Blockchain based energy transactions for a prosumer community.

by

Nikita Karandikar

A dissertation submitted in partial satisfaction of
the requirements for the degree
PHILOSOPHIAE DOCTOR (PhD)

University of
Stavanger

Faculty of Science and Technology
Department of Electrical Engineering and Computer Science
September 2021

University of Stavanger
N-4036 Stavanger
NORWAY
www.uis.no

ISBN 978-82-8439-018-5
ISSN 1890-1387

PhD Thesis UiS no. 599

Dedicated to my dear mother,
Shilpa Karandikar

Acknowledgements

I would like to express my gratitude to my supervisor, Dr. Antorweep Chakravorty for his supervision, support and encouragement during the course of my research. I am especially grateful for his insightful comments that greatly improved the quality of this study. My sincere thanks also go to my co-supervisor Prof. Dr. Chunming Rong for his comments and suggestions on the work presented in this dissertation.

I acknowledge with gratitude, the financial and consulting support I received from the Research Council of Norway, International Research Institute of Stavanger (IRIS) as well as partners Lyse, DNV GL and Statnett.

Additionally, I thank my colleagues at UiS, Dr. Jayachander Subiryala, Dr. Aida Mehdipourpirbazari, Dhanya Therese Jose, Dr. Faraz Barzideh and Albana Roci for their valuable support and friendship during the course of my research.

I especially thank my mother and my sister for support and encouragement during and beyond the period of this research. Finally, I am grateful to my wonderful husband Dr. Rockey Abhishek for his help and support every step of the way. The completion of this work would not have been possible without them.

Nikita Karandikar, September 2021

Preface

This dissertation is submitted in partial fulfillment of the requirements for the degree of Philosophiae Doctor (PhD) at the University of Stavanger, Norway. The work presented in this thesis was based on research during February 2018 to May 2021 inclusive. Four published research papers form the basis of this dissertation. The publications are reformatted to align with the thesis pagination. The articles are self-contained.

Abstract

Integration of solar micro-generation capabilities in domestic contexts is on the rise, leading to the creation of prosumer communities who generate part of the energy they consume. Prosumer communities require a decentralized, transparent and immutable transaction system in order to extract value from their surplus energy generation and usage flexibility. The aim of this study is to develop frameworks and methods to create such a prosumer transaction system with self enforcing smart contracts to facilitate trading of energy assets such as electricity units, energy flexibility incentives and storage credits.

Blockchain is a transparent, distributed ledger for consensus based transaction processing maintained by a network of peer nodes. Hyperledger Fabric is a blockchain platform that offers the added benefits of lower operating cost, faster transaction processing, user authentication based access control and support for self enforcing smart contracts.

This thesis investigates the applicability of Hyperledger Fabric to tokenize and transact energy assets in a unified transaction system. Data driven approaches to implement an incentive based energy flexibility system for peak mitigation on the blockchain are also investigated.

To this end, the stakeholders for such a transaction management system were identified and their business relationships and interactions were described. Energy assets were encapsulated into blockchain tokens and algorithms were developed and encoded into self enforcing smart contracts based on the stakeholder relationships. A unified transaction framework was proposed that would bring on board all the stakeholders, their trading relationships and the assets being transacted. Tokens and methods in the transaction system were implemented in fungible and non fungible versions and the versions were critically compared in terms of application area, design, algorithmic complexity, performance, advantages and disadvantages. Further, with a focus on energy flexibility applications, a prosumer research dataset was analysed to gain insights into the production and consumption behaviors. Based on these insights, a data driven approach for peak mitigation was proposed and implemented on the

Hyperledger Fabric blockchain.

The thesis thus addresses different aspects of a blockchain based prosumer transaction system, and shows the feasibility of proposed approaches through implementation and performance testing of proofs of concept.

Contents

Acknowledgements **iv**

Preface **v**

Abstract **vi**

List of Figures **xiii**

List of Tables **xiii**

List of Papers **xv**

1 Introduction **1**
1.1 Blockchain applications to prosumer communities . . 1
1.2 Problem Description and Motivation 4
1.3 Research Objective and Questions 6
1.4 Research publications 8
1.5 Thesis Outline 10

2 Background **11**
2.1 Precursors to Blockchain 11
2.2 Blockchain 13
2.3 Hyperledger Fabric 13
2.3.1 Key Concepts in Hyperledger Fabric 14
2.3.2 Hyperledger Fabric Transaction Flow 16
2.4 Hyperledger Caliper 19

3 Contributions **21**
3.1 Overview 21

3.2 Paper I 22
3.3 Paper II 26
3.4 Paper III 30
3.5 Paper IV 38

4 Conclusions and Future Work **47**
4.1 Conclusions 47
4.2 Future Work 48

Paper I: Transactive energy on Hyperledger Fabric **57**
1 Introduction 60
1.1 Microgrids 60
1.2 Smart Grids 60
1.3 Blockchain 61
1.4 Hyperledger Fabric 63
2 Hyperledger Fabric Architecture 63
2.1 Key Components of Hyperledger Fabric 63
2.2 Transaction flow in Hyperledger Fabric 64
3 Proposed Architectures 66
4 Use cases 70
4.1 Value To Prosumers 70
4.2 Value To DSO 70
4.3 Value To prosumer community 72
4.4 Value To EVs 74
5 Related Works 75
6 Conclusion 77

Paper II: RenewLedger : Renewable energy management powered by Hyperledger Fabric **83**
1 Introduction 86
2 Overview of System 88
2.1 Application entities 88
2.2 Architecture 89
3 Implementation 91
3.1 Tokens 91
4 Experiments 93
4.1 Setup 93

4.2 Transaction flow and Performance 94
5 Related Works 101
6 Conclusion 102

Paper III: Blockchain-based prosumer incentivization for peak mitigation through temporal aggregation and contextual clustering. 105
1 Introduction 109
2 Proposed system 111
2.1 System participants and requirements 111
2.2 Data driven approach 113
3 Ausgrid Dataset and Analysis 114
3.1 Ausgrid Dataset Overview 114
3.2 Aggregated Energy Profile for all customers . 116
3.3 Seasonal Energy Profile for all customers . . . 118
4 Smart energy transaction analytic 122
4.1 Semantic linking and contextualisation 123
4.2 Contextual clustering and labelling 127
4.3 Cross-contextual similarity 130
5 Blockchain based reward system 130
5.1 Smart contract 132
5.2 Implementation 135
5.3 Results 136
6 Related Works 137
7 Conclusion 139

Paper IV: Blockchain based energy management system with fungible and non-fungible tokens. 147
1 Introduction 150
2 System Overview 155
2.1 System Participants and Tokens 155
3 Design and Implementation 158
3.1 Structure of the Token 159
3.2 Design Requirements 160
3.3 Implementing Energy Assets as NFT 162
3.4 Implementing Energy Assets as FT 168
3.5 Complexity of the Algorithms 173

3.6 Token lifecycle 177
4 Experimental Setup, Results and Discussion 179
4.1 Experimental Setup 180
4.2 Results and Discussion 182
4.3 Comparison of Non Fungible Tokens and Fungible Tokens 188
4.4 Limitations and Future Work 191
5 Related Works 192
6 Conclusions 196

List of Figures

1.1 Relation between appended papers and research questions 8

2.1 Transaction flow in Hyperledger Fabric 17

3.1 Interconnections between papers 22
3.2 Orderer nodes in a separate location [35] 24
3.3 Orderer nodes collocated with peer organizations [35] 25
3.4 Relationship between organizations and tokens [36] . 28
3.5 Schematic outline of the system. [37] 32
3.6 Locations of customers [37] 32
3.7 Structure of transaction token [37] 36
3.8 Experimentation testbed [37] 37
3.9 Structure of token [38] 40
3.10 Lifecycle for Fungible Tokens (FT) and Non Fungible Tokens (NFT). [38] 42
3.11 Sequence of experiments [38] 45

List of Tables

3.1 Simplified example of dataset 34
3.2 Simplified example of multilayer structure 34

List of Papers

The following papers are included in this thesis:

- **Paper I**

  Transactive energy on Hyperledger Fabric
  N. Karandikar, A. Chakravorty, C. Rong
  Published in the proceedings of 2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)

- **Paper II**

  RenewLedger : Renewable energy management powered by Hyperledger Fabric
  N. Karandikar, A. Chakravorty, C. Rong
  Published in the proceedings of 2020 IEEE Symposium on Computers and Communications (ISCC)

- **Paper III**

  Blockchain-based prosumer incentivization for peak mitigation through temporal aggregation and contextual clustering.
  N. Karandikar, R. Abhishek, N. Saurabh, Z. Zhao, A. Lercher,

N. Marina, R. Prodan, C. Rong, A. Chakravorty
Published in the journal Blockchain: Research and Applications, Elsevier Publications

- **Paper IV**

Blockchain based energy management system with fungible and non-fungible tokens.
N. Karandikar, A. Chakravorty, C. Rong
Published in the special issue Blockchain Applications in Smart Energy Grids of the journal Sensors, MDPI Publications

Chapter 1

Introduction

This chapter presents an introduction into the thesis and is structured as follows. Section 1.1 explains the need for a decentralized transaction system for a community based renewable energy infrastructure and proposes the use of blockchain as the platform for creating such a system. Section 1.2 presents the research problem and the motivations of the study. In section 1.3, the primary objective of the research is described along with the research questions this thesis investigates. In section 1.4, a list of the research articles is presented and finally in section 1.5 an outline of the thesis is provided.

1.1 Blockchain applications to prosumer communities

Solar energy micro generation is being increasingly adopted by residential consumers, partly due to economic benefits such as savings on energy bills and government subsidies [1] and partly due to rising awareness of the environmental benefits [2]. Energy consumers who generate a portion of the energy they consume and buy from the grid as needed are called prosumers [3]. Prosumers living in close proximity to each other can organize into prosumer communities or microgrids [4] and create a local market for sale and purchase of surplus renewable energy. Prosumer communities can also explore alternatives for community level storage of energy using the services

of storage providers who take over the logistics of setting up and maintaining a large scale energy storage facility and abstract away the intricacies of these activities from the end users [5]. Prosumers are billed based on the storage they use, thus effectively receiving energy storage capacity as a service. Storing energy in such a community level storage will facilitate transactions between members of the community as it reduces the need for wired connections between all community members. Additionally, Electric vehicles (EV) also represent additional customers for renewable energy. This community level energy storage can function as EV charging stations allowing for more opportunities for monetizing on the surplus energy. Including EVs in the prosumer community can allow for EV batteries to be rented out when not in use to add to the storage capacity and energy flexibility of the community, thus benefiting both parties. Mahmud et al. [6] studied the usage of a community level storage facility for EV charging and proposed a decision tree based algorithm for reduction of peak load through management of EVs, microgeneration and community level energy storage facilities.

Peak demand periods are a challenge for the power companies, who may need to over-provision generation capacity in order to ensure grid stability thus increasing the marginal cost of electricity [7]. Thus, grid operators can benefit from peak shaving mechanisms such as demand response [8]. Demand response for a prosumer community can be implemented by incentivizing prosumers for energy flexibility by offering them reward tokens. As the prosumers are collocated, considering them as a prosumer community would allow the power company to calculate the required flexibility of the community as a whole [9] as well as to increase engagement by creating game based energy flexibility tasks [10].

Prosumers, EV owners, Power Companies and Storage Providers together form a business network due to their transactive relationships with one another. Such a business network includes several small scale prosumers and EV owners and thus must be decentralized in order to prevent the decision making from being concentrated in the hands of a centralized party. A system that encapsulates this business network is required in order to facilitate transactions between the members. Such a transaction system must be transparent, immutable

and enable provenance tracing of assets transacted on the network in order to be trustworthy. Business relationships must be encapsulated in agreed upon self enforcing contracts in order to automate and facilitate transactions.

Blockchain [11] is a decentralized immutable ledger that satisfies these requirements as it prevents any member from unilaterally processing transactions or making decisions on the network [12, 13]. Blockchain was created in 2008 as the ledger for Bitcoin, a new cryptocurrency, but due to its interesting properties has been found to be applicable to other areas as well [14]. Blockchain is implemented as an append-only decentralized ledger where each node in a blockchain network maintains an identical copy of the ledger and nodes must seek consensus before adding any transactions to the ledger. Transactions are maintained in blocks such that each block contains the hash of the previous block, thus making the ledger verifiable. Provenance of any asset can be readily traced by reviewing its transaction history on the ledger. As ledger is duplicated at every node of the network, transactions are transparent to all members. Due to its distributed nature, and the cryptographic linking of blocks, the ledger is also immutable as any unilateral attempts to change the transactions will create inconsistency and thus be rejected by the network. Double spending is a concern for digital assets that can be replicated and unscrupulous parties may attempt to spend the same asset multiple times. This problem is mitigated by the use of a decentralized and immutable ledger that requires consensus for each transaction. In Bitcoin, consensus is achieved by implementing a cryptography based mechanism called Proof -of- Work that requires nodes to complete extensive computations requiring considerable resources in order to add a new block of transactions.

Blockchain networks can be broadly categorized as permissioned or permissionless. Permissionless networks such as Bitcoin and Ethereum [15] are open to anyone to join and propose transactions. Permissioned networks, such as Hyperledger Fabric [16] only permit authenticated parties to join and participate in the network. Identity based access control mechanisms as well as traceability can be implemented to define each nodes' privileges in the network as well as introduce accountability as the invoker for each transaction is known. Self

enforcing smart contracts can be created to encapsulate agreed upon business logic in order to automate transactions.

As the network participants are identified and authenticated, resource intensive mechanisms such as Proof of Work are not required. Operating cost of the system is thus reduced and the need to implement cryptocurrency to incentivize nodes to process transactions is removed. Tokens and assets in Hyperledger Fabric are not restricted to cryptocurrency but can be used to represent anything of value. Authentication of participants of financial transactions is required due to Know your Customer (KYC) [17] [18] and anti money laundering (AML) regulations. Moreover, Hyperledger Fabric has the concept of organizations that can be used to reflect the real world network participants and the relationships between them. Encapsulating business relationships and participants can provide a unified method of transaction, provenance tracing and identity management. Hyperledger Fabric has a modular architecture due to which the trust models, consensus mechanisms and transaction format can be chosen and implemented as suitable for the specific application. Gur et al. [19] implemented an energy metering system for smart grids with privacy protection on the Hyperledger Fabric. Che et al. [20] used Hyperledger Fabric to implement an authentication focused prosumer transaction system. Thus, the features of Hyperledger Fabric make it especially suited for developing a peer to peer energy transaction system.

