Consumer starts uploading his energy consumption readings to Smart Contract 1.
Event 5:BalanceExhausted	Smart Contract 2	Consumer unsubscribes, or balanceExhausted	DSO	Stop supplying the Consumer.

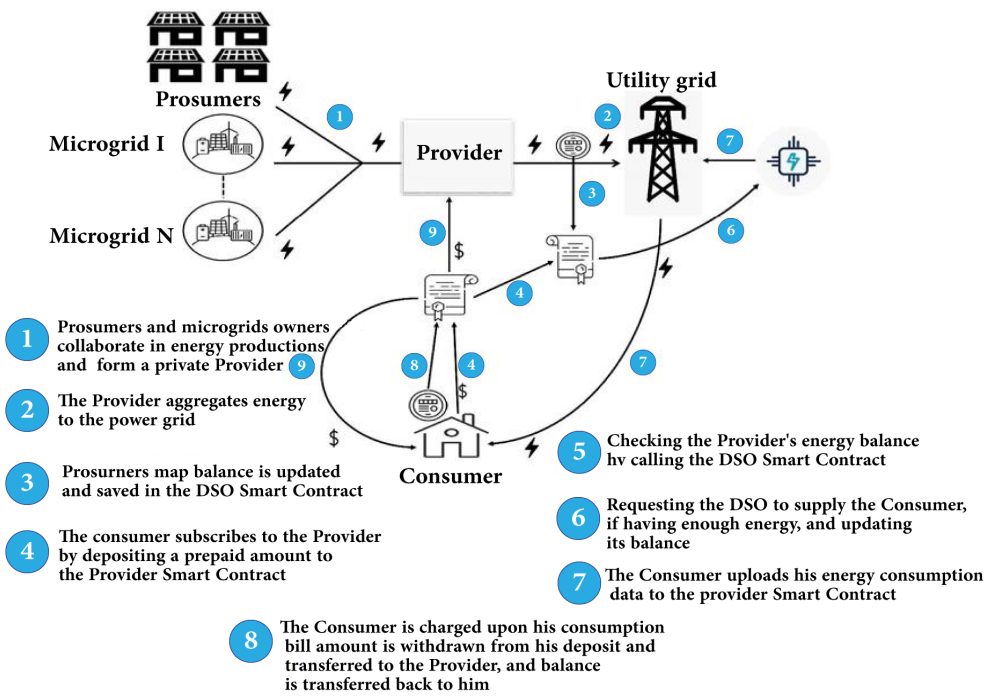

Figure 10. Interactions between the entities of the proposed P2P trading Platform.

4.4. The State Diagram Representation

Each time a smart contract is called, it is executed on the Ethereum virtual machine and its new state is stored on the blockchain. Thus, is suitable to represent the smart contract as a finite state machine as depicted in the diagram of Figure 11. Inputs represent the different function calls that modify the state of the smart contract.

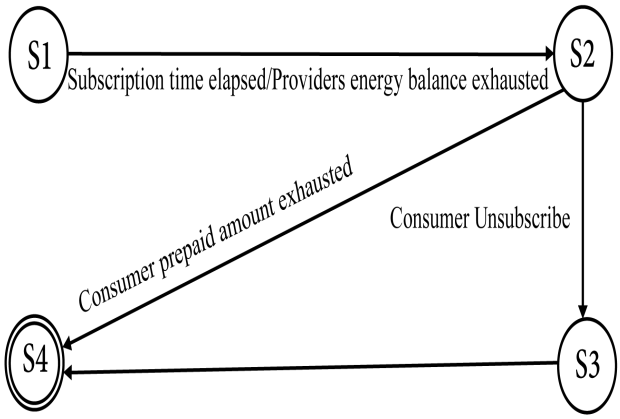

Figure 11. State diagram.

4.5. Consensus Protocol

Blockchain is principally a distributed architecture with no central authority, the validity of the shared ledger is agreed upon consensus held between nodes in the network. Thus, consensus protocol stands as a key feature in such an architecture-based application. It ensures the integrity of the data, which is uploaded to the blockchain without a central trusted authority. Ethereum private blockchain proposes two options for consensus protocols. Ethash, which is a Proof of Work-based protocol and Clique Proof-of-Authority consensus protocol. Moreover, in this work, we preferred to use Ethash to keep the platform decentralized, since PoA tends toward centralization by having a set of predefined validators. What is suggested is to use Ethash with a fixed difficulty and no mining rewards. All consumer nodes in the network are potential miners. All miner nodes would be having the same hashing power, thus, if a node attempts to tamper with the ledger, it would have to find new valid hashes for all the blocks preceding the blocks tampered with, and will be certainly leftover in the mining race. The only scenario where a node can successfully be fast enough to tamper with the already committed ledger is to use hardware having higher hashing power, which is supposedly not allowed, given that the nodes' hardware need to be certified. For that purpose, the proposed consensus algorithm for this architecture is that blocks are not accepted if the time difference between the last block mining time and the time of proposing a new valid block is less than a defined threshold. To define the threshold time, a PoW algorithm is developed in Java and run on a four (4) cores CPU: Intel(R) Core(TM) i5-9300H CPU @ 2.40 GHz and a GPU: NVIDIA GeForce GTX 1650 with Max-Q Design. The mining function is called in a loop for several iterations. In each iteration, the mining time is pushed in an array then the median of the set is deduced. $M = M_0, \ldots, M_k$ is the set of mining times where the mining function has been called for k times with a fixed difficulty and a random string of fixed size to be hashed. Tthreshold represents the minimum threshold time for accepting a new block. It is defined to be the median of the set M. Tthreshold is computed for several sets M of different sizes. Tthreshold is picked when convergence is met, as illustrated in Table 3.

Table 3. Tthreshold for different sizes of mining sets.

Set Size	Median of the Set [ms]
50	69
100	49
150	39
200	39

According to Table 3, the deduced Tthreshold is 39 ms. To test the proposed mining protocol and assess its performance, it is executed on 10 different threads running

concurrently on the same machine, as depicted in Figure 12. Each thread represents a miner. Fraudulent miners' nodes are represented by threads running mining protocol with less difficulty, to act as nodes having more hashing power. In the testing experiment, the certified miners are represented by threads mining with difficulty 5, whereas fraudulent miners are represented by threads mining with difficulty 4. The experiment consists of running the threads for 100 rounds corresponding to 100 blocks. The number of fraudulent miner nodes goes from a single node to 6 nodes representing more than 50% of the network. Threads with the shortest execution time in a round represent nodes successfully mining the new block for that particular round. If thread execution time is less than the accepted minimum threshold, its execution time is set to be Tthreshold.

