

Marshall University

Marshall Digital Scholar

Theses, Dissertations and Capstones

2021

Peer-to-Peer Energy Trading for Networked Microgrids

Jonathan David Warner

Follow this and additional works at: <https://mds.marshall.edu/etd>



Part of the [Digital Communications and Networking Commons](#), [Electrical and Computer Engineering Commons](#), and the [Environmental Engineering Commons](#)

PEER-TO-PEER ENERGY TRADING FOR NETWORKED MICROGRIDS

A thesis submitted to
the Graduate College of
Marshall University
In partial fulfillment of
the requirements for the degree of
Master of Science

In
Electrical & Computer Engineering
by

Jonathan David Warner

Approved by

Dr. Tarek Masaud, Committee Chairperson

Dr. Taher Ghomian

Dr. Pingping Zhu

Marshall University
May 2021

APPROVAL OF THESIS

We, the faculty supervising the work of Jonathan David Warner, affirm that the thesis, *Peer-To-Peer Energy Trading For Networked Microgrids*, meets the high academic standards for original scholarship and creative work established by the Electrical & Computer Engineering Masters program and the College of Engineering and Computer Sciences. The work also conforms to the formatting guidelines of Marshall University. With our signatures, we approve the manuscript for publication.

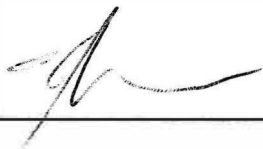


Dr. Tarek Masaud, CSEE

Committee Chairperson

3/18/2021

Date



Dr. Taher Ghomian, CSEE

Committee Member

3/18/2021

Date



Dr. Pingping Zhu, CSEE

Committee Member

3/18/2021

Date

© 2021
JONATHAN DAVID WARNER
ALL RIGHTS RESERVED

ACKNOWLEDGMENTS

I want to thank all the people who helped and supported me to achieve this milestone. I cannot begin any acknowledgement without first thanking my parents, David and Denise, whose love, support, and encouragement have carried me through every stage of my journey. I must also thank my brother, Tim, for being my inspiration to overachieve and my life-long target to outperform. I also want to thank my many friends who have encouraged, counseled, and accommodated me on this journey. Lastly, I want to thank my adviser, Dr. Masaud, who has provided me with and guided me through many opportunities to grow as a researcher and a scholar.

TABLE OF CONTENTS

List of Nomenclature	vii
List of Tables	viii
List of Figures	ix
Abstract	xi
Chapter 1: Introduction	1
1.1. Background	1
1.2. Literature Review	5
1.3. Contributions	10
1.4. Thesis Outline	11
Chapter 2: Peer-to-Peer Energy Trading In Distribution Networks: An Overview	12
2.1. Introduction	12
2.2. Peer-To-Peer Energy Trading Mechanism	16
2.2.1. Game Theory	18
2.2.2. Auction Markets	19
2.2.3 Constrained Optimization	20
2.3. Blockchain-Based P2P Trading	21
Chapter 3: The Proposed P2P Energy Trading Model	27
3.1. Islanded Operation Mode – Multiagent Model	27
3.1.1. Seller and Buyer Identification	28
3.1.2. Price Adjustment and Contract Matching Mechanism	29
3.2. Grid Connected Operation Mode – Multiagent Model	35
3.2.1. Formulation of the Optimization Scheduling Problem	35

3.2.2. Energy Trading and Price Adjustment Model – Model I	36
3.2.3. Energy Trading and Price Adjustment Model – Model II	41
3.3. Two-Phase Blockchain Consensus Protocol.....	43
Chapter 4: Simulation Results	48
4.1. Islanded Operation Mode – Multiagent Model.....	48
4.2. Grid Connected Operation – Multiagent Models.....	57
4.2.1. Model I.....	57
4.2.2. Model II	62
Chapter 5: Conclusion.....	68
5.1. Future Work	69
5.2. Outcome Publications	69
References	71
Appendix A: Approval Letter from the Office of Research Integrity	80
Appendix B: Tabulated Data	81
Appendix C: Simulation Codes	83

LIST OF NOMENCLATURE

Sets:

m	Seller
n	Buyer
r	Price adjustment round
t	Hour

Parameters:

NL	Net load
$P_{g,max}$	Maximum generation capacity of microgrid
C_{ps}	Maximum storage capacity of microgrid
C^g	Cost of generation for DG
C^{pk}	Cost of peak plant generation
C^{bss}	Cost of battery storage
C^{cur}	Energy and load curtailment cost
C^{sh}	Energy and load shedding cost
SU	Startup cost
SD	Shutdown cost
$p_{g,min}$	Minimum generation limit of DG
$p_{g,max}$	Maximum generation limit of DG
$p_{bss,dch,max}$	Maximum discharge power from BSS
$p_{bss,dch,min}$	Minimum discharge power from BSS
$p_{bss,ch,max}$	Maximum charge power to BSS
$p_{bss,ch,min}$	Minimum charge power to BSS
E^{min}	Minimum state of charge of BSS

E^{max}

Maximum state of charge of BSS

$p^{tie,max}$

Maximum capacity of utility tie line

p^{pk}

Maximum energy from peak generation

p^{cur}

Maximum energy to curtail

C^{res}

Cost of spinning reserve energy

Variables:

P^g

Generation of DG

p^{bss}

Power charged to or discharged from BSS

SOC

State of charge of BSS

p^{grid}

Power exchanged with the utility

p^{res}

Power from spinning reserve

λ^{grid}

24-hour dynamic utility price

$\lambda^{utility}$

Fixed utility price

p^{net}

Amount of power to be traded

u

Binary variable describing the dispatch of DG

y

Binary variable describing startup status

z

Binary variable describing shutdown status

d

Binary variable describing discharge state

c

Binary variable describing charge state

λ^{sell}

Desired selling price

λ^{buy}

Desired buying price

LIST OF TABLES

Table 3.1	Price adjustment and contract execution algorithm - islanded operation	33
Table 3.2	Price adjustment and contract execution algorithm - grid- connected operation model I.....	41
Table 3.3	Price adjustment and contract execution algorithm - grid- connected operation model II.....	43
Table 3.4	Two-phase blockchain consensus protocol	47
Table 4.1	Amount of power traded in each contract over the 24-hour time horizon	52
Table 4.2	Validation of obtained results in comparison to results from the literature	57

LIST OF FIGURES

Figure 2.1	Microgrid concept	12
Figure 2.2	Group of interconnected microgrids forming a microgrid network	16
Figure 2.3	Prosumer definition	17
Figure 2.4	General blockchain structure	22
Figure 3.1	System model (islanded operation)	28
Figure 3.2	Flowchart of the proposed trading model (grid connected operation)	37
Figure 3.3	System model (grid connected operation).....	38
Figure 3.4	pBFT two round voting process with faulty node tolerance	44
Figure 3.5	Flowchart of the proposed model	47
Figure 4.1	Number of executed contracts and amount of traded power in the case of 10 interconnected microgrid system.....	50
Figure 4.2	A successful price adjustment process for a selected block	51
Figure 4.3	Variation in the computation time with respect to the change of the number of microgrids in the network	53
Figure 4.4	Variation in the number of executed contracts with respect to the change of the number of microgrids in the network.....	54
Figure 4.5	A sample of generated blocks containing contract data	55
Figure 4.6	The change in average validation time with respect to the change of the number of microgrids in the network.....	56
Figure 4.7	24-hour dynamic energy prices	59
Figure 4.8	Excess and deficit power to be traded from seven networked microgrids	59

Figure 4.9	Amount of deficit power vs. satisfied deficit power resulting from the peer-to-peer trading model.....	60
Figure 4.10	Comparison of the model trading price to the utility price	61
Figure 4.11	An example of the price adjustment process of two buyers and two sellers	62
Figure 4.12	A sample of generated blocks containing contract data	63
Figure 4.13	Comparison of energy deficit to traded energy	64
Figure 4.14	Comparison of the model trading price to utility price	64
Figure 4.15	An example of price adjustment process of two buyers and two sellers.....	66
Figure 4.16	Total energy traded in each hour, including primary and spinning reserve power	67
Figure 4.17	A sample of generated blocks containing contract data	68
Figure 4.18	A sample of generated blocks including a spinning reserve contract block.....	68

ABSTRACT

Considering the limitations of the existing centralized power infrastructure, research interests have been directed to decentralized smart power systems constructed as networks of interconnected microgrids. Therefore, it has become critical to develop secure and efficient energy trading mechanisms among networked microgrids for reliability and economic mutual benefits. Furthermore, integrating blockchain technologies into the energy sector has gained significant interest among researchers and industry professionals. Considering these trends, the work in this thesis focuses on developing Peer-to-Peer (P2P) energy trading models to facilitate transactions among microgrids in a multiagent network. Price negotiation mechanisms are proposed for both islanded and grid-connected microgrid networks. To enable a trusted settlement of electricity trading transactions, a two-stage blockchain-based settlement consensus protocol is also developed. Simulation results have shown that the model has successfully facilitated energy trading for networked microgrids.

CHAPTER 1

INTRODUCTION

This chapter will include a discussion of the background and motivation which inspired the undertaking of this research. Additionally, this chapter includes an extensive literature review of the topic, and a summary of the thesis scope and contributions.

1.1 Background

For decades, energy consumers have relied on the large-scale power grid to supply electrical power for their homes and businesses. The power grid has traditionally consisted of utility-owned and operated electric power generation plants connected to high voltage transmission lines delivering electricity up to hundreds of miles away from the power source. The highly centralized nature of these power grids has been a significant concern for system reliability and resiliency, where a single-equipment failure can have extensive effects on the rest of the grid, leading to localized power disruptions, generation outages, and even large-scale blackouts [1]. Additionally, the power grid has traditionally relied on conventional, non-renewable energy sources such as coal and natural gas. These fossil fuels, in addition to being a depleting resource, have been shown to contribute greatly to environmental pollution and growing crisis of climate change [2].

In recent years, the concept of a smart grid has begun to emerge which aims to alleviate and eliminate many of the concerns of the traditional power grid. Smart Grid is the term for the next generation of the power grid which is currently being researched, developed, and deployed around the world. The Smart Grid includes many technological upgrades and additional control features to the grid including smart meter technology, smart appliances, and a focus on renewable energy and other energy efficient resources [3]. The goal of the Smart Grid is to

optimize the economic and operational efficiency of the grid by improving grid reliability and demand response flexibility. Because of this emphasis on efficiency and integration of renewable energies, it is common for the future smart grid to be imagined as a system of interconnected smart microgrids [4]. IEEE standard 1547.4 has confirmed that representing large power grids by a group of interconnected microgrids significantly enhances the reliability, resiliency, and sustainability of the network [5]. Thus, significant attention has been paid to the concept of networked microgrid operation in recent years.

Microgrid is defined by the US Department of Energy as “a group of interconnected loads and distributed energy resources within clearly defined electrical boundaries that acts as a single controllable entity with respect to the grid. A microgrid can connect and disconnect from the grid to enable it to operate in both grid-connected or island mode” [5]. Microgrid applications have been developing steadily over the past few years and deployment of microgrids are anticipated to increase even more in the near future, especially with an increasing interest in the use of smart grid technology.

The most important aspect of any power grid operation is the ability to maintain generation-demand balance at all times. However, due to the intermittent nature of the renewable energy sources present in the microgrids and the unique energy demands of each grid, all microgrids will have different energy profiles. It is an essential principle of any grid operation to maintain a generation-demand balance. However, for any given interval in a 24-hour day, some microgrids will be experiencing energy deficit conditions where generation is less than demand, while others may be experiencing energy surplus conditions where generation is greater than demand. Therefore, in order to maintain real-time balance of local power generation and demand, microgrids may seek to establish energy trading coalitions to share energy resources.

Coalition formation and the energy trading mechanism used to assure adequate power sharing among networked microgrids are crucial elements to the success of the interconnected microgrid network. Energy trading helps to ensure generation-demand imbalance is mitigated as much as possible based on available resources while also minimizing microgrid operation costs by maximizing the economic efficiency of the microgrid network.

Centralized energy trading models are relatively simplistic and convenient; however they are hard to scale for a large number of entities. Furthermore, centralized trading schemes make networks highly susceptible to cyber-attacks [6]. Because of these drawbacks, the emergence of Blockchain and the great amount of attention given to it has led to a tremendous amount of interest in using the technology within information infrastructures to assure secure and decentralize energy trading [6]. In the U.S., the Brooklyn Microgrid project is an example of a first generation successful peer-to-peer blockchain system operating through smart meters, where prosumers are able to trade energy based on pre-determined bid prices [7].

As previously noted, the concept of blockchain is becoming a popular choice for secure transactions in decentralized networks. First proposed in 2008, a blockchain is a simple growing list of public records known as blocks which are securely linked to one another to form a chain. The original application of the blockchain was to provide a distributed public transaction record for the Bitcoin cryptocurrency, but the concept has been adapted to be used in any case where transactional security is a concern. The primary advantage of the blockchain is its distributed, decentralized structure, which allows for direct peer-to-peer interaction without the concern of a potentially compromised central entity. The blockchain also employs a consensus method, whereby blocks must be validated by varying means before being appended to the blockchain.

Due to the immutability of the blockchain record and the consensus methods used to validate information appended to the chain, blockchain is a perfect candidate for tracking financial transaction records [8]. These blockchains are constructed as digital ledgers, where each block contains the details of a financial transaction. The data contained in a block includes the identification of the parties participating in the transaction, the amount of goods being transacted, the timestamp of the execution of the transaction, and an alpha-numeric string called a hash, which is taken from the previous block. This data is input into a hashing function, generating a new and unique hash which is appended to the block, separate from the transaction data. All information is then updated synchronously to the entire network so that each peer (node in the network) keeps a record of the same ledger. The next block includes the hash from the previous block as part of its data and generates a new hash, which is then used by the next block. Therefore, the blocks are chained together by these uniquely generated hashes. If any of the data in a chained block is modified, the hash associated with that block will change, no longer matching the hash used in the next block and thereby breaking the chain. Because of its consensus method, integrity of the data recorded in the ledger can be guaranteed without a trusted third party [9]. Various consensus algorithms have been developed such as Proof of Work (PoW), Proof of Stake (PoS), Delegated Proof of Stake (DPoS), Ripple Protocol Consensus Algorithm (RPCA) and AlgoRand [9]. The consensus algorithm is the most important factor of the entire blockchain system, not only because it is the primary method of blockchain security, but also because the efficiency of the consensus algorithm is the primary factor influencing the blockchain's performance.

1.2 Literature Review

In the existing literature, numerous models have been developed for Peer-to-Peer (P2P) energy trading with and without considering the integration of blockchain technology, which can be classified into three main areas: game theory models [10] – [13], auction models [14] – [17], and analytical models [18] – [20]. Researchers in [10] propose a novel game-theoretic model for P2P energy trading using direct interaction between prosumers with a particular consideration for demand response capability and privacy. Price negotiations are modeled as a non-cooperative game, with a novel iterative algorithm developed to reach price equilibrium. The work proposed in [11] suggests a strategy for energy storage allocation utilizing a Stackelberg game. In this strategy, each participant submits an initial bid price which is evaluated by a central, third-party participant. The third party then calculates a target price between the minimum and maximum bid prices which seeks to maximize the average cost savings among participants. After the selected target price is proposed, participants may adjust the amount of energy storage allocation they are willing to purchase, which would in turn affect the target price calculation. The target price and participant energy storage amounts are continuously updated until an equilibrium is reached where all bids are finalized and payments are exchanged. A primary drawback of the proposed strategy is the requirement of a centralized entity to control price adjustments for trading. Additionally, the economic efficiency is not maximized for each participant. The model proposed in [12] formulates a cooperative game to solve energy trading with bilateral decision making. In this model, buyers and sellers determine their maximum and minimum acceptable costs and maximum and minimum acceptable payoff. The game is solved when an appropriate bilateral price is determined for both parties. The efficiency of the model is shown for energy trading between single seller single buyer and single seller multiple buyers; however, the model

is not extended to scenarios which consider multiple sellers in the marketplace. The work in [13] proposes a Stackelberg game-based dynamical pricing strategy for energy sharing. A central trading operator sets the buying and selling prices for the microgrid, while the energy prosumers are able to determine their level of energy consumption from the microgrid and from the utility.

Researchers in [14] propose an energy market auction for a community microgrid. The community microgrid establishes trading between its prosumers in an effort to purchase energy at a price below the utility price. The utility participates as an infinite energy source in the market, therefore excess generated power in the microgrid will remain unsold if it cannot beat the utility price. Additionally, the proposed trading market is conditioned for noniterative bidding, which does not allow for energy prices and bids to be dynamically updated to encourage trading. In [15], an iterative double auction price adjustment mechanism is proposed in an effort to maximize social welfare. A third-party energy broker entity acts as an auctioneer, determining the execution of energy trades based on buying and selling prices submitted by participants. In this way, the third-party entity ultimately determines the final trading price and amount.

Researchers in [16] propose a continuous double auction-based electricity market utilizing a predictive trading optimization model. In this strategy, an optimization problem which determines optimal grid operation is integrated into the auction model in order to optimize the bidding behaviors of buyers and sellers. This optimized auction model relies on market pricing predictions based on historical transaction data. The work in [17] proposes a multiagent method for energy trading utilizing a strict reverse auction bidding scheme. In this method, an aggregator acts as a central entity interfacing with generation entities in the microgrid which bid to fulfill the microgrid demand with the aggregator having control over the selection of bids. If the available energy supply is higher than the demand, the aggregator lowers the purchasing price and accepts

updated bids from willing participants. This process continues until only one participant remains and the trade is executed at the final price. The primary drawback of this work is also the reliance on a central trade control entity.

The work in [18] develops a P2P linear programming trading model for applications in decentralized and centralized storage energy markets. The model is designed as a multi-energy management strategy with a goal to minimize electricity costs in the trading community. The authors admit that price determination is highly dependent on the assumed consumption costs, which in the proposed modeled are determined from grid prices. Authors in [19] propose an optimization model for the operation of PV systems in the context of P2P energy trading. The proposed model is designed as a mixed integer linear programming model used to optimize operational decisions of a distributed energy market which allows P2P energy trading. In this way, the pricing mechanism for energy trading is directly incorporated in the optimization problem. Research in [20] proposes a dynamical pricing model for energy sharing considering the supply and demand ratio of shared PV energy. Prosumers calculate desired prices in an effort to minimize economic and inconvenience costs. Buyers attempt to match their prices to the seller price, with both updating their prices iteratively based on the cost function until converging on their finalized prices. If prices do not match, the excess demand is satisfied by the utility. None of the trading models proposed in [10] – [20] consider costs associated with islanded microgrid operation, such as energy curtailment and load shedding. Additionally, the existing models fail to consider other factors which influence the buying and selling prices such as spinning reserve and energy storage costs.

In terms of integrating blockchain technology, few studies have focused on developing decentralized energy trading using blockchain technology [21] – [25]. For instance, the work in

[21] proposes a two-layer algorithm for blockchain-based energy trading negotiation and transaction settlement among grid connected networked prosumers. The first layer is a contract chain, which contains data related to the energy transaction. The second layer is a ledger chain, which tracks the trading balance of the microgrid network. Because the data in these two chains directly influence one another, the immutability of their records is further ensured by a high frequency verification mechanism between corresponding blocks in each chain. Researchers in [22] suggest the use of a smart contract method based on energy tokens, where the energy token represents a unit of power at a fixed price. This work shows that utilizing blockchain for smart contract allows automatic execution of energy trading contracts in a secure, decentralized network. Authors in [23] extend the energy token method by using a linear time-based value depreciation model for the energy tokens. This method stimulates energy trading by incentivizing the buying and selling of tokens within a time limit. An incentivizing method utilizing Nash bargaining theory is presented in [24]. In [25], the impact of applying load management on reducing the energy cost bought from a blockchain-based peer-to-peer energy trading market is studied. PoW is used as the blockchain consensus method in this work.

System security is crucial to the successful operation of interconnected energy trading systems, and recently proposed models have turned to blockchain technology to address these concerns [8, 26-30]. The work in [26] proposes a blockchain model for detecting data corruption produced by third party intrusions based on a collaborative intrusion detection approach. The use of blockchain as a means of intrusion detection removes the security risk of a central authority while improving the speed and accuracy of detection. A modified blockchain approach utilizing directed acyclic graphs is proposed in [8] to ensure the preservation of security in networked microgrids in order to minimize operational costs. The proposed method additionally includes a

data restoration technique for the event of a corruption on the blockchain due to third-party intrusion. A unified energy blockchain based on consortium blockchain for secure peer-to-peer energy trading in industrial internet of things is presented in [27]. This unified energy blockchain utilizes a traditional proof of work consensus method to validate blocks on the chain. To ease the limitation of execution time, a credit-based payment scheme is also proposed to stimulate fast peer-to-peer energy trading. To increase the system security and privacy, differentially private energy trading auction using consortium blockchain for microgrids systems is proposed in [28]. This model seeks to increase the security and privacy of traditional auction-based peer-to-peer trading by utilizing consortium blockchain technology. Researchers in [29] propose the use of consortium blockchain to ensure privacy protection of direct transactions between microgrids. In an effort to improve the efficiency of blockchain transactions, the use of practical byzantine fault tolerance (pBFT) is proposed as an alternative to traditional consensus methods in this work. Peer-to-peer energy trading based on Blockchain implementation using Hyperledger Fabric considering different energy transaction scenarios and crowdsources is presented in [30]. This work considers blockchain utilizing a modified pBFT consensus method but requires a central managing entity in large-scale crowdsourced implementation.

Based on the conducted literature review, it was found that there are three aspects that need further investigation:

- (i) How the energy trading price and amount is determined for both islanded and grid-connected microgrid networks.
- (ii) Once a trading transaction is completed, there is a need to develop more secure energy- and time-efficient consensus algorithms to settle those transactions in the blockchain.

Additionally, the existence of malicious nodes that might invalidate the voting process of

the consensus mechanism and manipulate the recorded data need to be considered in the algorithm's development. Otherwise, the blockchain system may become insecure, unreliable, and inefficient.

- (iii) In spite of the numerous advantages of using blockchain in developing a trustable trading environment, the concern of network privacy is still a primary concern restricting blockchain implementation, especially in peer-to-peer- trading mechanisms.

1.3 Contributions

To resolve the above challenges, the work in this thesis focuses on developing P2P energy trading models to facilitate transactions among microgrids in a multiagent network. Price negotiation mechanisms are proposed for both islanded and grid-connected microgrid networks. Additionally, a two-stage blockchain-based energy trading algorithm for a group of networked microgrids is considered to ensure the security of the energy trading mechanism. The two-stage algorithm develops an energy trading-based smart contract mechanism in stage one, and a transaction settlement method is developed in the second stage. The contribution of this work can be summarized as follows.

1. Development of a pre-conditioned smart contract-based energy trading mechanism to allow microgrids to establish coalitions, negotiate the electricity trading price, and the amount of energy to be traded. Significantly distinguished from the work in the literature, this method is uniquely developed in such a way that the pre-determined smart contracts are executed autonomously in a blindly traded energy marketplace, where peers do not share their data, including the energy prices during the trading process. This contributes to the need to develop more privacy-preserving negotiation mechanisms for peer-to-peer trading oriented processes.

2. Distinguished from the work in the literature [16] – [26], this work proposes a new blockchain-based contract settlement protocol utilizing a two-phase consensus algorithm consisting of pBFT and a modified PoS to ensure system security, energy and time efficiency.
3. The proposed algorithm establishes a price adjustment mechanism for islanded operation and two distinct price adjustment mechanisms for grid-connected operation of microgrid networks. In contrast to the work in the literature, the cost of microgrid local resources (storage cost, curtailment cost, load shedding cost, and spinning reserve) are considered in the price adjustment process to incentivise the P2P energy trading.

1.4 Thesis Outline

This thesis is structured in five chapters. Chapter 1 provides background and motivation, literature review, research contributions, and a thesis outline. Chapter 2 contains an extended discussion of peer-to-peer energy trading applications for networked microgrids and blockchain technology. Chapter 3 presents the proposed energy trading model for the networked microgrid applications, including the development of the pricing mechanisms for islanded and grid-connected operation and the blockchain-based settlement protocol. Chapter 4 details the simulation of the proposed models with an extensive presentation and analysis of the obtained results. Additionally, the results of the simulation are compared against existing solutions found in the literature. Finally, Chapter 5 provides the conclusions of the thesis work and proposes future research to extend the work contained in the thesis.