1.2 Problem Description and Motivation

The integration of blockchain with a community based energy infrastructure allows the parties involved to transact energy assets such as energy units, reward tokens and storage credits with each other in a transparent and decentralized manner. Due to the decentralized nature of the proposed system, small scale prosumers will also be involved in the decision making along side the power company and the storage provider. Due to the relationships between the stakeholders described in section 1.1, it is essential to create a unified transaction system that brings on board all the stakeholders and

provides a platform to facilitate transactions between them.

First, the relationships among the stakeholders must be well understood through the interactions involved in transactions among them. Stakeholders must be then grouped into organizations based on the relationships and energy assets must be tokenized for transactions based on use case. The network and token architecture needs to take into account the requirements of the transaction platform as well as the characteristics of the Hyperledger Fabric blockchain platform.

Blockchain tokens can be broadly classified as fungible tokens (FT) and non fungible tokens (NFT). FT only encapsulate value and are thus interchangeable and can be broken up and traded in parts. NFT encapsulate value as well as unique information such as an identifier, due to which they are not interchangeable. Energy assets that have a Guarantee of Origin [21] have a unique identifier and are not interchangeable and can thus be represented as NFT. However, energy assets that do not encapsulate any unique information can be represented as FT. Both token types are relevant to energy transaction systems. Due to the different characteristics of each token type, type specific methods and algorithms would be needed to take the tokens through the lifecycle. A comparative analysis of both types of tokens in terms of design, implementation, algorithmic complexity, performance, limitations, advantages and disadvantages is essential when considering adoption and can provide a guide for future implementations of a blockchain based transaction system.

Section 1.1 identified peak shaving as an important consideration for power companies. Analysis of real world data sets published by power companies can offer insight into the production and consumption behavior which can guide peak shaving strategies. Extraction and analysis of aggregated user energy production and consumption behavior in temporal contexts as well as semantic linking and contextual clustering of the data can identify different aspects of peak net consumption periods. Based on this analysis, a data driven approach for peak mitigation can be proposed that incentivizes prosumers for low consumption during identified peak periods. This prosumer incentivization system can be implemented on the Hyperledger Fabric blockchain and the smart contract logic can be based on the analysis

performed.

Several works [22] [23] [24] study different aspects of design and implementation of peer to peer energy transaction systems. A large number of these works use Ethereum as the blockchain platform and employ approaches specific to the characteristics of that platform. Furthermore, an important focus of this work is to create frameworks and transaction logic for use case determined tokenization of energy assets. However, only a few works found studied these aspects of a blockchain based energy transaction system. Tokenization is a way to encapsulate energy assets into blockchain tokens and it thus is an important aspect of facilitating energy transactions on the blockchain. Therefore this work attempts to address these gaps found in literature.

1.3 Research Objective and Questions

The primary objective of this work is to develop frameworks, methods and algorithms to create a decentralized and transparent energy transaction system with self enforcing smart contracts for prosumer communities. We propose the use of blockchain, a decentralized and transparent ledger to encapsulate energy assets as well facilitate and record transactions. However, in addition to decentralization, such a prosumer system would have several requirements.

Firstly, the underlying transaction platform must support the encapsulation and encoding of the identified relationships among the stakeholders, the energy assets to be transacted among them as well as the business logic and rules for transaction. These relationships and the business logic of transactions must be agreed upon by all stakeholders and changes to the logic must also be mutually agreed. Implementation and performance testing of the proposed approaches is essential in order to show their feasibility. The approaches developed must be sufficiently general for adaptation into future implementations. An investigation into the suitability of blockchain to satisfy all these requirements is thus essential.

Secondly, for peak mitigation through incentive driven usage flexibility, we proposed a blockchain based reward system. Analysis of prosumer consumption and production patterns from a real world

dataset could be exploited to create the reward logic of the incentive system in a data centric approach. Studying such a dataset, creating business logic, encoding the business logic into smart contracts and performance testing the implementation would not only offer an insight into the different consumption and production behaviors of interest, but also delineate some of the challenges of such a system and evaluate whether the blockchain implementation can address these challenges.

Thirdly, as assets on the blockchain would be encapsulated in tokens which would be manipulated by transactions, research into tokenization and transfer of value using the blockchain is valuable. In a unified system, there would several assets of different types, owned by different entities. Each asset type would be subject to different business logic and would involve a different subset of stakeholders. Each individual asset would have an owner who must be the only entity authorized to redeem the value in the asset or initiate transfer to another entity. The design of the token, its lifecycle and methods must reflect the real world requirements for storage, access and manipulation of the encoded assets.

Based on the objectives of the study, we investigate the following research questions.

(1) RQ 1. Is blockchain applicable to address transaction management for energy trading, flexibility and storage?

(2) RQ 2. What data driven approaches can be adopted to implement a blockchain based prosumer incentivization system for peak mitigation?

(3) RQ 3. How can blockchain be used to tokenize and transact multiple types of energy assets in a unified system?

The approaches developed in this work would be useful to address challenges in peer to peer transaction systems for prosumer communities, power companies and storage providers.

1.4 Research publications

Research on the identified research questions was organized into four publications as shown in figure 1.1.

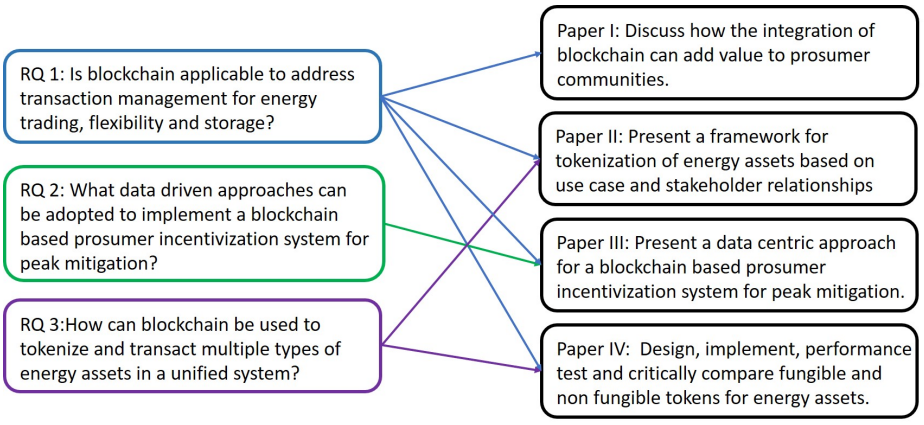


Figure 1.1: Relation between appended papers and research questions

All four papers have been published. Paper I identified the need for a transaction system for energy assets in a prosumer community and discussed the applicability of blockchain as the platform on which to build such a system. Paper II identified the stakeholders and organizations in the business network and defined blockchain tokens to represent the different types of energy assets being transacted. Paper III analysed a popular research dataset for energy transactions in order to identify patterns in consumption and production behaviour and used the findings to create business logic to implement a blockchain based prosumer incentivization system for peak mitigation. Paper IV identified FT and NFT as the two broad categories of blockchain tokens and designed and implemented both versions in order to compare them in terms of use case, design, performance, algorithmic complexity, advantages and disadvantages. Below is a brief description of the research publications.

- Paper I "Transactive energy on hyperledger fabric." was published in the proceedings of the 2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS). IEEE, 2019.

In this paper we discussed the applicability of blockchain, specifically Hyperledger Fabric to add value to a community based renewable energy infrastructure. We proposed and compared two architectures for placement of the different member nodes of the Hyperledger Fabric network. Finally, we presented the interactions that would occur between the transacting parties for transacting each type of energy asset.

- Paper II "RenewLedger: Renewable energy management powered by Hyperledger Fabric." was published in the proceedings of the 2020 IEEE Symposium on Computers and Communications (ISCC). IEEE, 2020.

  In this paper we divided the stakeholders into organizations and defined blockchain tokens mapped to each type of energy asset to be transacted. We designed, implemented and performance tested a proof of concept of the proposed framework on the Hyperledger Fabric.

- Paper III "Blockchain-based prosumer incentivization for peak mitigation through temporal aggregation and contextual clustering." was published in the journal Blockchain Research and Applications, Elsevier Publications.

  This paper analyses a popular energy consumption and production dataset published by the Ausgrid corporation. Aggregated user energy behavior was extracted in temporal contexts and semantic linking and contextual clustering was performed. Insight into consumption and production patterns obtained from this analysis was translated into business logic to incentivize prosumers for reduction in net consumption during identified peak periods. This business logic was encoded into smart contracts and a proof of concept prosumer incentivization system was built on the Hyperledger Fabric blockchain, which demonstrated the applicability of the proposed data driven approach to this problem.

- Paper IV "Blockchain Based Transaction System with Fungible and Non-Fungible Tokens for a Community-Based Energy

Infrastructure" was published in the special issue "Blockchain Applications in Smart Energy Grids" of the peer reviewed journal Sensors, MDPI publications.

This paper presents a comparative study on FT and NFT for energy assets. First, the stakeholder relationships were encoded into the trust model and the energy assets to be transacted based on these relationships were encapsulated as blockchain tokens. Methods and algorithms were developed to manage the lifecycle of the tokens. A proof of concept was developed and performance tested on the Hyperledger Fabric implementing the methods as smart contracts. Based on the experimental evaluation and analysis the two implementations were critically compared in terms of design, algorithmic complexity, performance and limitations.

1.5 Thesis Outline

The remainder of this thesis is organized as follows. Chapter 2 presents a background of technologies and methodologies used in this thesis. Chapter 3 summarizes and briefly discusses the research articles presented, while Chapter 4 concludes the discussion and presents avenues for future research. The research papers are placed at the end of the thesis.

Chapter 2

Background

This chapter provides a context for the papers presented in the thesis. The concept of Blockchain in broadly explained in the context of its history, and the merits of Hyperledger Fabric, the blockchain platform used in this work are highlighted. The key concepts and components and a typical transaction flow in a Hyperledger Fabric network are explained. Hyperledger Caliper, the tool used for the performance testing experiments in this work is also briefly described.

2.1 Precursors to Blockchain

The concepts used in modern blockchain were first found to be used in circa fifteenth century by a community in Micronesia, an archipelago in the South Pacific Ocean. The inhabitants of the island of Yap used stones called Rai stones [25] as money. These stones were valued based on their size, history and difficulty of excavating and thus, tended to be enormous in size and weight. These were rarely moved because of their weight and the risk of damage. Thus, the inhabitants needed to trace the value exchanges and ownership of the stones without physically moving them. A ledger was thus started that noted who owned a given stone. The owner of a stone, thus had a reason to quickly update the ledger and tell others that they owned the stone, in order to prevent another person from doing so first. Thus, the concepts of decentralization through a shared ledger were

already in use in this ancient society.

Moving forward to the 20th century, David Chaum, a cryptography professor in his paper [26] published in 1983, introduced a cryptography based protocol for an automated payments system that made it possible to conduct payments without revealing to third parties except under exceptional circumstances, the payee, time or amount of the transaction while still allowing individuals to provide proof of payment. Moreover, payments could be stopped if the value was shown to be stolen. Thus Bitcoin concepts of anonymity of transacting parties were investigated in this work.

Then, in 1991, researchers Stuart Haber and W. Scott Stornetta, proposed a procedure [27] for certifying the creation or modification time of a digital document. Their method made it infeasible for a user even with the collusion of a time-stamping service to either back-date or forward-date a document. The procedure did not require record keeping by the time stamping service and preserved the privacy of the documents themselves. Thus, they proposed a method to tackle the problem of tampering in digital records. The foundations of tamper proofing, that are used in modern blockchains were laid out in this work.

A decade later, David Mazières and Dennis Shasha in their paper [28] published in 2002, proposed a protocol for a multi user network file system for detection of tampering attacks and stale data. They introduced the concept of fork consistency for data integrity, where if the server delays a user from viewing a single change by another user, the two users would not see each other's changes anymore. This would be readily detected with online communication. These concepts created the framework for modern blockchain.

Another breakthough came in 2005 when Nick Szabo proposed BitGold [29], one of the earliest decentralized cryptocurrencies. He introduced innovations such as "client puzzle function" and "proof of work function" and time stamping of transactions, but could not overcome the problem of double spending, wherein digital currency users could issue duplicate transactions, spending the same digital currency multiple times.

Finally, in 2008, an anonymous inventor, only known by the pseudonym Satoshi Nakamoto introduced Bitcoin [11], the world's

first successful cryptocurrency. They succeeded in solving the double spending problem by implementing a distributed network of peers generating computational proof for the order of transactions. The underlying decentralized, immutable ledger for Bitcoin was called "block chain", which is now usually written as "blockchain".

2.2 Blockchain

Blockchain [11] was created in 2008 as a ledger for Bitcoin transactions and Bitcoin remains its most well known implementation. Blockchain is a distributed shared ledger, such that nodes in the blockchain network maintain identical copies of the ledger, thus requiring nodes to seek consensus before adding any transactions to the ledger. The blocks of transactions are maintained in the form of a chain such that each block contains the hash of its previous block, which makes it not only verifiable but also makes it so that any attempts to change the order of transactions would create a snowball effect of inconsistency and be thus rejected by the network. Blockchain was envisioned as a public ledger where anyone could perform transactions and eliminated the need for trust or a central authority. It accomplished this by implementing a cryptography based consensus mechanism called Proof of Work that required nodes in the network to expend considerable resources in order to add a block of transactions to the blockchain. The success of Bitcoin and the interesting features of its underlying technology have piqued the interest of research and industry alike and blockchain, as envisioned initially or a modified understanding of the original technology has been found applicable to many domains and use cases [30]. The next section explains why Hyperledger Fabric was chosen as the blockchain platform for the works presented in the thesis. Key concepts and the transaction flow of Hyperledger Fabric are also described.

2.3 Hyperledger Fabric

In order to adapt blockchain to enterprise use, several additional requirements must be satisfied. For business applications, financial

transactions are subject to KYC and AML regulations [17] [18]. Due to this, the identification and authentication of participants is a requirement. Moreover, business transactions need to be deterministic, so that transactions posted are final and correct. Several blockchain implementations, rely on probabilistic consensus algorithms that have a high probability of eventual ledger consistency but are vulnerable to the possibility of divergent ledgers also known as ledger forks where different members may have different accepted order of transactions [31]. Hyperledger Fabric implements deterministic consensus so that any block validated and committed is final, so the possibility of ledger forks is eliminated.