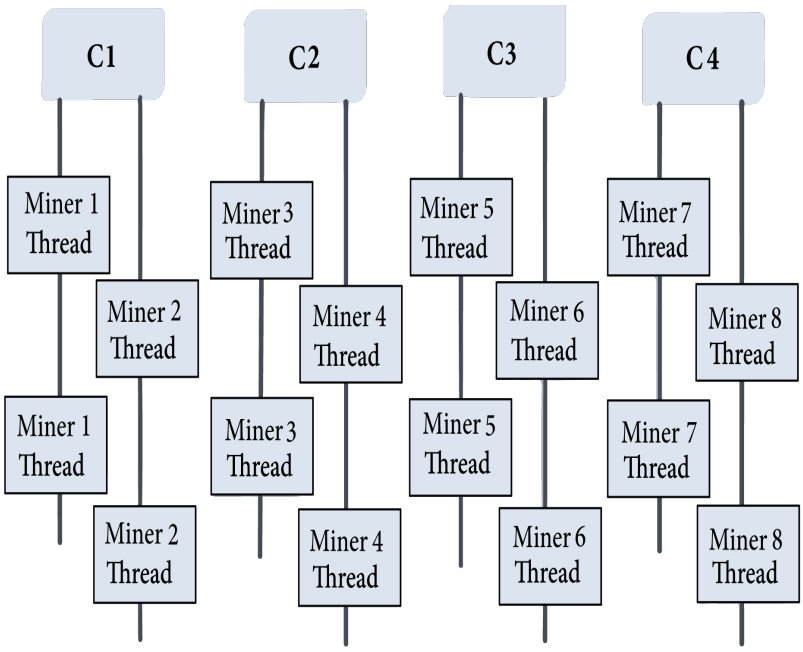

Figure 12. Threads running concurrently on CPU cores, simulating miners competing to add a new bloc.

Miners for each round are defined according to the pseudocode of Algorithm 14. The obtained results showed that in each mining round there are always at least two successful miners, which corresponds to a fork situation. This is expected, since all miners are supposed to have the same hashing power. Results in Figure 13 describe miner involvement in forks given a different number of fraudulent nodes.

Algorithm 14 Miner threads.

```
for i ← 0 to NumberOf(threads) do
    if thread.ExecutionTime ≤ Threshold then
        t ← Ttreshold
    else
        t ← thread.ExecutionTime
    end if
end for
Min ← Math.min(array)
for i ← 0 to Size(array) do
    if array[i] ≤ Min then
        MinerIndexes.push(i)
    end if
end for
return MinerIndexes
```

Fraudulent nodes are threads given less difficulty than the rest, to emulate nodes that have higher hashing power than allowed. It can be seen that fraudulent nodes are always ahead of others in the number of blocks mined successfully, even when the number of fraudulent nodes exceeds half of the total network. Hence, a fraudulent node can be easily spotted. Since the nodes in the proposed architecture are not anonymous, they can be held accountable for any proven fraudulent behavior.

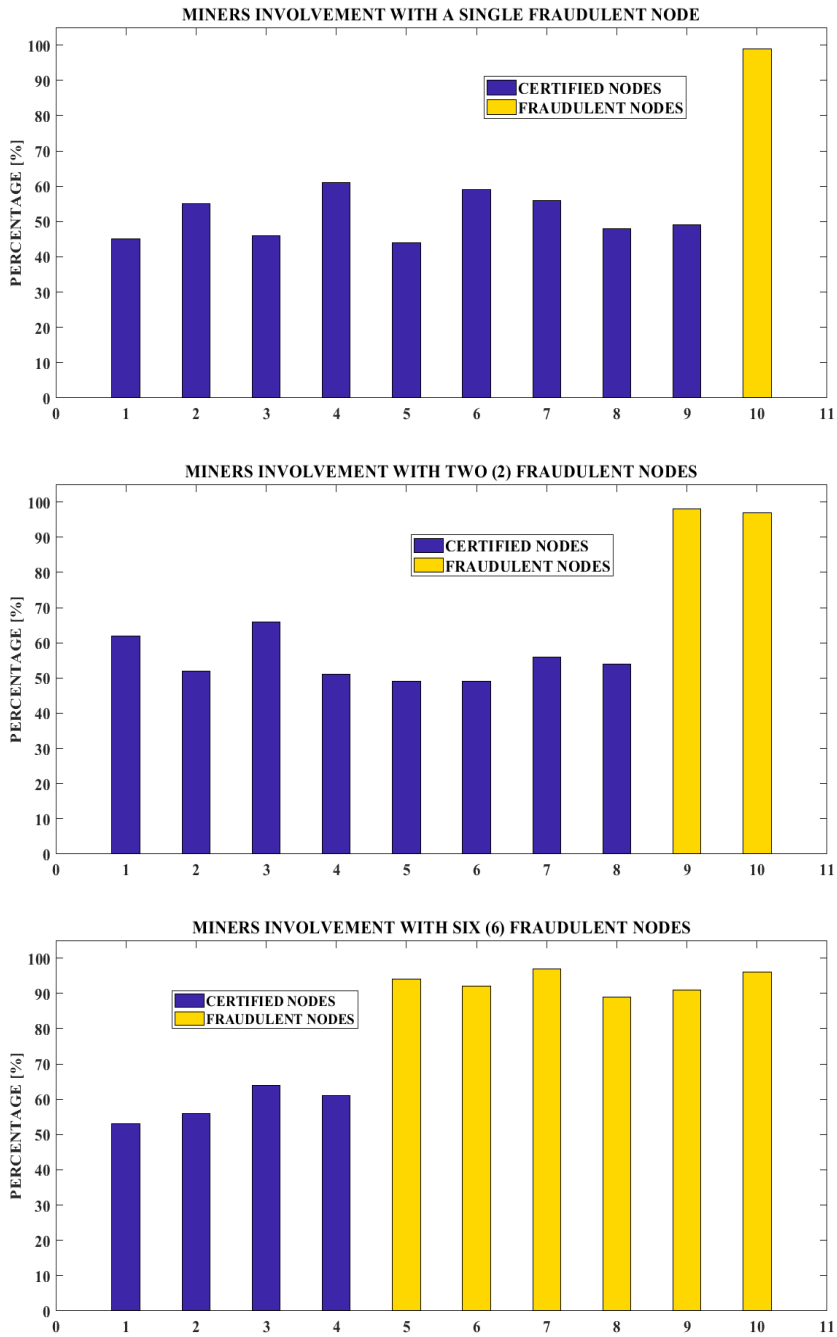

Figure 13. Miners involvement in forks using the proposed consensus protocol.

To implement the proposed consensus protocol on an Ethereum private blockchain, the go-Ethereum source code needs to be slightly overwritten. This can be done by performing minor additions to the source-code in the file: "https://github.com/ethereum/go-ethereum/blob/master/consensus/ethash/consensus.go" accessed on 23 March 2022

The following Algorithm 15 is a code snippet with the original line members from go-ethereum source code on Github; the applied additions are highlighted.