CHAPTER 2

PEER-TO-PEER ENERGY TRADING IN DISTRIBUTION NETWORKS: AN OVERVIEW

2.1. Introduction

The United States Department of Energy defines a microgrid as, “a group of interconnected loads and distributed energy resources within clearly defined electrical boundaries that acts as a single controllable entity with respect to the grid. A microgrid can connect and disconnect from the grid to enable it to operate in both grid-connected or island mode [5].” Fig. 2.1 shows a schematic diagram of the microgrid concept. From this definition we can identify three major distinguishing characteristics of a microgrid: (1) microgrids exist at the distribution level, consisting of distributed energy sources serving localized loads, (2) microgrids are capable of operating both with and without a connection to the power grid, and (3) microgrids have a layer of intelligent control, enabling the microgrid to actively manage its resources to operate successfully whether it is islanded or grid connected.

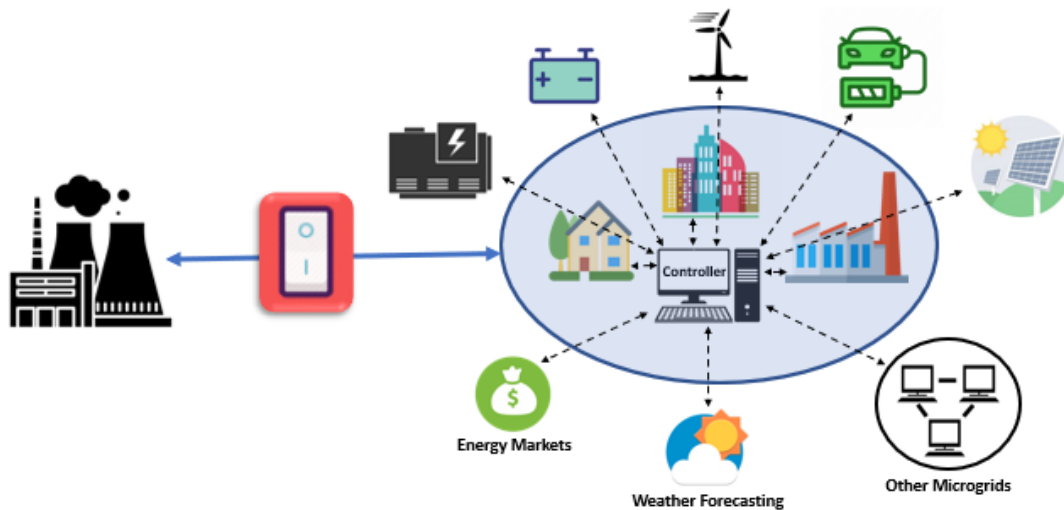


Figure 2.1. Microgrid concept

As mentioned previously, the prevailing method of energy distribution for the last century has consisted of large-scale, centralized energy production facilities generating and transmitting power over long distances up to hundreds of miles away from the generation. To mitigate losses over these long distances, power is transmitted at very high voltages up to 750 kV. Stepping up to this voltage requires large, expensive step-up transformers at the generating facilities, and similarly expensive step-down transformers to gradually step down from transmission to distribution voltage, and then finally to residential service levels. According to the US Energy Information Administration, the amount of power lost in transmission and voltage transformation is on average 5% of the total energy production [31]. Microgrids avoid the issue of transmission power loss altogether by relying on distributed generation technology (DG). DG can be defined as small energy generating resources installed at the distribution level. DG is already localized to the loads it serves, and therefore requires little to no power transmission to deliver energy to the consumer.

An additional drawback of the current power grid system is the centralized nature of the power grid. Large-scale utility generation plants service thousands of individual customers through a small number of transmission lines, the loss of which would impact those customers. Conversely, the islanding capability enables microgrid to maintain the power supply during power grid outages, and reliably supply power to its local loads.

The second unique characteristic of the microgrid is its ability to operate with or without a connection to the power grid. When the microgrid has a connection to the power grid, it is said to be grid-connected. This grid connection gives the microgrid the ability to exchange energy with the grid, either by purchasing energy from the grid in the event that the microgrid cannot

generate enough energy to satisfy its demand, or by selling energy to the grid in the event that the microgrid has generated more energy than it has demand. When the microgrid is disconnected from the power grid, it is said to be islanded, or operating in islanded mode. A microgrid operating in islanded mode does not have the capability of exchanging energy with the grid, and therefore must rely on its own generation to satisfy the local demand.

Islanded operation can be either voluntary or involuntary. In the majority of voluntary cases, the microgrid is islanded because a connection to the grid is impractical, such as when the microgrid is servicing loads in a remote community [4]. Additionally, a microgrid can choose to enter islanded operation when the utility grid experiences disturbances. A microgrid may be involuntarily islanded due to a loss of the tie line between the microgrid and the power grid as a result of equipment failure or a severe weather event [4]. Also, a system blackout in the power grid could be considered an involuntary islanding. Fig. 2.1 illustrates the ability of a microgrid to operate in either grid connected or islanded mode.

The third unique characteristic of the microgrid is the ability to intelligently control its energy resources to maximize the efficiency and increase the deployment of renewable distributed energy sources (RES). As mentioned previously, microgrids are equipped with an array of local energy sources (energy storage, dispatchable generation units, controllable loads, etc) which can be utilized to accommodate the variable nature of renewable generation. Microgrids manage its local resources in one of two primary ways: centralized control or distributed control. The microgrid utilizes a central controller which is tasked with scheduling the dispatch of the all the various energy sources and managing all the loads in the grid [32]. For distributed control schemes, the microgrid relies on the local control of each resource in the microgrid [33]. Distributed schemes are often structured as Multi-Agent Systems (MAS), where

each resource is equipped with a controller agent that coordinates the generation or demand of its resource among the other agents in the grid [34].

To achieve optimal management and scheduling of microgrid local resources in both control schemes, it is imperative of the microgrid operator to forecast the grid energy profile, which include demand and renewable generation forecast. In addition to its ability to forecast, the microgrid must also be equipped to respond to real-time forecasting errors and sudden contingency scenarios, such as an unscheduled islanding. Microgrid controllers must be able to self-stabilize through voltage and frequency control to ensure the microgrid remains operational [35].

Beyond the operational characteristics and concerns of a single microgrid, much research attention is being paid to the operation and interaction of groups of microgrids. Due to the rapidly increasing deployment of microgrids, a primary focus of microgrid research is now turning to the microgrid network. Networked microgrid is a large group of interconnected microgrids which may operate cooperatively among themselves while still remaining distinct from the utility grid as shown in Fig. 2.2. The ability of networked microgrids to operate cooperatively leads to increased energy efficiency, reduced emissions, and lower energy consumer costs [36]. Additionally, transitioning the utility power grid toward a system of microgrid networks provides more efficient, and reliable smart grid [4].

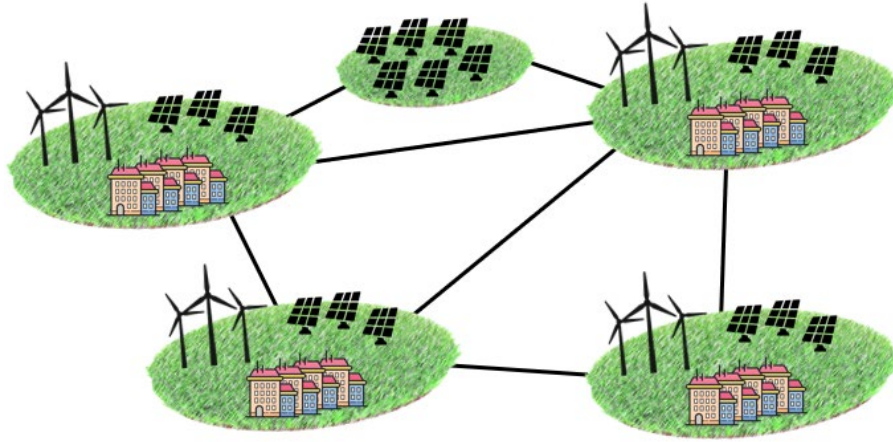


Figure 2.2. Group of interconnected microgrids forming a microgrid network

2.2. Peer-to-Peer Energy Trading Mechanism

Peer-to-Peer (P2P) energy trading represents direct energy trading between peers. For networked microgrids applications, energy is traded among all microgrids (peers) in the network to achieve a desired social welfare (e.g., reliability or economic benefits). P2P markets can also be defined as energy exchange platforms that create a transactive energy market for all peers to bid and offer for transacting energy. P2P energy trading in the microgrid network service towards achieving generation-demand balance while reducing the instances of load shedding and energy curtailment, thereby improving the energy and economic efficiency of each microgrid.

In P2P energy trading markets, participants known as prosumers buy and sell energy from one another in an effort to gain economic and reliability benefits. A prosumer is an emerging category of energy consumer which not only consumes energy but also has the capacity to produce energy through installed distributed generation (most commonly renewable generation) as illustrated in Fig. 2.3. The primary energy source of the prosumer is therefore its own generation, and any energy demand above the generation capacity of the prosumer must be sourced from external generation, such as the power grid. Conversely, if the amount of energy

produced exceeds the prosumer demand, the prosumer can sell the excess energy. In this way it becomes easy to define a microgrid as an energy prosumer, which utilizes its own generation to satisfy loads, while also being able to exchange energy with external entities [37].

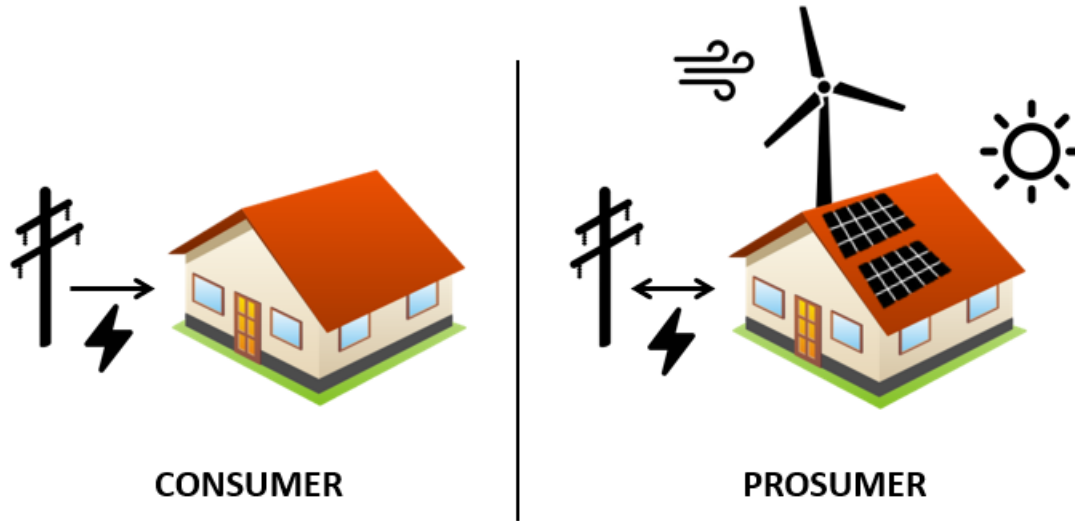


Figure 2.3. Prosumer definition

For reliability purposes, each prosumer aims at achieving demand-generation balance. However, due to fluctuations in both renewable generation and demand, achieving generation-demand balance becomes a challenging task. Therefore, there are three distinct scenarios exist for the prosumer: (1) when generation-demand balance is achieved, (2) when generation is less than demand, and (3) when generation exceeds demand. In the first scenario, the prosumer has satisfied all of its loads and has no remaining energy, therefore no action is required. In the second scenario, the prosumer was not able to satisfy all of its loads and therefore must seek to purchase additional energy or be forced to shed his unsatisfied load. In the third scenario, the prosumer has not only satisfied all of its demand, but it also has excess energy which it may seek to sell at a profit, store in an energy storage system, or curtail. P2P energy trading between prosumers seeks to resolve the energy deficits and surpluses of scenarios (2) and (3).

Among the numerous P2P models developed in the literature, three primary techniques are employed [37]: 1) game theory, 2) auction markets, and 3) constrained optimization.

2.2.1 Game Theory

Game theory has been defined as the mathematical study of the interaction among rational decision makers. The modern development of game theory was heavily influenced by its use among mathematicians and economists as a tool for determining the optimal equilibrium among game participants [38]. In basic terms, game theory analyzes the decision-making process among game participants in competitive situations for which the actions of one participant both effect and depend on the actions of the other participants. Therefore, game theory has become a popular technique for developing P2P trading models where each participant is seeking to optimize their situation. Two general categories of games exist: cooperative games and non-cooperative games [38].

In cooperative games, participants are motivated by both their self-interests and the interests of one, multiple, or all other participants in the game. Participants operate under coalition, making decisions that benefit all coalition members while fairly distributing revenue. Cooperative game P2P trading models involve situations where trading participants sacrifice their individual economic optimization to achieve an equilibrium which satisfies a larger group of trading participants [39]. In cooperative games, the optimal trading price is not always achieved.

In non-cooperative games, all participants seek to optimize their situation based on their self-interests without directly communicating among other members [38]. The end goal of a non-cooperative game is to achieve a Nash equilibrium, which is a stable state wherein no game participant can improve their standing by deviating from their current states. The most popular

non-cooperative game strategy utilized in P2P trading is the Stackelberg game. In the Stackelberg game, a single participant is designated as the leader and makes the first decision, and all other participants have the benefit of optimizing their decision based on the leader's initial decision [40]. If the game is dynamic, all participants then update their decisions strategically to arrive at an equilibrium. In practice, this trading model would involve the leader offering an initial price, and all other participants providing counter-prices. After the initial set of prices is known, participants are able to adjust their pricing strategy to obtain the best possible economic outcome, i.e. selling at the highest price a buyer is willing to purchase, or purchasing at the lowest price a seller is willing to sell.

2.2.2 Auction Markets

Auction refers to the process of buying and selling a commodity by offering bids that establish the price at which an auction participant is willing to buy or sell the commodity being offered [41]. Traditionally, three types of auctions exist, depending on the number and makeup of the participants: forward auction, reverse auction, and double auction [42]. The most common type of layman's auction is the forward auction, where a single seller is offering an item that receives bids from multiple potential buyers and the highest bid succeeds. The reverse auction is the opposite, where a single buyer accepts bids from multiple sellers where the lowest bid succeeds. While both the forward and reverse auction types can be used to develop P2P trading models, the most popular is the double auction.

In a double auction, multiple buyers and multiple sellers participate, where buyers submit their bids and sellers submit their asking prices. A third-party market controller then evaluates all the bids and selects a price which clears the market; that is, where all sellers who asked below

the selected price and all buyers who bid above the selected price succeed in their sale or purchase [43].

Ideally, the selected price is obtained from the intersection of the aggregated supply and demand curves of the available bids, known as the average mechanism [11]. However, in practice P2P energy trading double auctions can utilize different price determination mechanisms which can be designed to comply with a set of system constraints.

2.2.3 Constrained Optimization

Constrained optimization is the process of optimizing an objective function with respect to one or more variables considering limitations of those variables. For the purposes of P2P trading, the objective function is a cost function, which is solved to minimize the decision variables. Several optimization techniques have been used to develop P2P energy trading models, including linear programming (LP), mixed integer linear programming (MILP), and alternating direction method of multipliers (ADMM).

In mathematics, LP is the method of achieving the optimal scenario in a model represented entirely by linear relationships. In other terms, LP optimizes a model defined as a linear equation. LP are capable of being solved very efficiently using the simplex algorithm. In the case of P2P energy trading, LP is utilized to determine the optimal trading decisions which minimize trading costs. This optimization can be done considering a single prosumer, or an entire P2P trading market.

MILP is a special case of integer LP where some variables are constrained as integers while other variables are capable of being non-integers. Unlike LP, because of the mixture of integer and non-integer variables, MILP problems cannot be solved efficiently using the simplex

algorithm. MILP is used in much the same way as LP to develop P2P trading models, with the main difference being the number and mixture of decision variable types being solved.

ADMM is a variant of the augmented Lagrangian method for solving distributed convex optimization problems. The ADMM works by decomposing a constrained optimization problem into distinct unconstrained problems which can then be solved more easily and adding a term to consider the error of decomposing the original problem. In the case of P2P energy trading, ADMM is particularly useful for dual price adjustment optimization.

2.3 Blockchain-Based P2P Trading

One of the fastest growing technological advances of the 21st century is the continued development and application of the blockchain and blockchain technology. A blockchain is a digital database of information containing an immutable ledger of transactions distributed in a decentralized network [9]. In the simplest terms, a blockchain is a growing list of records that cannot be changed that is shared among a group of users.

A fairly recent concept, blockchain was first proposed in 2008 by an individual (or group) using the alias Satoshi Nakamoto and was originally intended as a public transaction ledger for the popular cryptocurrency bitcoin [44]. Since its invention, blockchain technology has been utilized in many applications, including cryptocurrency, banking, supply chain logistics, smart contracts, and energy trading.

Blockchain takes its name from the structure of its data as a chain of discrete information blocks which form a digital ledger. Each block in the chain contains the informational details of a discrete transaction record, with each transaction being recorded on a new block in a time-linear manner [9]. The information contained in each block includes the identification of the parties participating in the transaction, the amount of the commodity being transacted, the price

of that transaction (if applicable), the precise time of the transaction, and often a nonce, which is a random string of numbers. Additional information can be included to suit the particular application. Fig. 2.4 illustrates a simple schematic diagram of the blockchain structure.

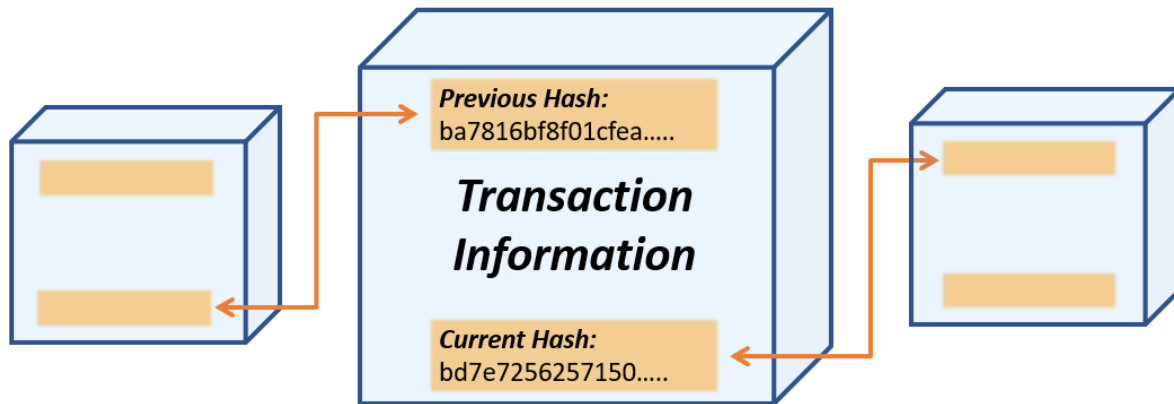


Figure 2.4. General blockchain structure

In addition to the transactional information and the nonce, each block contains an alphanumeric string known as a hash, which is obtained by inputting the information from the block into a hash function (a special mathematical function) which generates a fixed-length hash that is deterministic with regard to the input [46]. The hash of the block is then published at the end of the block. Furthermore, the next block in the chain will add the previous hash to its block data before hashing, which will contribute to the input of the hash function to generate the hash that will be published to the end of the second block. The third block will adopt the hash of the second, and so on. This process of relating each block by their hashes is where the chain concept originates [9].

When a block is added to the blockchain, all information is then updated synchronously to the entire network that shares the blockchain. This ensures that all participants have the same blockchain record. If any of the data in a chained block is modified, even by a single character,

the hash generated with that block will change. Since the tampered block's hash has changed, it no longer matches the hash found in the data of next block and indicates the tampering of the block. In this way, a malicious actor that wishes to tamper with a block would also have to update the information in the next block (and also the previous block). However, altering the information in the second block will cause hash mismatch between the second and third blocks, and so the information of the third block must also be updated. Therefore, altering the information of just one block causes an increasing ripple effect. Additionally, since the blockchain is distributed among all network participants, the malicious actor would have to alter all copies of the blockchain simultaneously. This would require individual access to all decentralized machines with a copy of the blockchain, which is also practically infeasible.

The immutability of blocks, coupled with the decentralized distribution of the blockchain record, make blockchain an attractive solution to cybersecurity concerns for financial transactions in energy trading [9]. Additionally, blockchains employ a consensus method as a requirement for adding blocks to the blockchain, which protects the blockchain from malicious actors that would seek to record false information on the blockchain. These consensus methods require that a form of proof be provided that the block being added to the blockchain is valid. Various consensus methods have been developed, including PoW, PoS, Proof of Authority (PoA), and pBFT [16].

PoW is the original and most widely used algorithm for determining blockchain consensus. By definition, PoW describes a system that requires a not-insignificant amount of computational effort to complete a task, which acts as a deterrent to malicious abuse. For blockchain applications, PoW relies on the addition of the previously mentioned nonce, where blockchain participants called miners guess the nonce value that will produce a hash for the

block below a certain target hash value. Determining a nonce value which will satisfy the given condition is non-trivial, and the target hash is most often selected such that it requires approximately 10 minutes to solve for the nonce. Once the nonce has been solved for, the valid hash is broadcast to the network and the validated block is added to the chain.

PoS was proposed as an alternative to the PoW algorithm, which is both energy and time inefficient due to the high computational requirements. PoS works on the basis of “stake”, wherein a validator for a proposed block is chosen randomly from a group of validators that have a certain level of stake in the blockchain. Individuals who participate in the blockchain and validate blocks more often accrue more stake, which in turn gives them a higher chance of being selected to validate new blocks.

Conceptually, tying the validation of new blocks to stake deters malicious actors from attempting to accrue high levels of stake in order to manipulate the blockchain, since accruing stake also means the individual is increasingly reliant on the accuracy of the chain, therefore making it self-detrimental to manipulate the chain. The concept of stake was originally established based on cryptocurrency but can also be defined according to the specific blockchain application. For instance, when PoS is used for energy trading, stake can be defined as the number of blocks the individual has previously participated in.

PoA is a consensus method typically reserved for permissioned blockchain. In contrast to traditional blockchains which are publicly distributed, permissioned blockchains are only distributed among a group of trusted peers [46]. Among the group of blockchain participants, PoA requires a subset of accounts which have the authority to validate blocks for the blockchain. In this way, PoA is a more centralized consensus method since it relies on a controlled group of privileged blocks. While this method provides a much faster and more automatic method of

validation, it also requires that authorized participants be highly trusted and have very strong individual security, since a malicious actor can target these authorized participants in order to gain validation privileges.

Unlike the other consensus methods, pBFT is not a direct proof method. Instead, pBFT relies on applying the concept of BFT for the purposes of block validation. BFT refers to the ability for a distributed network to reach an assured consensus despite the presence of faulty or malicious nodes that propagate false data. The distinguishing feature of pBFT is the use of two voting rounds to reach consensus. When a block requires validated, the block is compiled by the transacting parties and sent to a participant known as the primary. The primary then broadcasts the block proposal to the network. This commences the first voting round, wherein each participant broadcasts an acknowledge message to the network that it has obtained the block to be validated. This is the first vote, and each network participant is then waiting to receive votes from a pre-determined threshold (usually $2/3$) of the network [47]. When a participant receives the requisite number of votes, it sends a second vote which acknowledges whether it approves the block being validated. Once the second vote has been broadcast for more than $2/3$ of the network the block is validated and the primary appends the block to the chain [47]. In this way, the network is capable of tolerating up to $1/3$ of network not participating or behaving contrary to the other network members. Malicious actors would need to gain control over more than $1/3$ of all network participants to manipulate the blockchain [47].

The blockchain is becoming a popular choice for secure P2P energy trading in decentralized networks due to the immutability of the blockchain record and the consensus methods used to validate energy transactions appended to the chain [9]. For instance, smart contract architectures for decentralized P2P energy trading based on Blockchain is a commonly

used method. The smart contract is defined as a computerized transaction protocol that executes the terms of a contract [37]. By converting contractual conditions into code and embedding them into property that enables self-executing of trusted transactions and agreements between different, anonymous nodes without the need for a central authority. For blockchain applications, smart contracts are scripts stored on the blockchain with a unique hash [37]. A smart contract is triggered by addressing a transaction to it. It then executes independently and automatically in a prescribed manner on every node in the network, according to the data that was included in the triggering transaction.