Consensus mechanisms such as proof of work have a prohibitively high operation cost which may be untenable for a business application. Moreover, in a business application as all participants must be authenticated, consensus mechanisms like proof of work that were designed to offset the risks due to anonymity are no longer necessary. Additionally, blockchain implementations such as Bitcoin are notorious for the low transaction throughput they offer [32]. In enterprise applications where users are accustomed to high transaction throughput with lower latency offered by databases, this may be a concern. Hyperledger Fabric uses a consensus mechanism based on member nodes endorsing transactions where the required endorsements are dictated by the business relationships. Moreover, Hyperledger Fabric can offer a much higher throughput and lower latency than many other blockchain platforms [33]. The modular architecture of Hyperledger Fabric allows for aspects such as consensus mechanisms and transaction formats to be chosen by the blockchain operator. Additionally, self enforcing smart contracts are supported in popular general purpose languages such as Java, Golang and Node.js. Due to these features, all the works presented in this thesis were developed using Hyperledger Fabric with the smart contracts were written in Golang.

2.3.1 Key Concepts in Hyperledger Fabric

Peer nodes in Hyperledger Fabric are grouped into organizations which intuitively mirror organizations in the real world. Organizations in a

Hyperledger Fabric network are members of one or more channels. Channels are a sub network within the network that include member organizations that have a business relationship with one another. Transaction data in a channel is isolated from other channels and is only visible to channel members.

Peers store a copy of the ledger for each channel of which they are members. The ledger consists of two interlinked but separate parts, namely the world state and the blockchain. The world state is a database of key value pairs that stores the values corresponding to the current ledger state. These key value pairs represent the current state of attributes of the business objects or assets being transacted on the network. The world state is modified by transactions that occur in the channel. Transaction logic for defining, creating, modifying, reading or deleting assets is encoded in the chaincode. A chaincode can have several smart contracts that contain the executable code for transactions. A chaincode must be instantiated on a channel and approved by the member organizations before use. The blockchain stores the immutable transaction log that resulted in the ledger state. Thus, the blockchain can be used to establish provenance of any asset.

Aside from peers, organizations can have users or clients who propose transactions without needing to host the ledger. In order to access the network resources, a client identity would need to be created for each client. Hyperledger Fabric is a permissioned blockchain platform with support for access control. Access to network resources and data is governed by identities encoded in an X.509 digital certificate. Certificate authorities of each organization issue identities to its members by generating a key pair of public and private keys for each member. A peer would use its private key to sign a transaction proposal which is matched to the corresponding public key by the membership service provider to check the authenticity of the digital signature.

Clients of an organization can create transaction proposals for modifications to the world state. However, the transaction would need to be simulated and endorsed by other members of the network in order to achieve consensus in this decentralized system. Endorsement policies stipulate which and how many members of the network must simulate and endorse transactions. Endorsement policies can be

specified at the channel level, the chaincode level as well as at the key level. The chaincode level endorsement policy overrides the channel level policy for a given chaincode. Similarly, the key level endorsement policy overrides the chaincode level for a given key - value pair in the world state. As the key value pair represents a real world asset it is also called a blockchain token for that asset and its associated endorsement policy is also known as token level endorsement policy.

Once a transaction proposal receives the required number of endorsements, the transactions must be placed in the correct order and divided into blocks. Hyperledger Fabric uses separate nodes for providing the ordering service. In this thesis, the ordering service nodes are implemented as members of a separate organization called the Orderer organization. The orderer organization is administered by the certificate authorities of the peer organizations. The recommended orderer service implementation in Hyperledger Fabric is Raft [34], which is a crash fault tolerant ordering service based on the Raft protocol. Raft has a leader-follower model where the followers replicate the decisions of the elected leader node. Raft can continue to function even when nodes, including leader nodes are lost, while the majority of the ordering nodes (quorum) continue to function. These components of the Hyperledger Fabric network participate in the transaction flow that is explained next.

2.3.2 Hyperledger Fabric Transaction Flow

The transaction flow in Hyperledger Fabric begins when a client of an organization creates a transaction proposal and sends it to the peers. Some peers only maintain the ledger and do not participate in endorsement. Such peers are called committing peers to distinguish them from endorsing peers. Figure 2.1 shows the steps involved in the transaction flow.

The Fabric peers receive the transaction proposal for endorsement. The peers check to see that the proposal is correctly formed, that it is not a duplicate proposal, that the client signature is valid and that the client submitting the transaction is authorized to perform the requested transaction. The peer then uses the values from the transaction proposal as input parameters for the requested chaincode

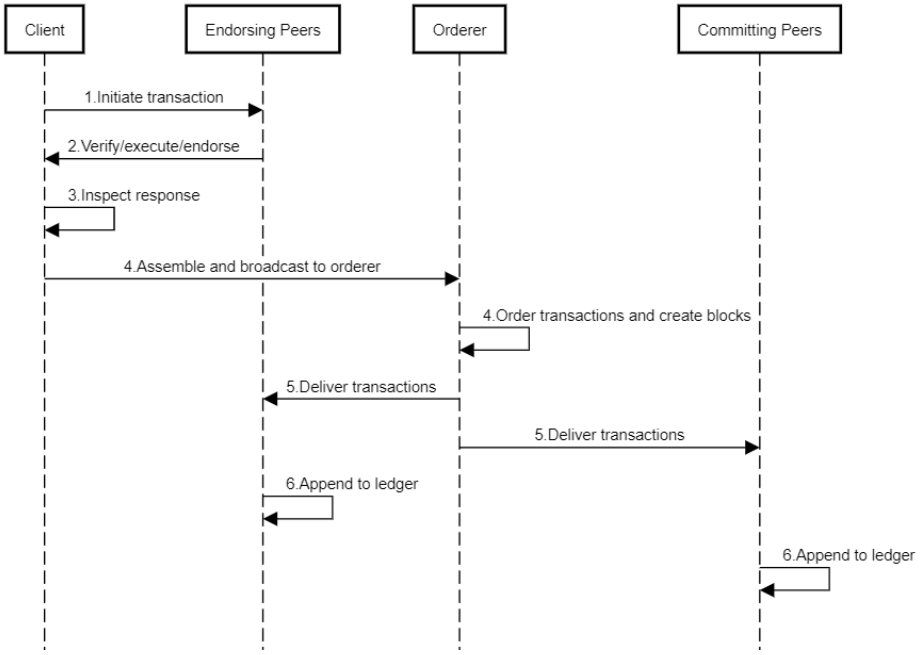


Figure 2.1: Transaction flow in Hyperledger Fabric

and executes the requested operation against the world state and generates a read/write set and a response value. The peer constructs a proposal response using these generated values, signs it with its unique credentials and sends it back to the client. The client inspects the response. If the request was a read operation, the transaction will not be written to the ledger (neither the world state not the blockchain). If the operation requested was a write, the client checks to see if the proposal has received sufficient endorsements as per the endorsement policy. The Client then sends the transaction proposal with the endorsements and the Channel ID to the orderer. The orderer sorts the received transactions in each channel in a chronological order and creates blocks of transactions. These blocks are then sent to the leader peer in each organization which sends them to the other peers in the organization using gossip protocol. The peers validate the transactions contained in the block to ensure that the endorsement policy was satisfied and that the read/write set is unchanged since it was generated.

Transactions are marked as valid or invalid at this stage. Transactions may be marked as invalid due to collision. Hyperledger Fabric implements multi version concurrency control to prevent double spending. Due to this, if two uncommitted transactions modify the same key in the world state, a collision is created and at most one modification will be marked as valid.

Consider three individuals, Alice, Bob and Charlie. Bob has a blockchain token, which Bob transfers to Alice in transaction T1. Now, before the transaction is committed at the peers, Bob creates another transfer in transaction T2, for the same token, this time to Charlie. Bob in this case is trying to spend the same value twice, a problem known as double spending. In order to prevent such a scenario, a transaction is only considered valid if the version of each key that was present in the read set of the transaction during simulation matches the current version of the same key. In other words, if T1 is committed first, it will change the value of the key, from version 1 to version 2. If T1 is already committed to the ledger when T2 is simulated at the peers for endorsement, then T2 will be rejected as Bob will not have sufficient value to complete the transfer. However, in the second case, if T1 is not committed when T2 is simulated, but is committed before T2 is validated, then T2 will fail. When T2 was simulated in the second case, the key was at version 1, but when T2 is being validated the key would be at version 2, after modification by T1. This means that value of the key in the read set during validation of T2 would not match the value of the key during simulation. On the other hand, if T2 is committed before T1, the converse would occur.

The peers then append the block to their ledgers and update their world state with the valid transactions. Once the updates are successfully committed, the peers emit an event to inform the client of the transaction completion.

Performance measurement of transaction processing in Hyperledger Fabric is essential, to show its feasibility for adoption into real world scenarios. Hyperledger Caliper is a load generation and performance measurement tool for blockchain.

2.4 Hyperledger Caliper

Hyperledger Caliper is a performance benchmarking tool for different blockchain platforms. Papers II, III and IV included in this thesis all present performance testing experiments conducted using Hyperledger Caliper.

Caliper generates a workload for the configured system under test (SUT), in this case Hyperledger Fabric and monitors the responses. Caliper thus functions as a client using client identities to generate requests for the Hyperledger Fabric based transaction systems presented in this thesis. The rate at which transactions are sent to a blockchain implementation is a key factor for performance tests. Caliper provides several different rate controllers that can be configured to generate transaction requests based on configured rules. Paper II used the fixed rate controller (Version 0.2) which allows configuration of a request send rate. Send rate is expressed in transactions per second (TPS) and is defined as the number of transaction requests that are sent to a blockchain implementation each second. This rate controller is the most basic controller, which sends transactions at the configured send rate, until the configured total number of transactions are sent or the configured total time to send transactions has elapsed.

Paper III and IV used the fixed load rate controller (Version 0.4.2) which maintains the configured queue of unfinished transactions in the system by modifying the request send rate. The fixed load rate controller drives the tests at a target number of backlog transactions. The target count of backlog transactions is also known as Queue Length. The driven send rate will be modified by the controller in response to the observed Queue Length, in order to maintain the configured Queue Length. This rate controller thus, aims to achieve the maximum possible send rate for the system whilst maintaining the configured Queue Length. This rate controller sends transactions, maintaining the configured Queue Length, until the configured total number of transactions are sent or the configured total time to send transactions has elapsed.

Caliper generates a report based on the performance testing conducted. Throughput, measured in TPS is the rate at which transactions are committed to ledger. Latency is the time in seconds that

elapses between the sending of a transaction request to the blockchain (by an application, such as Caliper) and the transaction being committed to the ledger. Request send rate (in TPS), transaction throughput (in TPS), transaction latency (in second) are metrics included in this report.

Chapter 3

Contributions

This chapter presents an overview of four separate but interlinked research articles. A summary of each research article is provided in the subsequent sections and the specific contributions are delineated.

3.1 Overview

This dissertation addresses the need for new solutions to facilitate transaction management of renewable energy and energy flexibility in prosumer communities. The interconnections between the works forming this dissertation are shown in figure 3.1.

Paper I [35] presents a theoretical framework for this thesis by highlighting the need for a decentralized transactive system for prosumer communities and discussing the applicability of Hyperledger Fabric to this problem.

Paper II [36] builds upon paper I and presents RenewLedger, a blockchain-based framework for tokenizing energy assets such as electricity units, energy flexibility incentives and storage credits. Paper II implements a proof of concept of the system described in paper I.

Paper III [37] presents a data driven approach for implementing a incentive based energy flexibility system for prosumers on the blockchain. Paper III focuses on the reward-based prosumer usage flexibility scenario described in paper I.

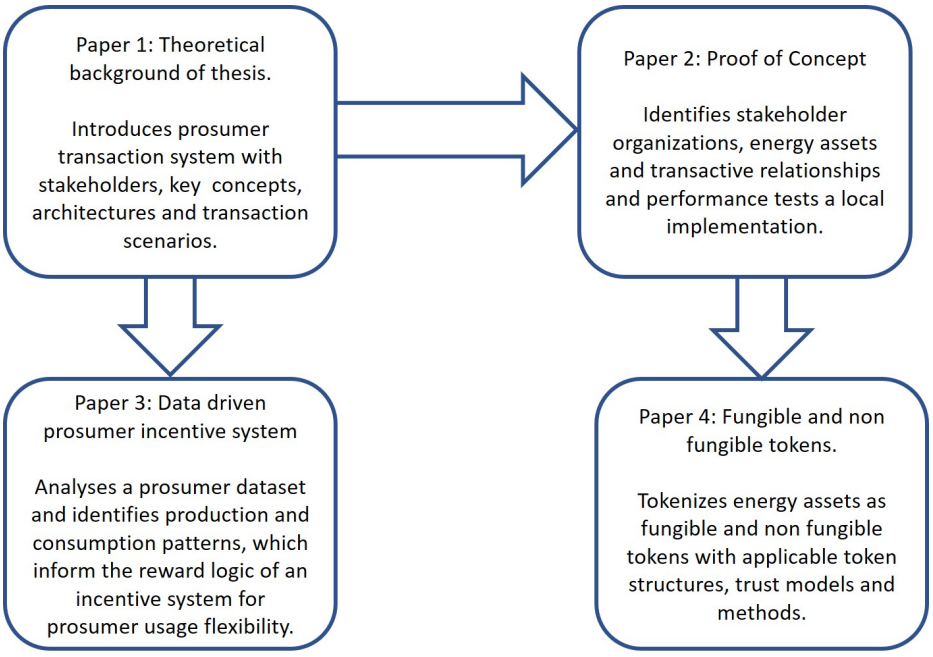


Figure 3.1: Interconnections between papers

Paper IV [38], presents a comparative study of energy assets represented as fungible and non fungible blockchain tokens based on application area, design, algorithmic complexity, performance, advantages and disadvantages. Paper IV thus, introduces both fungible and non fungible tokens in the system implemented in paper II with the applicable token design, trust model and methods.

The remainder of this chapter, summarizes each research paper.

3.2 Paper I

Transactive Energy on Hyperledger Fabric

This paper was published in Proceedings of 2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS).

Microgeneration of solar energy is gaining popularity especially in residential contexts giving rise to prosumers [3]. Several prosumers

in a neighborhood can organize into prosumer communities or microgrids and explore avenues for extracting value out of their surplus energy through energy trading or storage. Renewable energy microgeneration can offer financial benefits such as reduction in energy bills or earnings from selling the surplus, as well as environmental benefits. Electric vehicles (EV) if included in the prosumer community would be additional customers for the surplus energy. Moreover, EV owners can monetize on the batteries when not in use by sharing or renting them out as additional storage for the community. Communities can set up community level storage facilities by using the services offered by storage providers [5] who take over the task of setting up and maintaining the storage facility for a usage based fee structure. This would reduce the barrier for entry in such a system as it removes the need to have personal energy storage. Locally generated power can cut down on transmission losses inherent in transmitting energy over large distances [39] as the consumption site is collocated with the production. Microgrid capabilities can also give some degree of energy security to a community and stored energy can be directed to life saving efforts or disaster recovery in times of crisis.