Algorithm 15 Modifications performed on PoW consensus in Go-Ethereum

```
// Various error messages to mark blocks invalid. These should be private to prevent
engine specific errors from being referenced in the remainder of the codebase, inherently
breaking if the engine is swapped out. Please put common error types into the consensus
package.
var (
        errOlderBlockTime = errors.New("timestamp older than parent")
        errTooManyUncles = errors.New("too many uncles")
        errDuplicateUncle = errors.New("duplicate uncle")
        errUncleIsAncestor = errors.New("uncle is ancestor")
        errDanglingUncle = errors.New("uncle's parent is not ancestor")
        errInvalidDifficulty = errors.New("non-positive difficulty")
        errInvalidMixDigest = errors.New("invalid mix digest")
        errInvalidPoW = errors.New("invalid proof-of-work")
        errInvalidMininegTime = errors.New("invalid mining time")

func (ethash *Ethash) verifyHeader(chain consensus.ChainHeaderReader, header, parent
*types.Header, uncle bool, seal bool, unixNow int64) error {
// Ensure that the header's extra-data section is of a reasonable size
if uint64(len(header.Extra)) > params.MaximumExtraDataSize {
    return fmt.Errorf("extra-data too long: %d > %d", len(header.Extra),
params.MaximumExtraDataSize
// Verify the header's timestamp
        if header.Time > uint64(unixNow+allowedFutureBlockTimeSeconds) {
          return consensus.ErrFutureBlock
        }
}
if header.Time <= parent.Time {
    return errOlderBlockTime
}
if header.Time - parent.Time < Tthreshold {
      return errInvalidMininegTime
}
```

4.6. Proof of Location (PoL)

Consumer node deployment and setting need to be supervised and certified by the DSO so that the nodes are allowed to join the trading network. Consumer node transactions are identified by their digital signature. However, since consumers are billed according to the consumption data uploaded by their respective assigned nodes, which are set to upload the consumer's consumption reading from the respective smart meter to the smart contract, it is crucial to ensure that the data uploaded is indeed coming from the certified nodes and not from any malicious clone or identity thief who could gain access to the consumer node private key. What is proposed is to add a proof of location (PoL) that must be included in every transaction issued by the consumer node to prove the transaction is coming from the known location of the certified node. Outdoor localization mainly uses Global Positioning System (GPS) as the main localization technique. However, GPS is not suitable for trustless decentralized scenarios. What is suggested is to use trilateration to localize the consumer nodes. Trilateration is a technique for node localization that determines object position, knowing its distance from at least 3 reference points.

Considering 2-dimensional trilateration, a 2-dimensional frame needs to be set with an origin O (0,0) and three reference points, R_{p1}, R_{p2} and R_{p3} with known coordinates (x_1,y_1), (x_2,y_2) and (x_3, y_3), knowing the distances d_1, d_2 and d_3 between a target point $T_p(x_T, y_T)$

and R_{p1}, R_{p2} and R_{p3}, respectively, the couple (x_T, y_T) can be deduced by solving the set of three expressions of Equation (3), as illustrated in Figure 14.

$$\begin{aligned} d_1^2 &= (x_t - x_1)^2 + (y_t - y_1)^2 \\ d_2^2 &= (x_t - x_2)^2 + (y_t - y_2)^2 \\ d_3^2 &= (x_t - x_3)^2 + (y_t - y_3)^2 \end{aligned} \tag{3}$$

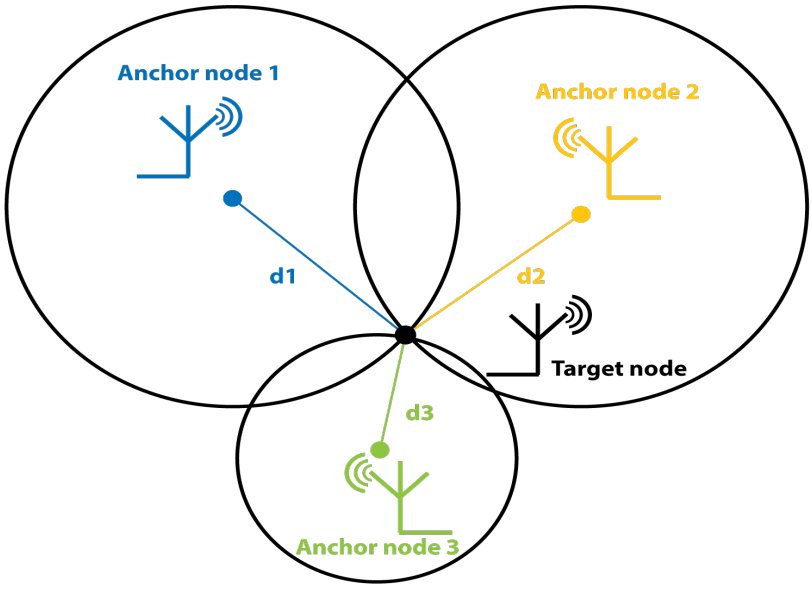

Figure 14. Trilateration technique.

In the proposed PoL model, three anchor nodes are deployed to cover a defined area with a defined number of consumer nodes. Consumer nodes send a PoL request to the 3 anchor nodes; each anchor node computes the distance separating him from the consumer node, then sends back the measured distance signed using his private key K_j^{Pr}. Consumer nodes need to include the three signed locations they received from the 3 anchors as a PoL. Given a defined geographical location with a set of n Consumer nodes $CN_i = CN_0, CN_1, \ldots, CN_n$ and a set of 3 anchor nodes $An_j = An_0, An_1, An_3$ each consumer node i should send a PoL request $Pol\_Req_{i \to j}$ to the anchor j and receive Pol response $Pol\_Res_{j \to i}$, accordingly, as in Equation (4).

$$Pol_{Res_{j \to i}} : \left\{ \begin{matrix} Distance\ between\ An_j\ and\ CN_i \\ Timestamp \end{matrix} \right\} K_j^{Pr} \tag{4}$$

There are two main techniques to measure the distance between a reference anchor and a target node, which use either the Time of Arrival (ToA) or the Received Signal Strength Indicator (RSSI). Although assessing the most suitable distance measurement technique is not within the scope of our paper, we found some reference papers proposing RSSI techniques for outdoor localization as in [67–70] and we aimed to present a blueprint for Wi-Fi RSSI-based node localization, that potentially can be used for PoL in blockchain-based decentralized P2P energy trading. Given the RSSI of a received signal, the distance between the transmitter and the receiver nodes can be deduced according to the expression in Equation (5).

$$d = 10^{-(RSS - P_t + Pl_0 - \delta)/10\gamma} \tag{5}$$

where Pl_0 is the path loss at a reference distance of (1 m) measured using the expression in (6), λ is the wave length of the emitted signal and γ is the path loss exponent and depends on the environment.

$$Pl_0 = 20log10\left(\frac{4\pi}{\lambda}\right) \tag{6}$$

δ is the standard deviation and depends on the hardware and antenna used as well as on the noise. We tried to define the location of 2 smartphones through Wi-Fi RSSI. RSSI values of a hotel Wi-Fi access point were measured in an urban area of Kuala Lumpur using two different smartphones, a SAMSUNG GALAXYAS30 and an OPPO F11. RSSI readings were recorded each 2-meter interval walking from the hotel main entrance following the itinerary illustrated in Figure 15 for 50 times, to deduce the standard deviation model:

Figure 15. The itinerary followed.