CHAPTER 3

THE PROPOSED P2P ENERGY TRADING MODELS

3.1 Islanded Operation Mode – Multiagent Model

The entire interconnected islanded microgrid system is modeled as a distributed multi-agent network, where each agent (microgrid) is a node of the network. Multi-agent coalition refers to a way to cooperate agents to complete a task, where none of them can complete it independently [21]. Based on this definition, it was assumed that each microgrid (agent) consists of only renewable distributed generation and power demand. Since all microgrids are connected to each other and disconnected from the power grid, thus, the grid back up is unavailable. Therefore, the task of all microgrid operators in the islanded system is to balance local renewable generation and demand. Hence, achieving zero net load is used to measure the level of satisfaction of all participants in the P2P trading. It should be noted that the net load is defined as demand minus renewable generation. All microgrids in the islanded system share a common interest which is satisfying their net load; hence, they agree to work in a collaborative manner to satisfy their net load. It is also assumed that each microgrid does not have sufficient non-renewable local resources (e.g, dispatchable units, storage, controllable loads). Therefore, each microgrid is extensively incentivized to participate in the P2P trading to balance their net load. This incentive mechanism can be justified based on the fact that reliability benefits are main drivers of microgrids operation in islanded mode [48]. Limited capacity of local resources can be used only as a back-up if the power exchanged in the P2P trading is insufficient to balance their net load. Therefore, it is not required to formulate a scheduling optimization problem since dispatchable units, storage and controllable loads are not primarily used to balance the net load.

In terms of the model architecture, each node represents a microgrid consisting of renewable generators, and power demand, local controller, and trade controller in a layered architecture. The renewable distributed generation and power demand are located in the physical resource layer. On top of that there is the local controller agent (LCA), which is mandated to manage the load and renewable generation data (forecasting hourly net load). In the event of energy deficit or surplus, the local controller forwards the information to the trade agent (TA) which is tasked with buying or selling energy to satisfy the microgrid hourly net load for the day ahead 24-hour time horizon. A graphical illustration of the system model is given in Fig. 3.1. The overall trading model is described in detail in the following sections.

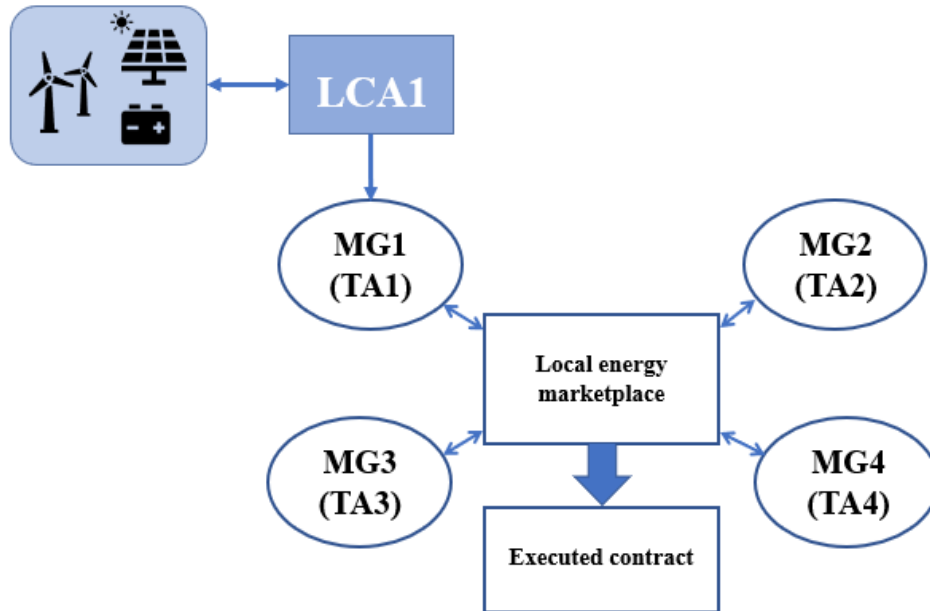


Figure 3.1. System Model (islanded operation)

3.1.1 Seller and Buyer Identification

It is assumed that all microgrids have the capability to forecast their power demand and generation for a particular time slot t , and is beyond the scope of this work. Hence, the energy production and consumption for each time interval t in the time horizon T is generated randomly

by utilizing the Mersenne Twister pseudo-random number generator [49], The Mersenne Twister outputs a statistically uniform distribution between the upper and lower bounds detailed in equations (1), and (2) obtained from [50] with a slight modification. Utilizing the Mersenne Twister pseudo-random number generator was based on the fact that for each time horizon T , there is a maximum value for the renewable generation and electric load, below which the sub-horizon values are permitted to vary in a quasi-random fashion dictated by a Mersenne Twister pseudo-random number generator [51].

$$P_g(t) = rand \left[P_{Rg,max} - \frac{C_{ps}}{T}, P_{Rg,max} + \frac{C_{ps}}{T} \right] \quad (1)$$

$$P_l(t) = rand \left[P_{Rg,max} - \frac{C_{ps}}{T}, P_{Rg,max} + \frac{C_{ps}}{T} \right] \quad (2)$$

The local controller determines the renewable generation-based net load of each microgrid by:

$$P^{net}(t) = P_i(t) - P_g(t) \quad (3)$$

where $P^{net}(t) < 0$ denotes an energy surplus while $P^{net}(t) > 0$ indicates an energy deficit. If $P^{net} = 0$ the microgrid has reached generation-demand balance. For all time intervals where $P^{net} \neq 0$, the local controller notifies the trade controller of the need to buy or sell energy. A microgrid with a negative net load is identified as a seller, whereas a microgrid with a positive net load is identified as a buyer as shown in equations (4) and (5).

$$P^{net}(t) < 0 \Rightarrow P^{net} = P_{sell}^{net} \quad (4)$$

$$P^{net}(t) > 0 \Rightarrow P^{net} = P_{buy}^{net} \quad (5)$$

3.1.2 Price Adjustment and Contract Matching Mechanism

After determining the remaining net load, the local controller forwards the hourly energy deficit or surplus information to the trade controller to interface with the other microgrids participating in the local energy trading marketplace. For each round r in the time interval t

(hourly time interval) in 24-hour day ahead scenario, formulated energy selling contracts are offered at fixed prices by microgrids with surplus power, and energy buyers bid for these contracts with prices offered by microgrids with power deficit. Both sellers and buyers aim to get their contracts matched and executed to satisfy their net load since grid backup is absent. The contract matching process is developed as follows:

1) Sellers start with high fixed energy prices and make progressively lower offers in an effort to match the price of potential buyers after each unsuccessful offering round. Conversely, buyers start with low fixed prices and making progressively higher offers in an effort to match the prices offered in seller contracts.

2) For each round r in the time interval t , an autonomous contract matching round is done in the marketplace considering the following possible scenarios:

- (i) If $P_{sell} \leq P_{buy}$ the contract is automatically executed.
- (ii) If $P_{sell} > P_{buy}$, the buyer moves on to the next available contract.
- (iii) If after the first trading round the offered contract did not receive a match from a potential buyer, the seller must lower its contract selling price using (6) and the buyer must increase the desired purchase price using (7). The contract will automatically execute when the buyer and seller prices converge in a future round.

The contract price adjustment mechanism is developed as follows:

1) For time intervals when $P^{net} < 0$, the microgrid is designated as a seller (P^{net} is identified as P_{sell}^{net}) and the trade controller authors an energy contract containing the amount of surplus power for sale and the price per kW of the power being sold. The seller calculates the desired selling price for each contract offering round as shown in (6).

$$\lambda_{sell}(t, r) = \lambda_{sell}(t, r - 1) - \tau \left(C^{bss} P^{bss} + \alpha C^{cur} (P_{sell}^{net} - P^{bss}) \right) + A_{i,j} C^{tr} P_{sell}^{net} \quad (6)$$

It should be noted that t indicates the hourly time interval of the considered 24-hour time horizon, and r represents the trading round in hour t . Since grid tie is unavailable for the islanded microgrid network, the seller initially attempts to sell with a price higher than the utility price in the first round ($r = 1$), denoted as λ_{sell}^{fixed} , where λ_{sell}^{fixed} is designed such that $\lambda_{sell}^{fixed} > \lambda^{utility}$ (the seller intends to sell at higher than the utility price). Therefore, when $r = 1$, $\lambda_{sell}(t, r - 1)$ is equal to the initial price λ_{sell}^{fixed} . To ensure price fairness and avoid price adjustment manipulation by the seller, a maximum threshold for λ_{sell}^{fixed} is specified by the marketplace and agreed on by all microgrid operators.

If the offered contract did not receive a match in the first round, the seller must lower its selling price according to (6). The price is reduced considering the operation cost of battery storage, curtailment cost, and transmission cost, where the second term in (6) represents the battery operation cost for each charging cycle, and the third term represents the energy curtailment cost. The cost of curtailment is modeled as a loss of revenue where $C^{cur} = \lambda_{sell}^{fixed}$ in the first round and $\lambda_{sell}(r - 1)$ for all sequential rounds. It should be noted that curtailment is applied only when the surplus power (P_{sell}^{net}) is higher than the battery charging limit for each round ($|P_{sell}^{net}| > P^{bss, ch, max}$). Hence, the curtailed amount of power for each round is a percentage of the difference between the surplus power and the power charged in the battery ($\alpha|P_{sell}^{net}| - P^{bss}$). It should be noted that τ is a binary variable with a value of 0 in the first round and 1 in all subsequent rounds for each hour. The fourth term indicates the transmission cost, where A is the distance matrix that represents the distance between any two microgrids in the network, i is the buyer microgrid index and j is the seller microgrid index; hence $A_{i,j}$ is the distance between microgrid i , and microgrid j . The price adjustment process in (6) is developed based on the fact that sellers would tend to charge and curtail surplus power to satisfy its net load

if they did not sell their excess power. Microgrids are motivated to utilize energy trading to avoid these high costs, therefore the seller will go back and adjust its selling price after each round until it gets its contract matched and executed.

2) For time intervals where $P^{net} > 0$, the microgrid is designated as a buyer (P^{net} is identified as P_{buy}^{net}) and the trade controller enters the marketplace to evaluate potential contract purchases.

The buyer enters the marketplace with a desired purchase price calculated using (7).

$$\lambda_{buy}(t, r) = \lambda_{buy}(t, r - 1) + \tau \left(C^g P^g + \beta C^{sh} (P_{buy}^{net} - P^g) \right) \quad (7)$$

Since grid tie is unavailable for the islanded microgrid network, the buyer initially attempts to purchase power with a price lower than the utility price in the first round ($r = 1$), denoted as λ_{buy}^{fixed} , where λ_{buy}^{fixed} is designed such that $\lambda_{buy}^{fixed} < \lambda^{utility}$ (the buyer intends to pay less than the utility price).). Therefore, when $r = 1$, $\lambda_{buy}(t, r - 1)$ is equal to the initial price λ_{buy}^{fixed} . To avoid manipulation of the price adjustment by the buyer, a minimum threshold for λ_{buy}^{fixed} is specified by the marketplace and agreed upon by all microgrid operators. In addition, all cost parameters (e.g, $C^g, C^{sh}, C^{cur}, C^{tr}$) used in price adjustment equations are constant and determined by the local marketplace in which all peers are trading.

If the offered buying contract did not receive a match, the buyer will increase its offered buying price. The price is increased considering the operation cost of dispatchable units and load shedding cost, where the second term in (7) denotes the dispatchable unit operation cost for a committed cycle, and the third term indicates the load shedding cost. It should be noted that load shedding is applied only when the deficit power (P_{buy}^{net}) is larger than the backup dispatchable unit output power limit for each round ($|P_{buy}^{net}| > P^{g,max}$). Hence, the amount of deficit power to

be shed is a percentage of the difference between the deficit power and the power supplied by the dispatchable unit ($\beta(|P_{sell}^{net}| - P^{bss})$).

The price adjustment process in (7) is designed based on the fact that buyers would have to get power from back up dispatchable units, as well as applying load shedding to balance its deficit net load if they did not buy power. Microgrids with energy deficits are motivated to avoid these high costs, and therefore the buyer will go back and adjust (increase) its buying price after each round until it gets its contract matched and executed. The complete contract price adjustment and execution algorithm is shown in Table 3.1.

Table 3.1. Price adjustment and contract execution algorithm – islanded operation

Algorithm 1. Islanded Operation	
1:	when $P^{net} < 0 \Rightarrow P^{net} = P_{sell}^{net}$
2:	when $P^{net} > 0 \Rightarrow P^{net} = P_{buy}^{net}$
3:	$r = 1$
4:	while contract = FALSE
5:	if $r = 1$
6:	$\lambda_{sell}(t, r) = \lambda_{sell}^{fixed} - \tau(C^{bss}P^{bss} + \alpha C^{cur}(P_{sell}^{net} - P^{bss})) + A_{i,j}C^{tr}P_{sell}^{net}$
7:	$\lambda_{buy}(t, r) = \lambda_{buy}^{fixed} + \tau(C^gP^g + \beta C^{sh}(P_{buy}^{net} - P^g))$
8:	else
9:	$\lambda_{sell}(t, r) = \lambda_{sell}(t, r - 1) - \tau(C^{bss}P^{bss} + \alpha C^{cur}(P_{sell}^{net} - P^{bss})) + A_{i,j}C^{tr}P_{sell}^{net}$
10:	$\lambda_{buy}(t, r) = \lambda_{buy}(t, r - 1) + \tau(C^gP^g + \beta C^{sh}(P_{buy}^{net} - P^g))$
11:	if $\lambda_{buy} \geq \lambda_{sell}$
12:	contract = TRUE
13:	else
14:	$r = r + 1$
15:	end

The blockchain in this work is used only as a secure settlement protocol after the contract execution. Even though the autonomous contract execution using blockchain is beyond the scope of this work, it can be done as follows:

Sellers and buyers prepare their contract condition off-chain, and then they compile and deploy their smart contract for possible execution to the local marketplace using an appropriate blockchain architecture that supports smart contract and deterministic consensus protocols. Hyperledger is a private blockchain software [52] that provides a modular architecture that makes it simple to implement smart contracts and deterministic pBFT-bases distributed consensus [53]. For instance, the energy trading model proposed in [54] adopted Hyperledger platform for implementing the proposed model. Furthermore, to avoid trade manipulation, prospective energy buyers do not share their desired purchase prices with energy sellers. Sellers that are aware of desired buying prices can manipulate trade by i) overvaluing their contracts by holding to a higher price knowing that prospective buyers will raise the desired buying price to meet energy demands or ii) under-valuing their contracts in order to undercut competition and execute more contracts. The converse is true for buyers manipulating buying prices. Therefore, prices should not be shared between buyers and sellers during contract's matching process. This can be done by encrypting the data included in the contract before it is broadcast in the marketplace. For instance, the confidential transactions technique discussed in [55] can be adopted where the buyer and seller have contract s that contain price and other confidential information. The technique of confidential transactions is to keep the price amount secret and to grant verifiers the ability to check the validity of amounts [56]. In this case, buyer and seller perform a two-stage encryption process. At first, buyer and seller perform cryptographic hash operation on their contracts to preserve the confidentiality and authenticity of data to each other. Then, the buyer adds a public-key or asymmetric cryptography to further protect data from 3rd party (intruder) intervention/malicious party. In particular, each of the buyers and sellers generate a public-key and a private-key. The buyer then encrypts its (cryptographic hashed)

contracts with the public key of the seller. The seller then decrypts data using its own private key. Once the seller decrypts the other party's data, its smart contract system performs price-matching. Note that this price-matching operation contains a smart algorithm that can work on the cryptographic-hash (price) amounts from buyer and seller and make a decision [55]. Once this contract matching operation is done, it informs the buyers and the sellers about its decision. In this way, buyer and seller are not exposed to the price of each other and hence the overall confidentiality is preserved.

3.2 Grid Connected Operation Mode – Multiagent Model

3.2.1 Formulation of The Optimization Scheduling Problem

In the first stage of the energy trading model for grid-connected operation, each microgrid solves a local energy resource scheduling problem. Since all microgrids in the network are independent entities with unique self-interests, each microgrid solves its own local optimal scheduling problem with an objective to minimize its operation cost. The scheduling problem is modeled as Mixed Integer Linear Programming (MILP) problem. It was assumed that all microgrid are connected to each other and connected to the main utility grid.

$$\min \sum_{t=1}^{24} (C^g P_t^g u_t + SU_t y_t + SD_t z_t + C^{bss} P_t^{bss} + pr_t^{grid} P_t^{grid} + C^{cur} |P_t^{net}|) \quad (8)$$

The first term of the objective function shown in (8) includes the cost of a dispatchable generation unit, followed by the startup and shutdown costs associated with the dispatchable generation unit. The fourth term is the operational cost of the Battery Storage System (BSS). The fifth term is the cost of energy exchanged with the grid (either purchased or sold), and the sixth term is cost the curtailed power (curtailed excess generation or curtailed excess load), where P_t^{net} is the hourly curtailed net load. It should be noted that P_t^{net} is negative for excess generation (curtailed power), and positive for deficit power (curtailed load). The net load is

defined as the hourly demand minus hourly renewable generation. Each microgrid will trade its excess net load P_t^{net} with other microgrids to avoid generation and load curtailment cost, hence; minimize its operation cost. The objective function is subject to the following constraints:

$$P_t^g + P_t^{bss} + P_t^{grid} + P_t^{net} = NL_t \quad (9)$$

$$P_t^{g,min} u_t \leq P_t^g \leq P_t^{g,max} \quad (10)$$

$$y_t - z_t = u_t - u_{t-1} \quad (11)$$

$$y_t + z_t \leq 1 \quad (12)$$

$$P_t^{bss} \leq P^{bss,dch,max} d_t - P_t^{bss,ch,min} c_t \quad (13)$$

$$P_t^{bss} \geq P^{bss,dch,min} d_t - P_t^{bss,ch,max} c_t \quad (14)$$

$$d_t + c_t \leq 1 \quad (15)$$

$$SOC_t = SOC_t - P_t^{bss} \quad (16)$$

$$E^{min} \leq SOC_t \leq E^{max} \quad (17)$$

$$|P_t^{grid}| \leq P^{tie,max} \quad (18)$$

The power balance equation (9) represents the power balance equation of each microgrid. Equation (10) ensures that the dispatchable generator operates within its operational limits. The constraint (11) ensures that a startup or shutdown only occurs when there is a change in the operating state of the generator from ON to OFF and vice versa. Equation (12) ensures that the generation unit cannot start-up and shutdown in the same hourly time interval. Equations (13) and (14) show the charge and discharge constraints of the BSS, while equation (15) ensures the BSS is not simultaneously charging and discharging in the same hourly time interval. Equation (16) shows that the SOC of the BSS is affected by the amount of energy charged or discharged from the BSS at hour t . The SOC of the battery storage system is limited by (17). The constraint

in (18) ensures that the power exchanged with the utility grid is limited by the maximum capacity of the tie line.

3.2.2 Energy Trading and Price Adjustment Model – Model I

After the scheduling problem has been solved, each microgrid has determined its remaining net load, modeled as P^{net} , which represents the surplus or deficit power for each hourly time interval t . Each microgrid intends to satisfy P^{net} by trading in the marketplace to avoid generation curtailment cost. If $P^{net} < 0$ (surplus power), the microgrid will sell the surplus power. Whereas, if $P^{net} > 0$ (deficit power), the microgrid will purchase power from the market to satisfy its deficit load and avoid the high cost of load curtailment. Hence, for each hourly time interval, microgrids with excess power will be identified as sellers, and microgrids with deficit power will be identified as buyers. Fig. 3.2 shows a flowchart of the overall trading model.

It is worth noting that the deficit and excess power cannot be traded with the utility grid because the maximum power that can be traded with the utility grid is limited by the tie line maximum capacity limit.

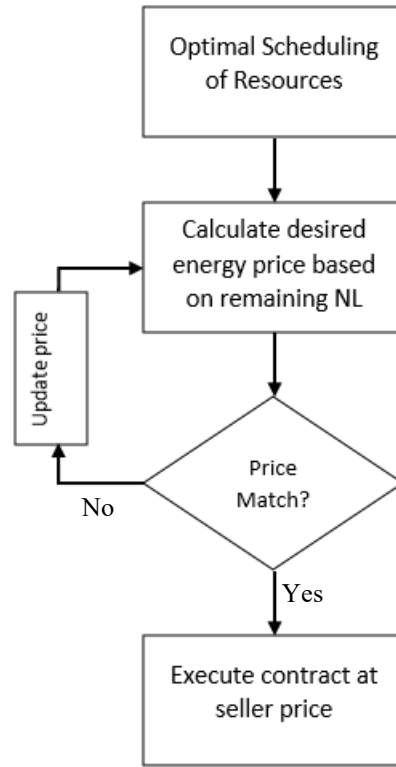


Figure 3.2. Flowchart of the proposed trading model (grid connected operation)

Similar to the system described for islanded operation, the microgrid network is modelled as a multi-agent system, where each microgrid in the network represents an agent. Each agent (microgrid) includes one *local coordination agent* (LCA), and one *trading agent* (TA). The LCA performs autonomous energy management by solving a local scheduling problem to determine optimal control actions of the local energy resources and the net load to be traded. The LCA forwards the net load information to the TA, which is tasked with resolving each microgrid net load by facilitating energy trading among marketplace participants. In contrast to the system model for islanded operation, in grid-connected operation the trading agent is also capable of exchanging energy with the utility grid. A system model for the proposed system is shown in Fig. 3.3.

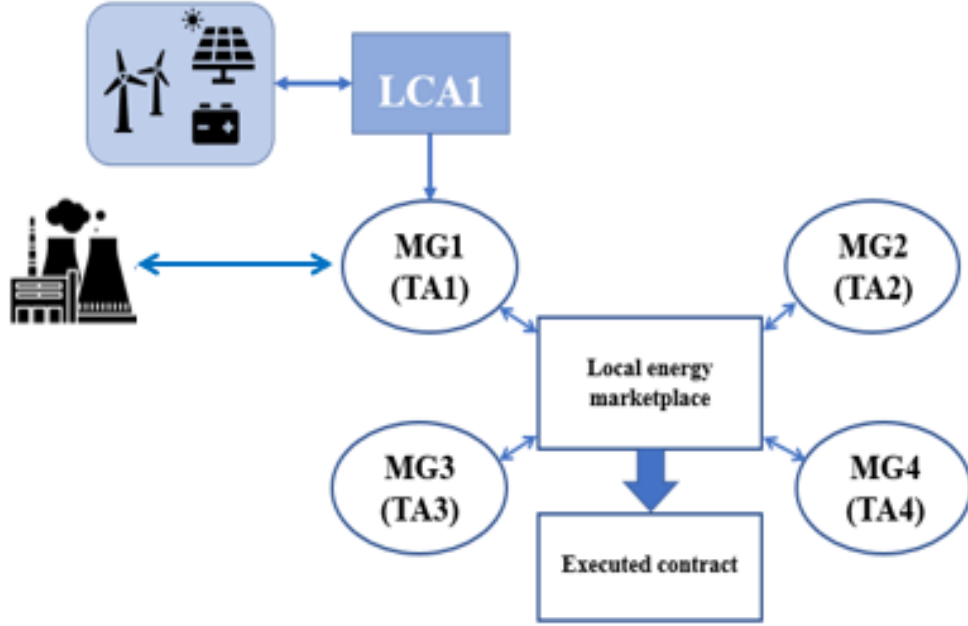


Figure 3.3. System model (grid-connected operation)

Before offering their energy contracts in the marketplace, microgrids which will be selling energy calculate their initial offering price as a function of their excess power according to the developed formula shown in (19). Similarly, buyers calculate their initial offered price as a function of their deficit power demands according to equation (20).

$$\lambda_{mt}^{sell}(t, r) = \left(1 + \frac{\sqrt{|P_{mt}^{sell}|}}{3} \right) \lambda_t^{grid} \quad (19)$$

$$\lambda_{nt}^{buy}(t, r) = \left(1 - \frac{\sqrt{|P_{nt}^{buy}|}}{3} \right) \lambda_t^{grid} \quad (20)$$

Equations (19) and (20) are designed in such a way that microgrids with a lower magnitude of P^{net} are more willing to exchange energy at a price closer to the utility prices, while microgrids with higher magnitudes of P^{net} are more motivated to maximize their economic benefits (sellers maximizing profits, and buyers minimizing energy purchase cost).

Sellers will then publish their initial contracts to the marketplace detailing their power available for purchase and the offered price per unit of power. Seller contracts on the marketplace are sorted, with the lowest contract price considered first, followed by the second lowest contract price, and continuing in that fashion. Trading rounds for hour t begin with buyers attempting to purchase power by matching their prices to the initial offering prices of the available contracts starting at seller index $m = 0$. The contract matching process can be summarized as follows:

- 1) If $\lambda_n^{buy} \geq \lambda_m^{sell}$ the contract is a match for the buyer-seller pair (n, m) and the transaction will be executed at the price λ_m^{sell} . If $P_{mt}^{sell} > P_{nt}^{buy}$ then the amount of exchanged power will equal P_{nt}^{buy} , resulting in buyer n fully satisfying its load and dropping from the market for hour t , while the remaining balance of P_{mt}^{sell} will remain available for purchase in the marketplace. Whereas, If $P_{mt}^{sell} < P_{nt}^{buy}$ then the amount of exchanged power will equal P_{mt}^{sell} , resulting in seller m fully depleting its excess power and dropping from the market for hour t , while the buyer n with remaining demand equal to $P_{nt}^{buy} - P_{mt}^{sell}$ will continue to purchase available power in the marketplace to satisfy its load.
- 2) If $\lambda_n^{buy} < \lambda_m^{sell}$, the contract is not a match, and buyer n moves to the next available contract.