Microgrids can be integrated with smart grids in order to provide even more value. A United Nations Economic Commission for Europe Report [40] defines smart grid as follows "A SmartGrid is an electricity network that can intelligently integrate the actions of all users connected to it – generators, consumers, and those that do both – in order to efficiently deliver sustainable, economic and secure electricity supplies." Smart grids enable a bidirectional flow of electricity data which allows real time energy generation and consumption data to be used for decision making. Smart grids can also enable more efficient demand response strategies by power companies such as incentive based energy flexibility which can reduce peaks in energy consumption and further add value to the grid by increasing grid stability and reducing cost.

Transactions between prosumers, EVs, storage providers and power companies that correspond to the described exchanges of value require a transparent transaction system. This transaction system must be decentralized in order to represent the interests of the small scale prosumers and EV owners. Immutability, transparency, provenance

tracing and self enforcing business contracts will further increase the trustworthiness and usability of such a system.

In this paper, we argued that blockchain specifically Hyperledger Fabric [16], a popular open source, enterprise grade and permissioned blockchain platform is well suited to creating such a transaction system. We discussed how the characteristics of Hyperledger Fabric directly satisfy the requirements described, a discussion that is also presented in section 1.1 of this thesis. The paper further describes the Hyperledger Fabric architecture with its key components and the workflow of transaction processing as was described in chapter 2 of this thesis.

Two architectures were proposed for setting up the Hyperledger Fabric network. Three organizations are considered in this network

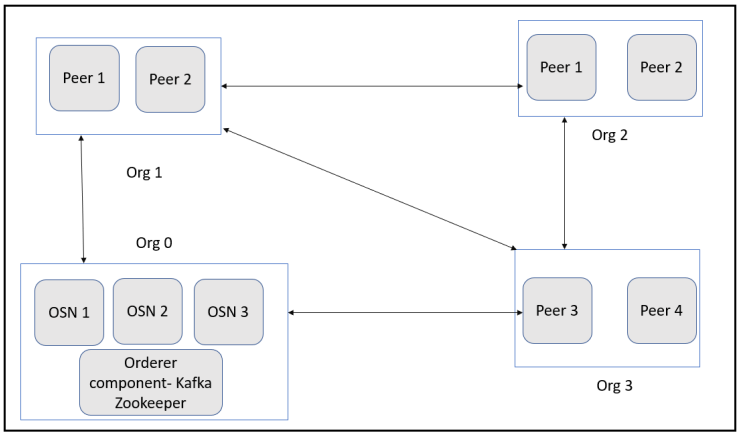


Figure 3.2: Orderer nodes in a separate location [35]

corresponding to the Transaction Platform, Power Company and Storage Provider. Additionally an orderer organization is created which is administered by the certificate authorities of the other three organizations. The first architecture shown in figure 3.2 puts the orderer organizations and all the ordering service nodes (OSN) on a location that it does not share with any of the other organizations. Alternatively, one OSN could be collocated with each of the peer organizations as shown in figure 3.3. In case of intra organization transactions, if the OSN collocated with the organization is the leader for a separate channel created for transactions within the organization,

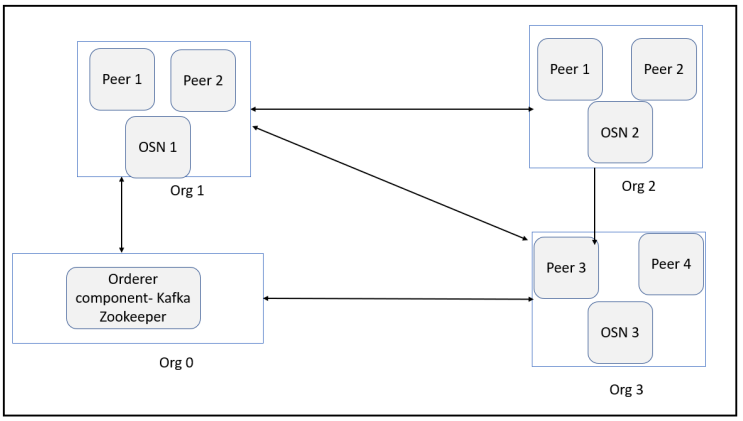


Figure 3.3: Orderer nodes collocated with peer organizations [35]

the ordering would be done by the local OSN, which may help reduce overhead.

For papers II, III and IV, we chose the former option as we consider a single channel with a single OSN leader at any given time. The proposed system is a unified system for all types of energy assets, thus a single channel was chosen. However, an implementation where this is not a requirement can consider separate channels for each type of energy asset. In paper I, we considered the ordering service to be a Kafka Zookeeper cluster as it was the ordering service implementation recommended by the Hyperledger Fabric v 1.4 documentation at the time of writing. This changed with the addition of Raft [34] as a supported and recommended ordering service implementation for Hyperledger Fabric v 1.4.1 onwards. As Raft, like Kafka follows a leader and follower model, the presented architectures are valid for Raft as well. In paper I we proposed the use of CouchDB as the state database due to its rich querying capabilities. However LevelDB offers better performance [41], and we found that it was possible to circumvent the need for rich querying for the proposed system. Thus, we chose LevelDB for papers II, III and IV.

Finally, the paper has an in depth discussion of the use cases of such a transaction system. Use cases considered are energy unit transactions between prosumers, EV battery charging, prosumer invoked energy storage, reward tokens for EV battery sharing as

well as demand response tokens for prosumers who estimate their own consumption in advance or perform energy flexibility tasks in a gamified context. The benefits provided to prosumers, EVs, the power company and the prosumer community are discussed. Interactions that would occur between the transacting parties for value exchange are detailed with sequence diagrams to further illustrate this point.

In this paper we discussed the applicability of blockchain in general and Hyperledger Fabric in particular, to creating a prosumer transaction system. We described a preliminary understanding of the stakeholders and their relationships which form the conceptual basis of our other works.

3.3 Paper II

RenewLedger : Renewable energy management powered by Hyperledger Fabric

This paper was published in the proceedings of the 2020 IEEE Symposium on Computers and Communications (ISCC).

A prosumer community enabled by a smart grid can earn monetary benefits from their surplus energy by selling it to members of the community such as other prosumers or EV owners. Energy storage is another alternative available to prosumer communities who can store the surplus energy for later use. A community level energy storage owned and operated by a storage provider can be considered, which will reduce the up front costs of setting up and maintaining personal energy storage. Such a storage would be made available to the prosumer community by way of storage credits that when purchased entitle the prosumer to store a certain amount of energy for a certain amount of time. EVs batteries when not in use can also be included in community storage in exchange for incentives. Such a prosumer community can also earn rewards for participating in energy flexibility tasks. Tasks can also have penalties for reneging, which means agreeing to complete a task and then failing to do so. Power companies can implement incentive based direct to consumer demand response strategies for peak shaving thus reducing the need to invest in over provisioning capacity. This reduction in cost can benefit

both, the prosumer community and the power company. The tasks performed by prosumers to earn these incentives could be accurately estimating their own consumption or performing energy flexibility tasks in a gamified context. Stored energy could also be used for peak shaving by discharging during peak times [6].

Energy assets such as energy units, reward tokens and storage credits would thus be transacted on such a system between three organizations namely, the prosumer community, the power company and the storage provider. This paper presents RenewLedger, a blockchain based framework to facilitate decentralized and transparent transactions of energy assets between these parties.

Six types of energy assets are considered on the system that would be transacted between the three organizations.

(1) EUnit- energy units transacted between prosumers

(2) EVUnit- energy units sold by prosumer to EVs

(3) InUnit- rewards offered by power company to prosumers for consumption estimation

(4) GaUnit - rewards offered by power company to prosumers for energy flexibility tasks in a gamified context

(5) StUnit - Storage credits used by prosumers to store energy with the storage provider

(6) EStUnit - rewards offered by storage provider to EVs for battery sharing

Figure 3.4 shows the relationships between organizations in the network as well as the tokens that encapsulate the energy assets.

These energy assets were encapsulated into tokens on the Hyperledger Fabric represented as key value pairs on the state database. We used LevelDB as the state database as it offers better performance than CouchDB [41] which is the other supported state database in Hyperledger Fabric. LevelDB does not support rich querying, so we include information required to query the token in the key. The key would include the type of energy asset and a unique serial number and could also include other information needed to query the token.

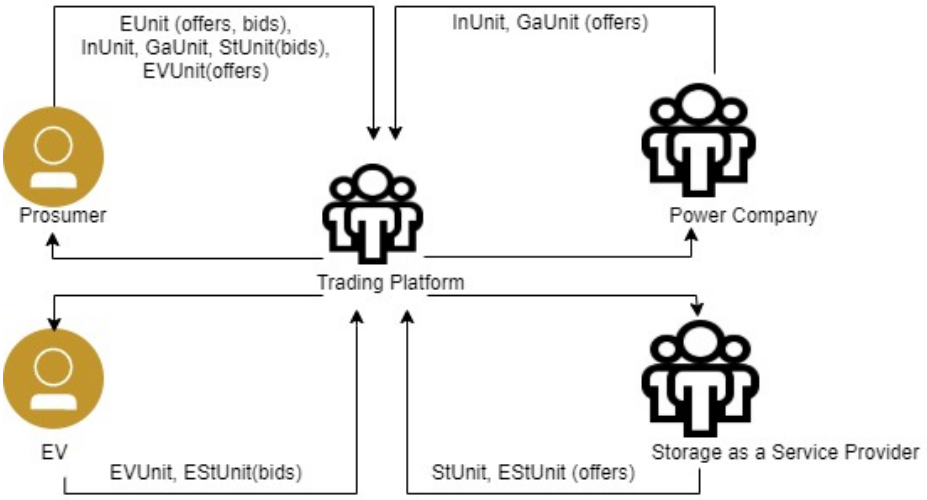


Figure 3.4: Relationship between organizations and tokens [36]

Each token value has token specific fields in addition to a bid flag and a text field storing the owner of the token. The bid flag is set to true when a bid is placed on a token. We consider bidding as registering a binding interest in a token without negotiating the price of the token. EUnit and EVUnit have fields price (price of token), location (location of seller) and value (amount of electricity units in token). InUnit and GaUnit have fields reward (incentive offered for task), penalty (penalty for reneging) and requirement (what the task entails). StUnit has fields price (price of the token), value (number of energy storage credits) and conditions (of storage). EStUnit has fields reward (incentive offered), amount (storage capacity required) and conditions (of storage). Other fields can be added as required to the value of the key value pair of the token.

In addition to the peer organizations which are transaction platform, power company and storage provider, we have a separate orderer organization implemented as a Raft ordering service which is the recommended ordering service implementation in Hyperledger Fabric. A client is setup for each peer organization to enable the users of that organization to access the network. The transaction platform client is used by members of the prosumer community to buy and sell energy

units, earn reward tokens and to store surplus energy. Members of the power company offer incentivization and gamification tasks and process the rewards using the client. The storage provider client is used by members of that organization to process storage requests and rewards.

In order to performance test the proposal, a chaincode in Golang with two functions, one for writing a single key value pair to the state and another for reading a single key value pair from the state was implemented. The endorsement policy for the chaincode was that one peer from each organization would be required to endorse each transaction. For EUnits and EVUnits, both the buyer and seller would be members of the Transaction Platform organization, so a network to transact these could be implemented with a single organization namely the transaction platform. InUnits and GaUnits, are offered by the power company to members of the transaction platform organization, so a network for transaction of these tokens could be implemented with two organizations, the transaction platform and the power company. Finally, the StUnits and EStUnits are transacted between members of the transaction platform and the storage provider. However, if the stored energy is used for peak shaving then the power company would also be involved in these transactions. Thus, a transaction system for StUnits and EStUnits would be implemented using three organizations. We implemented proofs of concept with one, two and three organizations in the network and performance tested them with varying request send rates. In each implementation, we configured two peers in each organization, so that if one peer goes down, the other peer can continue to process transaction requests. We found that increasing the send rate increased the throughput as well as the latency. Moreover, comparing the performance of networks with one, two and three organizations we found that increasing the size of consortium negatively impacted performance. The performance of the implementation was found to be promising for a local implementation with low resources. In Paper III and Paper IV, we implemented transaction systems with more resources and found that the performance was much higher. For instance, the implementation in Paper III could support 792,540 customers with a reasonably low infrastructure footprint. Horizon-

tal and vertical scaling of the computing resources would scale the implementation further as described in Thakkar et al. [42].

3.4 Paper III

Blockchain-based prosumer incentivization for peak mitigation through temporal aggregation and contextual clustering

This paper has been published in the journal Blockchain: Research and applications, Elsevier Publications.

Peak demand is rising due to the increase in the number of retail users [7]. Periods of peak demand present a challenge to the power company who must maintain grid stability in presence of variation [43]. If the peak demand approaches the supply, it can compromise grid stability and lead to blackouts.

Power companies can deal with this problem by either supplementing the supply or by reducing the demand. Supplementing the supply may require the power company to over provision capacity which may increase the marginal cost of electricity. Moreover, fossil fuel powered peaker plants that are generally used to add capacity during peak periods can be polluting [44]. The increasing integration of solar energy into the grid provides an opportunity to address this issue. Locally generated solar energy can be either directly used to add to the supply or used to charge grid connected batteries which can be discharged when needed. As the generation and consumption sites are collocated, energy loss due to transmission will be reduced [39] when charging the grid battery.

Demand side response [8] is another way to reduce the magnitude of peaks by shifting some of the peak usage to off peak times. If the magnitude of the peak is reduced, the power company may be able to postpone or avoid investment in additional capacity, which can lead to savings for the consumer. Ausgrid Corporation, a large power company operating in Australia is conducting a pilot study [45] to achieve peak shaving through the use of grid connected batteries owned and operated by the power company.

Demand response can be implemented by discouraging customers from using electricity at peak times by implementing variable pricing

depending on time of use. Alternatively, prosumers can be encouraged to contribute energy to charging the grid connected battery as well as to contribute energy flexibility for demand response. In this paper we propose a two pronged data driven approach for incentive based peak mitigation.

(1) Customers who keep their energy consumption below a calculated threshold during the identified peak consumption periods are awarded reward tokens.

(2) The top surplus producing customers in the network earn reward tokens based on the amount of surplus they produce.