Since there is no standard industrial norm for computing RSSI, its measurement for the same distance is different in the two phone devices. In such an architecture with no unified protocol in RSSI measurement, the anchor node needs to have a reference database with a d(RSSI) model for each device. We kept the same distance function as in Equation (1) with personalized λ and δ and for each device as shown in Figure 16.

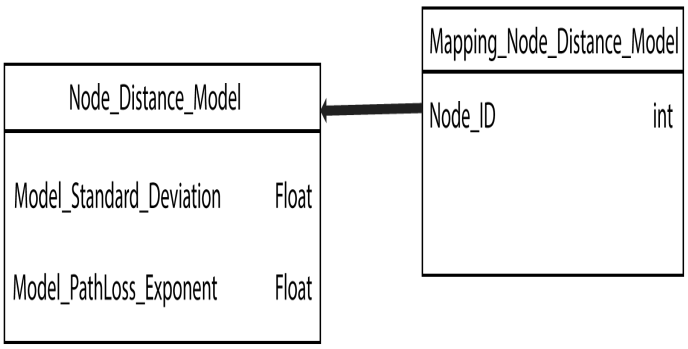

Figure 16. Mapping diagram illustrating how each consumer node is mapped to its own distance computation model in the Anchor server node.

The Wi-Fi access point is of 5 GHz with a wavelength of 6 cm, the PoL is deduced according to Equation (3), the access point Wi-Fi signal power transition is $P_T = 25$ dBm. The measured RSSI corresponding to each distance in the entire 50 testing sets was used to compute the distance standard deviation for each device. In our case, the RSSI in a given position varies randomly over time from one test set to another, the values that are measured

for each distance in each training set are provided in Equation (7). The corresponding results are reported in Figures 17 and 18.

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N=50} RSSI_i - u} \tag{7}$$

where, $u = \frac{1}{N}\sum_{i=1}^{N=50} RSSI_i$

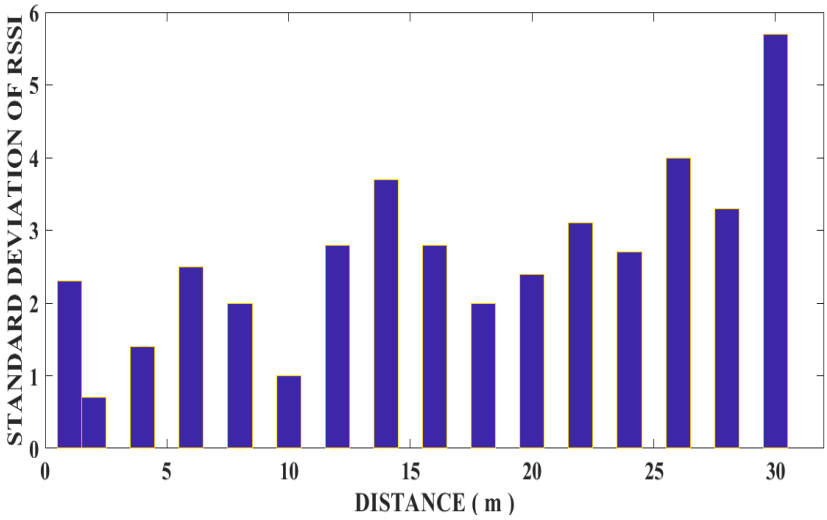

Figure 17. Standard deviation for device 1.

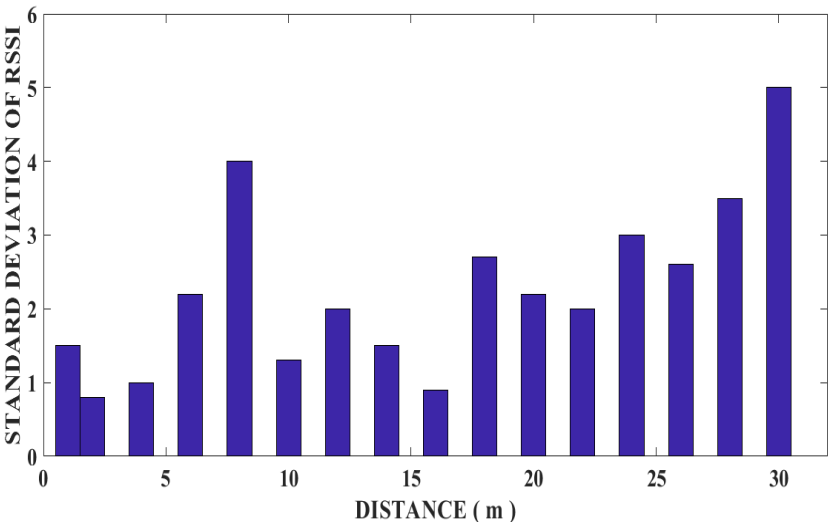

Figure 18. Standard deviation for device 2.

The mean standard deviation for device 1 is 2.2625, whereas it is 2.6500 for device 2. By fixing δ for each device, the next step is to deduce γ for each one of them using best fitting curve method between the set of distances $d_1, d_2, \ldots, d_{15}$ and the set of corresponding RSSI mean averages $RSSI_{av0}, RSSI_{av1}, \ldots, RSSI_{av15}$, using Equation (5). This is depicted in Figures 19 and 20.

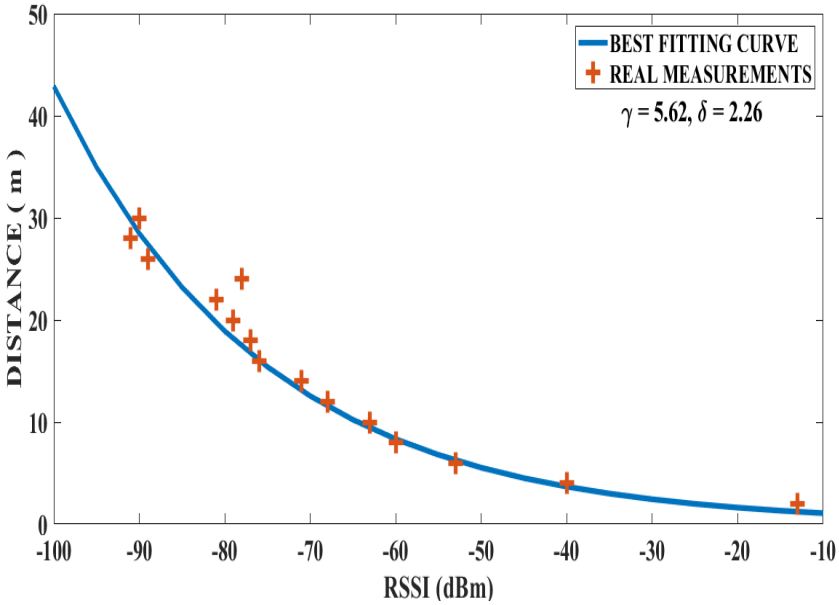

Figure 19. Pathloss exponent for device 1 deduced using best fitting curve.