If after all matching contracts are executed in round r there still exists sellers with surplus power and buyers with deficit power, then the remaining sellers and buyers will update their prices according to equations (21) and (22), respectively, and begin the next trading round. Equations (21) and (22) are designed to motivate market participants to trade by incrementally reducing the price of sellers and increasing the price of buyers after each trading round until their prices converge. The rationale behind (21) and (22) is that the seller price decreases with the

increase in the number of matching rounds, while the buyer price increases with the increase of the number of matching rounds.

$$\lambda_{mt}^{sell}(r) = \lambda_{mt}^{sell}(r-1) - \frac{r-1}{r+r!} \lambda_{mt}^{sell}(r-1) \quad (21)$$

$$\lambda_{nt}^{buy}(r) = \lambda_{nt}^{buy}(r-1) + \frac{r-1}{r+r!} \lambda_{nt}^{buy}(r-1) \quad (22)$$

The remaining seller contracts are resorted by price low to high, and the buyers then attempt to match their prices with available contracts in the same manner for subsequent rounds until either $\sum P_{mt}^{net} = 0$ or $\sum P_{nt}^{net} = 0$. If there is no available energy left in the marketplace, the remaining microgrids with energy deficits will curtail their remaining loads. Similarly, if there are no willing buyers left in the marketplace, the remaining microgrids with surplus power will curtail their excess generation. Table 3.2 depicts the contract offering and matching algorithm.

Table 3.2. Price adjustment and contract execution algorithm – grid-connected operation model I

Algorithm 2. Grid-connected operation – model I	
1:	when $P^{net} < 0 \Rightarrow P^{net} = P_{mt}^{sell}$
2:	when $P^{net} > 0 \Rightarrow P^{net} = P_{nt}^{buy}$
3:	$r = 1$
4:	while contract = FALSE
5:	if $r = 1$
6:	$\lambda_{mt}^{sell}(r) = \left(1 + \frac{\sqrt{ P_{mt}^{sell} }}{3}\right) \lambda_t^{grid}$
7:	$\lambda_{nt}^{buy}(r) = \left(1 - \frac{\sqrt{ P_{nt}^{buy} }}{3}\right) \lambda_t^{grid}$
8:	else
9:	$\lambda_{mt}^{sell}(r) = \lambda_{mt}^{sell}(r-1) - \frac{r-1}{r+r!} \lambda_{mt}^{sell}(r-1)$
10:	$\lambda_{nt}^{buy}(r) = \lambda_{nt}^{buy}(r-1) + \frac{r-1}{r+r!} \lambda_{nt}^{buy}(r-1)$
11:	if $\lambda_{nt}^{buy} \geq \lambda_{mt}^{sell}$
12:	contract = TRUE
13:	else
14:	$r = r + 1$
15:	end

3.2.3 Energy Trading and Price Adjustment Model – Model II

The calculation of initial offering prices for buyers and sellers is conducted in the same manner as above with equations (19) and (20). Similarly, the contract matching process proceeds as described above, with one of two possible outcomes:

- 1) If $\lambda_{nt}^{buy} \geq \lambda_{mt}^{sell}$ the contract is a match for the buyer-seller pair (n, m) and the transaction will be executed at the price λ_m^{sell} .
- 2) If $\lambda_{nt}^{buy} < \lambda_{mt}^{sell}$, the contract is not a match, and buyer n moves to the next available contract.

Diverging from the previous model: if after all matching contracts are executed for round $r = 1$ there still exist sellers with surplus power and buyers with deficit power, then all microgrids still participating will update their prices according to equation (23) for buyers and equation (24) for sellers.

$$\lambda_{nt}^{buy}(r) = \lambda_{nt}^{buy}(r-1) \left(1 + \alpha(k P_n^{buy}) \right) + \tau \left((C^{pk} P^{pk}) + (\beta C^{sh} (P_n^{net} - P^{pk})) \right) \quad (23)$$

$$\lambda_{mt}^{sell}(r) = \lambda_{mt}^{sell}(r-1) - \tau (C^{cur} P^{cur} + C^{bss} P^{bss}) \quad (24)$$

Buyers calculate their price adjustment considering transmission losses in the second term of (23), where the transmission cost is calculated as a fixed percentage of the traded energy. The third term in (23) considers the cost of peak plant generation and load shedding, while the fourth term includes the cost of load shedding. The binary variable τ has a value of 0 when $r = 1$ and a value of 1 otherwise. Likewise, the binary variable α has the opposite conditions, having a value of 1 when $r = 1$ and 0 otherwise. The constant β limits the amount of load shedding to 1 percent of the remaining load. Sellers calculate their price adjustment considering energy curtailment in the second term and battery storage costs in the third term. The maximum energy available from peak generation is defined as (25), while the amount of energy curtailed is defined

in (26), and the amount of energy charged to the battery is limited by (27). Table 3.3 depicts the contract offering and matching algorithm for grid-connected operation model II.

$$P^{pk} = 0.1 \times P_n^{buy} \quad (25)$$

$$P^{cur} = 0.05 \times P_m^{sell} \quad (26)$$

$$P^{bss} \leq P^{bss,ch,max} \quad (27)$$

Table 3.3. Price adjustment and contract execution algorithm – grid-connected operation model II

Algorithm 3. Grid-connected operation – model II	
1:	when $P^{net} < 0 \Rightarrow P^{net} = P_{sell}^{net}$
2:	when $P^{net} > 0 \Rightarrow P^{net} = P_{buy}^{net}$
3:	$r = 1$
4:	while contract = FALSE
5:	if $r = 1$
6:	$\lambda_{mt}^{sell}(r) = \lambda_{mt,fixed}^{sell} - \tau(C^{cur}P^{cur} + C^{bss}P^{bss})$
7:	$\lambda_{nt}^{buy}(r) = \lambda_{nt,fixed}^{buy} \left(1 + \alpha(k P_n^{buy})\right) + \tau\left((C^{pk}P^{pk}) + (\beta C^{sh}(P_n^{net} - P^{pk}))\right)$
8:	else
9:	$\lambda_{mt}^{sell}(r) = \lambda_{mt}^{sell}(r - 1) - \tau(C^{cur}P^{cur} + C^{bss}P^{bss})$
10:	$\lambda_{nt}^{buy}(r) = \lambda_{nt}^{buy}(r - 1) \left(1 + \alpha(k P_n^{buy})\right) + \tau\left((C^{pk}P^{pk}) + (\beta C^{sh}(P_n^{net} - P^{pk}))\right)$
11:	if $\lambda_{nt}^{buy} \geq \lambda_{mt}^{sell}$
12:	contract = TRUE
13:	else
14:	$r = r + 1$
15:	end

After adjusting prices, buyers and sellers again attempt to match prices to execute contracts. This process continues until there is no more surplus or deficit energy available in the marketplace. If after all buyer and seller contract matches there still exists an energy deficit for the final buyer, the final seller will offer an additional amount of energy above the initial contract offer by utilizing a spinning reserve. Spinning reserve is the extra generating capacity available in a dispatched generator which can be utilized to compensate for power shortages. It is assumed that the seller is equipped with a spinning reserve margin from dispatchable generation which

can supply up to an additional 20% of the available energy. This additional energy is offered at a higher price according to the cost of reserve energy (C_{res}). If the second selling price is less than the penalty cost of load shedding for the buyer and the buyer still needs to purchase energy to satisfy its deficit, then a subsequent contract will be executed which transfers to the buyer an amount of reserve generation up to 20% of the final seller's original excess power balance. The contract will be executed with an agreed upon price for the cost of the reserve energy. The total cost of the reserve contract is defined by (28). The amount of reserve energy exchanged in the contract is defined by (29) where z is defined by (30) as a percentage of the seller energy and limited to be less than 0.2.

$$C_{contract} = zP_m^{sell}C^{res} \quad (28)$$

$$P^{res} = zP_m^{sell}, \quad z \leq 0.2 \quad (29)$$

$$z = \frac{p_{buy_psell}}{p_{sell}} \quad (30)$$

3.3 Two-Phase Blockchain Consensus Protocol

To enable a trusted settlement of electricity trading transactions, a smart blockchain-based contracts protocol for transaction settlement is developed. The proposed blockchain method uses a traditional distributed ledger consisting of blocks of data that are connected in a single chain. These blocks of data contain the details of the finalized contract from the trading marketplace, including the network address of the buyer and seller, the amount of energy being trading, the price per kilowatt of the contract, the timestamp when the contract was executed, the hash from the previous block, and a new hash generated using the SHA-256 hashing algorithm. Because this ledger chain is a distributed ledger, each node of the network maintains a copy of the ledger.

Before a block is appended to the ledger chain, it must be validated using a consensus method. A two-phase consensus process method is proposed. In the first phase, a pBFT is adopted. pBFT has been proposed in recent years as a viable alternative to popular consensus methods such as PoW and PoS. Byzantine Fault Tolerance (BFT) refers to the ability for a distributed network to reach an assured consensus despite the presence of faulty or malicious nodes that propagate false data. The consensus process developed in this work is shown in Fig. 3.4.

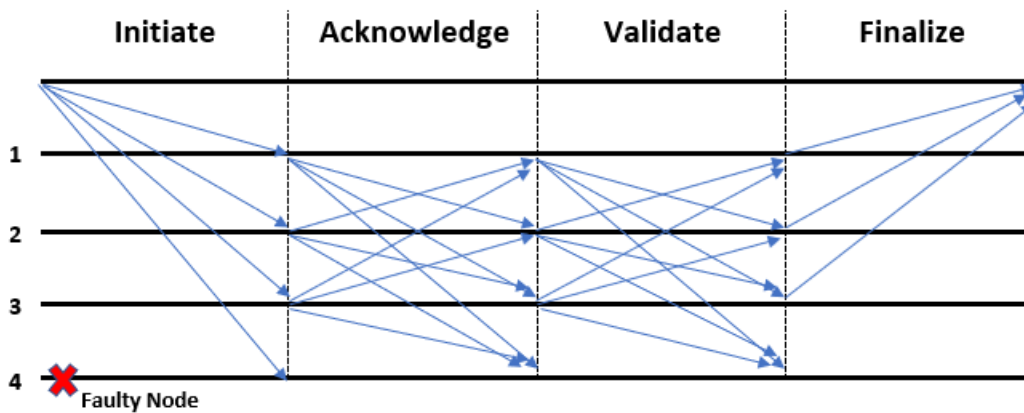


Figure 3.4. pBFT two round voting process with faulty node tolerance

The pBFT is an optimized application of traditional BFT method, which ensures consensus for any network of size $3f + 1$ when there exists $2f + 1$ validating responses (where f denotes the maximum number of faulty nodes).

The pBFT works by a voting consensus where each node has an equally weighted vote value. For each block validation process the following steps are implemented:

- 1) *Initiate*: a random node (microgrid) is selected to be the primary node. The primary node broadcasts the proposed block including the contract data to each of the secondary nodes in the network.

- 2) *Acknowledge*: Each of the secondary nodes broadcasts a vote to acknowledge their receipt of the proposed block to each node.
- 3) *Validate*: After receiving $2f + 1$ approval messages, a node will broadcast a validation message if the data in the proposed block is valid.
- 4) *Finalize*: When $2f + 1$ validation messages are received, the block has been validated and is moved to the second phase of the consensus process.

It should be noted that according to traditional pBFT implementation, the network is secure for any network of size $3f + 1$ where f is the maximum number of faulty nodes. Therefore, in a system where greater than $1/3$ of nodes are faulty (corrupted or non-functioning), the pBFT no longer ensures a secure consensus. In the proposed model, the voting criteria is modified from the traditional $1/3$ fault protection to be $2f + 1$, because this criteria is greater than $2/3$ of the network size. While a $2/3$ criteria is sufficiently safe, the margin of error is a motivating factor for introducing the modified PoS as a second phase of consensus.

To ensure a high level of security, a simultaneous second phase of consensus is conducted using a modified version of PoS consensus. For this consensus method, each microgrid is assigned a semi-random value generated using a weighting factor. This weighting factor corresponds to the recent history of participation in the energy trading marketplace, where microgrids with higher levels of participation are assigned higher weighting factors. After stakes values have been generated, the microgrid with the highest stake value during each consensus round is chosen as a special independent validator node. This validator node constructs a block using the same contract data as was broadcast in the pBFT consensus round and compares its block to the one validated using the pBFT consensus. If these two blocks match, the validator broadcasts a final confirmation that the block is valid, and it is appended to the public chain. If

the two blocks do not match, it indicates a fault that has manipulated the formed block, and an notification will be propagated to report a data manipulation incident. The overall two-phase consensus algorithm is illustrated in Table 3.3, where each microgrid is described as a prosumer.

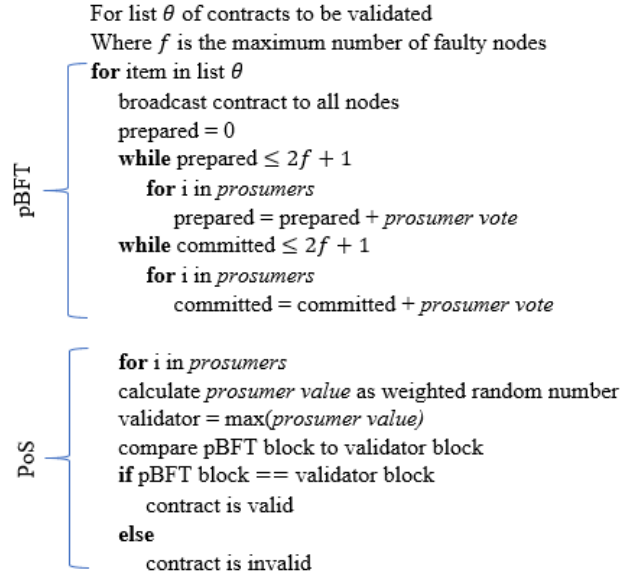


Table 3.4. Two-phase blockchain consensus protocol

In comparison with other common blockchain consensus methods, pBFT shows several advantages. Firstly, pBFT has no fixed time requirements before consensus can be reached. PoW and the traditional PoS both have fixed time interval requirements before a proposed block can be validated. Additionally, pBFT does not require additional resources specific to the blockchain creation, instead utilizing existing network technology and processing capabilities to perform the block validation function. PoW requires expensive, special purpose computing hardware to perform block validation tasks which consume a significant amount of energy and financial resources. Furthermore, traditional PoS requires participants to be willing to use expendable financial resources in order to wager for validation rights.

It is important to mention, however, that pBFT-based consensus methods have scalability issues regarding its use for a massive number of nodes. However, methods including partitioning

the network into smaller groups called federates have been shown to result in improved scaling up to 1000 nodes [4].

The complete two-stage energy trading model which summarizes the trading process for all proposed models, including price negotiation mechanism and blockchain-based contract settlement, is detailed in the flowchart shown in Fig. 3.5.

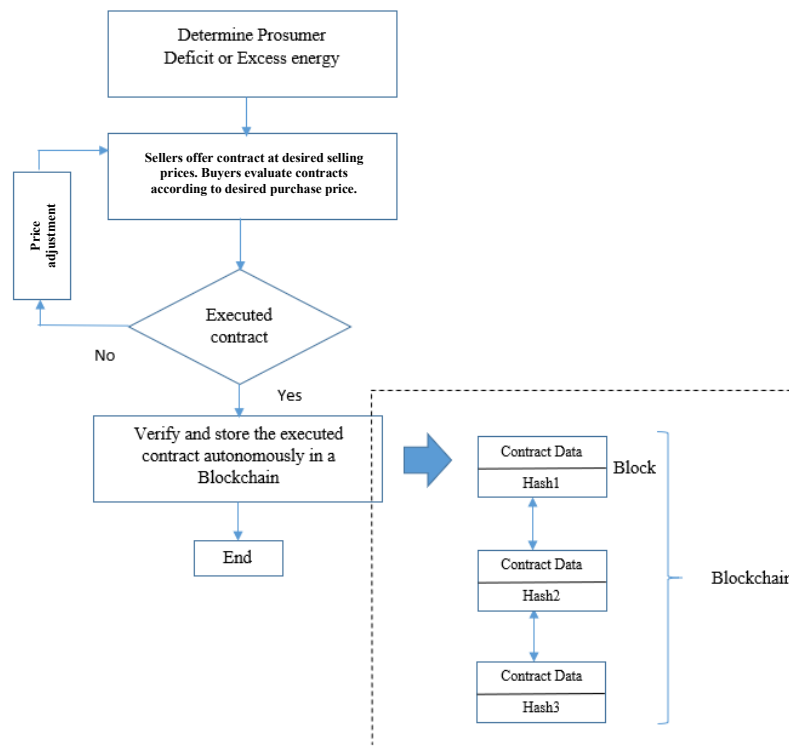


Fig. 3.5. Flowchart of the proposed model

CHAPTER 4

SIMULATION RESULTS

4.1 Islanded Operation Mode – Multiagent Model

The proposed model was simulated using Python 3.6 in Microsoft Visual Studio Professional 2017 on a quad-core 2 GHz CPU equipped with 16 GB RAM. For adjusting the contract price, the charge cost of the battery storage is considered to be \$0.03/kW [57] with a charging limit per round of 2 kW. The adopted \$0.25/kWh operation cost of dispatchable unit is obtained from [58] with a slight modification, considering a ramp rate of 5 kW per round. The load shedding cost adopted in this study is \$1.0/kW [59]. The maximum curtailment ratio for each round is taken as 1% of the hourly surplus net load value. The load shedding ratio for each round is taken as 4% of the deficit hourly net load value. Transmission cost is considered to be 2.8×10^{-6} \$/(kWh. km) [60]. The simulation is carried out with $T = 24$ hours, $t = 1$ hour and scenarios of 5, 10, 15, and 20 node microgrid networks. Unless otherwise noted, trading results for each scenario are similar and congruent, and samplings of results from selected scenarios are reported for brevity.

Using equations (6) and (7), microgrids participating in energy trading successfully adjusted their bid prices, executing a total of 216 contracts for a total of 419.63 kW of traded power during for the scenario of a 10 node microgrid network, whereas 456 successful contracts were executed with a total traded power of 937.17 kW for the scenario of a 20 node microgrid network. A graphical representation of each executed contracts for the case of 10 microgrids is depicted in Fig. 4.1, where the blue line represents the hourly total amount of traded power. The vertical bars show the accumulation of contracts in each hour, where each colored section of a bar represents a separate contract and the size of the section representing the amount of energy

traded in the contract. Table 4.1 details the data obtained from simulation for the scenario of 20 microgrids, showing the amount of power traded in each formed contract over the 24-hour period. A similar table of data with the results for the scenario of 10 microgrids was used to generate fig. 4, and conversely a similar figure could be generated for the scenario of 20 microgrids using the data from Table 4.1.

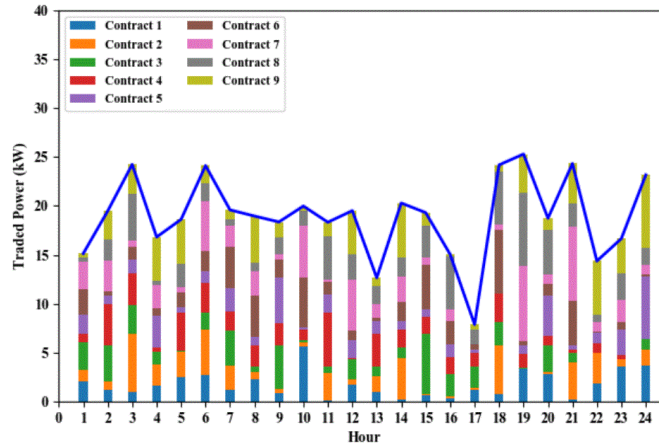


Figure 4.1 number of executed contracts and amount of traded power in the case of 10 interconnected microgrid system

Fig. 4.2 shows an example of the progressive price adjustments of a buyer and seller negotiating an energy contract. In this sample, the desired buyer (microgrid 2) and seller (microgrid 5) adjust their prices according to (6) and (7) over successive contract matching rounds for one contract. It can be clearly seen that the seller decreases their asking price after each round (blue line), while the buyer increases their offered purchase price after each round (red line), with a contract match occurring at a price of \$0.198/kW. According to the agreed upon market conditions, the seller offered their initial selling price at a value of \$0.25/kW, and buyer sets their initial price to \$0.15/kW. With the price of the buyer lower than the price of the seller, there is no match in the first round. In the second round the seller decreases their price to \$0.2329/kW according to (6), and the buyer increases their price to \$0.175/kW according to (7).

However, there was still no contract price match. Similarly in the third round the seller decreases their price to \$0.215/kW and buyer increases their price to \$0.2/kW with no contract match. Finally, in the fourth round the seller reduces their price to \$0.198/kW, and the buyer increases their price to \$0.225/kW. Hence, a contract is executed at the offered seller price of \$0.198/kW (for an exchange of 1.69 kW) since the seller price is now less than the buyer price.

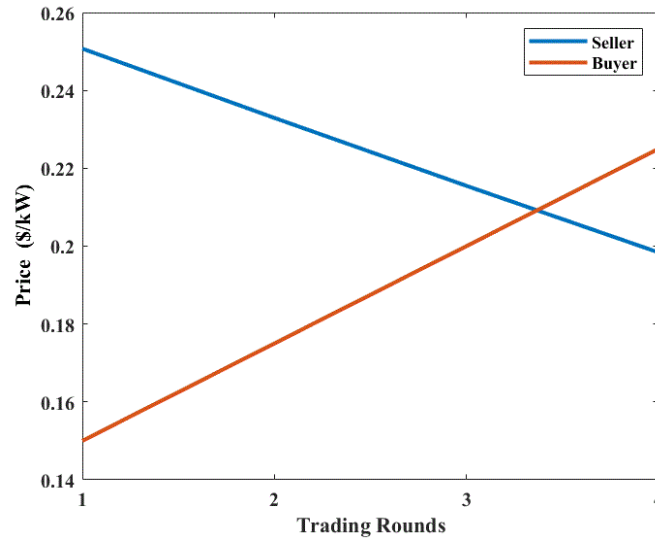


Figure 4.2. A successful price adjustment process for a selected block

Table 4.1. Amount of power traded (kW) in each executed contract for the 24-hour time horizon

Contract no Hour	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	3.67	0.83	1.87	1.08	0.41	0.24	2.9	0.02	2.28	0.32	7.59	2.55	1.84	1.3	0.1	0.36	3.27	5.64	3.4
2	3.2	0.15	0.77	1.93	0.19	0.5	0.99	2.78	1.96	0.88	2.56	1.09	0.49	4.2	1.97	0.22	2.26	0.31	1.92
3	1.67	3.13	3.07	1.3	2.8	0.17	0.43	0.63	0.74	0.73	0.41	4.93	0.26	0.73	2.76	1.96	0.59	2.84	3.03
4	0.22	2.81	1.05	0.56	1.79	0.18	3.38	1.53	0.62	5.52	1.92	2.35	3.56	0.37	0.3	1.19	2.42	1.77	8.54
5	1.77	3.67	2.12	0.47	1.9	3.18	0.16	0.17	0.18	2.92	1.83	1.36	1.37	1.98	4.06	5.36	7.24	9.57	4.97
6	2.44	1.13	2.6	0.32	0.09	1.09	2.7	0.14	1.47	5.5	0.08	0.2	1.21	0.33	0.15	1.98	1.18	5.28	3.56
7	2.05	0.46	1.23	0.44	0.72	0.24	1.73	1.73	1.02	1.01	2.41	1.02	0.06	1.68	0.9	1.03	2.06	4.21	1.83
8	0.74	0.85	0.52	2.69	0.32	1.68	4.05	0.85	3.09	1.69	2.14	0.56	3.23	1.03	0.46	0.27	1.32	4.74	4.05
9	0.82	2.98	0.93	2.34	0.88	0.74	0.16	2.9	0.14	0.63	2.93	1.67	0.32	0.06	2.01	0.88	0.82	0.15	7.04
10	4.06	0.6	2.02	3.3	1.61	1.45	2.43	1	0.31	1.99	1.77	2.53	3.7	2.7	1.47	0.4	0.9	5.15	3.45
11	3.29	0.05	2.32	0.54	6.15	1.84	0.13	1.93	0.58	3.07	0.29	6.07	1.01	1.73	4.07	0.37	1.97	3.8	5.83
12	1.18	2.36	0.07	2.6	3.49	0.32	0.63	2.19	1.62	1.68	0.43	0.72	5.85	0.85	4.32	6	3.5	0.81	3.97
13	1.89	2.46	1.89	0.07	1.91	1.27	3.48	3.82	1.05	0.79	1.85	0.09	1	1.82	0.96	5.87	3.24	4.51	2.03
14	2.96	2.32	4.22	0.82	0.44	2.14	2.57	1.01	1.01	1.01	0.16	0.36	0.03	0.86	1.71	2.54	0.97	4.46	2.66
15	0.79	1.01	3.24	2.96	3.05	0.46	1.1	1.41	0.02	0.03	0.46	1.67	3.08	1.54	4.83	1.23	0.71	4.38	6.46
16	0.67	2.18	2.09	1.08	1.29	0.51	0.54	3.61	0.1	1.88	0.38	0.04	0.63	2.09	0.24	8.49	4.89	6.5	2.24
17	0.91	2	0.21	0.43	1.55	1.68	3.1	0.42	3.54	2.2	1.44	6.29	0.7	6.5	5.39	6.47	1.77	3.09	3.22
18	2.85	2.53	0.01	4.97	0.62	1.22	2.62	3.1	3.04	1.31	2.75	0.49	1.93	0.27	1.84	0.5	6.19	1.91	1.25
19	2.25	2.39	0.45	0.49	0.01	0.07	0.01	6.68	0.9	0.19	4.22	0.83	0.79	0.47	4.04	2.17	0.6	0.37	3.61
20	2.32	3.67	0.87	3.12	1.81	1.14	0.81	5.26	0.13	0.12	1.34	1.07	3.69	0.63	4.46	0.26	1.08	4.49	0.5
21	7.13	0.13	0.94	3.31	0.35	2.49	0.04	3.3	2.03	0.17	1.85	0.66	5.01	2.22	0.07	4.73	2.16	0.07	1.43
22	4.03	1.39	3.87	0.43	2.65	1.68	2.72	4.92	1.94	6.22	1.48	0.59	3.48	1.34	0.3	7.14	2.69	1.78	4.66
23	0.37	3.92	1.1	3.06	0.14	2.52	0.96	1.16	6.06	1.74	7.77	6.01	7.01	0.47	4.18	0.67	0.13	0.72	3.8
24	5.34	0.17	0.13	4.09	1.08	4.19	0.13	2.23	2.75	1.35	2	0.07	1.35	0.48	0.45	7.61	2.8	3.44	4.06

Since each instance of offering a price and executing a contract is a discrete computational task, the average computation time required to complete discrete tasks required to execute a final contract are reported for a varying number of microgrids in the network, as shown in Fig. 4.3. The average computation time to execute all contracts in the case of 20 microgrids is found to be less than one second (around 15.25 ms), which demonstrates the time efficiency of the proposed trading model. It can also be noted that the average contract matching time increases with the increase in the number of microgrids in the network in a nearly linear relationship. This is because with an increase in the number of microgrids in the network, the amount of traded contracts increases accordingly. The linearity of the increase in contract negotiation time shows that the proposed trading model is easily scalable for differing numbers of microgrids. This is also depicted in Fig. 4.4, which shows a similarly linear relationship between the number of formed contracts and the number of interconnected microgrids.