An incentivization system would be required to transparently calculate and process rewards. Such a system would involve three organizations namely, a transaction platform for the prosumer community, the power company and the storage provider. All three organizations must agree on the logic of the incentivization system. The transaction platform organization represents the prosumer community so it must be able to check and approve all transactions. The storage provider must be able to check that the energy shown as produced is actually recorded in the grid battery and is available for use. Moreover, a decentralized system will prevent the management from being taken over by a centralized entity. Thus, a blockchain based system must be considered in order to fulfil these requirements. Moreover, as the system must only be open to authenticated members of the prosumer community Hyperledger Fabric [16], a popular permissioned blockchain platform would be suitable. This blockchain implementation would be the transaction platform infrastructure for this incentivization system. Figure 3.5 presents an overview of the proposed data driven approach.

The dataset used in this work is the solar home electricity dataset from the Ausgrid Corporation, Australia which includes data for 300 random customers in the Ausgrid network from July 1 2012 to July 30 2013 recorded at half hour intervals. The location of these customers is shown in the map in figure 3.6. First, customers with anomalous and inconsistent data are removed. Then the user data is aggregated and user energy behavior is extracted in temporal contexts. Seasonal

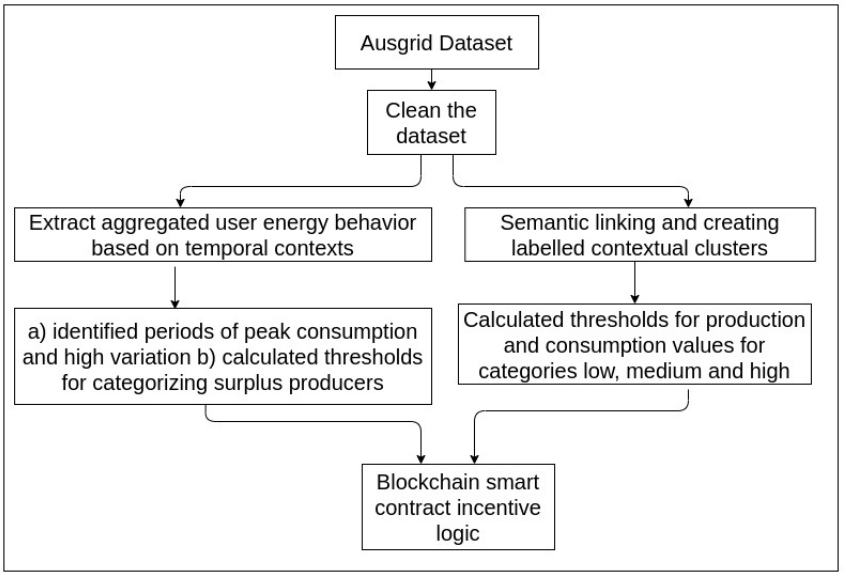


Figure 3.5: Schematic outline of the system. [37]

Figure 3.6: Locations of customers [37]

(winter, summer, autumn and spring), day of the week (weekday or weekend) and time of the day are the temporal contexts studied. Temporal variation in the fields of the dataset was of particular interest. Coefficient of variation, which is calculated as the ratio of standard deviation to the mean was used to characterize variation. Additionally, surplus energy production was aggregated and used to calculate thresholds for categorizing the surplus generation.

Further, the rows of the dataset were semantically linked and clustered based on production and consumption contexts. The fields in the dataset corresponding to solar production, heating consumption, energy consumption, regional price and regional demand were each considered to a separate layer in a multilayer graph linked by the common attributes, which were Customer, TimeStamp and PostCode, which constituted an inter layer edge in the graph. Thus, each dataset row was split over the layers, connected by the common attributes. Each layer had multiple clusters and C was the set of clusters in a layer. These clusters were labelled based on calculated thresholds.

In order to illustrate this process described in section 4 of paper III, consider the simplified example dataset shown in table 3.1. The values here are to illustrate the process and are not representative of actual values from the dataset. Here we have five columns and four rows. Of the columns, Customer, PostCode and TimeStamp together are the common attributes that allow a row, split across layers to be linked, thus forming an edge of the graph where the edge is referenced by the Row column shown in table 3.1. The contexts here, are Production (column Prod.) and Consumption (column Cons.) that would be put in two separate layers for each row, connected by an edge, so that there exists one edge connecting each row of the dataset across layers, as shown in table 3.2.

Table 3.1: Simplified example of dataset

Row	Customer	PostCode	TimeStamp	Prod.	Cons.
1	10	2000	01-01-12	0.5	23
2	10	2000	02-01-12	10	22
3	20	3000	01-01-12	11	1
4	20	3000	02-01-12	0.6	2

Table 3.2: Simplified example of multilayer structure

Layer	Edge	Value
Production	1	0.5
Consumption	1	23
Production	2	10
Consumption	2	22
Production	3	11
Consumption	3	1
Production	4	0.6
Consumption	4	2

Now, for a simplified explanation of the clustering process, consider that in each layer, clusters are formed so that values that are close together are part of the same cluster, and each cluster in each layer has two values. So, for the Production layer, we would have two clusters, such that cluster 1 would have values 0.5 and 0.6 and cluster 2 would have values 10 and 11. Similarly, in the Consumption layer we would have two clusters as well, so that cluster 1 would include values 1 and 2 and cluster 2 would have values 22 and 23. Even though the dataset row is split across layers and clustered by value, it is still connected via the edge. So, for instance, consider row 1 from table 3.1. The common attributes (Customer, PostCode and TimeStamp) form edge 1 that connects the Production value 0.5 to the consumption value 23, across layers. Now, consider that we need to label the clusters using two labels, “High” and “Low”. So then for

both layers, cluster 1 would be labelled "Low" and cluster 2 would be labelled as "High". The numbering of the clusters described here is simply to refer to them in this explanation and does not provide any information about the values in the cluster. In paper III, section 4.1, step 3 the first sentence says, "creates a semantic link between layers in the multilayer network M by utilising attribute similarity between nodes in the same layer." This was intended to mean that nodes are linked across layers by using the common attributes (Customer, PostCode and TimeStamp) in each layer, to form edges.

Based on the analysis of the dataset, of the several patterns identified we focused on the following observations to formulate the smart contract logic.

(1) Seasonal peak consumption was observed in summer and winter, which was attributable to climate control needs. Summer also showed the highest variation in energy consumption.

(2) Peak consumption during the week was seen on weekends when residential customers are home.

(3) Two peaks in consumption were observed in a day, one before the start of the work day in the morning and one in the evening at the end of the work day.

(4) Thresholds t_a, t_b and t_c were identified that corresponded to surplus production in the 25th, 50th and 75th percentile respectively. The values for the dataset were 0.134 kWh, 0.284 kWh and 0.477 kWh for t_a, t_b and t_c respectively

(5) For the user demand the thresholds t_1 and t_2 were identified to categorize clusters into categories of low, medium and high consumption. For the dataset used the values were $t_1 = 1.8306$ kWh and $t_2 = 3.6608$ kWh.

We considered each row in the dataset as a transaction, which would be encapsulated in a blockchain token. The blockchain token representing the dataset transaction would be composed of a key value pair on the state database created by a blockchain transaction. The dataset rows have fields describing aspects of the transaction such

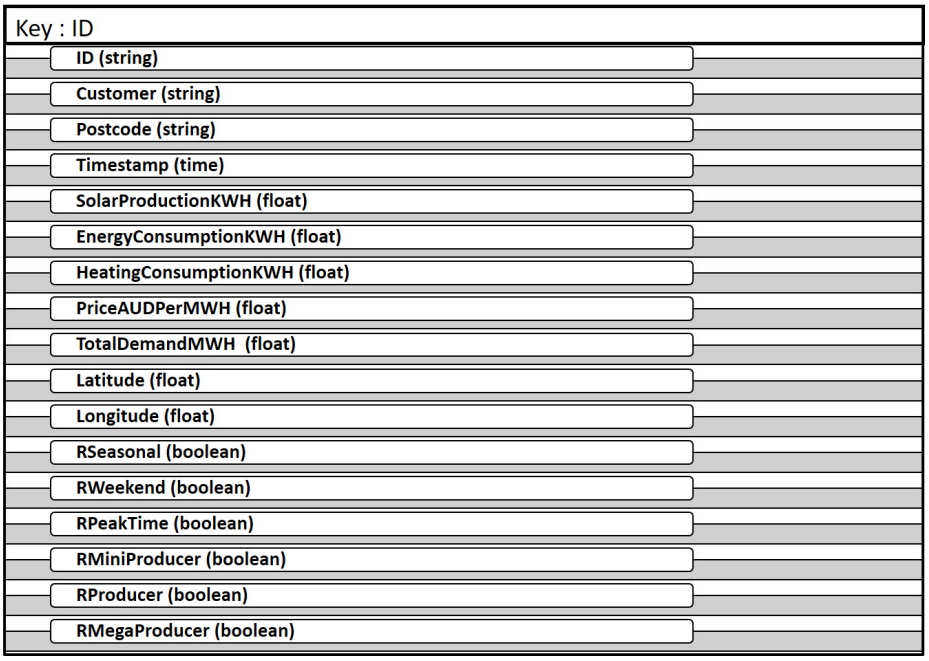


Figure 3.7: Structure of transaction token [37]

as customer ID, timestamp of recorded values, solar production in kWh and energy consumption in kWh. The blockchain token would include all the fields from the dataset transaction in the value of the key value pair. Additionally six fields are added to the value, each corresponding to a reward. These are Boolean values which would be set to true if the conditions of a reward of the given type are met by a particular dataset transaction. The token structure is shown in figure 3.7. The key of the token is composed of the customer ID and the timestamp of the dataset row.

RSeasonal, RWeekend and RPeakTime are reward fields to incentivize low consumption (based on thresholds identified in observation 5) during summer/winter seasons (based on observation 1), during weekends (based on observation 2) and during daily peak consumption times (based on observation 3) respectively.

RMiniProducer, RProducer and RMegaProducer are reward fields to incentivize surplus production and are set to true if a transaction shows surplus generation greater than or equal to t_a, t_b and t_c respec-

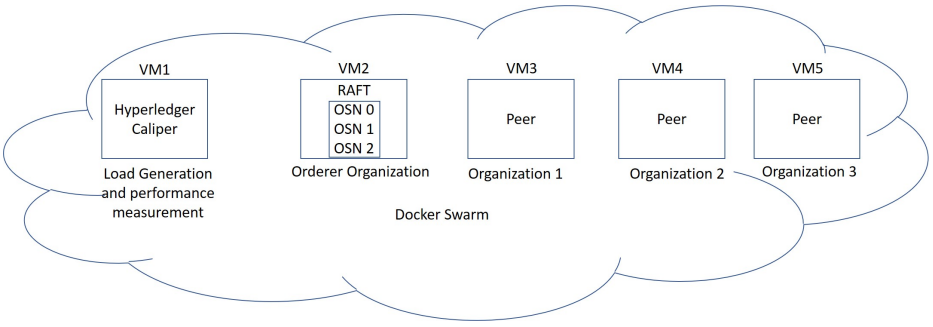


Figure 3.8: Experimentation testbed [37]

tively (based on observation 4). A transaction could earn multiple rewards.

The chaincode is implemented in Golang and implements two algorithms. Algorithm 1 is a helper algorithm that checks if a given key exists already in the state. Algorithm 2 includes the logic to create the transaction token. First the organization of the invoking client is checked. As the power company has access to the generation and consumption information and is the organization that offers the rewards, it must be the organization which initiates each transaction. Thus, if the invoker client does not belong to the power company the transaction does not proceed. Next, the token to be created is checked against the state (using Algorithm 1) to make sure that there is no collision. Each customer has a dataset row and thus blockchain token created each half an hour. If the token for a given customer for a given half an hour time does not exist already, the rewards associated with the transaction are calculated as explained and the transaction is updated to the blockchain.

In order to test how many customers our implementation would support, it was important to test the performance. We build a proof of concept of the proposed system on the Hyperledger Fabric platform and deployed it in a cloud infrastructure. The testbed is shown in figure 3.8 . Each node of the network was implemented as a Docker container connected in a Docker Swarm for high availability. Each organization was deployed on a separate virtual machine (VM). Hyperledger Caliper was run on a separate VM to generate the load and measure the performance. Thus five VMs were used in the

network, three for the peer organizations, one for the Raft based ordering service organization and the fifth for the load generator.

The fixed load rate control mechanism of Hyperledger Caliper was used which starts with a configured request send rate and maintains the number of unfinished transactions below the configured value by modifying the send rate. The starting send rate was set to 1000 transactions per second (TPS) and the configured queue length was varied from 125 unfinished transactions to 650 unfinished transactions. It was observed that the send rate achieved increased with the increase in queue length. Due to the increase in send rate, transaction throughput and latency increased as well. Our implementation achieved a throughput of 440.3 TPS with sub second latency at a send rate of 443 TPS when the queue permitted was 650 unfinished transactions. As a transaction for each customer is processed each half an hour, extrapolating we can say that our implementation can support 792,540 customers with a reasonably low infrastructure.

3.5 Paper IV

Blockchain Based Transaction System with Fungible and Non-Fungible Tokens for a Community-Based Energy Infrastructure

This paper was published in the journal Sensors, MDPI publications.

In this paper, a blockchain based transaction system was proposed for prosumer communities to extract value from their surplus generation and energy flexibility in a transparent, decentralized and immutable manner. Hyperledger Fabric was used as it offers the additional benefits of access control, lower operating costs, self enforcing smart contracts as well as configurable trust models, transaction format and consensus mechanisms. Hyperledger Fabric does not have a native cryptocurrency and allows encapsulation of any asset into a blockchain token.

Tokens on the blockchain can broadly be categorized into two categories. Fungible tokens (FT) encapsulate value only and are identical for all intents and purposes. They can be broken down and transacted in parts. In an energy transaction system, if the

energy assets of a type are interchangeable then they can be modelled as FT. For instance, Bitcoin tokens could be considered fungible, as one Bitcoin is equal to another Bitcoin and a Bitcoin can be divided into smaller parts and traded. Non fungible tokens (NFT) on the other hand encapsulate value as well as unique information, such as an identifier. Energy assets with an attached Guarantee of Origin [21] certification, for instance have a unique identifier and are not interchangeable. Such energy assets would be represented as NFT. Another example of NFT is Crypto Kitties [46], which are unique collectibles so that no two are the same. Implementing energy assets as NFT allows assets of the same type to have different prices. However, if uniformity is required in the implementation of assets of the same type, FT would be chosen.

In this paper we implement two versions of a unified energy transaction system for transacting several types of energy assets such that one version considers energy assets as FT while the other considers them as NFT . Paper II discussed six types of energy assets (EUnit, EVUnit, InUnit, GaUnit, StUnit and EStUNit) to be transacted among the three organizations (Transaction Platform, Power Company and Storage Provider) in the business network. In this work we consider the same six types of energy assets to design and implement both versions of the transaction system for the three organizations. The goal of this paper is to critically compare FT and NFT based on design, algorithmic complexity, performance, application area, advantages and disadvantages.