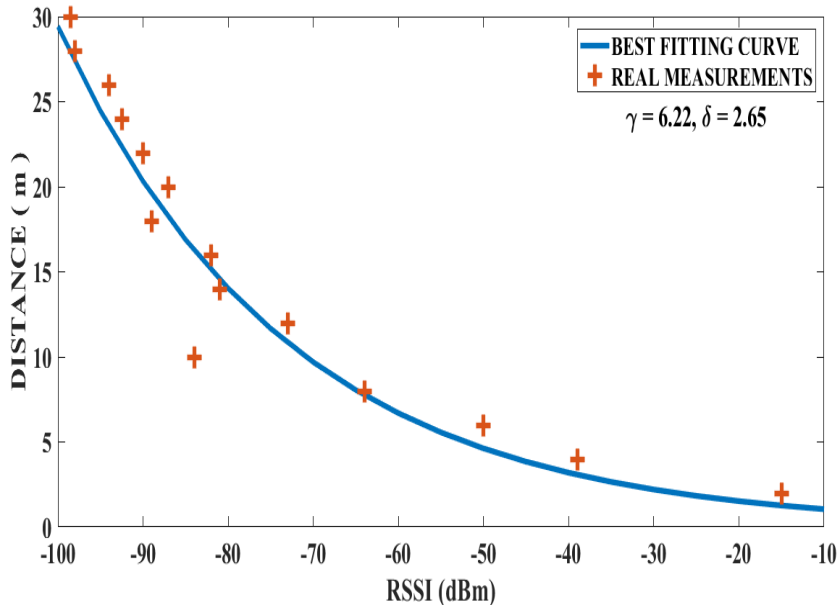

Figure 20. Pathloss exponent for device 2 deduced using best fitting curve.

5. Experiment Results and Analysis

To assess the performance of the Proposed platform, an Ethereum private blockchain has been set on a machine having 4 cores CPU: Intel(R) Core(TM) i5-9300H CPU @ 2.40 GHz and a GPU: NVIDIA GeForce GTX 1650 with Max-Q Design. It is worthy to mention that the purpose of the experiment is not to validate the functioning of the platform, since it has been already accomplished during the development stage on Remix IDE. We decided to put the focus on evaluating its performance when increasing subscribing consumers as well as their consumption data sending rate. Accordingly, blockchain nodes are created running on different ports and are connected to each other as peers. The created nodes represent, as a whole, subscribing consumers uploading their electricity consumption data on the provider's smart contract. The evaluated metric is transactions latency, which consists of time in seconds between the transaction being issued by the node calling the smart contract and when being committed to the ledger. As this is illustrated in Figure 21, transaction latency increases with the increase of the sending rate. This is due to the block mining

time, as well as to the fact that block size in Ethereum blockchain is limited to a maximum of 20 million gas and thus to a limited number of transactions in a single block. The less gas a transaction consumes, the more transaction can be contained in a single block and consequently, the less transaction latency. The slope of latency increase varies according to the block mining time. Figures 22 and 23 show gas consumption for each function in both smart contracts.

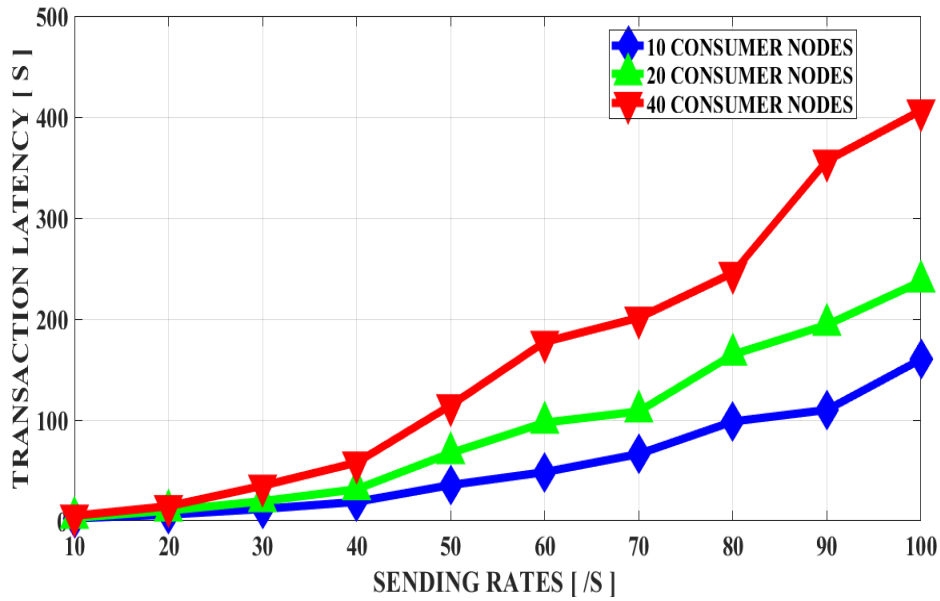

Figure 21. Transaction's latency with increasing send rates for different numbers of consumer node.

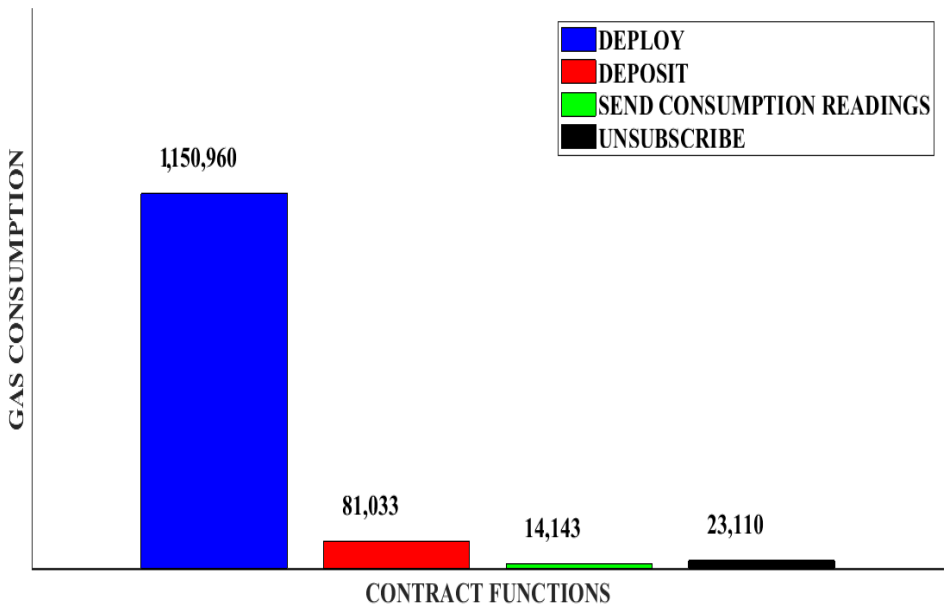

Figure 22. Gas consumption for Provider Smart Contract.