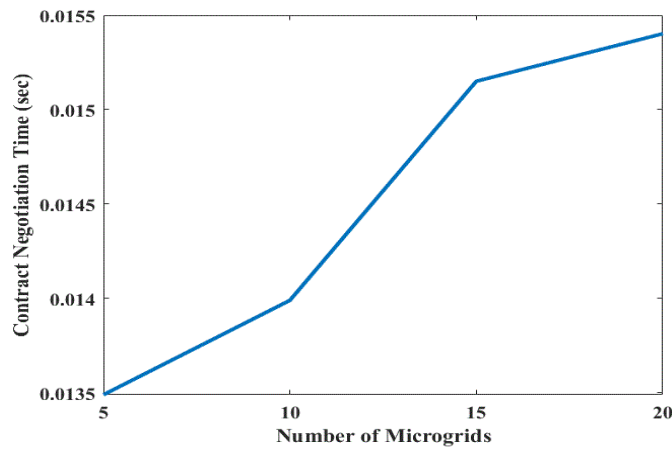


Figure 4.3. Variation in the computation time with respect to the change of the number of microgrids in the network

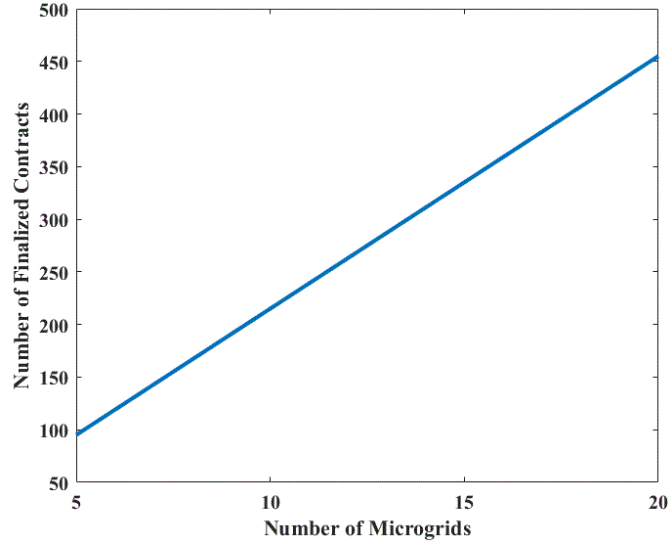


Figure 4.4. Variation in the number of executed contracts with respect to the change of the number of microgrids in the network

It was found that the model has successfully incentivized efficient energy trading among islanded networked microgrids, primarily to satisfy outstanding net loads, and secondarily to obtain the economic benefit of trading at a negotiated price. Each microgrid with power deficit successfully purchased power to meet demands while microgrids with power surplus sold off their excess power hence, demand-generation balance for the islanded interconnected system has been achieved and the costs associated with load shedding and power curtailment have been avoided.

After all contract negotiations have been finalized, the details of each contract are recorded as blocks, verified using a novel two-phase consensus mechanism, and placed on a contract records blockchain. Each validated block in the chain contains (i) buyer and seller identification IDs, (ii) the amount of power being traded, (iii) a transaction price expressed in \$/kW, (iv) the timestamp of the execution of the transaction, and (v) an alpha-numeric string called a hash, which is taken from the previous block on the blockchain. A sample of data included in two sequentially generated blocks is shown in Fig. 4.5.

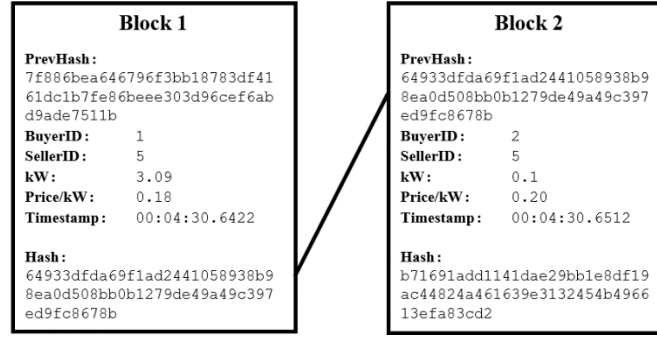


Figure 4.5. A sample of generated blocks containing contract data

In order to investigate the impact of the number of microgrids in the network on the validation time required for the proposed consensus method, the validation time is measured as the time elapsed between transaction submission and block confirmation. The average validation time for the validation method was calculated for an increasing number of microgrids, as shown in Fig. 4.6. It can be observed that the validation time increases accordingly (approximately a linear increase rate) with the increase in the number of networked microgrids. This is due to the fact that with an increase in the number of microgrids in the network, there is a corresponding increase in the number of executed contracts (see Fig. 4.4), which results in a longer validation period. The validation time required to validate all created blocks in a 20 microgrids network is found to be around 1.9 seconds as shown in Fig 4.6.

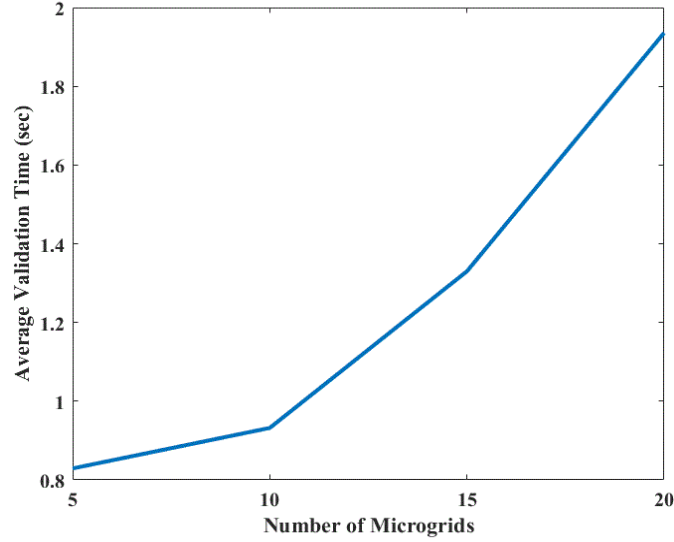


Figure 4.6. The change in average validation time with respect to the change of the number of microgrids in the network

To ensure the effectiveness of the proposed model, the obtained results were compared with the results of recent work proposed in [21] and [61]. Table 4.2 depicts the full comparison, which demonstrates the time efficiency of the proposed energy trading model (less negotiation time for the same number of nodes, and improvement in the success rate of the transaction, where all deficit and surplus power is satisfied).

It was found that the proposed method is time efficient compared to more traditional methods that apply a direct price negotiation between peers. In traditional direct price negotiation methods, both negotiators are fully dedicated to take advantage of the offered contracts by the other peer and bring the other peer closer to their offered price. This increases the contract determination computation time and might lead to an unsuccessful negotiation process, which can cause a reliability problem when the grid back up is absent for islanded networked microgrids. To provide a brief comparison of the proposed negotiation method with a commonly used game theory-based developed algorithm in the literature, the results of the proposed method are compared with the result of the two algorithms proposed in [61]. In the case of 20

interconnected microgrids, the results in [61] show an average convergence times of 0.025 sec, and 0.05 sec, respectively. However, the negotiation method proposed in this thesis offers a shorter negotiation time of 0.0155 sec for the same number of microgrids as shown in Fig. 4.3, which demonstrates the time efficiency of the proposed energy trading model.

Table 4.2. Validation of obtained results in comparison to results from the literature

Comparison Aspects	Proposed Model	The Model Proposed in [21]
Computation time for contract determination	10 ms for six nodes system	~14 ms for six nodes system (700 ms for 300 nodes)
Consensus protocol	Less energy and time-consuming protocol based on pBFT and modified PoS	Utilizes a time-consuming consensus method contains three components: <i>contract-chain</i> , <i>ledger-chain</i> , and a <i>high frequency verification module</i> that requires all nodes to solve puzzle problems and vote for verification (validation time is not reported).
Comparison aspect	Proposed model	The model proposed in [61]
Transaction negotiation time (Average convergence time)	0.0115 sec for 20 node system	Algorithm 1: 0.025 sec for 20 node system Algorithm 2: 0.05 sec for 20 node system

With regard to justifying the fast validation time of our proposed consensus method, the work in [62] confirms that in networked computer systems a pBFT algorithm can be executed in the order of milliseconds. Furthermore, in the modified proof of stake algorithm being proposed in the second phase of the consensus process, the stake is calculated automatically based on pre-existing data without a pre-determined time constraint. Therefore, the PoS algorithm does not significantly impact the validation time.

It is also worth mentioning that although there are many advantages to pBFT-based consensus, it has been categorized as communication bound, hence it has a scalability issue when

utilized for networks with a very large number of nodes. To overcome these limits without sacrificing safety, the method proposed in [63] includes an approach to partition large networks into smaller groups called federates; thus resulting in an improved scalability up to 1000 nodes [53].

4.2 Grid Connected Operation – Multiagent Models

4.2.1 Model I

The scheduling problem is modeled as a Mixed Integer Linear Program (MILP) optimization problem using IBM CPLEX 12.7, and the energy trading model was simulated for a case of 7 networked microgrids using Python 3.6. The generation and load data were modified from compiled wind energy and load data for 7 interconnected microgrids with an average installed renewable energy capacity of 7.5 MW for each microgrid. Each microgrid is equipped with a dispatchable generator with a fuel cost of 61.3 \$/MW [64]. The maximum power generation per round for the dispatchable generator was assumed to be 0.15 MW. Each microgrid is additionally equipped with a battery storage system with a charge and discharge cost of 70 \$/MW [64]. The maximum charging limit per round is considered to be 0.3 MW. The maximum capacity of the tie line connecting each microgrid with the utility grid is considered to be 0.5 MW. A 24-hour dynamic energy price for solving the scheduling optimization problem is adopted as shown in Fig. 4.7 [65]. The cost of load shedding and renewable energy curtailment is assumed to be 1,000 \$/MW.

After solving the local scheduling problem for a group of seven networked microgrids, there was a total excess load of 18.92 MW (positive net load) and an excess generation of -16.97 (negative net load) over the 24-hour time horizon. Fig. 4.8 shows the total net load to be traded for each time interval. The trading model is then simulated for each hour in the day ahead 24-

hour time horizon. The total amount of power traded over the 24-hour period was 8.85 MW, distributed over 58 contracts. The maximum power traded in a single contract was 0.56 MW exchanged at 2:00 am, while the minimum power traded in a single contract was 0.0018 MW exchanged at 3:00 am. Fig. 4.9 shows the amount of deficit power in each hour that was satisfied by the trading model (labelled in orange). Since there is more deficit power than excess power, not all deficit power is satisfied, where the remaining deficit power will be curtailed.

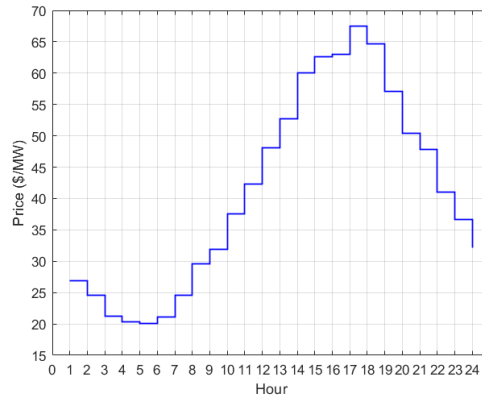


Figure 4.7. 24-hour dynamic energy prices

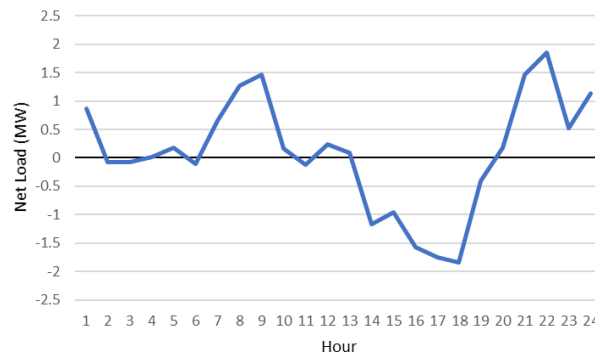


Figure 4.8. Excess and deficit power to be traded from seven networked microgrids

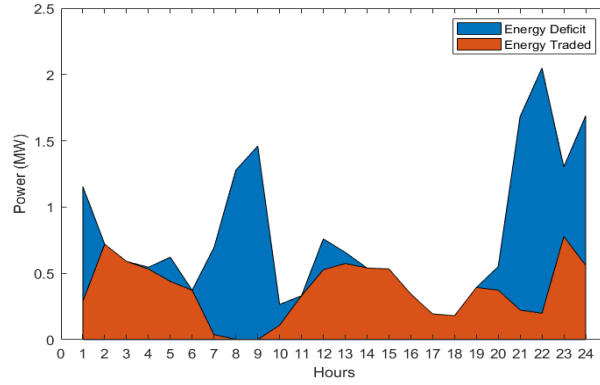


Figure 4.9. Amount of deficit power vs satisfied deficit power resulting from the peer-to-peer trading model

The model has successfully facilitated efficient energy trading among the networked microgrids, and reduced the amount of curtailed power, hence, the costs associated with load shedding and power curtailment have been reduced. It should be noted that almost all deficit power is satisfied through energy trading for early morning hours, namely 2:00 am through 6:00 am and afternoon hours 11:00 am through 8:00 pm. Conversely, in the morning hours from 7:00 am to 10:00 am and nighttime hours from 9:00 pm to 12:00 midnight, there is insufficient excess generation available to satisfy the demand during these periods. This result correlates well with Fig. 4.8, where the total net load of the microgrid network was largely positive (deficit power) in the later morning and night, and negative (excess power) in the early morning and mid-day through the evening. We can conclude from Fig. 4.9 that, when excess generation is available, the proposed trading model is effective to satisfy all deficit net load. It is also worth noting that no contracts were executed at 9:00 am, which was a result of no excess generation being available in the marketplace during that hour. The total cost of energy trading in the 24-hour period was \$321.78, with the average contract price of 36 \$/MW and average cost of a contract at \$5.55. The maximum contract price was 63.98 \$/MW executed at 3:00 pm, while the

minimum contract price was 16.13 \$/MW executed at 2:00 am. Fig. 4.10 shows a comparison between the final contract trading price and the dynamic utility price. This comparison shows that, even without limiting the upper bound of the negotiated price, the pricing mechanism consistently arrives at a price close to the utility price at each hour, indicating that the pricing mechanism used in the trading model is fair for both buyers and sellers.

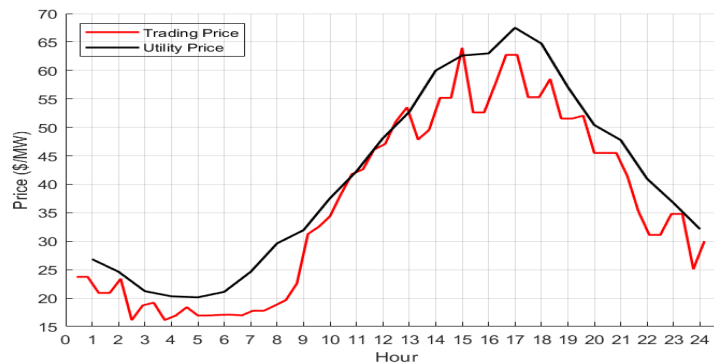


Figure 4.10. Comparison of the model trading price to the utility price

Fig. 4.11 shows an example of the price adjustment of marketplace participants during hour 2. During this hour, two sellers offered contracts in the marketplace, seller 1 having 0.64 MW excess power and seller 2 having 0.157 MW excess power. Two buyers attempted to satisfy their loads by purchasing energy from the marketplace, with buyer 1 having a 0.12 MW deficit and buyer 2 having a 0.599 MW deficit. The initial selling prices offered were 31.19 \$/MW for seller 1 and 27.88 \$/MW for seller 2. Initial buying prices were set at 21.77 \$/MW for buyer 1 and 18.27 \$/MW for buyer 2. No contract matches occurred in the first round, so each participant adjusted their price for a second round. During the second trading round, buyer 1 matched with seller 2 and a contract was executed which sold 0.12 MW to buyer 1 at a price of 20.91 \$/MW. This contract fully satisfied the load of buyer 1 and reduced the available power for sale from seller 2 to 0.036 MW. Because buyer 1 was no longer participating in the trading and seller 2 still had the lowest available contract price, buyer 2 also matched with seller 2, exchanging the

remaining excess energy of 0.036 from seller 2 at the same selling price. Since buyer 2 still had an unsatisfied load of 0.56 MW, they also match with seller 1 at a price of 23.39 \$/MW. A contract was executed and fully satisfied the demand of buyer 2. Since Seller 1 still has a remaining excess power and there are no buyers during this hour, seller 1 curtailed the remaining excess power of 0.076 MW.

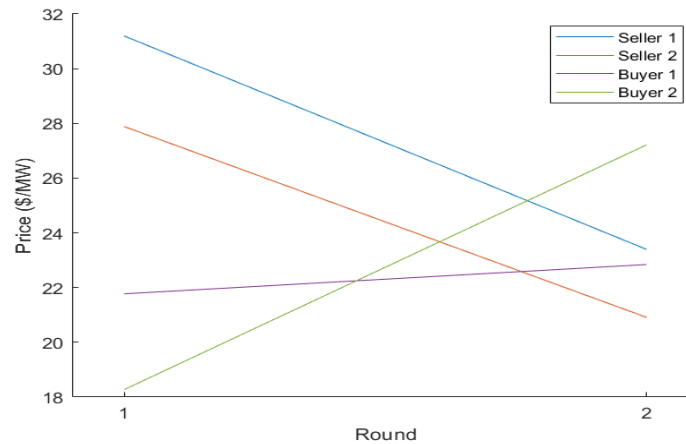


Figure 4.11. An example of price adjustment process of two buyers and two sellers

After all contract negotiations have been finalized, the details of each contract are recorded as blocks, verified using the proposed two-phase consensus mechanism, and placed on a contract records blockchain. Similarly, as with the simulation in section 4.1, each validated block in the chain contains (i) buyer and seller identification IDs, (ii) the amount of power being traded, (iii) a transaction price expressed in \$/MW, (iv) the timestamp of the execution of the transaction, and (v) an alpha-numeric hash, which is taken from the previous block on the blockchain. A sample of two sequentially generated blocks is shown in Fig. 4.12.

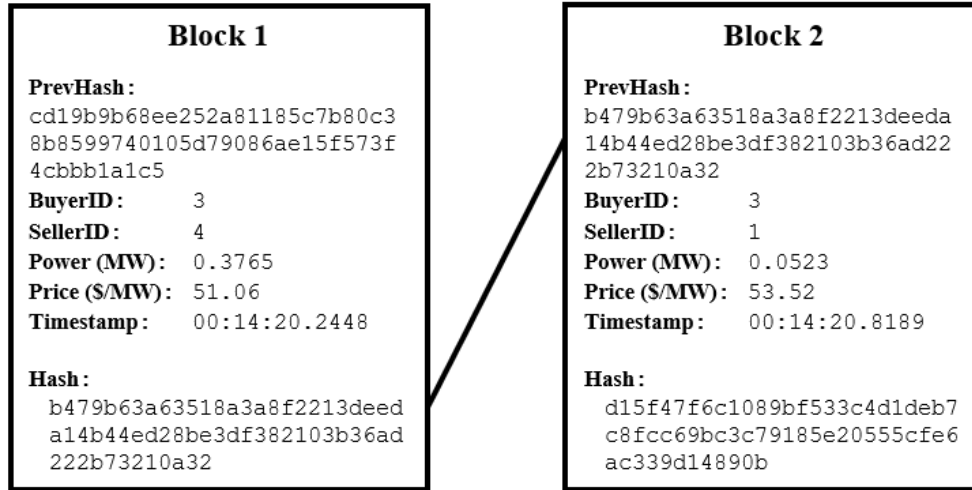


Figure 4.12. A sample of generated blocks containing contract data

For grid-connected operation model I, the test system consisted of a 7-node microgrid network. The proposed blockchain settlement protocol was simulated for a system of this size, and it was found that the average validation time for a single block in this system was 0.705 seconds, or 705 ms.

4.2.2 Model II

The same scheduling problem, under the same parameters detailed in section 3.2.1, is utilized for the second non-islanded trading model. After solving the local scheduling problem for the seven microgrid network, there was a total excess load of 18.92 MW (positive net load) and an excess generation of -16.91 (negative net load) over the 24-hour time horizon. Again Fig. 4.8 shows the total net load to be traded for each time interval.

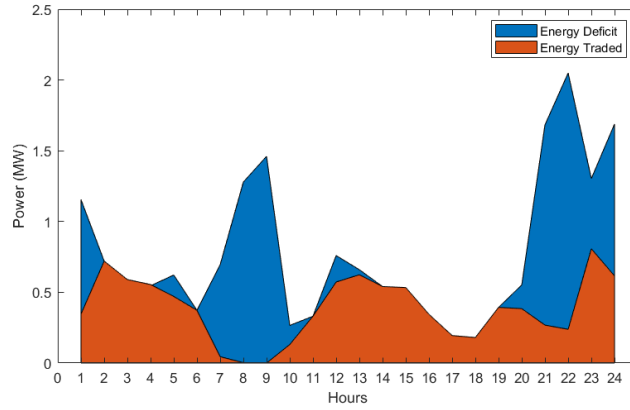


Figure 4.13. Comparison of energy deficit to traded energy

The trading model is then simulated each hour in a day ahead 24-hour time horizon. It was found that the model has successfully facilitated efficient energy trading among the networked microgrids and reduced the amount of curtailed power. Thus, the proposed model improves the operation cost of each microgrid by reducing the high costs associated with load shedding and power curtailment. The total amount of power traded over the 24-hour period was found to be 9.26 MW, distributed over 59 contracts. The maximum power traded in a single contract was 0.599 MW exchanged at hour 2 while the minimum power traded in a single contract was found to be 0.0005 MW (0.5 kW) exchanged at hour 8. Fig. 4.12 shows the amount of deficit power in each hour that was satisfied by the trading model (labelled in orange).