First, we identified trading relationships between the three organizations for transactions of the six types of energy assets. For EUnit and EVUnit, the Transaction Platform and the Storage provider organizations are involved. This is because these tokens are issued by members of the Transaction Platform. Moreover, we considered that the prosumers do not have any personal storage and the storage provider would store the surplus energy. The transaction of energy units would thus occur through the storage provider. Due to this the storage provider must verify and endorse that the surplus energy is actually physically available. For InUnit and GaUnit the power company organization invokes the creation of tokens, while the members of the Transaction Platform would bid upon these tokens.

Finally, StUnit and EStUnit would involve the Storage Provider and the Transaction Platform for the same reasons. Thus, each energy asset is mapped to two organizations, one that would create the token and the other that would endorse it. This mapping is used to set the key level endorsement policy of each token.

The energy assets would be encapsulated in blockchain tokens. A token, whether FT or NFT would be taken through its lifecycle by four methods. First the token is created by a seller. Second, a buyer bids upon the token. Third, the seller transfers the ownership of the token to the buyer, and finally the owner of a token redeems the value of the token. Redeeming a token destroys the value of the token by the amount redeemed. So, when a NFT is redeemed, the entire token is destroyed as it cannot be redeemed in parts. In contrast, when a FT is redeemed, the value that is redeemed is reduced from the value of the token, but the token itself is not destroyed. In the proposed energy system, redeeming a token would allow the holder to use the value, for instance by consuming the energy associated with an EUnit. The amount of energy associated with the EUnit would be available for use with the Storage Provider, as the Storage Provider verifies and endorses creation and transactions of EUnits.

A token is implemented on the Hyperledger Fabric as a key value pair. Both FT and NFT implementations would have the same basic token structure which is shown in figure 3.9.

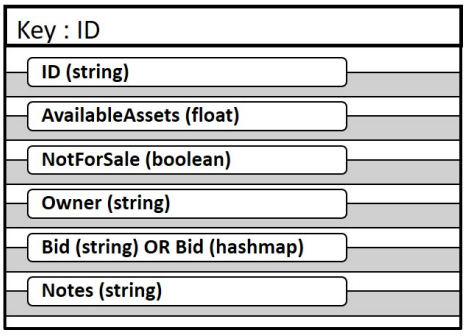


Figure 3.9: Structure of token [38]

The key of the token in the state database would uniquely identify the token. We used LevelDB as the state database as it offers better

performance [41]. However, it does not support rich querying. For FT, as the tokens are identical, a total of six tokens would be created for each client, one for each type of energy asset. Thus, for FT, the tokens would function like accounts and as a client only has six accounts, they would not need rich queries. The key of a FT token was of the form **Key: clientid_tokentype**. For NFT, there can be several tokens of the same token type owned by the same client. In order to distinguish between these NFTs and in order to support querying for each client by use of range queries, NFT keys would be of the form **Key: clientid_tokentype_transactionID**, where transaction ID is a unique identifier for a transaction in a channel scope.

The value of the key value pair of the token included AvailableAssets (number count of energy assets in token), NotForSale (boolean, set to true if the token is not for sale), owner (string, identity of the token owner) , Bid (string OR hashmap) and Notes (string, for any information about the token). In case of NFT, the Bid would be a string field accepting the identity of a single bidder. In case of FT, as the token can be broken up, multiple bids are accepted and stored in a hashmap (key: bidder identity, value: bid amount). If the bidder bids on a FT multiple times, the value of the bid is updated with the total of the bids. For every bid placed on a FT, the count of AvailableAssets field is reduced by the bid amount.

Algorithms 1 to 11 were developed to take the tokens through the lifecycle which is shown in figure 3.10. Algorithm 1 sets the key level endorsement policy of a token. This policy overrides the chaincode level endorsement policy and includes endorsers for each token based on the transaction relationships as described before. Algorithm 2 reads a single token from the state. Algorithms 1 and 2 are helper algorithms and are used by the other algorithms. Algorithms 3-7 correspond to methods for NFT while 8-11 correspond to FT.

Algorithm 3 creates a NFT. Each energy asset is mapped to an organization which is permitted to invoke creation as described before. For instance EUnit and EVUnit can only be created by the Transaction Platform. The create algorithm checks that the combination of energy asset type and invoker organization is valid, sets the owner of the asset to the invoker (if creating for self) or the buyer (if creating for transfer), adds the new token to the state and sets the key level

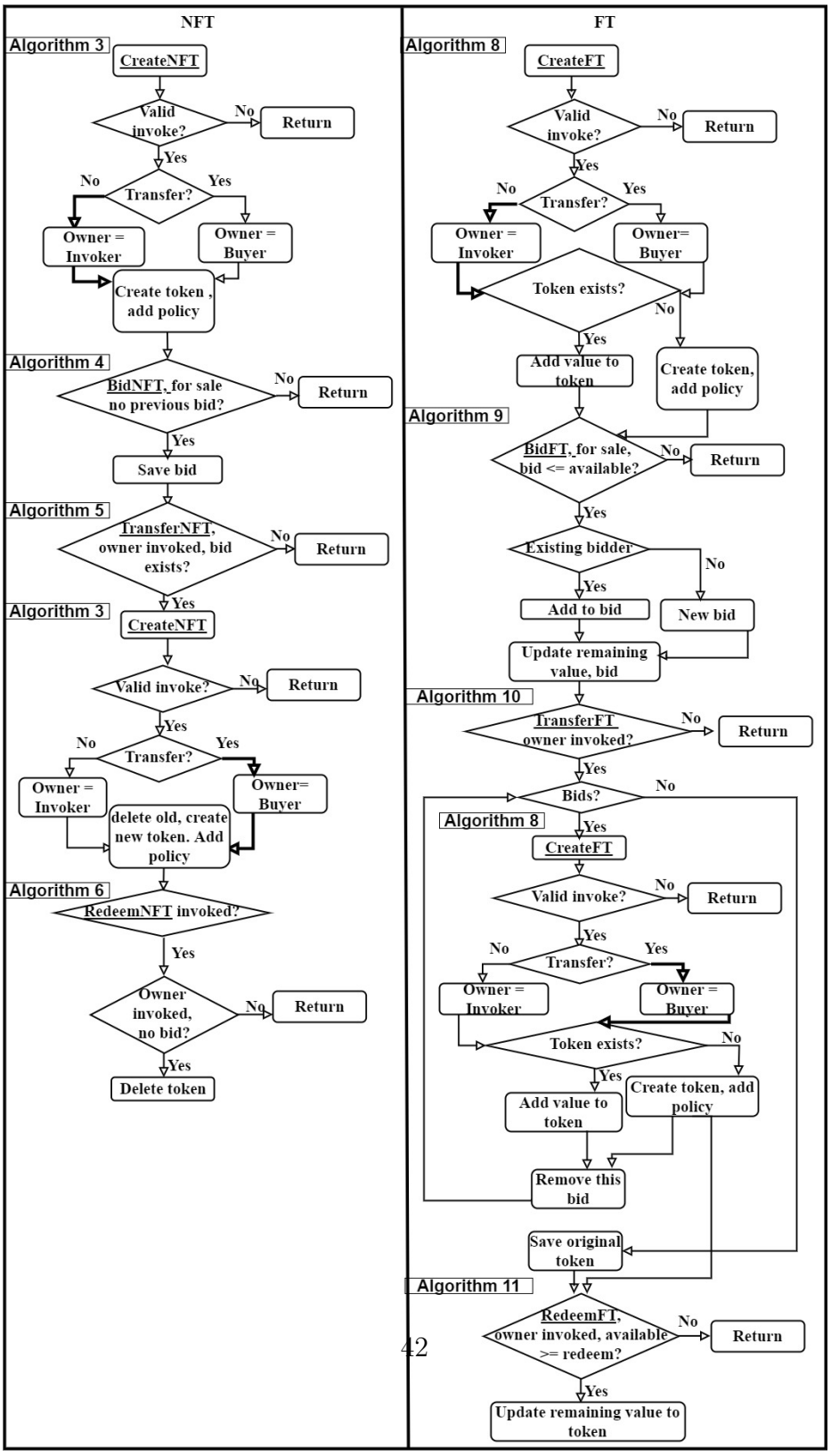


Figure 3.10: Lifecycle for Fungible Tokens (FT) and Non Fungible Tokens (NFT). [38]

endorsement policy (Algorithm 1). A buyer would bid on this token using Algorithm 4 if it is for sale and there is no other bid upon it. The seller can now transfer the token to the buyer using Algorithm 5, by creating a new token owned by the buyer (Algorithm 3) and deleting the original token from the state. Finally a token can be redeemed by the owner if there is no bid present and deleted from the state using Algorithm 6. Algorithm 7 is a bulk read algorithm that retrieves all tokens of a given token type owned by the invoking client using a range query.

Algorithm 8 creates a FT. The create algorithm checks that the invoker and token type combination is permitted and sets the owner of the token as explained for Algorithm 3. "Ownerorg" is mentioned in the algorithm 8 in the paper, which is intended to mean invokerorg, similar to algorithm 3. Then, it reads the token (Algorithm 2) and only creates a token if a token for the given client and client type does not already exist, else it updates the existing token with the additional value. Finally, in case of token creation, it sets the key level endorsement policy (Algorithm 1). This token can accept multiple bids, each of which is stored in a hashmap using Algorithm 9. When a bid is placed, the bid amount is deducted from the AvailableAssets field of the token. The seller can complete transfers for all bids on the token using Algorithm 10 by creating (or updating if it already exists) a token for the buyer and removing the processed bids from the hashmap. Even if the seller's token has no more assets, it is not deleted from the state as it functions like an account. An owner of a FT can redeem value equal to or less than the value available on the token using Algorithm 11. Unlike NFT, the token is not deleted when redeemed, but the redeem value is deducted from the token value (AvailableAssets).

NFT and FT thus have operations Create, Bid, Transfer and Redeem. FT operations Create and Transfer may create a token and set its endorsement policy or simply update a token. In order to distinguish between the these we call the former operations Create and Transfer and the latter ReCreate and ReTransfer.

NFT Create, Bid, Transfer and Redeem as well as FT Create, ReCreate, Bid and Redeem have time and space complexities O(1) as they perform a constant number of operations when executed each

time. NFT Bulk Read (Algorithm 7) retrieves all the assets owned by a client which would depend on and grow with the number of tokens owned by them. Each token returned by the bulk read operation can be considered as a separate read. The space used would also depend on the number of tokens retrieved. Thus the time and space complexities for NFT Bulk Read are O(n) which was expected to be a bottleneck if executed on chain. FT Transfer and FT ReTransfer would also execute the loop of creating/updating the buyer's token multiple times, depending on the number of bids present. The time and space complexities for these methods are thus O(n). However, as these operations complete transfer of value to multiple distinct recipients they can be compared to the same number of separate transfer executions for NFT. Due to this, these operations cannot be considered a bottleneck. A fair comparison between FT and NFT would thus consider a single bid per token.

In order to performance test the proposal, we implemented two chaincodes one each for FT and NFT implementations with the algorithms described encoded in smart contracts. The infrastructure of the testbed is the same as was described for Paper III. Two clients Alice and Bob were created as members of two different organizations where Alice was the seller and Bob was the buyer. The NFT implementation was tested with the lifecycle methods Create, Bid, Transfer and Redeem in order, while the FT implementation was tested with Create, Bid, Transfer, Redeem, ReCreate, Bid, ReTransfer and Redeem in that order. The experiments are described in detail in figure 3.11.

Hyperledger Caliper was used to submit the transactions and measure the performance. The fixed load rate control mechanism was used, similar to Paper III which maintains the defined queue of unfinished transactions by modifying the send rate. The queue length was varied from 100 to 1000 unfinished transactions with a step of 100. Additionally, the Read algorithms were implemented for both FT and NFT chaincodes with the queue length varied from 100 to 400 unfinished transactions. Finally the Bulk Read algorithm for NFT was tested which retrieved 10,000 tokens in each transaction. The queue lengths for Bulk Read were also varied from 100 to 400 unfinished transactions. The performance was characterized using

NFT experiment sequence		FT experiment sequence	
	Balance		**Balance**
Alice: Creates 10,000 NFT of 10 assets each	Alice: 10,000 NFT *10	Alice: Creates 10,000 FT of 10 assets each	Alice: 10,000 FT *10
Bob: Bids on 10,000 NFT for full value	Alice: 10,000 NFT *10	Bob: Bids on 10,000 FT, 1 bid per token, 5 assets per bid	Alice: 10,000 FT *10
Alice: Creates 10,000 NFT of 10 assets each for Bob, Deletes 10,000 NFT for Alice	Alice: 0 NFT Bob: 10,000 NFT *10	Alice: Creates 10,000 FT for Bob with 5 assets each. Reduces Alice's 10,000 FT by 5 assets each	Alice: 10,000 FT *5 Bob: 10,000 FT *5
Bob: Redeems 10,000 NFT	Alice: 0 NFT Bob: 0 NFT	Bob: Redeems 10,000 FT for 2 assets each	Alice: 10,000 FT *5 Bob: 10,000 FT *3
		Alice: Adds 10 assets each to 10,000 FT accounts	Alice: 10,000 FT *15 Bob: 10,000 FT *3
		Bob: Bids on 10,000 FT, 1 bid per token, 5 assets per bid	Alice: 10,000 FT *15 Bob: 10,000 FT *3
		Alice: Adds 5 assets each to Bob's 10,000 FT. Reduces Alice's 10,000 FT by 5 assets each	Alice: 10,000 FT*10 Bob: 10,000 FT *8
		Bob: Redeems 10,000 FT for 2 assets each	Alice: 10,000 FT *10 Bob: 10,000 FT *6

Figure 3.11: Sequence of experiments [38]

the metrics transaction throughput and latency.

Based on the performance testing we draw the following conclusions. Increasing the permitted queue of unfinished transactions increased the request send rate which in turn increased the throughput as well as the latency. Increasing the transaction complexity, defined as the number of write operations per transaction negatively impacted the performance. The operations that had the the same transaction complexity showed similar performance. NFT Bid, NFT Redeem, FT Bid, FT ReCreate and FT Redeem all had a transaction complexity of one write operation and showed similar performance. Operations with transaction complexity of two write operations namely NFT Create, FT Create and FT ReTransfer also showed similar perfromance. Finally, FT Transfer and NFT Transfer had transaction complexity of three write operations and showed similar performance. Thus, the performance of operations for NFT and FT was similar for most major operations. However, FT operations ReCreate and ReTransfer were faster than Create and Transfer operations for both, FT and NFT. Read operations for FT and NFT were comparable as well.