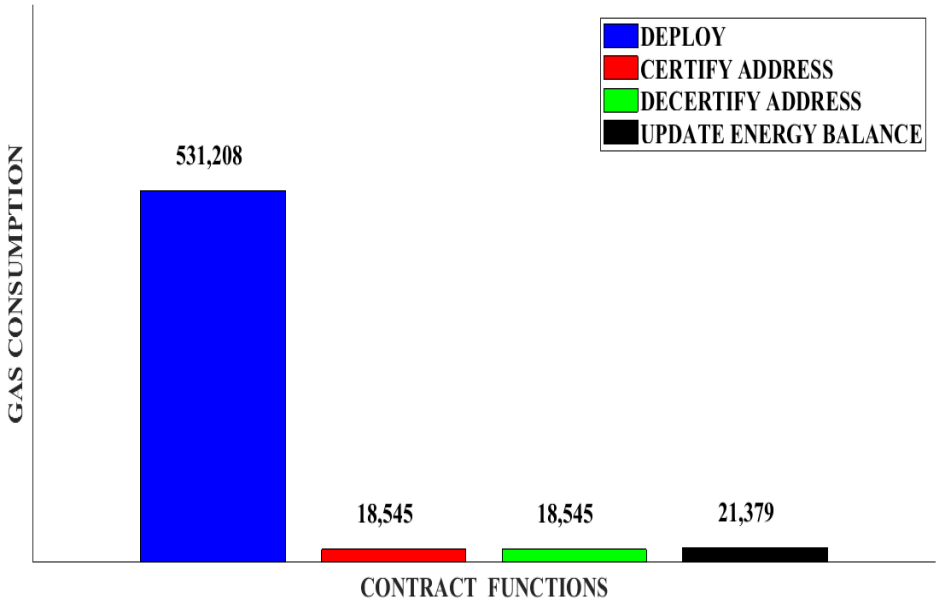

Figure 23. DSO Smart Contract Gas consumption.

6. Conclusions

In this paper, an Ethereum-based energy platform has been introduced where prosumers cooperate to form a single provider supplying client consumers with a prepaid blockchain-based billing system. The potential and the feasibility of such a platform have been presented in this paper as well as its assets over the traditional energy tokenization energy trading scheme present in the literature. An extensive description of the architecture and functioning of this platform as well as its implementation have been carried out throughout this study. Moreover, a ToU pricing model based on machine learning and autonomous clustering performed on a smart contract are presented. The present work aimed to spotlight the relevance of the proposed platform, as well as evaluate it based on pertinent metrics.

Author Contributions: Conceptualization, methodology, and writing, Y.M. and M.H.H.; software, Y.M.; validation, supervision & Revision: M.H.H., M.R.I., T.S.G., E.A.A.E., F.M.S. and M.M. Funding: M.H.H., M.R.I., E.A.A.E. and F.M.S. All authors have read and agreed to the published version of the manuscript.

Funding: The authors extend their appreciation to the Deanship of Scientific Research at King Khalid University for funding this work through Research Group Program under grant number (R.G.P.1/219/42). Yaine Merrad is grateful to IIUM Tuition Fee Waiver program.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Acknowledgments: This work was conducted at the IoT and Wireless Communication Protocols Laboratory at the ECE department, International Islamic University Malaysia (IIUM). It is partially sponsored by the Malaysian Ministry of Higher Education (MoHE) Prototype Research Grant Scheme number PRGS19-0013-0057. Yacine Merrad is grateful to IIUM Tuition Fee Waiver program.

Conflicts of Interest: The authors declare that they have no conflict of interest.

References

1. Carson, B.; Romanelli, G.; Walsh, P.; Zhumaev, A. Blockchain beyond the Hype: What Is the Strategic Business Value? 19 June 2018. Available online: https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/blockchain-beyond-the-hype-what-is-the-strategic-business-value#= (accessed on 11 August 2020).
2. Khan, S.; Loukil, F.; Ghedira-Guegan, C.; Benkhelifa, E.; Hani, H.B. Blockchain smart contracts: Applications, challenges, and future trends. *Peer-Netw. Appl.* **2021**, *14*, 2901–2925. [CrossRef] [PubMed]