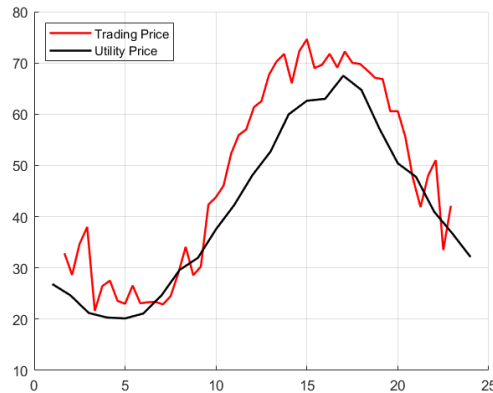


Figure 4.14. Comparison of the model trading price to utility price

The total cost of energy trading in the 24-hour period was \$433.17, with the average contract price of 47.59 \$/MW and average cost of a contract at \$8.33. The maximum contract price was 74.63 \$/MW executed at hour 15, while the minimum contract price was 21.63 \$/MW executed at hour 3. Fig. 4.13 shows a comparison between the final contract trading price and the dynamic utility price. Looking to this figure, it is clear that the proposed model facilitates the trading of contracts at prices close to the utility price, indicating that the pricing mechanism used in the trading model is fair for both buyers and sellers.

Fig. 4.14 shows an example of the price adjustment of marketplace participants during hour 19. During this hour, two sellers offered contracts in the marketplace, seller 1 having 0.4179 MW excess power and seller 2 having 0.3744 MW excess power. Two buyers attempted to satisfy their loads by purchasing energy from the marketplace, with buyer 1 having a 0.2467 MW deficit and buyer 2 having a 0.1468 MW deficit. The initial selling prices offered were 69.39 \$/MW for seller 1 and 68.73 \$/MW for seller 2. Initial buying prices were set at 48.23 \$/MW for buyer 1 and 50.16 \$/MW for buyer 2. No contract matches occurred in the first 10 rounds, so each participant adjusted their price incrementally to attempt to execute a contract. During the 11th trading round, buyer 1 matched with seller 2 and a contract was executed which sold 0.2467 MW to buyer 1 at a price of 67.23 \$/MW. This contract fully satisfied the load of buyer 1 and reduced the available power for sale from seller 2 to 0.1277 MW. Because buyer 1 fully satisfied their load deficit, they no longer participated in subsequent trading rounds. Additionally, since no other prices matched in round 11, price adjustments continued until round 16 when buyer 2 matched prices with seller 1. The executed contract traded 0.1468 MW to buyer 2 at a price of 66.84 \$/MW, fully satisfying the load deficit of buyer 2 and leaving seller 1 with a remaining excess of 0.2711 MW. Since both seller 1 and seller 2 still have remaining excess

power and there are no more buyers participating on the marketplace for this hour, seller 1 is forced to curtail their remaining excess power of 0.2711 MW and seller 2 is forced to curtail their remaining excess power of 0.1277 MW.

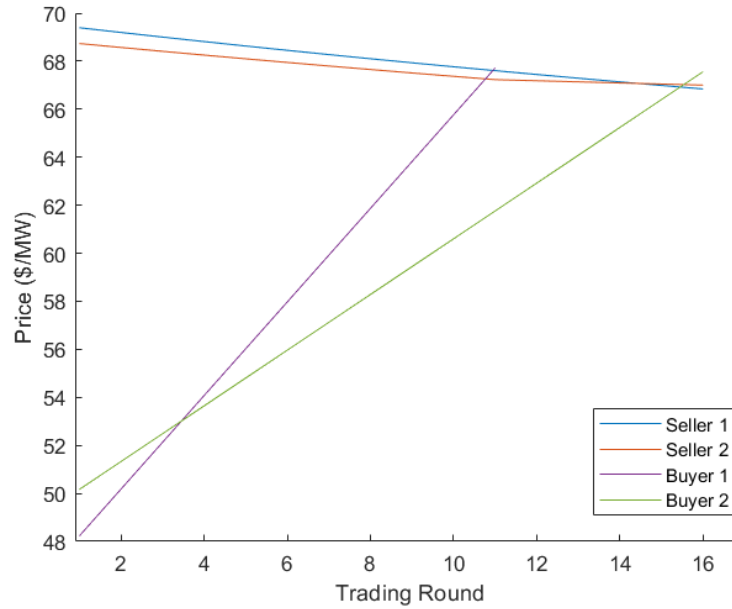


Figure 4.15. An example of price adjustment process of two buyers and two sellers

For hours in which all excess energy was sold in the marketplace but there still existed an energy deficit with the final buyer, a final contract was executed utilizing the spinning reserve energy of the final seller. For this contract, the amount of energy sold could not exceed 20% of the final seller's net load, according to equations (29) and (30) from section 3.2.3. In total, 13 reserve energy contracts were executed over the 24-hour simulation which traded 0.359 MW of spinning reserve energy, at an average of 7.33 kW of spinning reserve energy traded per contract. Each of these contracts was executed at the designated spinning reserve price of 63 \$/MW, with the total cost of spinning reserve contracts being \$22.63, for an average contract cost of \$0.45. The results shown in Fig. 4.15 show the total amount of traded power for each hour in the 24-hour time horizon, and highlights the contribution of spinning reserve power in each hour where

a spinning reserve contract was executed. As can be seen in the figure, spinning reserve contracts (highlighted in red) contributed to the traded energy in 13 of 24 hours.

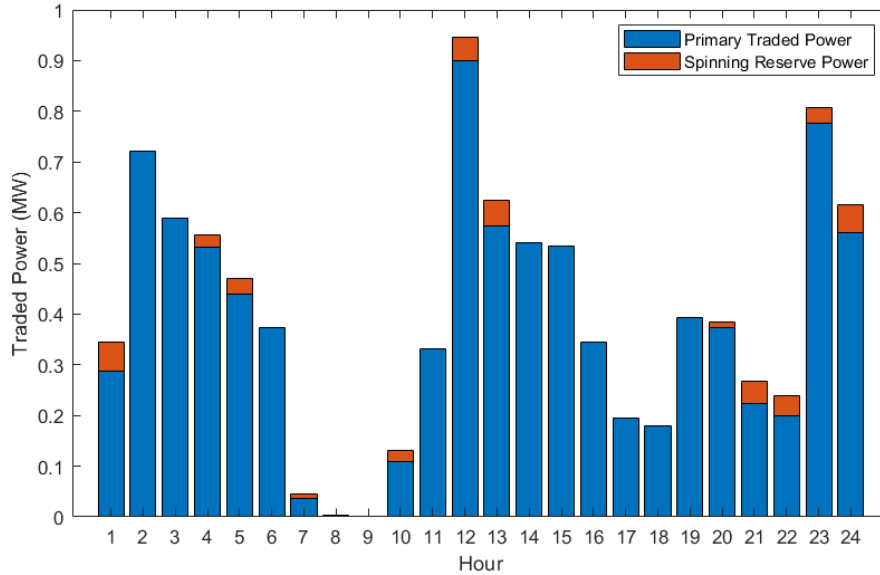


Figure 4.16. Total energy traded in each hour, including primary and spinning reserve power

After all contract negotiations have been finalized, the details of each contract are recorded as blocks, verified using the proposed two-phase consensus mechanism, and placed on a contract records blockchain. Similarly, as with the simulation in section 4.1, each validated block in the chain contains (i) buyer and seller identification IDs, (ii) the amount of power being traded, (iii) a transaction price expressed in \$/MW, (iv) the timestamp of the execution of the transaction, and (v) an alpha-numeric hash, which is taken from the previous block on the blockchain. A sample of two sequentially generated blocks is shown in Fig. 4.12. Whereas Fig. 4.14 shows a sample of generated block for the spinning reserve contract (Block 2). The block contains (i) buyer and seller identification IDs, (ii) the amount of power being traded using the seller spinning reserve in MW and as percentage of the seller excess power, (iii) a transaction

price expressed in \$/MW, (iv) the timestamp of the execution of the transaction, and (v) an alpha-numeric hash, which is taken from the previous block on the blockchain.

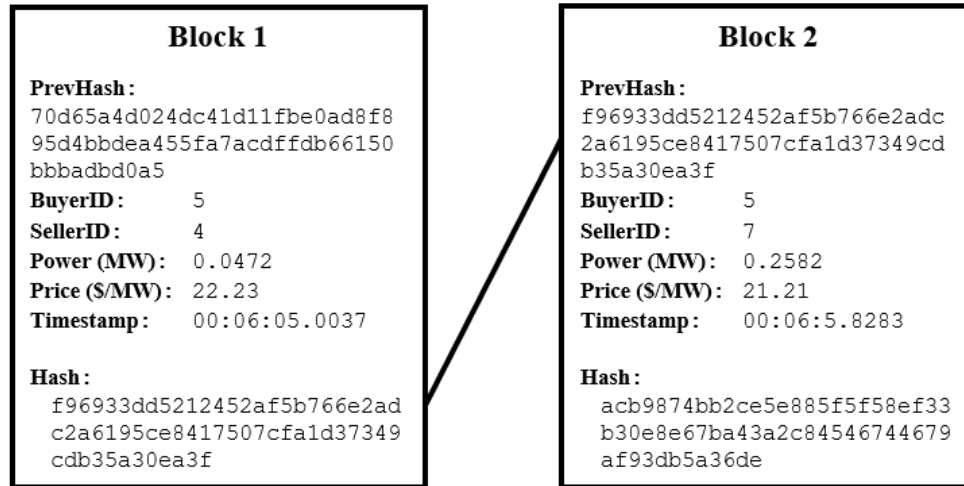


Figure 4.17. A sample of generated blocks containing contract data

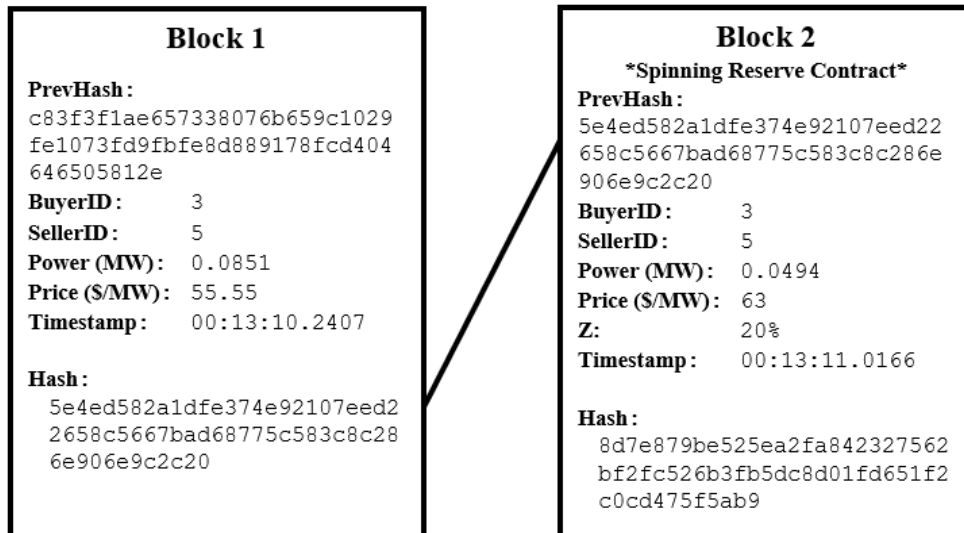


Figure 4.18. A sample of a generated block including a spinning reserve contract block

CHAPTER 5

CONCLUSION

In this work, multiple Peer-to-Peer (P2P) energy trading mechanisms for groups of both islanded and grid-connected microgrid networks are proposed. Additionally, a two-stage blockchain-based energy transaction settlement protocol is developed to ensure the security of the energy trading transactions. Simulation results show that all of the proposed electricity trading mechanisms can efficiently facilitate energy trading between networked microgrids. The conclusion of this work can be summarized as follows:

- In the case of islanded interconnected microgrids, simulation results showed that the proposed energy trading model and price adjustment mechanism effectively facilitated fair energy transactions between microgrids in a cost and time efficient manner. The proposed model ensured grid network reliability by ensuring the satisfaction of the deficit and excess power of all trading participants.
- In the case of grid-connected microgrid networks, simulation results showed that the proposed energy trading models and price adjustment mechanisms effectively facilitated energy trading and assured price fairness for all trading participants. The optimization problem formulation and price adjustment mechanisms assured minimum operation cost for each microgrid in the network.
- The novel two-phase blockchain-based transaction settlement protocol promoted system security and records immutability through the use of a two-stage consensus protocol. Simulation results showed that the proposed consensus algorithm is time-efficient compared with traditional consensus methods.

5.1 Future Work

Further research which can be undertaken to extend the conducted work may include:

- Improving the scalability of the proposed consensus protocol by partitioning a large network into smaller networks (federates) as proposed in [63].
- Simulation of the proposed energy trading models utilizing different and more comprehensive generation and load data collected from an operational microgrid network.
- A unified energy trading and price adjustment mechanism for both islanded and grid-connected operation which includes specific considerations for energy trading during transitions between islanded and grid-connected operation.
- Development of new hybrid two-phase consensus protocols that combines pBFT with different consensus methods such as Proof of Authority and Proof of Work.

5.2 Outcome Publications

Portions of the research presented in this thesis have been peer-reviewed and published as shown below:

- 1) T. M. Masaud, J. Warner and E. F. El-Saadany, "A Blockchain-Enabled Decentralized Energy Trading Mechanism for Islanded Networked Microgrids," IEEE Access, vol. 8, pp. 211291-211302, November 2020.
- 2) J. Warner and T. M. Masaud, "Decentralized Peer-to-Peer Energy Trading Model for Networked Microgrids," in IEEE Conference on Technologies for Sustainability, Orange County, CA, USA (Virtual), April 22-24, 2021. Accepted.

REFERENCES

- [1] M. S. Saleh, A. Althaibani, Y. Esa, Y. Mhandi, A. A. Mohamed, "Impact of clustering microgrids on their stability and resilience during blackouts," *2015 International Conference on Smart Grid and Clean Energy Technologies (ICSGCE)*, pp. 195–200, October 2015.
- [2] J. Lelieveld, K. Klingmüller, A. Pozzer, R. T. Burnett, A. Haines, V. Ramanathan, "Effects of fossil fuel and total anthropogenic emission removal on public health and climate," *Proceedings of the National Academy of Sciences of the United States of America*, 116 (15), pp. 7192-7197, March 2019.
- [3] A. P. Sakis Meliopoulos et al., "Smart Grid technologies for autonomous operation and control," in *IEEE Transactions on Smart Grid*, vol. 2, no. 1, pp. 1-10, March 2011, doi: 10.1109/TSG.2010.2091656.
- [4] S. Parhizi, H. Lotfi, A. Khodaei, and S. Bahramirad, "State of the art in research on microgrids: A review", *IEEE Access*, vol. 3, pp. 890 - 925, June 2015.
- [5] IEEE Guide for Design, Operation, and Integration of Distributed Resource Island Systems with Electric Power Systems, IEEE Standard 1547.4, pp. 1–54, 2011.
- [6] N. Aitzhan and D. Svetinovic, "Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 5, pp. 840–852, Sep./Oct. 2016.
- [7] LO3 Energy: The Future of Energy, [Online]. Available: <https://lo3energy.com/>. Accessed: Nov. 20, 2019.
- [8] B. Wang, M. Dabbaghjamesh, A. K. Fard and S. Mehraeen, "Cybersecurity enhancement of power trading within the networked microgrids based on blockchain and directed

- acyclic graph approach," in *IEEE Transactions on Industry Applications*, p. In Press., 2019.
- [9] A. S. Musleh, G. Yao and S. M. Muyeen, "Blockchain applications in smart grid—review and frameworks," in *IEEE Access*, vol. 7, pp. 86746-86757, 2019, doi: 10.1109/ACCESS.2019.2920682.
- [10] A. Paudel, K. Chaudhari, C. Long and H. B. Gooi, "Peer-to-peer energy trading in a prosumer-based community microgrid: A game-theoretic model," in *IEEE Transactions on Industrial Electronics*, vol. 66, no. 8, pp. 6087-6097, Aug. 2019, doi: 10.1109/TIE.2018.2874578.
- [11] W. Tushar, B. Chai, C. Yuen, S. Huang, D. B. Smith, H. V. Poor, and Z. Yang, "Energy storage sharing in smart grid: a modified auction based approach", *IEEE Transactions on Smart Grid*, vol. 7, no. 3, May. 2016. pp. 1462 – 1475.
- [12] A. Anees, T. Dillon and Y.-P. P. Chen, "A novel decision strategy for a bilateral energy contract", *Applied Energy*, vol. 253, pp. 1-13, Nov. 2019.
- [13] S. Cui, Y. Wang and N. Liu, "Distributed game-based pricing strategy for energy sharing in microgrid with PV prosumers," *IET Renewable Power Generation*, vol. 12, no. 3, pp. 380-388, 2018.
- [14] P. Shamsi, H. Xie, A. Longe and J. Joo, "Economic dispatch for an agent-based community microgrid," in *IEEE Transactions on Smart Grid*, vol. 7, no. 5, pp. 2317-2324, Sept. 2016, doi: 10.1109/TSG.2015.2487422.
- [15] J. Kang, R. Yu, X. Huang, S. Maharjan, Y. Zhang and E. Hossain, "Enabling localized peer-to-peer electricity trading among plug-in hybrid electric vehicles using consortium

- blockchains," in *IEEE Transactions on Industrial Informatics*, vol. 13, no. 6, pp. 3154-3164, Dec. 2017, doi: 10.1109/TII.2017.2709784.
- [16] K. Chen, J. Lin, Y. Song, "Trading strategy optimization for a prosumer in continuous double auction-based peer-to-peer market: A prediction integration model," *Applied Energy*, vol. 242, pp. 1121-1133, 2019.
- [17] M. H. Cintuglu, H. Martin and O. A. Mohammed, "Real-time implementation of multiagent-based game theory reverse auction model for microgrid market operation," *2016 IEEE Power and Energy Society General Meeting (PESGM)*, Boston, MA, USA, pp. 1-1, 2016, doi: 10.1109/PESGM.2016.7741074.
- [18] A. Luth, M. Zepter, P. C. del Granado, R. Egging, "Local electricity market designs for peer-to-peer trading: The role of battery flexibility," *Applied Energy*, vol. 229, pp. 1233-1243, 2018.
- [19] S. Nguyen, W. Peng, P. Sokolowski, D. Alahakoon, X. Yu, "Optimizing rooftop photovoltaic distributed generation with battery storage for peer-to-peer energy trading," *Applied Energy*, vol. 228, pp. 2567-2580, 2018.
- [20] N. Liu, X. Yu, C. Wang, C. Li, L. Ma, and J. Lei, "An energy sharing model with price-based demand response for microgrids of peer-to-peer prosumers," *IEEE Transactions on Power Systems*, vol. 32, no. 5, pp. 3569 – 3583, Jan. 2017.
- [21] F. Luo, Z. Y. Dong, G. Liang, J. Murata and Z. Xu, "A distributed electricity trading system in active distribution networks based on multi-agent coalition and blockchain," *IEEE Transactions on Power Systems*, vol. 34, no. 5, pp. 4097-4108, September 2019.
- [22] S. J. Pee, E. S. Kang, J. G. Song and J. W. Jang, "Blockchain based smart energy trading platform using smart contract," *2019 International Conference on Artificial Intelligence*

- in Information and Communication (ICAHC)*, Okinawa, Japan, pp. 322-325, 2019, doi: 10.1109/ICAHC.2019.8668978.
- [23] M. T. Devine and P. Cuffe, "Blockchain electricity trading under demurrage," *IEEE Transactions on Smart Grid*, vol. 10, no. 2, pp. 2323-2325, March 2019.
- [24] H. Wang, J. Huang, "Incentivizing energy trading for interconnected microgrids," *IEEE Transactions on Smart Grid*, vol. 9, no. 4, pp. 2647-2657, July 2018.
- [25] C. Dang, J. Zhang, C. Kwong, L. Li, "Demand side load management for big industrial energy users under blockchain-based peer-to-peer electricity market," in *IEEE Transactions on Smart Grid*, vol. 10, no. 6, pp. 6426-6435, March 2019.
- [26] B. Hu, C. Zhou, Y. C. Tian, Y. Qin and X. Junping, "A collaborative intrusion detection approach using blockchain for multimicrogrid systems," in *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 49, no. 8, pp. 1720-1730, August 2019.
- [27] Z. Li, J. Kang, R. Yu, D. Ye, Q. Deng and Y. Zhang, "Consortium blockchain for secure energy trading in industrial internet of things," in *IEEE Transactions on Industrial Informatics*, vol. 14, no. 8, pp. 3690-3700, August 2018.
- [28] M. U. Hassan, M. H. Rehmani and J. Chen, "DEAL: Differentially private auction for blockchain-based microgrids energy trading," in *IEEE Transactions on Services Computing*, vol. 13, no. 2, pp. 263-275, 1 March-April 2020, doi: 10.1109/TSC.2019.2947471.
- [29] S. Zhang, M. Pu, B. Wang, B. Dong, "A privacy protection scheme of microgrid direct electricity transaction based on consortium blockchain and continuous double auction," *IEEE Access*, vol. 7, pp. 151746-151753, October 2019.

- [30] S. Wang, A. F. Taha, J. Wang, K. Kvaternik, and A. Hahn, "Energy crowdsourcing and peer-to-peer energy trading in blockchain-enabled smart grids", *IEEE Transactions on Systems, Man, and Cybernetics Systems*, vol. 49. no.8, August 2019.
- [31] US Energy Information Administration, "State Electricity Profiles," US EIA, Washington, DC, USA, 2020. Accessed on: Dec. 15, 2020. [Online].
- [32] "IEEE Standard for the Specification of Microgrid Controllers," in IEEE Std 2030.7-2017 , vol., no., pp.1-43, 23 April 2018, doi: 10.1109/IEEESTD.2018.8340204
- [33] M. Pulcherio, M. S. Illindala, J. Choi and R. K. Yedavalli, "Robust microgrid clustering in a distribution system with inverter-based DERs," in *IEEE Transactions on Industry Applications*, vol. 54, no. 5, pp. 5152-5162, Sept.-Oct. 2018, doi: 10.1109/TIA.2018.2853039.
- [34] Y. Han, K. Zhang, H. Li, E. A. A. Coelho and J. M. Guerrero, "MAS-based distributed coordinated control and optimization in microgrid and microgrid clusters: A comprehensive overview," in *IEEE Transactions on Power Electronics*, vol. 33, no. 8, pp. 6488-6508, Aug. 2018, doi: 10.1109/TPEL.2017.2761438.
- [35] J. A. P. Lopes, C. L. Moreira, and A. G. Madureira, "Defining control strategies for microgrids islanded operation," *IEEE Transaction on Power Systems*, vol. 21, no. 2, pp. 916–924, May 2006.
- [36] H. Kanchev, F. Colas, V. Lazarov and B. Francois, "Emission reduction and economical optimization of an urban microgrid operation including dispatched PV-based active generators," in *IEEE Transactions on Sustainable Energy*, vol. 5, no. 4, pp. 1397-1405, Oct. 2014, doi: 10.1109/TSTE.2014.2331712.

- [37] W. Tushar, T. K. Saha, C. Yuen, D. Smith and H. V. Poor, "Peer-to-peer trading in electricity networks: An overview," in *IEEE Transactions on Smart Grid*, vol. 11, no. 4, pp. 3185-3200, July 2020, doi: 10.1109/TSG.2020.2969657.
- [38] T. Basar and G. J. Olsder, *Dynamic Noncooperative Game Theory*. New York, NY: Academic Press, 1995.
- [39] W. Saad, Z. Han, H. V. Poor, and T. Basar, "Game-theoretic methods for the smart grid: An overview of microgrid systems, demand-side management, and smart grid communications," *IEEE Signal Processing Magazine*, vol. 29, no. 5, pp. 86–105, Sept. 2012.
- [40] W. Tushar, W. Saad, H. V. Poor, and D. B. Smith, "Economics of electric vehicle charging: A game theoretic approach," *IEEE Transactions on Smart Grid*, vol. 3, no. 4, pp. 1767–1778, Dec. 2012.
- [41] McAfee, R. Preston; McMillan, John (1987). "Auctions and Bidding". *Journal of Economic Literature*. 25 (2): 699–738.
- [42] D. Ganguly and S. Chakraborty, "E commerce - forward and reverse auction - A managerial tool to succeed over business competitiveness," *2008 Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, Phuket, pp. 447-452, 2008, doi: 10.1109/SNPDP.2008.51.
- [43] W. Saad, Zhu Han, H. V. Poor and T. Başar, "A noncooperative game for double auction-based energy trading between PHEVs and distribution grids," *2011 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, Brussels, pp. 267-272, 2011, doi: 10.1109/SmartGridComm.2011.6102331.