However, Bulk Read operation for NFT showed a very poor performance (Peak throughput 13 TPS, Latency 18.44 second) and would thus have to be executed off chain. In contrast, tokens in the FT implementation function like accounts, due to which the total energy assets of a type owned by a client can be retrieved through a single Read operation, which is an advantage. However, the FT implementation has limitations with concurrency, due to the Multi Version Concurrency Control in Hyperledger Fabric intended to prevent double spending. When multiple operations try to update the same token, contention arises which means that at most one can succeed. Thus, when two sellers try to transfer value to the same buyer token at the same time only one can succeed. This problem does not arise in NFT as it avoids contention by creating new tokens for every Create and Transfer operation. Measures such as application based queuing could be considered to solve this. Thus, both implementations have comparable performance with specific advantages and disadvantages and no performance based reason was found to choose one over the other. The choice of implementation would thus depend on the use case.

Chapter 4

Conclusions and Future Work

This chapter concludes this thesis. The research questions identified in chapter 1 are presented along with a discussion of how specific publications addressed these questions. Directions for future research are described which could add value to the proposed blockchain based transaction system.

4.1 Conclusions

The thesis investigated the following research questions.

RQ 1. Is blockchain applicable to address transaction management for energy trading, flexibility and storage?

Papers I, II, III and IV investigated different aspects of this research question. In paper I, the applicability of blockchain for energy transactions for trading, flexibility and storage was conceptually presented. Paper II identified the specific stakeholders and their business relationships and encapsulated the energy assets into blockchain tokens for facilitating transactions. Paper III focused on the applicability of blockchain to implement a incentive based energy flexibility system for prosumers using consumption and production behaviors identified from a real world dataset. Finally, Paper IV, showed that token specific business relationships can be encapsulated on the blockchain and critically compared transactions of energy assets implemented as

fungible and non fungible tokens.

RQ 2. What data driven approaches can be adopted to implement a blockchain based prosumer incentivization system for peak mitigation?

Paper III presented a data centric approach for design and implementation of a incentive based system to encourage energy flexibility among prosumers for peak mitigation. A two part strategy was proposed using a prosumer research dataset to reward prosumers who recorded a higher amounts of surplus production as well as prosumers who recorded a low consumption during peak periods. The insights from analysis of the dataset informed the logic of the smart contracts implemented in the blockchain based system.

RQ 3. How can blockchain be used to tokenize and transact multiple types of energy assets in a unified system?

Paper II identified six different types of energy assets and encapsulated them into tokens implemented as key-value pairs for trading on the blockchain. Each energy asset had the value of the key value pair composed of different fields specific to the type of energy asset. These different key-value pairs were then stored on the same world state thus implementing a unified transaction system for diverse assets. Paper IV demonstrated the tokenization and transactions of the six types of energy assets in fungible and non fungible versions, which are the two broad categories of blockchain tokens. The design, implementation and performance testing showed feasibility of using the proposed approaches to implement both types of tokens.

4.2 Future Work

Several directions for future work emerge from this thesis. Of these, prediction and recommendation capabilities would add significant value to the proposed transaction system. Prediction of individual as well as neighborhood energy consumption, storage needs as well as energy flexibility if made, could be used to create and periodically update the smart contract logic. This would bring the benefits of

machine learning into the blockchain based transaction system.

Individualized recommendation would help a prosumer choose energy flexibility tasks or storage alternatives. Recommendation for buyer and seller matching based on location, pricing or source of generation would also add value and allow the buyers to choose from a smaller subset of options that already fulfil their requirements.

Going a step beyond recommendations, if the user preferences are known, transaction choices could be automated by individualized client side code that executes and creates the transaction proposal that invokes the appropriate the smart contract when certain conditions are met.

Although the work done in this thesis focuses on peer to peer transactions for renewable energy, several other domains could also adapt these approaches to other domains. The approach described in paper IV where only the owner of a token can initiate transactions can be coupled with private data collections in Hyperledger Fabric for consent based data management. For instance, in hospitals, a patient consent system can be implemented for managing access to their medical data.

An interdisciplinary study with the social sciences can be considered, that conducts and analyses user surveys on the proposed gamification and incentivization mechanisms for demand response. This study can target an understanding of prosumer attitudes to different forms of incentive-based usage behavior modification strategies and analyse correlations with factors such as age, socio-economic status, education, number of adults or children in the household and geographic location. Such analyses can help identify the most effective strategies for demand response in relation to other personal factors. The findings from such a survey could be validated in a pilot project. Based on user interactions, the pilot project could verify if the opinions expressed by individuals in the survey are actually reflected in their decision making. Such a study, if conducted could be used to predict how a given user would respond to a particular strategy. This could be used to guide future incentive-based usage flexibility strategies.

As a transaction system involves user interactions for exchange of value, previous user experiences with buying or selling may help

future buyers and sellers make decisions. As the proposed transaction system is for a prosumer community, where people living in close proximity transact with each other, users may be reluctant to leave reviews, especially negative reviews about their friends and neighbors, especially if the review could be traced back to the reviewer. However, complete anonymization may encourage users to leave under-deservedly mean and unhelpful or spam reviews. This is often seen in the comments users leave on content, products or services online, which often requires a human moderator to flag problematic comments. Thus, such reputation management systems often face a struggle balancing anonymization with traceability. A combination of public and private blockchains as suggested in Lisi et al. [47] could be considered in a solution.

References

[1] Xiaolei Yang, Lingyun He, Yufei Xia, and Yufeng Chen. "Effect of government subsidies on renewable energy investments: The threshold effect." In: *Energy Policy* 132 (2019), pp. 156–166.

[2] Weihua Su, Mengling Liu, Shouzhen Zeng, Dalia Štreimikienė, Tomas Baležentis, and Ilona Ališauskaitė-Šeškienė. "Valuating renewable microgeneration technologies in Lithuanian households: A study on willingness to pay." In: *Journal of Cleaner Production* 191 (2018), pp. 318–329.

[3] Axel Gautier, Julien Jacqmin, and Jean-Christophe Poudou. "The prosumers and the grid." In: *Journal of Regulatory Economics* 53.1 (2018), pp. 100–126.

[4] Allison Lantero. "How microgrids work." In: *US Department of Energy* 17 (2014).

[5] Forbes. *The Rising Popularity of Energy Storage as a Service*. https://www.forbes.com/sites/pikeresearch/2019/12/06/the-rising-popularity-of-energy-storage-as-a-service/?sh=223abe19a3a7. Accessed: 2021-04-23. 2021.

[6] Khizir Mahmud, M Jahangir Hossain, and Graham E Town. "Peak-load reduction by coordinated response of photovoltaics, battery storage, and electric vehicles." In: *IEEE Access* 6 (2018), pp. 29353–29365.

[7] Moslem Uddin, Mohd Fakhizan Romlie, Mohd Faris Abdullah, Syahirah Abd Halim, Tan Chia Kwang, et al. "A review on peak load shaving strategies." In: *Renewable and Sustainable Energy Reviews* 82 (2018), pp. 3323–3332.

[8] Wujing Huang, Ning Zhang, Chongqing Kang, Mingxuan Li, and Molin Huo. "From demand response to integrated demand response: Review and prospect of research and application." In: *Protection and Control of Modern Power Systems* 4.1 (2019), pp. 1–13.

[9] Kaveh Paridari, Alessandra Parisio, Henrik Sandberg, and Karl Henrik Johansson. "Demand response for aggregated residential consumers with energy storage sharing." In: *2015 54th IEEE conference on decision and control (CDC)*. IEEE. 2015, pp. 2024–2030.

[10] Tarek AlSkaif, Ioannis Lampropoulos, Machteld Van Den Broek, and Wilfried Van Sark. "Gamification-based framework for engagement of residential customers in energy applications." In: *Energy Research & Social Science* 44 (2018), pp. 187–195.

[11] Satoshi Nakamoto et al. "Bitcoin: A peer-to-peer electronic cash system." In: (2008).

[12] Subhasis Thakur and John G Breslin. "Peer to peer energy trade among microgrids using blockchain based distributed coalition formation method." In: *Technology and Economics of Smart Grids and Sustainable Energy* 3.1 (2018), pp. 1–17.

[13] Ayman Esmat, Martijn de Vos, Yashar Ghiassi-Farrokhfal, Peter Palensky, and Dick Epema. "A novel decentralized platform for peer-to-peer energy trading market with blockchain technology." In: *Applied Energy* 282 (2021), p. 116123.

[14] Sherali Zeadally and Jacques Bou Abdo. "Blockchain: Trends and future opportunities." In: *Internet Technology Letters* 2.6 (2019), e130.

[15] Vitalik Buterin et al. "A next-generation smart contract and decentralized application platform." In: *white paper* (2014).

[16] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. "Hyperledger fabric: a distributed operating system for permissioned blockchains." In: *Proceedings of the thirteenth EuroSys conference*. 2018, pp. 1–15.

[17] Christian Bührer, Ivo Hubli, and Eliane Marti. "The regulatory burden in the Swiss wealth management industry." In: *Financial Markets and Portfolio Management* 19.1 (2005), pp. 99–108.

[18] Paul Makin and Consult Hyperion. "Regulatory Issues Around Mobile Banking." In: *The Development Dimension ICTs for Development Improving Policy Coherence: Improving Policy Coherence* 139 (2010).

[19] Ahmet Önder Gür, Şafak Öksüzer, and Enis Karaarslan. "Blockchain based metering and billing system proposal with privacy protection for the electric network." In: *2019 7th International Istanbul Smart Grids and Cities Congress and Fair (ICSG)*. Ieee. 2019, pp. 204–208.

[20] Zheng Che, Yu Wang, Juanjuan Zhao, Yan Qiang, Yue Ma, and Jihua Liu. "A distributed energy trading authentication mechanism based on a consortium blockchain." In: *Energies* 12.15 (2019), p. 2878.

[21] Statnett. *Elcertificates and guarantees of origin*. https://www.statnett.no/en/for-stakeholders-in-the-power-industry/system-operation/the-power-market/elcertificates-and-guarantees-of-origin/. Accessed: 2021-04-23. 2021.

[22] Zhetao Li, Jiawen Kang, Rong Yu, Dongdong Ye, Qingyong Deng, and Yan Zhang. "Consortium blockchain for secure energy trading in industrial internet of things." In: *IEEE transactions on industrial informatics* 14.8 (2017), pp. 3690–3700.

[23] Keke Gai, Yulu Wu, Liehuang Zhu, Meikang Qiu, and Meng Shen. "Privacy-preserving energy trading using consortium blockchain in smart grid." In: *IEEE Transactions on Industrial Informatics* 15.6 (2019), pp. 3548–3558.

[24] Nurzhan Zhumabekuly Aitzhan and Davor Svetinovic. "Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams." In: *IEEE Transactions on Dependable and Secure Computing* 15.5 (2016), pp. 840–852.

[25] Jacob Goldstein and David Kestenbaum. *The Island Of Stone Money*. `https://www.npr.org/sections/money/2011/02/15/131934618/the-island-of-stone-money?t=1631444682532`. Accessed: 2021-09-12. 2021.

[26] David Chaum. "Blind signatures for untraceable payments." In: *Advances in cryptology*. Springer. 1983, pp. 199–203.

[27] Stuart Haber and W Scott Stornetta. "How to time-stamp a digital document." In: *Conference on the Theory and Application of Cryptography*. Springer. 1990, pp. 437–455.

[28] David Mazieres and Dennis Shasha. "Building secure file systems out of Byzantine storage." In: *Proceedings of the twenty-first annual symposium on Principles of distributed computing*. 2002, pp. 108–117.

[29] Nick Szabo. "Bit gold." In: *Website/Blog* (2008).

[30] Michael Crosby, Pradan Pattanayak, Sanjeev Verma, Vignesh Kalyanaraman, et al. "Blockchain technology: Beyond bitcoin." In: *Applied Innovation* 2.6-10 (2016), p. 71.

[31] Till Neudecker and Hannes Hartenstein. "Short paper: An empirical analysis of blockchain forks in bitcoin." In: *International Conference on Financial Cryptography and Data Security*. Springer. 2019, pp. 84–92.

[32] Mauro Conti, E Sandeep Kumar, Chhagan Lal, and Sushmita Ruj. "A survey on security and privacy issues of bitcoin." In: *IEEE Communications Surveys & Tutorials* 20.4 (2018), pp. 3416–3452.

[33] Christian Gorenflo, Stephen Lee, Lukasz Golab, and Srinivasan Keshav. "FastFabric: Scaling hyperledger fabric to 20 000 transactions per second." In: *International Journal of Network Management* 30.5 (2020), e2099.

[34] Diego Ongaro and John Ousterhout. "In search of an understandable consensus algorithm." In: *2014 USENIX Annual Technical Conference (USENIXATC 14)* (2014), pp. 305–319.

[35] Nikita Karandikar, Antorweep Chakravorty, and Chunming Rong. "Transactive Energy on Hyperledger Fabric." In: *2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*. IEEE. 2019, pp. 539–546.

[36] Nikita Karandikar, Antorweep Chakravorty, and Chunming Rong. "RenewLedger: Renewable energy management powered by Hyperledger Fabric." In: *2020 IEEE Symposium on Computers and Communications (ISCC)*. IEEE. 2020, pp. 1–6.

[37] "Blockchain-based prosumer incentivization for peak mitigation through temporal aggregation and contextual clustering." In: *Blockchain: Research and Applications* (2021).

[38] Nikita Karandikar, Antorweep Chakravorty, and Chunming Rong. "Blockchain Based Transaction System with Fungible and Non-Fungible Tokens for a Community-Based Energy Infrastructure." In: *Sensors* 21.11 (2021), p. 3822.

[39] Stefan Ivanov Sulakov. "The cross-border trade impact on the transmission losses." In: *2017 15th International Conference on Electrical Machines, Drives and Power Systems (ELMA)*. IEEE. 2017, pp. 115–118.

[40] "United Nations Economic Commission for Europe Electricity System Development: A focus on Smart Grids." In: ().

[41] Parth Thakkar, Senthil Nathan, and Balaji Viswanathan. "Performance benchmarking and optimizing hyperledger fabric blockchain platform." In: *2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE. 2018, pp. 264–276.

[42] Parth Thakkar and Senthil Nathan. "Scaling Hyperledger Fabric Using Pipelined Execution and Sparse Peers." In: *arXiv preprint arXiv:2003.05113* (2020).

[43] VK Mehta and Rohit Mehta. *Principles of power system: including generation, transmission, distribution, switchgear and protection*. S. Chand, 2005.