3. Thukral, M.K. Emergence of blockchain-technology application in peer-to-peer electrical-energy trading: A review. *Clean Energy* **2021**, *5*, 104–123. [CrossRef]
4. Wang, B.; Zhao, S. Design of a privacy-preserving decentralized energy trading scheme in blockchain network environment. *Int. J. Electr. Power Energy Syst.* **2021**, *125*, 106465. [CrossRef]
5. Lu, X.; Guan, Z.; Zhou, X.; Wu, L.; Du, X.; Guizani, M. An Efficient and Privacy-Preserving Energy Trading Scheme Based on Blockchain. In Proceedings of the 2019 IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 9–13 December 2019; pp. 1–6.
6. Ekanayake, J.N.J.; Strbac, G. *Distributed Generation*; IET: London, UK, 2010.
7. Ahl, A.; Masaru, Y.; Kenji, T.; Daishi, S.; Review of blockchain-based distributed energy: Implications for institutional development. *Renew. Sustain. Energy Rev.* **2019**, *107*, 200–211. [CrossRef]
8. Taha, W.S.A.; Wang, J.; Kvaternik, K.; Hahn, A. Energy Crowdsourcing and Peer-to-Peer Energy Trading in Blockchain-Enabled Smart Grids. *IEEE Trans. Syst. Man Cybern. Syst.* **2019**, *49*, 1612–1623.
9. Bahramirad, L.Z.S.; Paaso, A.; Yan, M.; Shahidehpour, M. Blockchain for decentralized transactive energy management system in networked microgrids. *Electr. J.* **2019**, *32*, 58–72.
10. Yang, N.S.W.; Guo, M.; Koen, H.; Dam, V.; Wang, X. Energy demand side management within micro-grid networks enhanced by blockchain. *Appl. Energy* **2018**, *228*, 1385–1398.
11. ISGAN. Smarter & Stronger Power Transmission: Review of Feasible Technologies for Enhanced Capacity and Flexibility. ISGAN. Available online: https://www.iea-isgan.org/smarter-stronger-power-transmission-review-offeasible-technologies-for-enhanced-capacity-and-fl (accessed on 23 March 2022).
12. Cioara, P.C.T.; Antal, M.; Anghel, I.; Salomie, I.; Bertoncini, M. Blockchain based decentralized management of demand response programs in smart energy grids. *Sensors* **2018**, *18*, 162.
13. Park, C.; Yong, T. Comparative review and discussion on P2P electricity trading. *Energy Procedia* **2019**, *128*, 3–9. [CrossRef]
14. Ikpehai, J.O.A.; Anoh, K.; Adebisi, B.; Hammoudeh, M.; Son, S. State-of-the-art and prospects for peer-to-peer transaction-based energy system. *Energies* **2017**, *10*, 2106.
15. Silvestre, M.L.D.; Gallo, P.; Guerrero, J.M.; Musca, R. Blockchain for power systems: Current trends and future applications. *Renew. Sustain. Energy Rev.* **2020**, *119*, 109585. [CrossRef]
16. Foti, M.; Manolis, V. What blockchain can do for power grids? *Blockchain Res. Appl.* **2021**, *2*, 100008. [CrossRef]
17. Siano, P. Demand response and smart grids—A survey. *Renew. Sustain. Energy Rev.* **2014**, *30*, 461–478. [CrossRef]
18. O'Connell, N.; Pinson, P.; Madsen, H.; O'Malley, M. Benefits and challenges of electrical demand response: A critical review. *Renew. Sustain. Energy Rev.* **2014**, *39*, 686–699. [CrossRef]
19. Dalhues, S.; Zhou, Y.; Pohl, O.; Rewald, F.; Erlemeyer, F.; Schmid, D.; Zwartscholten, J. Research and practice of flexibility in distribution systems: A review. *CSEE J. Power Energy Syst.* **2019**, *5*, 285–294.
20. Song, J.; Kang, E.; Shin, H.; Jang, J.A. Smart Contract-Based P2P Energy Trading System with Dynamic Pricing on Ethereum Blockchain. *Sensors* **2021**, *21*, 1985. [CrossRef] [PubMed]
21. Hare, J.; Gathercole, M.; Foster, B.; Garrick, E. Whitepaper BRIKCOIN. Global Decentralized Property Platform, July 2019. Available online: https://icobench.com/ico/brikcoin/whitepaper (accessed on 23 March 2022).
22. WePower UAB. We Power. 27 February 2019. Available online: https://wepower.com/media/WhitePaper-WePower_v_2.pdf (accessed on 23 March 2022).
23. Devine, M.T.; Russo, M.; Cuffe's, P. *Blockchain Electricity Trading Using Tokenised Power Delivery Contracts*; ESRI Working Paper: Dublin, Ireland, 2019; pp. 1–11.
24. Cali, U.; Fifield, A. Towards the decentralized revolution in energy systems using blockchain technology. *Int. J. Smart Grid Clean Energy* **2019**, *8*, 245–256. [CrossRef]
25. Castellanos, J.A.F.; Coll-Mayor, D.; Notholt, J.A. Cryptocurrency as guarantees of origin: Simulating a green certificate market with the Ethereum Blockchain. In Proceedings of the 2017 IEEE International Conference on Smart Energy Grid Engineering (SEGE), Oshawa, ON, Canada, 14–17 August 2017; pp. 367–372.
26. Bitquery. Ethereum Mainnet 0x178c820f862b14f316509ec36b13123da19a6054 in Ethereum Mainnet. 3 December 2021. Available online: https://explorer.bitquery.io/ethereum/token/0x178c820f862b14f316509ec36b13123da19a6054 (accessed on 23 March 2022).
27. Bitquery. 0x4cf488387f035ff08c371515562cba712f9015d4. 3 December 2021. Available online: https://explorer.bitquery.io/ethereum/token/0x4cf488387f035ff08c371515562cba712f9015d4 (accessed on 23 March 2022).
28. Bitquery. 0xf4134146af2d511dd5ea8cdb1c4ac88c57d60404. 3 December 2021. Available online: https://explorer.bitquery.io/ethereum/token/0xf4134146af2d511dd5ea8cdb1c4ac88c57d60404 (accessed on 23 March 2022).
29. T Team. Etherscan. 2021. Available online: https://etherscan.io/tx/0x3421fd298722a02e2eae65f78904f6acb92d7a4c3d4f256fe7ccd782ca717786 (accessed on 23 March 2022).
30. Muzumdar, A.; Modi, C.; Madhu, G.; Vyjayanthi, C. A trustworthy and incentivized smart grid energy trading framework using distributed ledger and smart contracts. *J. Netw. Comput. Appl.* **2021**, *183–184*, 103074. [CrossRef]
31. Gai, K.; Wu, Y.; Gai, K.; Zhu, L.; Qiu, M.; Shen, M. Privacy-preserving energy trading using consortium blockchain in smart grid. *IEEE Trans. Ind. Inform.* **2019**, *15*, 3548–3558. [CrossRef]