- [44] S. Nakamoto. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. Accessed: Dec. 15, 2020. [Online]. Available: <https://bitcoin.org/bitcoin.pdf>
- [45] K. Toyoda, K. Machi, Y. Ohtake and A. N. Zhang, "Function-level bottleneck analysis of private proof-of-authority ethereum blockchain," in *IEEE Access*, vol. 8, pp. 141611-141621, 2020, doi: 10.1109/ACCESS.2020.3011876.
- [46] R. Beck, "Beyond bitcoin: The rise of blockchain world," *Computer*, vol. 51, no. 2, pp. 54_58, Feb. 2018.
- [47] A. Andrey and C. Petr, "Review of existing consensus algorithms blockchain," 2019 *International Conference "Quality Management, Transport and Information Security, Information Technologies" (IT&QM&IS)*, Sochi, Russia, pp. 124-127, 2019, doi: 10.1109/ITQMIS.2019.8928323.
- [48] Z. Wang, B. Chen, J. Wang and J. kim, "Decentralized energy management system for networked microgrids in grid-connected and islanded modes," in *IEEE Transactions on Smart Grid*, vol. 7, no. 2, pp. 1097-1105, March 2016, doi: 10.1109/TSG.2015.2427371.
- [49] X. Tian and K. Benkrid, "Mersenne twister random number generation on FPGA, CPU and GPU," in *Proceedings of NASA/ESA Conference on Adaptive Hardware Systems*, pp. 460–464, Jul. 2009.
- [50] W. Hou, L. Guo, and Z. Ning, "Local electricity storage for blockchain based energy trading in industrial Internet of Things," *IEEE Trans. Industrial Information*, vol. 15, no. 6, pp. 3610–3619, Jun. 2019.
- [51] B. Brooks, J. Jiang, and H. Sun, "Incorporating user utility in a smart microgrid with distributed generation and elastic demand," in *Proc. IEEE International Conference on Communication Workshops*, pp. 1–5, May 2017.

- [52] Hyperledger. (2020). *Blockchain Technologies for Business*. [Online]. Available:
<https://www.hyperledger.org>
- [53] T. T. A. Dinh, R. Liu, M. Zhang, G. Chen, B. C. Ooi and J. Wang, "Untangling blockchain: A data processing view of blockchain systems," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 7, pp. 1366-1385, 1 July 2018, doi: 10.1109/TKDE.2017.2781227.
- [54] X. Chen and X. Zhang, "Secure electricity trading and incentive contract model for electric vehicle based on energy blockchain," *IEEE Access*, vol. 7, pp. 178763-178778, 2019.
- [55] G. Maxwell. (2017). *Confidential Transactions*. [Online]. Available:
<http://diyhpl.us/wiki/transcripts/gmaxwell-confidential-transactions/>
- [56] H. Zhang, F. Zhang, B. Wei, and Y. Du, "Implementing confidential transactions with lattice techniques," *IET Information Security*, vol. 14, no. 1, pp. 30-38, Jan. 2020.
- [57] K. Mongird, "Energy storage technology and cost characterization report," Pacific Northwest Nat. Lab., Seattle, WA, USA, Tech. Rep. PNNL-28866, Jul. 2019. [Online]. Available:
https://www.energy.gov/sites/prod/files/2019/07/f65/Storage%20Cost%20and%20Performance%20Characterization%20Report_Final.pdf
- [58] D. T. Nguyen and L. B. Le, "Optimal bidding strategy for microgrids considering renewable energy and building thermal dynamics," in *IEEE Transactions on Smart Grid*, vol. 5, no. 4, pp. 1608-1620, July 2014, doi: 10.1109/TSG.2014.2313612.
- [59] Z. Jianjun, S. Dongyu, Z. Dong and G. Yang, "Load shedding control strategy for power system based on the system frequency and voltage stability," *2018 China International*

- Conference on Electricity Distribution (CICED)*, Tianjin, 2018, pp. 1352-1355, April 2018, doi: 10.1109/CICED.2018.8592262.
- [60] N. Wang, W. Xu, Z. Xu, and W. Shao, "Peer-to-peer energy trading among microgrids with multidimensional willingness," *Energies*, vol. 11, no. 12, p. 3312, Nov. 2018.
- [61] A. M. Jadhav, N. R. Patne and J. M. Guerrero, "A novel approach to neighborhood fair energy trading in a distribution network of multiple microgrid clusters," in *IEEE Transactions on Industrial Electronics*, vol. 66, no. 2, pp. 1520-1531, Feb. 2019, doi: 10.1109/TIE.2018.2815945.
- [62] T. Distler, C. Cachin and R. Kapitza, "Resource-efficient byzantine fault tolerance," in *IEEE Transactions on Computers*, vol. 65, no. 9, pp. 2807-2819, Sept. 2016, doi: 10.1109/TC.2015.2495213.
- [63] D. Mazieres, "The stellar consensus protocol: A federated model for internet-level consensus," Dec. 2017, [Online]. Available: <http://www.scs.stanford.edu/17au-cs244b/notes/scp.pdf>, Last accessed: 2020.
- [64] R. D. Mistry, F. Eluyemi, T. Masaud, "Impact of aggregated EVs charging station on the optimal scheduling of battery storage system in islanded microgrid," *IEEE North American Power Symposium*, Morgantown, WV, pp. 1-5, 2017.
- [65] National Grid Website, [Online]. Available: https://www9.nationalgridus.com/niagaramohawk/business/rates/5_hour_charge.asp

APPENDIX A



Office of Research Integrity

February 22, 2021

Jonathan Warner
Electrical & Computer Engineering
WAEC 3101
Marshall University

Dear Mr. Warner:

This letter is in response to the submitted thesis abstract entitled "*Peer-to-Peer Energy Trading for Networked Microgrids*." After assessing the abstract, it has been deemed not to be human subject research and therefore exempt from oversight of the Marshall University Institutional Review Board (IRB). The Code of Federal Regulations (45CFR46) has set forth the criteria utilized in making this determination. Since the information in this study does not involve human subjects as defined in the above referenced instruction, it is not considered human subject research. If there are any changes to the abstract you provided then you would need to resubmit that information to the Office of Research Integrity for review and a determination.

I appreciate your willingness to submit the abstract for determination. Please feel free to contact the Office of Research Integrity if you have any questions regarding future protocols that may require IRB review.

Sincerely,

Bruce F. Day, ThD, CIP
Director

APPENDIX B

TABULATED DATA

Table B.1. Net load of each microgrid (10 microgrid scenario) from randomly generated data for islanded operation

		Microgrid No.									
		1	2	3	4	5	6	7	8	9	10
Hour in the 24-hour time horizon	1	2.18	-7.51	2.21	1.21	5.98	1.15	-3.91	2.23	-0.75	-5.69
	2	0.88	3.74	8.5	-4.95	-1.44	-6.5	-0.43	0.34	5.51	-8.23
	3	-7.16	0.48	3.91	-1.25	4.61	-3.12	0.14	-3.81	-8.65	0.47
	4	0.34	8.59	-3.08	-2.48	2.45	-3.24	5.08	0.17	3.58	-6.34
	5	-3.59	0.41	1.4	0.19	0.86	6.72	-8.17	-2.28	-0.03	-0.72
	6	4.4	-2.97	-1.28	1.07	-2.67	-0.87	4.89	-2.55	4.54	0.95
	7	-0.42	6.22	4.26	-0.27	-1.75	-3.64	3.1	0.4	7.84	-1.56
	8	0.27	2	-4.05	5.07	-4.15	3.37	-4.46	-3.8	-6.12	1.97
	9	-1.16	-0.25	-3.84	0.61	0.6	0.96	0.3	6.36	-1.02	-3.05
	10	-1.65	3.51	-1.67	1	0.77	-4.13	-2.69	-4	1.86	1.4
	11	8.49	-1.88	3.94	2.12	-5.51	6.71	4.53	-2.29	-4.02	-1.39
	12	4.12	-1.94	-5.15	-0.73	6.36	7.96	4.47	4.09	0.55	0.02
	13	-1.33	6.55	4.46	-1.52	-2.8	9.06	6.7	-2.16	-6.71	-3.95
	14	8.9	1.82	-3.42	-6.31	-3.17	5.23	0.04	-1.9	-5.29	-2.22
	15	-4.05	-5.41	-7.13	-0.77	3.26	-0.74	7.02	4.39	-5.48	1.08
	16	6.39	7.69	4.74	-0.8	8.03	5.73	0.83	-4.34	8.23	2.3
	17	-0.72	-6.01	8.34	2.63	6.25	1.09	4.96	-8.16	-0.67	2.78
	18	-2.53	-2.12	-0.87	2.56	6.81	-2.43	1.13	-0.24	6.15	1.24
	19	-8.36	-1.38	-4.57	1.68	-0.41	2.67	-6.94	-1.37	3.68	3.44
	20	-5.59	4.79	-9.37	-2.27	-0.2	-2.4	-3.81	-3.62	4.28	-0.97
	21	0.46	1.53	-1.42	-1.06	7.03	-1.19	-5.74	-1.45	0.77	-2.47
	22	-0.6	-0.18	8.38	8.4	4.68	2.82	3.45	-6.24	-2.55	-2.57
	23	0.06	-4.46	4.57	8.3	1.92	-5.62	-5.43	-2.19	-1.75	-2.71
	24	1.77	-3.45	6.02	-6.77	-3.5	-2.21	-0.46	4.16	-5.22	6.89

Table B.2. Microgrid net load data used for grid-connected operation

	Microgrid No.							
		1	2	3	4	5	6	7
Hour in the 24-hour time horizon	1	-1.0879	1.0658	1.2812	-0.9228	-0.2915	-0.6728	0.7178
	2	-1.4392	0.8188	1.0711	-1.1416	-0.4007	-0.9571	0.6673
	3	-1.1819	0.5483	0.8841	-1.1714	-0.4100	-1.0842	0.7177
	4	-0.8322	0.8929	0.6810	-1.3023	-0.4534	-1.1884	0.6878
	5	-0.9556	0.9716	0.8356	-1.3251	-0.3995	-0.9503	0.7179
	6	-0.9376	1.0173	0.7609	-1.2164	-0.6946	-1.0951	0.6406
	7	-0.8374	0.8963	0.9494	-1.0277	-0.3048	-0.6898	0.8310
	8	-0.6112	1.0060	1.3298	-0.9043	-0.1632	-0.8024	0.9565
	9	-0.4116	1.0337	1.2967	-1.1386	-0.0706	-0.4902	1.0514
	10	-0.3749	0.8225	1.1505	-1.1760	-0.1092	-0.6339	1.0156
	11	-0.6009	0.7378	1.2809	-1.2022	-0.3865	-0.8644	0.8634
	12	-1.0991	0.6847	1.7097	-0.9516	-0.2279	-0.6570	0.6802
	13	-1.1278	0.7894	1.6098	-1.0136	-0.2469	-0.7014	0.6679
	14	-1.1242	0.9071	1.4907	-1.3214	-0.2169	-0.6001	0.6581
	15	-0.8272	1.0511	1.3826	-1.4349	-0.1821	-0.5341	0.6617
	16	-0.9168	1.0542	1.1904	-1.6089	-0.3698	-0.5695	0.6822
	17	-0.5412	1.0832	1.0112	-1.7098	-0.4226	-0.4706	0.8817
	18	-0.6536	1.0284	0.8279	-1.5350	-0.4731	-0.9744	1.0521
	19	-0.4008	1.1967	0.8238	-1.5868	-0.3744	-0.4901	1.0968
	20	-0.5076	0.9942	1.0032	-1.5441	-0.6827	-0.4383	1.0871
	21	-0.6829	1.1671	1.3144	-1.3794	-0.0839	-0.5612	1.1353
	22	-0.9986	1.1823	1.7851	-1.0688	-0.0947	-0.7355	1.0267
	23	-1.4259	1.0406	1.4382	-0.9938	-0.2752	-0.9514	0.9374
	24	-1.3453	1.2199	1.5806	-0.8601	-0.2114	-0.8152	0.9498

APPENDIX C

SIMULATION CODES

C.1. Simulation Code for Islanded Operation

```
import hashlib as hasher
import datetime
import math
import random
import datetime as date
from openpyxl import Workbook

filename = "blockchain_test123.xlsx"
workbook = Workbook()
sheet = workbook.active

cost_batt = 30/1000
cost_shed = 1000/1000
cost_reserve = 250/1000
cost_trans = 0.0000028
t=1

A = ([ [0, 10.2, 14.5, 149.7, 162, 171.1, 50.3, 21.6, 175, 178.2, 182.3, 75,
184.3, 119.6, 87.6, 128, 155.2, 191, 66.2, 81],
[10.2, 0, 17.9, 159.2, 171.9, 183.4, 60.1, 17, 183.2, 188.3, 192, 85.6,
197.2, 131, 54.3, 31.6, 105.2, 142.5, 151, 162.2],
[14.5, 17.9, 0, 165.3, 177, 185.3, 5.6, 24.8, 186.8, 193.5, 196, 91.7,
202.9, 135.2, 62.2, 51, 107, 127.6, 191, 88.1],
[149.7, 159.2, 165.3, 0, 12.1, 220.5, 102.6, 175, 22.7, 27.1, 231.5,
125.2, 234.9, 172, 44.1, 191.5, 34, 202.3, 198, 215.1],
[162, 171.9, 177, 12.1, 0, 232, 112.2, 187.3, 34.8, 39.3, 242.6, 137,
247.5, 182.4, 22, 161.6, 124.2, 155, 104.5, 74.1],
[171.1, 183.4, 185.3, 220.5, 232, 0, 122.4, 95.9, 242, 247.7, 10.2,
145.6, 15, 192.1, 149.5, 86, 101.1, 62.5, 25, 36.2],
[50.3, 60.1, 5.6, 102.6, 112.2, 122.4, 0, 75.2, 122.3, 127.8, 130.5,
25, 135.9, 70.4, 178.2, 61.5, 140.2, 32, 73, 197.3],
[21.6, 17, 24.8, 175, 187.3, 95.9, 75.2, 0, 197.1, 202.3, 205.7, 100.4,
213.8, 145, 121.1, 167.5, 55, 31.4, 232.1, 14],
[175, 183.2, 186.8, 22.7, 34.8, 242, 122.3, 197.1, 0, 36.2, 252.3,
147.6, 257, 192.9, 165.5, 71, 41.4, 127.7, 182, 52.3],
[178.2, 188.3, 193.5, 27.1, 39.3, 247.7, 127.8, 202.3, 36.2, 0, 257,
152.4, 262.6, 197, 165.2, 77.7, 23.1, 112, 182.5, 69.2],
[182.3, 192, 196, 231.5, 242.6, 10.2, 130.5, 205.7, 252.3, 257, 0,
155.1, 15.9, 204.7, 68.4, 121.5, 182, 20.6, 142.2, 40],
[75, 85.6, 91.7, 125.2, 137, 145.6, 25, 100.4, 147.6, 152.4, 155.1, 0,
160.5, 95.1, 150.2, 217.5, 49.2, 108.6, 82, 77.4],
[184.3, 197.2, 202.9, 234.9, 247.5, 15, 135.9, 213.8, 257, 262.6, 15.9,
160.5, 0, 205.7, 188.2, 62.4, 89, 124.5, 160.1, 98.8],
[119.6, 131, 135.2, 172, 182.4, 192.1, 70.4, 145, 192.9, 197, 204.7,
95.1, 205.7, 0, 66.5, 221.4, 155.2, 181.3, 25, 98.2],
[87.6, 54.3, 62.2, 44.1, 22, 149.5, 178.2, 121.1, 165.5, 165.2, 68.4,
150.2, 188.2, 66.5, 0, 114.8, 202.2, 158.2, 76, 185.2],
```

```

        [128, 31.6, 51, 191.5, 161.6, 86, 61.5, 167.5, 71, 77.7, 121.5, 217.5,
62.4, 221.4, 114.8, 0, 19.5, 171.2, 207, 75],
        [155.2, 105.2, 107, 34, 124.2, 101.1, 140.2, 55, 41.4, 23.1, 182, 49.2,
89, 155.2, 202.2, 19.5, 0, 132.2, 168.4, 63],
        [191, 142.5, 127.6, 202.3, 155, 62.5, 32, 31.4, 127.7, 112, 220.6,
108.6, 124.5, 181.3, 158.2, 171.2, 132.2, 0, 212.4, 78.4],
        [66.2, 151, 191, 198, 104.5, 25, 73, 232.1, 182, 182.5, 142.2, 82,
160.1, 25, 76, 207, 168.4, 212.4, 0, 185.2],
        [81, 162.2, 88.1, 215.1, 74.1, 36.2, 197.3, 14, 52.3, 69.2, 40, 77.4,
98.8, 98.2, 185.2, 75, 63, 78.4, 185.2, 0]])

```

```

class Block:
    def __init__(self, index, timestamp, data, previous_hash):
        self.index = index
        self.timestamp = timestamp
        self.data = data
        self.previous_hash = previous_hash
        self.hash = self.hash_block()

    def hash_block(self):
        sha = hasher.sha256()
        sha.update((str(self.index) + str(self.timestamp) + str(self.data) +
str(self.previous_hash)).encode())
        return sha.hexdigest()

def create_genesis_block():
    # Manually construct a block with
    # index zero and arbitrary previous hash
    return Block(0, date.datetime.now(), "Genesis Block", "0")

def next_block(last_block):
    this_index = last_block.index + 1
    this_timestamp = date.datetime.now()
    this_data = "kW:" + str(kW) + " Price:" + str(price_seller) + " Time:"
+time
    this_hash = last_block.hash
    return Block(this_index, this_timestamp, this_data, this_hash)

class Prosumer():
    def __init__(self, index, pgen, pload, pnet, status):
        self.index = index
        self.pgen = pgen
        self.pload = pload
        self.pnet = pnet
        self.status = status

# Create the blockchain and add the genesis block
blockchain = [create_genesis_block()]
previous_block = blockchain[0]
temp_buyer = []
temp_seller = []
alpha = 0
beta = 0
gamma = 0
average = 0
phi = 1

```



```

price_retail = 0.12
j=0
k=0

# How many blocks should we add to the chain
# after the genesis block
num_of_blocks_to_add = 24
y=0
m=1
n=1
pgensum = 0
ploadsum = 0
while(y < num_of_blocks_to_add):
    genlist = list()
    loadlist = list()
    netlist = list()
    conlist = list()
    prodlist = list()
    prosumers = list()
    producers = list()
    consumers = list()

    for k in range(1, 20):
        pgen = round(random.uniform(5,15), 2)
        genlist.append(pgen)
        pload = round(random.uniform(5,15), 2)
        loadlist.append(pload)
        pnet = round(pload-pgen, 2)
        netlist.append(pnet)
        if(pgen > pload):
            status = 1
        else:
            status = 0
        prosumers.append(Prosumer(k, pgen, pload, pnet, status))

    sum3 = sum(netlist)
    print(sum3)
    if(sum3 > 0):
        prosumers.append(Prosumer(20, 0, sum3, -sum3, 1))
    else:
        prosumers.append(Prosumer(20, sum3, 0, 0-sum3, 0))

    phi += 1
    for i in range(0, len(prosumers)):
        print("Prosumer: {}".format(prosumers[i].index))
        print("Pgen: {}".format(prosumers[i].pgen))
        print("Pload: {}".format(prosumers[i].pload))
        print("Pnet: {}".format(prosumers[i].pnet))
        print("Status: {}".format(prosumers[i].status))
        sheet.cell(row=i+1, column=25+phi).value = prosumers[i].pnet

        if(prosumers[i].pnet > 0):
            consumers.append(prosumers[i])
            conlist.append(prosumers[i].pnet)
        else:
            producers.append(prosumers[i])

```

```

        prodlist.append(prosumers[i].pnet)

for i in range(0, len(consumers)):
    print("Consumers: {}".format(consumers[i].pnet))
    sheet.cell(row=t, column=15).value = consumers[i].pnet
    t+=1
for i in range(0, len(producers)):
    print("Producers: {}".format(producers[i].pnet))
    sheet.cell(row=t, column=16).value = producers[i].pnet
    t+=1

#Negotiate contract
have_contract = 0
average = 0
price_retail = 0.20
j=0
for j in range(0, len(consumers)):
    for k in range(0, len(producers)):
        if consumers[j].pnet>0 and producers[k].pnet<0:
            alpha = 0.01
            beta = 0.04
            tau = 0
            fixed_seller = 0.25
            fixed_buyer = 0.15
            cost_curt = 0
            batt_limit = 2
            reserve_limit = 5
            have_contract = 0
            tasks = 0
            gamma = 0.2
            count = 0

            while(have_contract < 1):
                if(count<5):
                    if(abs(producers[k].pnet) < batt_limit):
                        price_seller = (fixed_seller) -
tau*(cost_batt*abs(producers[k].pnet)*gamma + alpha*cost_curt*abs(0)) +
(A[j][k]*cost_trans*abs(producers[k].pnet))
                        batt_limit -= abs(producers[k].pnet)
                        tasks += 1
                    else:
                        price_seller = (fixed_seller) -
tau*(cost_batt*abs(batt_limit) + alpha*cost_curt*abs(producers[k].pnet -
batt_limit)) + (A[j][k]*cost_trans*abs(producers[k].pnet))
                        tasks +=1
                    if(abs(consumers[j].pnet) < reserve_limit):
                        price_buyer = (fixed_buyer) +
tau*(beta*cost_shed*abs(0) + cost_reserve*abs(consumers[j].pnet))
                        reserve_limit -= consumers[j].pnet
                        tasks +=1
                    else:
                        price_buyer = (fixed_buyer) +
tau*(beta*cost_shed*abs(consumers[j].pnet - reserve_limit) +
cost_reserve*abs(reserve_limit))
                        tasks += 1
                else:

```

```

        if(abs(producers[k].pnet) < batt_limit):
            price_seller = (fixed_seller) -
            tau*(cost_batt*abs(producers[k].pnet)*0 + alpha*cost_curt*abs(0)) +
            (A[j][k]*cost_trans*abs(producers[k].pnet))
            batt_limit -= abs(producers[k].pnet)
            tasks += 1
        else:
            price_seller = (fixed_seller) -
            tau*(0*cost_batt*abs(batt_limit) + alpha*cost_curt*abs(producers[k].pnet)) +
            (A[j][k]*cost_trans*abs(producers[k].pnet))
            tasks +=1
            if(abs(consumers[j].pnet) < reserve_limit):
                price_buyer = (fixed_buyer) +
                tau*(beta*cost_shed*abs(0) + 0*cost_reserve*abs(consumers[j].pnet))
                reserve_limit -= consumers[j].pnet
                tasks +=1
            else:
                price_buyer = (fixed_buyer) +
                tau*(beta*cost_shed*abs(consumers[j].pnet) +
                0*cost_reserve*abs(reserve_limit))
                tasks += 1

    if(price_seller <= price_buyer):
        have_contract = 1
        print("Price_seller: {}".format(price_seller))
        print("Price_buyer: {}".format(price_buyer))
        sheet.cell(row=m, column=1).value = price_seller
        sheet.cell(row=n, column=2).value = price_buyer
        m+=1
        n+=1
        tasks += 1
        time = date.datetime.now().strftime('%S.%f')[:-2]
        sheet.cell(row=m, column=7).value = tasks
    else:
        tau = 1
        count += 1
        print("Price_seller: {}".format(price_seller))
        print("Price_buyer: {}".format(price_buyer))
        sheet.cell(row=m, column=1).value = price_seller
        sheet.cell(row=n, column=2).value = price_buyer
        m+=1
        n+=1
        cost_curt = price_seller
        fixed_seller = price_seller
        fixed_buyer = price_buyer
        tasks += 1
        time = date.datetime.now().strftime('%S.%f')[:-2]
        sheet.cell(row=m, column=7).value = tasks

    sheet.cell(row=k+1, column=1).value = price_seller
    sheet.cell(row=k+1, column=2).value = price_buyer

    sheet.cell(row=m, column=9).value = consumers[j].pnet
    sheet.cell(row=m, column=10).value = producers[k].pnet

    if(abs(producers[k].pnet) >= abs(consumers[j].pnet)):