32. Hlaing, K.M.; Nyaung, D.E. Electricity Billing System using Ethereum and Firebase. In Proceedings of the 2019 International Conference on Advanced Information Technologies (ICAIT), Yangon, Myanmar, 6–7 November 2019; pp. 217–221.
33. Gür, A.Ö.; Öksüzer, Ş.; Karaarslan, E. Blockchain Based Metering and Billing System Proposal with Privacy Protection for the Electric Network. In Proceedings of the 2019 7th International Istanbul Smart Grids and Cities Congress and Fair (ICSG), Istanbul, Turkey, 25–26 April 2019; pp. 204–208.
34. Aiman, S.; Hassan, S.; Habbal, A.; Rosli, A.; Shabli, M. Smart Electricity Billing System Using Blockchain Technology. *J. Telecommun. Electron. Comput. Eng.* **2018**, *10*, 91–94.
35. GfK Belgium Consortium. *Study on Residential Prosumers in the European Energy Union*; JUST/2015/CONS/FW/C006/0127 Framework Contract EAHC/2013/CP/04; European Commision: Brussels, Belgium, 2 May 2017.
36. Administration, Official Website of the International Trade. Poland–Country Commercial Guide. 3 September 2021. Available online: https://www.trade.gov/country-commercial-guides/poland-energy-sector (accessed on 23 March 2022).
37. Department for Business, Energy & Industrial Strategy. National statistics, Digest of UK Energy Statistics (DUKES): Electricity. 29 July 2021. Available online: https://www.gov.uk/government/statistics/electricity-chapter-5-digest-of-united-kingdom-energy-statistics-dukes (accessed on 23 March 2022).
38. Statista Research Department. Installed Capacity for Electricity Plants in France in 2020, by Energy Source (in Megawatts). 15 September 2021. Available online: https://www.statista.com/statistics/1263346/electrical-production-capacity-france/ (accessed on 23 March 2022).
39. IRENA, International Renewable Energy Agency. Portugal Europe RE SP. 29 September 2021. Available online: https://www.irena.org/IRENADocuments/Statistical_Profiles/Europe/Portugal_Europe_RE_SP.pdf (accessed on 23 March 2022).
40. Institut Fraunhofer pour les Systèmes Energétiques Solaires ISE à Fribourg. Energy-Charts. Available online: https://www.energy-charts.de/power_inst.htm (accessed on 23 March 2022).
41. Energy in Finland, April 2021. Available online: https://energiavirasto.fi/toimitusvarmuus (accessed on 1 June 2020).
42. Fernández, L. Cumulative Installed Power Capacity in Spain in 2021, by Technology (in Megawatts). 15 February 2022. Available online: https://www.statista.com/statistics/1002759/installed-power-capacity-in-spain/ (accessed on 23 March 2022).
43. IRENA, International Renewable Energy Agency. Greece Europe RE SP. 29 September 2021. Available online: https://www.irena.org/IRENADocuments/Statistical_Profiles/Europe/Greece_Europe_RE_SP.pdf (accessed on 23 March 2022).
44. Alves, B. Statista, Installed Electricity Capacity in Belgium 2000–2019. 9 February 2022. Available online: https://www.statista.com/statistics/1186751/belgium-installed-electricity-capacity/ (accessed on 23 March 2022).
45. Bauwens, T.; Gotchev, B.; Holstenkamp, L. What drives the development of community energy in Europe? The case of wind power cooperatives. *Energy Res. Soc. Sci.* **2016**, *13*, 136–147. [CrossRef]
46. Hasse, F.; Perfall, A.V.; Hillebrand, T.; Smole, E.; Lay, L. Blockchain—An Opportunity for Energy Producers and Consumers. 2019. Available online: https://www.pwc.com/gx/en/industries/assets/pwc-blockchain-opportunity-for-energy-producers-and-consumers.pdf (accessed on 23 March 2022).
47. Legislation.gov.uk. Directive 2011/83/EU of the European Parliament and of the Council. 25 October 2011. Available online: https://www.legislation.gov.uk/eudr/2011/83/introduction# (accessed on 23 March 2022).
48. Merz, M. Enerchain—Decentrally Traded Decentral Energy. 2020. Available online: https://enerchain.ponton.de/ (accessed on 23 March 2022).
49. EUR-Lex. Directive (EU) 2018/2001 of the European Parliament and of the Council of 11 December 2018. 11 December 2018. Available online: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv:OJ.L_.2018.328.01.0082.01.ENG (accessed on 23 March 2022).
50. Butenko, A. Sharing energy: Dealing with regulatory disconnection in dutch energy law. *Eur. J. Risk Regul.* **2016**, *7*, 701–716. [CrossRef]
51. Reiner Lemoine Institut GmbH. Blockchain for a Smart Energy Market: BEST. 2021. Available online: https://reiner-lemoine-institut.de/en/blockchain-energy-market-best/ (accessed on 23 March 2022).
52. Kirli, D.; Couraud, B.; Robu, V.; Salgado-Bravo, M.; Norbu, S.; Andoni, M.; Antonopoulos, I.; Negrete-Pincetic, M.; Flynn, D.; Kiprakis, A. Smart contracts in energy systems: A systematic review of fundamental approaches and implementations. *Renew. Sustain. Energy Rev.* **2022**, *158*, 112013. [CrossRef]
53. Ikeda, K. Chapter Seven—Security and Privacy of Blockchain and Quantum Computation. *Adv. Comput.* **2018**, *111*, 199–228.
54. Kiktenko, E.O.; Pozhar, N.O.; Anufriev, M.N.; Trushechkin, A.S.; Fedorov, A.K. Quantum-secured blockchain. *Quantum Sci. Technol.* **2018**, *3*, 035004. [CrossRef]
55. Cai, Z.; Qu, J.; Liu, P.; Yu, J. A Blockchain Smart Contract Based on LightWeighted Quantum Blind Signature. *IEEE Access* **2019**, *7*, 138657–138668. [CrossRef]
56. Mavroeidis, V.; Vishi, K.; Zych, M.D.; Jøsang, A. The Impact of Quantum Computing on Present Cryptography. *Int. J. Adv. Comput. Sci. Appl.* **2018**, *9*, 405–414. [CrossRef]
57. Alexeev, Y.; Bacon, D.; Brown, K.R.; Calderbank, R.; Carr, L.D.; Chong, F.T.; DeMarco, B.; Englund, D.; Farhi, E.; Fefferman, B.; et al. Quantum Computer Systems for Scientific Discovery. *PRX Quantum A Phys. Rev. J.* **2021**, *2*, 017001. [CrossRef]
58. An Official Website of the United States Government. PQC Standardization Process: Third Round Candidate Announcement. 22 July 2020. Available online: https://csrc.nist.gov/News/2020/pqc-third-round-candidate-announcement (accessed on 23 March 2022).

59. Ducas, L.; Kiltz, E.; Lepoint, T.; Lyubashevsky, V.; Schwabe, P.; Seiler, G.; Stehlé, D. CRYSTALS-Dilithium: A Lattice-Based DigitalSignature Scheme. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2018**, *2018*, 238–268. [CrossRef]
60. Kiningham, K.; Levis, P.; Anderson, M.; Boneh, D.; Horowitz, M.; Shih, M. Falcon—A flexible architecture for accelerating cryptography. In Proceedings of the 2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems (MASS), Monterey, CA, USA, 4–7 November 2019; pp. 136–144.
61. Yasuda, T.; Sakurai, K. A multivariate encryption scheme with rainbow. In *International Conference on Information and Communications Security*; Springer: Cham, Switzerland, 2016; pp. 236–251.
62. Over 10 Years of Hourly Energy Consumption Data from PJM in Megawatts. Kaggle, Version 3. 2019. Available online: https://www.kaggle.com/robikscube/hourly-energy-consumption?select=AEP_hourly.csv (accessed on 23 March 2022).
63. Bişkin, O.T.; Çifci, A. Forecasting of Turkey's Electrical Energy Consumption using LSTM and GRU Networks. *BSEU J. Sci.* **2021**, *8*, 656–667. [CrossRef]
64. Tokgöz, A.; Ünal, G. A RNN Based Time Series Approach for Forecasting Turkish Electricity Load. In Proceedings of the 26th Signal Processing and Communications Applications Conference (SIU), Izmir, Turkey, 2–5 May 2018; pp. 1–4.
65. Niya, S.R.; Schüpfer, F.; Bocek, T.; Stiller, B. A Peer-to-peer Purchase and Rental Smart Contract-based Application (PuRSCA). *Inf. Technol.* **2018**, *60*, 307–320.
66. Tabarak, A.Y.Y.; Nunna, H.; Doolla, S. Decentralized transactive energy management system for distribution systems with prosumer microgrids. In Proceedings of the 2018 19th International Carpathian Control Conference (ICCC), Szilvasvarad, Hungary, 28–31 May 2018; pp. 553–558.
67. Anagnostopoulos, G.; Kalousis, A. A reproducible analysis of RSSI fingerprinting for outdoor localization using sigfox: Preprocessing and hyperparameter tuning. In Proceedings of the 2019 International Conference on Indoor Positioning and Indoor Navigation (IPIN), Pisa, Italy, 30 September–3 October 2019; pp. 1–8.
68. Larijani, H.; Gibson, R.M. LoRa RSSI based outdoor localization in an urban area using random neural networks. *Comput. Conf.* **2021**, *2*, 1032–1043.
69. Goldoni, E.; Prando, L.; Vizziello, A.; Savazzi, P.; Gamba, P. Experimental data set analysis of RSSI-based indoor and outdoor localization in LoRa networks. *Internet Technol. Lett.* **2018**, *2*, e75. [CrossRef]
70. Barai, S.; Biswas, D.; Sau, B. Sensors Positioning for Reliable RSSI-based Outdoor Localization using CFT. In Proceedings of the 2020 IEEE International Symposium on Sustainable Energy, Signal Processing and Cyber Security (iSSSC), Gunupur Odisha, India, 16–17 December 2020; pp. 1–5.