```

```

        kW = abs(consumers[j].pnet)
        producers[k].pnet += consumers[j].pnet
        consumers[j].pnet = 0
    else:
        kW = abs(producers[k].pnet)
        consumers[j].pnet += producers[k].pnet
        producers[k].pnet = 0

    time = date.datetime.now().strftime('%S.%f')[:-2]
    block_to_add = next_block(previous_block)
    blockchain.append(block_to_add)
    previous_block = block_to_add
    print("Block #{} has been added to the
blockchain!".format(block_to_add.index))
    print("Data: {}".format(block_to_add.data))
    print("Hash: {}\n".format(block_to_add.hash))
    sheet.cell(row=m, column=3).value = block_to_add.index
    sheet.cell(row=m, column=4).value = block_to_add.data
    sheet.cell(row=m, column=5).value = block_to_add.hash
    sheet.cell(row=m, column=6).value = time
    sheet.cell(row=m, column=7).value = tasks
    sheet.cell(row=m, column=8).value = kW

    sheet.cell(row=m, column=11).value = sum(conlist)
    sheet.cell(row=m, column=12).value = sum(prodlist)

    y +=1

workbook.save(filename=filename)

```

C.2. Simulation Code for Grid-Connected Operation – Model I

```
import hashlib as hasher
import datetime
import math
import random
import numpy as np
from openpyxl import load_workbook

# -----
# DATA HANDLING
#
wb2 = load_workbook('NET LOAD.xlsx', data_only=True)

z=1

ws1 = wb2["Sheet2"]
ws2 = wb2["Sheet3"]
sheet = wb2.active
cell_range = ws1['A3':'I26']

#-----
# MG1

Load1 = []
for x in range(3,27):
    Load1.append((ws1.cell(row=x,column=2).value))
Load1 = list(np.around(np.array(Load1),4))

print("Load1: ")
print(*Load1)

Load2 = []
for x in range(3,27):
    Load2.append((ws1.cell(row=x,column=3).value))
Load2 = list(np.around(np.array(Load2),4))

print("\nLoad2: ")
print(*Load2)

Load3 = []
for x in range(3,27):
    Load3.append((ws1.cell(row=x,column=4).value))
Load3 = list(np.around(np.array(Load3),4))

print("\nLoad3: ")
print(*Load3)

Load4 = []
for x in range(3,27):
    Load4.append((ws1.cell(row=x,column=5).value))
Load4 = list(np.around(np.array(Load4),4))

print("\nLoad4: ")
print(*Load4)
```

```

Load5 = []
for x in range(3,27):
    Load5.append((ws1.cell(row=x,column=6).value))
Load5 = list(np.around(np.array(Load5),4))

print("\nLoad5: ")
print(*Load5)

Load6 = []
for x in range(3,27):
    Load6.append((ws1.cell(row=x,column=7).value))
Load6 = list(np.around(np.array(Load6),4))

print("\nLoad6: ")
print(*Load6)

Load7 = []
for x in range(3,27):
    Load7.append((ws1.cell(row=x,column=8).value))
Load7 = list(np.around(np.array(Load7),4))

print("\nLoad7: ")
print(*Load7)

Load8 = []
for x in range(3,27):
    Load8.append((ws1.cell(row=x,column=9).value))
Load8 = list(np.around(np.array(Load8),4))

print("\nLoad8: ")
print(*Load8)

# -----
# GLOBAL CONSTANTS & COUNTERS
#

Cbat = 0.03
Csh = 0.8
Cres = 0.25
Ckw = 0.25
Cl = 0.08
Cpk = 0.5
line_limit = 1000
Ccurt = 1000
Cgrid = [26.84, 24.63, 21.21, 20.32, 20.14, 21.10, 24.62, 29.62, 31.94,
37.54, 42.26, 48.08, 52.71, 59.98, 62.62, 62.99, 67.50, 64.72, 57.09, 50.41,
47.77, 40.96, 36.71, 32.12]

t=1
r = 0
q=0
fp_buy = 5
fp_sell = 25
j=1

for x in range(len(Load1)):

```

```

print("Hour # {}".format(x+1))
traders = []
prod = []
cons = []
traders.append(Load1[x])
traders.append(Load2[x])
traders.append(Load3[x])
traders.append(Load4[x])
traders.append(Load5[x])
traders.append(Load6[x])
traders.append(Load7[x])
traders.append(Load8[x])

for i in range(len(traders)):
    if(traders[i] < 0):
        prod.append(traders[i])
    elif(traders[i] > 0):
        cons.append(traders[i])
print(prod)
print(cons)
alpha = 1
price_buy = 8
price_sell = 25
r = 1
q=1
fixed_buylist=[]
price_buylist=[]
fixed_selllist=[]
price_selllist=[]

while(sum(prod) != 0 and sum(cons) != 0):
    r=1
    q=1
    trade = 0
    while(sum(prod) != 0 and sum(cons) !=0):
        for j in range(len(prod)):
            sale = False
            if(r <= 10):
                if(r==1):
                    fixed_sell = (1 +
((abs(prod[j]))**(1/2))/3))*Cgrid[x]
                    fixed_selllist.append(fixed_sell)
                    print("Fixed_seller price:
{}".format(round(fixed_sell, 4)))
                    price_sell = fixed_sell - (((r-1)/(r +
math.factorial(r)))*price_sell)
                    price_selllist.append(price_sell)
                    print("Seller price: {}".format(round(price_sell,
4)))
                else:
                    price_sell = price_selllist[j] - (((r-1)/(r +
math.factorial(r)))*price_selllist[j])
                    price_selllist[j] = price_sell
                    print("Seller price: {}".format(round(price_sell,
4)))
            for k in range(len(cons)):
                if(r <= 10):

```

```

        if(r==1):
            fixed_buy = (1 -
((abs(cons[k])** (1/2))/3))*Cgrid[x]
            fixed_buylist.append(fixed_buy)
            print("Fixed_buyer price:
{}".format(round(fixed_buylist[k],4)))
            price_buy = fixed_buylist[k] + (((r-1)/(r +
math.factorial(r)))*price_buy)
            price_buylist.append(price_buy)
            print("Buyer price:
{}".format(round(price_buylist[k], 4)))
        else:
            price_buy = price_buylist[k] + (((r-1)/(r +
math.factorial(r)))*price_buylist[k])
            price_buylist[k] = price_buy
            print("Buyer price:
{}".format(round(price_buylist[k], 4)))

price_buylisttemp = price_buylist
price_buylisttemp.sort()
trade_price = min(price_selllist)
for k in range(len(price_buylisttemp)):
    while(price_buylisttemp[k] > trade_price):
        trade_price = min(price_selllist)
        j = price_selllist.index(min(price_selllist))
        if(price_buylist[k] > trade_price and (prod[j] != 0
and cons[k] != 0)):
            print("\nPresale Prod: {}".format(prod[j]))
            print("Presale Cons: {}".format(cons[k]))
            if(prod[j] <= 0 and cons[k] >= 0):
                prod1 = prod[j]
                if(abs(prod[j]) < cons[k]):
                    traded = prod[j]
                    price = price_selllist[j]
                    cons[k] = cons[k] + prod[j]
                    prod[j] = 0
                    price_selllist[j] = 1000
                    print("Prod: {}".format(prod[j]))
                    print("Cons: {}".format(cons[k]))
                    print("kW Traded: {}".format(traded))
                    print("Price of Trade: {}".format(price))
                    print("\n")
                    sheet.cell(row=z+1, column=13).value =
x+1
                    sheet.cell(row=z+1, column=14).value =
prod[j]
                    sheet.cell(row=z+1, column=15).value =
cons[k]
                    sheet.cell(row=z+1, column=16).value =
traded
                    sheet.cell(row=z+1, column=17).value =
price
                    z = z+1
            else:
                traded = cons[k]*-1
                price = trade_price
                prod[j] = prod[j]+cons[k]

```



```

cons[k] = 0
price_buylist[k] = 1000
price_buylisttemp[k] = 0
print("Prod: {}".format(prod[j]))
print("Cons: {}".format(cons[k]))
print("kW Traded: {}".format(traded))
print("Price of Trade: {}".format(price))
print("\n")
sheet.cell(row=z+1, column=13).value =
x+1
sheet.cell(row=z+1, column=14).value =
prod[j]
sheet.cell(row=z+1, column=15).value =
cons[k]
sheet.cell(row=z+1, column=16).value =
traded
sheet.cell(row=z+1, column=17).value =
price
z = z+1
else:
    trade = 0
    r+=1
    print("\n")
    print("Remaining Prod: {}".format(prod))
    print("Remaining Cons: {}".format(cons))
    print("Remaining NET: {}".format(sum(prod)+sum(cons)))
    sheet.cell(row=z+1, column=18).value = sum(prod)+sum(cons)

    sale = True

    print("\n")
wb2.save("NET LOAD.xlsx")

```

C.3. Simulation Code for Grid-Connected Operation – Model II

```
import hashlib as hasher
import datetime as date
import math
import random
import numpy as np
from openpyxl import load_workbook
from openpyxl import Workbook

t=1

# -----
# DATA HANDLING
#
wb2 = load_workbook('NET LOAD.xlsx')

z=1

ws1 = wb2["Sheet2"]
ws2 = wb2["Sheet3"]
sheet = wb2.active
cell_range = ws1['A3':'I26']

#-----
# MG1

Load1 = []
for x in range(3,27):
    Load1.append((ws1.cell(row=x,column=2).value))
Load1 = list(np.around(np.array(Load1),4))

print("Load1: ")
print(*Load1)

Load2 = []
for x in range(3,27):
    Load2.append((ws1.cell(row=x,column=3).value))
Load2 = list(np.around(np.array(Load2),4))

print("\nLoad2: ")
print(*Load2)

Load3 = []
for x in range(3,27):
    Load3.append((ws1.cell(row=x,column=4).value))
Load3 = list(np.around(np.array(Load3),4))

print("\nLoad3: ")
print(*Load3)

Load4 = []
for x in range(3,27):
    Load4.append((ws1.cell(row=x,column=5).value))
Load4 = list(np.around(np.array(Load4),4))
```

```

print("\nLoad4: ")
print(*Load4)

Load5 = []
for x in range(3,27):
    Load5.append((ws1.cell(row=x,column=6).value))
Load5 = list(np.around(np.array(Load5),4))

print("\nLoad5: ")
print(*Load5)

Load6 = []
for x in range(3,27):
    Load6.append((ws1.cell(row=x,column=7).value))
Load6 = list(np.around(np.array(Load6),4))

print("\nLoad6: ")
print(*Load6)

Load7 = []
for x in range(3,27):
    Load7.append((ws1.cell(row=x,column=8).value))
Load7 = list(np.around(np.array(Load7),4))

print("\nLoad7: ")
print(*Load7)

Load8 = []
for x in range(3,27):
    Load8.append((ws1.cell(row=x,column=9).value))
Load8 = list(np.around(np.array(Load8),4))

print("\nLoad8: ")
print(*Load8)

# -----
# GLOBAL CONSTANTS & COUNTERS
#

blockchain = [create_genesis_block()]
previous_block = blockchain[0]
Cbat = 30
Csh = 500
Cres = 63
Ckw = 25
Cl = 80
Cpk = 63
Ppk = 1
Pbat = 0.3
Pmaxline = 1
Ccurt = 800
Cgrid = [26.84, 24.63, 21.21, 20.32, 20.14, 21.10, 24.62, 29.62, 31.94,
37.54, 42.26, 48.08, 52.71, 59.98, 62.62, 62.99, 67.50, 64.72, 57.09, 50.41,
47.77, 40.96, 36.71, 32.12]

t=1
r = 0

```

```

q=0
fp_buy = 5
fp_sell = 25
m=1
n=1
num_of_blocks_to_add = 24

for x in range(len(Load1)):

    print("\nHour # {}".format(x+1))
    traders = []
    prod = []
    cons = []
    traders.append(Load1[x])
    traders.append(Load2[x])
    traders.append(Load3[x])
    traders.append(Load4[x])
    traders.append(Load5[x])
    traders.append(Load6[x])
    traders.append(Load7[x])
    traders.append(Load8[x])

    for i in range(len(traders)):
        if(traders[i] < 0):
            prod.append(traders[i])
        elif(traders[i] > 0):
            cons.append(traders[i])
    print(prod)
    print(cons)
    alpha = 0
    price_buy = 8
    price_sell = 25
    r = 1
    q=1
    fixed_buylist=[]
    price_buylist=[]
    fixed_selllist = []
    price_selllist = []

    while(sum(prod) != 0 and sum(cons) != 0):
        r=1
        q=1
        j=0
        k=0
        trade = 0
        while(sum(prod) != 0 and sum(cons) !=0):
            j=0
            for j in range(len(prod)):
                sale = False
                if(abs(prod[j])<sum(cons)):
                    beta = 1
                else:
                    beta = 0
                if(abs(prod[j])>sum(cons)):
                    gamma = 1
                else:
                    gamma = 0

```

```

        if(r <= 100):
            if(r==1):
                fixed_sell = (1 +
((abs(prod[j]))**(1/2))/3))*Cgrid[x]
                fixed_selllist.append(fixed_sell)
                price_sell = fixed_sell
                print("Fixed_seller price:
{}".format(round(fixed_sell, 4)))
                price_selllist.append(price_sell)
                print("Seller price: {}".format(round(price_sell,
4)))
            else:
                price_sell = price_selllist[j] -
(price_selllist[j]*0.05*(abs(prod[j])) + Cbat*0.1*abs(prod[j]))
                price_selllist[j] = price_sell
                print("Seller price:
{}".format(round(price_selllist[j], 4)))

        for k in range(len(cons)):
            if(abs(sum(prod))<cons[k] or cons[k]>0.5):
                alpha = 1
            else:
                alpha = 0
            if(r <= 100):
                if(r==1):
                    fixed_buy = (1 -
((abs(cons[k]))**(1/2))/3))*Cgrid[x]
                    fixed_buylist.append(fixed_buy)
                    print("Fixed_buyer price:
{}".format(round(fixed_buylist[k],4)))
                    price_buy = fixed_buylist[k] +
fixed_buy*(cons[k]*0.05)
                    price_buylist.append(price_buy)
                    print("Buyer price:
{}".format(round(price_buylist[k], 4)))
                else:
                    price_buy = price_buylist[k] +
((Cpk*0.05*cons[k] + Csh*0.01*(cons[k]-0.05*cons[k]))
                    price_buylist[k] = price_buy
                    print("Buyer price:
{}".format(round(price_buylist[k], 4)))

        trade_price = min(price_selllist)
        count = 0
        #while(count <= 5):
        for k in range(len(price_buylist)):
            count += 1

            while(price_buylist[k] > trade_price):
                trade_price = min(price_selllist)
                j = price_selllist.index(min(price_selllist))
                if(price_buylist[k] > trade_price and (prod[j] <
0 and cons[k] > 0)):

                    print("\nPresale Prod: {}".format(prod[j]))

```

```

print("Presale Cons: {}".format(cons[k]))

if(prod[j] < 0 and cons[k] > 0):

    if(abs(prod[j]) < cons[k]):
        traded = prod[j]
        price = price_selllist[j]
        cons[k] = cons[k] + prod[j]
        prod1 = prod[j]
        pricesell1 = price_selllist[j]
        prod[j] = 0
        price_selllist[j] = 1000
        print("Prod: {}".format(prod[j]))
        print("Cons: {}".format(cons[k]))
        print("kW Traded: {}".format(traded))
        print("Price of Trade:

{}".format(price))

        print("\n")
        sheet.cell(row=z+1, column=13).value

        sheet.cell(row=z+1, column=14).value

        sheet.cell(row=z+1, column=15).value

        sheet.cell(row=z+1, column=16).value

        sheet.cell(row=z+1, column=17).value

        z = z+1
    else:
        traded = cons[k]*-1
        price = trade_price
        prod[j] = prod[j]+cons[k]
        cons[k] = 0
        #price_buylist[k] = 1000
        price_buylist[k] = 0
        print("Prod: {}".format(prod[j]))
        print("Cons: {}".format(cons[k]))
        print("kW Traded: {}".format(traded))
        print("Price of Trade:

{}".format(price))

        print("\n")
        sheet.cell(row=z+1, column=13).value

        sheet.cell(row=z+1, column=14).value

        sheet.cell(row=z+1, column=15).value

        sheet.cell(row=z+1, column=16).value

        sheet.cell(row=z+1, column=17).value

        z = z+1

elif(sum(prod) == 0 and cons[k] > 0):
    p = abs((abs(prod1)-cons[k])/prod1)
    p_res = p*abs(prod1)

```

```

tempSeller = Cres
p_resalt = 0.2*prod1
if((20*price_buylist[k])>tempSeller):
    traded = min(p_res, p_resalt)
    traded = traded*-1
    price = tempSeller
    cons[k] = cons[k] + traded
    prod1 = 0
    #price_selllist[j] = 1000
    print("Prod: {}".format(prod[j]))
    print("Cons: {}".format(cons[k]))
    print("kW Traded: {}".format(traded))
    print("Price of Trade: {}".format(price))
    print("Traded @ 2nd price")
    print("Z: {}".format(p))
    print("\n")
    sheet.cell(row=z+1, column=13).value =
x+1
    sheet.cell(row=z+1, column=14).value =
round(prod[j], 5)
    sheet.cell(row=z+1, column=15).value =
round(cons[k], 5)
    sheet.cell(row=z+1, column=16).value =
round(traded, 5)
    sheet.cell(row=z+1, column=17).value =
round(price, 5)
    sheet.cell(row=z+1, column=18).value = 1
    sheet.cell(row=z+1, column=20).value =
trade_price
    sheet.cell(row=z+1, column=21).value =
p_res
    z = z+1
else:
    trade = 0
    trade_price = 1000

wb2.save("NET LOAD.xlsx")

```

C.4. Simulation Code for Blockchain Settlement Protocol

```
import hashlib as hasher
import datetime
import math
import random
import datetime as date
from openpyxl import Workbook
import time
import numpy as np

filename = "consensus_16.xlsx"
workbook = Workbook()
sheet = workbook.active

class Block:
    def __init__(self, index, timestamp, data, previous_hash):
        self.index = index
        self.timestamp = timestamp
        self.data = data
        self.previous_hash = previous_hash
        self.hash = self.hash_block()

    def hash_block(self):
        sha = hasher.sha256()
        sha.update((str(self.index) + str(self.timestamp) + str(self.data) +
str(self.previous_hash)).encode())
        return sha.hexdigest()

def create_genesis_block():
    # Manually construct a block with
    # index zero and arbitrary previous hash
    return Block(0, date.datetime.now(), "Genesis Block", "0")

def next_block(last_block):
    this_index = last_block.index + 1
    this_timestamp = date.datetime.now()
    this_data = "kW:" + str(pnet) + " Price:" + str(0) + " Time:" +nowtime
    this_hash = last_block.hash
    return Block(this_index, this_timestamp, this_data, this_hash)

class Prosumer():
    def __init__(self, index, pgen, pload, pnet, status):
        self.index = index
        self.pgen = pgen
        self.pload = pload
        self.pnet = pnet
        self.status = status

    def vote(self):
        return 1

    def value(self):
        val = random.uniform(0,10)
        return val
```



```

# Create the blockchain and add the genesis block
blockchain = [create_genesis_block()]
previous_block = blockchain[0]
temp_buyer = []
temp_seller = []

j=0
k=0

# How many blocks should we add to the chain
# after the genesis block
num_of_blocks_to_add = 58
y=0
m=1
n=1
pgensum = 0
ploadsum = 0
while(y < num_of_blocks_to_add):
    prosumers = list()
    producers = list()
    consumers = list()
    for k in range(1, 8):
        pgen = round(random.uniform(5,15), 2)
        pload = round(random.uniform(5,15), 2)
        pnet = round(pload-pgen, 2)
        if(pgen > pload):
            status = 1
        else:
            status = 0
        prosumers.append(Prosumer(k, pgen, pload, pnet, status))

#Negotiate contract
have_contract = 0
f = 2
prepare = 0
commit = 0
values = list()
messages = 0
nowtime = date.datetime.now().strftime('%S.%f')[:-2]
sheet.cell(row=y+1, column=8).value = nowtime

#pBFT
while(prepare <= (2*f +1)):
    for i in range(0, len(prosumers)):
        prepare += prosumers[i].vote()
        time.sleep(random.uniform(0, 0.1))
        messages += 1

while(commit <= (2*f +1)):
    for i in range(0, len(prosumers)):
        commit += prosumers[i].vote()
        time.sleep(random.uniform(0,0.1))
        messages += 1

#Modified PoS
for i in range(0, len(prosumers)):

```

```

        values.append(prosumers[i].value())
    validator = np.argmax(values)

    nowtime = date.datetime.now().strftime('%S.%f')[:-2]
    block_to_add = next_block(previous_block)
    blockchain.append(block_to_add)
    previous_block = block_to_add
    print("Block #{} has been added to the
blockchain!".format(block_to_add.index))
    print("Data: {}".format(block_to_add.data))
    print("Hash: {}".format(block_to_add.hash))
    sheet.cell(row=y+1, column=3).value = block_to_add.index
    sheet.cell(row=y+1, column=4).value = block_to_add.data
    sheet.cell(row=y+1, column=5).value = block_to_add.hash
    sheet.cell(row=y+1, column=6).value = nowtime
    sheet.cell(row=y+1, column=7).value = messages

    y +=1

workbook.save(filename=filename)

```

C.5. Optimal Scheduling Problem Code

```

{string}T=...;    //time horizon
{string}G={"G1"};    //Thermal generating units
float L[T]=...;    //net load data
float F[G]=[61.3];    //operational cost ($/MW)
float Pmin[G]=[0.01]; //minimum power generation
float Pmax[G]=[.5];    //max power generation
float SU[G]=[15];    //startup for each unit
float SD[G]=[2];    //shutdown for each unit
float Eessmin= 2;    //Minimum energy capacity (MWh)
float Eessmax= 10;    //Maximum energy Capacity (MWh)
float Pdchmax=0.3;    //max discharging power for the battery
float Pdchmin=0.01;    //min discharging power for the battery
float Pchmin=0.01;    //min charging power for the battery
float Pchmax=0.3;    //max charging power for the battery
float CB=70;
float CC=1000;
float Cgrid[T] = [26.84, 24.63, 21.21, 20.32, 20.14, 21.10, 24.62, 29.62,
31.94, 37.54, 42.26, 48.08, 52.71, 59.98, 62.62, 62.99, 67.50, 64.72, 57.09,
50.41, 47.77, 40.96, 36.71, 32.12];

//float Cr=85;
//float SU[G]=[18,10,15]; //startup for each unit
//float SD[G]=[1.5, 1, 2]; //shutdown for each unit

// DECISION variables
//dvar float Pgrid[T];
dvar float+ P[G][T]; //generation from microgrid
dvar float R[T]; // Spining Reserve
dvar float Bss[T]; // battery Power
dvar float C[T]; // Battery Energy
dvar float Pnet[T];
dvar boolean u[G][T]; //commitment state of dispatchable units
dvar boolean y[G][T]; // startup variable
dvar boolean z[G][T]; //shut down variable
dvar boolean dl[T]; // DISCHARGING STATE VARIABLE
dvar boolean cl[T]; // CHARGING STATE VARIABLE

//objective function
dexpr float cost = sum(j in T)(sum (i in G) (F[i]*P[i][j]*u[i][j] +
y[i][j]*SU[i] + z[i][j]*SD[i]) + CB*Bss[j] + Cgrid[j]*R[j] +
CC*abs(Pnet[j]));

// model
minimize cost;

subject to {
// Power balance constraint
CT1: forall (i in G, j in T){
(sum(i in G)P[i][j]+Bss[j]+R[j]+Pnet[j]==L[j]);}

//generation limits
CT2:forall(i in G,j in T){
Pmin[i]*u[i][j]<=P[i][j];}

```

```

CT3: forall (i in G, j in T){
    L[j]<=Pmax[i]=>Pmax[i]*u[i][j]>=P[i][j];
    L[j]>=Pmax[i]=>Pmax[i]*u[i][j]>=P[i][j];}

CT5:forall(i in G, j in T:j!=first(T)){
    y[i][j]-z[i][j]==u[i][j]-u[i][prev(T,j)];}

CT6:forall (i in G, j in T){
    y[i][j]+z[i][j]<=1;}

//Battery limits
//max discharge and min charge of MG1
CT7:forall (j in T){
    Bss[j]<=((Pdchmax*d1[j])-(Pchmin*c1[j]));}

//min discharge and max charge of MG1
CT8: forall (j in T){
    Bss[j]>=((Pdchmin*d1[j])-(Pchmax*c1[j]));}

//charging and discharging state of ESS for MG1
CT9: forall (j in T){
    d1[j]+c1[j]<=1;}

//ESS state of charge for MG1
CT10:forall (j in T:j!=first(T)){
    C[j]==C[prev(T, j)]-Bss[j];}

//ESS capacity constraints for MG1
CT11:forall(j in T){
    C[j]<=Eessmax;
    C[j]>=Eessmin;}

CT12: forall(j in T){
    abs(R[j]) <= 0.5;}
}

```