[44] Melike Erol-Kantarci and Hussein T Mouftah. "TOU-aware energy management and wireless sensor networks for reducing peak load in smart grids." In: *2010 IEEE 72nd Vehicular Technology Conference-Fall*. IEEE. 2010, pp. 1–5.

[45] Ausgrid. *Community Batteries*. https://www.ausgrid.com.au/In-your-community/Community-Batteries. Accessed: 2021-02-17. 2021.

[46] Lennart Ante. "Non-fungible token (NFT) markets on the Ethereum blockchain: Temporal development, cointegration and interrelations." In: *Available at SSRN 3904683* (2021).

[47] Andrea Lisi, Andrea De Salve, Paolo Mori, and Laura Ricci. "Practical Application and Evaluation of Atomic Swaps for Blockchain-based Recommender Systems." In: *2020 the 3rd International Conference on Blockchain Technology and Applications*. 2020, pp. 67–74.

Paper I:
Transactive energy on Hyperledger Fabric

Transactive energy on Hyperledger Fabric

N. Karandikar[1], **A. Chakravorty**[1], **C. Rong**[1]

[1] Department of Electrical Engineering and Computer Science, University of Stavanger

Abstract:

As home solar power generation continues to gain popularity, trading excess power within the community is the next logical step. Locally generated power can enrich communities, cut down on transmission losses and provide some measure of energy security. In order to facilitate intra-neighborhood energy transactions, we propose a model that leverages the capabilities of blockchain to provide a transparent, secure and decentralized platform. As the consensus mechanism used must be energy efficient and because the users of the network must be identified and authenticated, in order to build trust and satisfy the Know Your Customer regulations implemented in many countries, a permissioned blockchain is used. Hyperledger Fabric which is a Linux foundation project maintained by almost 200 developers from over 35 organizations is chosen. Hyperledger Fabric has modular architecture allowing the operator to switch out components for others. We propose two architectures for Hyperledger applications and discuss how smart grids in conjunction with Hyperledger Fabric can be used to provide value to prosumers, prosumer communities, Distribution System operators (DSO) and Electric Vehicles (EVs).

1 Introduction

1.1 Microgrids

A microgrid[1] is defined by Department of Energy, USA as a local energy grid with control capability, which means it can disconnect from the traditional grid and operate autonomously. A microgrid may be powered using generators, batteries or through renewable sources like wind or solar power. If the traditional grid is unable to supply power to the consumers, perhaps due to a weather event or due to repairs being underway, a microgrid operating in island mode can help provide some degree of energy security. Moreover, if the microgrid is powered by renewable sources like solar energy, it may help cut costs and be environmentally friendly. The increasing popularity of home solar power generation [2] has given rise to a new kind of electricity consumer- the prosumer. The prosumer produces electricity in their home to meet some of the need and buys from the grid as needed. Prosumers would see a reduction in their utility bills as they produce a portion of the energy they consume and this financial incentive could encourage them to participate in the microgrid. This concept of using a renewable and clean energy resource would resonate with the ethically conscious members of the community in addition to the other benefits it offers. Prosumers who produce more energy than they need could explore avenues for monetizing on the excess energy or storing it.

1.2 Smart Grids

The smart grid is defined in the United Nations Economic Commission for Europe Report as follows “A SmartGrid is an electricity network that can intelligently integrate the actions of all users connected to it – generators, consumers, and those that do both – in order to efficiently deliver sustainable, economic and secure electricity supplies.”[3] A smart grid allows a bidirectional flow of electricity data enabling real time data collection of energy demand and supply, making monitoring, maintenance, energy generation and energy consumption more efficient. Moreover, the integration of EVs into the

infrastructure has led to the increase in demand for electricity but also presents an opportunity. EVs can be seen as additional mobile batteries which can fill up when energy prices are low, and discharge as needed to power other devices in the home. If the electricity used to charge the EVs is generated using renewables, it furthers the goal of environment protection and sustainability. A project of this magnitude raises questions about the logistics of the transactions and a need to make them transparent and secure. As with any system that deals with automated transactions between parties, a blockchain implementation could provide the ledger that the prosumers may use to track their energy trades between neighbors in a completely transparent way leading to a frictionless trading system [4].

1.3 Blockchain

Blockchain was developed in 2008 to support the development of Bitcoin [5], a new cryptocurrency and it tackled many of the issues that had caused previous attempts at digital currency to be unsuccessful. By implementing blockchain as an immutable and public ledger, Bitcoin could mitigate the risk of double spending without the need for a central authority or trust between transacting parties. The unique features of blockchain have generated interest among researchers and there has been focused research on the possibility of leveraging these features to other domains and use cases [6].

Blockchain is a shared, distributed ledger used to record transactions by multiple untrusting nodes in a network. An identical copy of the ledger is maintained at each node of the network so any case in which a node unilaterally tries to change a transaction can be detected and rejected by other nodes in the network. The ledger maintains sequentially ordered transactions as a sequence or chain of blocks such that each block holds the hash of the previous block, thus making the blockchain immutable and verifiable.

Blockchain networks can be categorized as permissionless or permissioned. Permissionless or public blockchain networks such as Bitcoin or Ethereum [7] allow anyone to join and perform transactions on the system. As there is no trust between the transacting parties, computation intensive consensus mechanisms like Proof of Work [5]

are used. A permissioned network does not allow unknown entities to participate in the network. Each node in the network is authenticated and each transaction is traceable to the node that performed it. Moreover, a permissioned network can have access control mechanisms in place to define which nodes can propose transactions, have read-/write access or accept new nodes into the network. As processing transactions does not require nodes to expend considerable resources, it is also not dependent on a cryptocurrency to incentivize nodes to run smart contracts or to validate transactions. As there is no cryptocurrency to steal, the risk of attack by a malicious party is reduced. Mining, one of the biggest contributors to operation and energy cost is avoided. Many countries now require businesses to identify and validate customers and perform due diligence on the potential risks for illegal activity by means of Know your customer (KYC) and anti-money laundering (AML) regulations [8].

Businesses that transact with each other form a business network in the real world. Each business maintains its own separate record of the transactions. These centralized databases containing unique information each present a single point of failure. Moreover, when a transaction takes place, the assets that are being transacted must have their provenance established to ensure that the party trading it does in fact have the title of ownership to that asset. Establishing provenance in these diverse systems is time consuming and laborious.

Permissioned networks provide a unified system of managing identities of network participants and tracing provenance in a business network. The immutable sequence of transactions stored in the blockchain enable the system to quickly establish the provenance of any asset. Instead of many loosely coupled centralized systems that each have disparate ways of transacting with each other, businesses may find value in a system that spans a business network bringing trust to untrusted parties and visibility in the network. Smart contracts [9] can be used to create self-enforcing agreements between transacting parties and express complex data models.

1.4 Hyperledger Fabric

Hyperledger Fabric [10] is an enterprise level permissioned blockchain platform which has a modular design that supports pluggable configurations for many components. This allows the operator to choose the format for transaction data, the consensus mechanism and to tailor the trust model and identity management protocols to the application. Smart contracts, or chaincode as it is known in Hyperledger Fabric parlance is not required to be written in a Domain specific language, as popular general purpose programming languages such as Java, Go and Node.js are supported. This encourages easy adoption by programmers familiar with these languages. Hyperledger Fabric has seen a high degree of adoption and is currently being used in many use cases such as SecureKey [11] and Everledger [12]. The rest of the paper is organized as follows, Section 2 presents an overview of Hyperledger Fabric. In Section 3 we discuss the proposed architectures. We discuss the potential applications of a blockchain based transactive platform in Section 4. In Section 5 we discuss our work in the context of related works. We conclude the paper in Section 6.

2 Hyperledger Fabric Architecture

2.1 Key Components of Hyperledger Fabric

The end user submits requests to the blockchain through the Client nodes. Client nodes connect to peer nodes to communicate with the blockchain, create transaction proposals and submit transaction invocations to endorsing peers. The endorsement policy is defined to stipulate which and how many endorsing peers must agree on a transaction before it can be committed to the ledger. The client is also responsible for collecting all the endorsements from the peers and sending a well-formed transaction to the ordering service to be included in a transaction block and then delivered to all the peers for validation and commit. The ordering service also communicates with the client node after the transaction is added to the ledger. Ordering service nodes (OSN) are responsible for taking endorsed transactions, putting them in the correct order and then broadcasting them to

the peer nodes. Peer nodes receive ordered transaction blocks and maintain the state and the commit blocks to the ledger. A peer may be an endorsing peer in addition to being a committing peer. A chaincode has an endorsement policy defined which stipulates which and/or how many endorsing peers must endorse the transaction before it is committed. Peer nodes store the latest state of the blockchain as versioned key-value stores in a persistent implementation and updates to the state are logged. All valid transactions are ordered by the orderer nodes. A ledger is maintained at the peer nodes and also at some orderer nodes to record all successful state changes. The ledger is created as an ordered hashchain of blocks with each block holding an array of ordered transactions. In the architecture, the trust model for chaincodes is decoupled from the trust model for ordering. For each chaincode, a different set of endorsing nodes can be specified and an entirely different set of nodes can provide the ordering service in a crash tolerant manner. Parallel or simultaneous chaincode execution is possible when chaincodes refer to a disjoint set of endorsers. Moreover, as chaincode execution can be computation intensive, if it is not executed by the OSN, it will make these nodes available exclusively to provide ordering service, thus reducing the load on them. Thus, the system would be easier to scale than if these services were provided by the same nodes. The architecture is built to be modular and allow pluggable ordering service implementations.

2.2 Transaction flow in Hyperledger Fabric

Before any transactions can be proposed, a channel must be set up and running and the application that intends to request the transaction must be registered and authenticated by the organization‘s Certificate Authority (CA). A chaincode must be installed on the peers with an endorsement policy set and instantiated on the channel prior to being requested. Instead of the more widely used order-execute transaction model, Fabric uses a execute-order-validate model that is run simultaneously at the endorsing peers. The transaction flow in Hyperledger Fabric as shown in Figure 1 has the following steps: 1. A client in the network initiates a transaction and sends a proposal in order to invoke a chaincode function to read from or write to

the ledger. The proposal is sent to one or more endorsing peers in the network depending upon the endorsement policy defined in the chaincode. The client application uses a Software Development Kit (SDK) to construct a transaction proposal in the correct format. This proposal is then signed by a unique cryptographic signature created by using the user‘s credentials. 2. The endorsing peers check to make sure that the proposal is not duplicate or replayed and that it is well formed. They verify the validity of the signature on the proposal and that the client submitting the proposal has the authority to perform the requested transaction. Each endorsing peer for this transaction then executes the chaincode function requested against the current state and sends the signed proposal response composed of the generated read-write set and a response value back to the application. 3. The application then verifies the endorsing signatures on the response and compares the responses received. If the transaction was only a read request the application would not need to send the transaction to the orderer. In case of a ledger update request, the application must verify that the endorsement policy was met before submitting the transaction to the orderer. Even if the application decides not to verify the responses or tries to submit a transaction that is not correctly endorsed, the peers will verify endorsement during commit validation. 4. The application assembles and then broadcasts the transaction proposal which includes the read/write sets, the endorsements and the channel ID. The ordering service receives the transactions for each channel, puts them in a chronological order and creates blocks. 5. All the peers on the channel receive the transaction blocks from the orderer. They then validate the transactions to ensure that the endorsement policy requirements were met and the read set variables on the ledger have not been changed since the set was created. As a result of this validation, the transactions are marked as valid or invalid. 6. All the peers on the channel append the transaction block to their chain and the state database is updated atomically. The client application that proposed the transaction is notified of transaction completion.

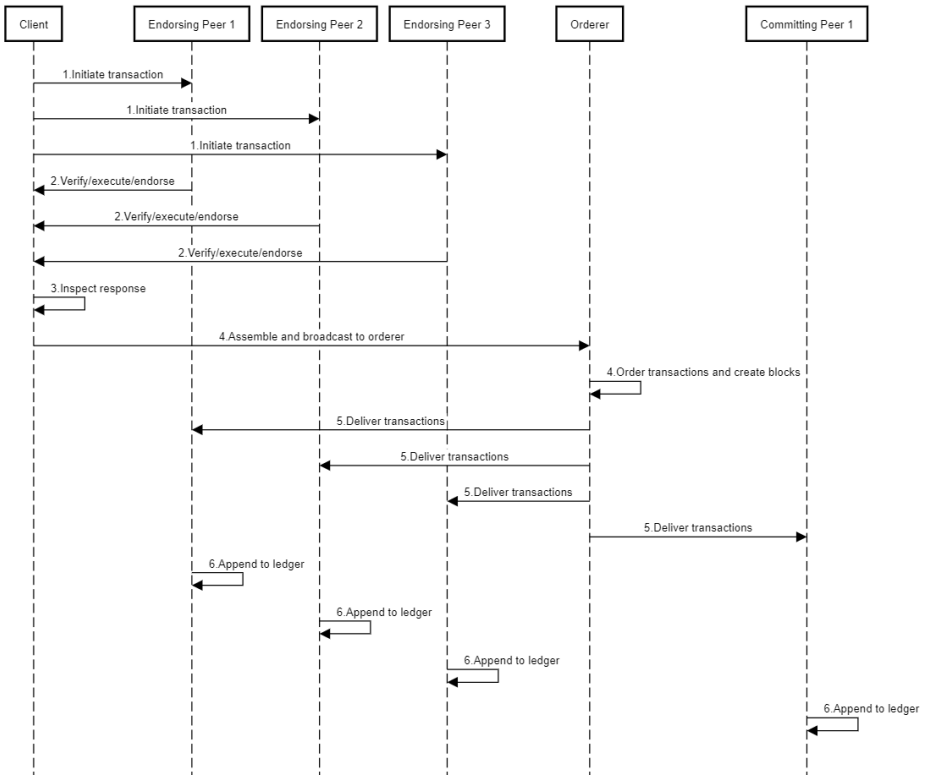


Figure 1: Transaction flow in Hyperledger Fabric

3 Proposed Architectures

We examine two different ways to configure the orderer. The orderer consists of Orderer service nodes and an orderer component. In architecture 1 as shown in Figure 2, the orderer is implemented as a separate organization backed by a Kafka [13]-Zookeeeper [14] cluster with 4 Kafka nodes and 5 Zookeeper nodes as recommended in the Fabric documentation [15].

Kafka is a Publish-Subscribe model message handling system that guarantees that all messages are sequentially ordered. Peers can subscribe to the service to receive new messages published by the Producer. Kafka stores all messages for a set amount of time or until a maximum size threshold. Peers can poll Kafka for transactions they want to read and this ensures that when a crashed node comes back

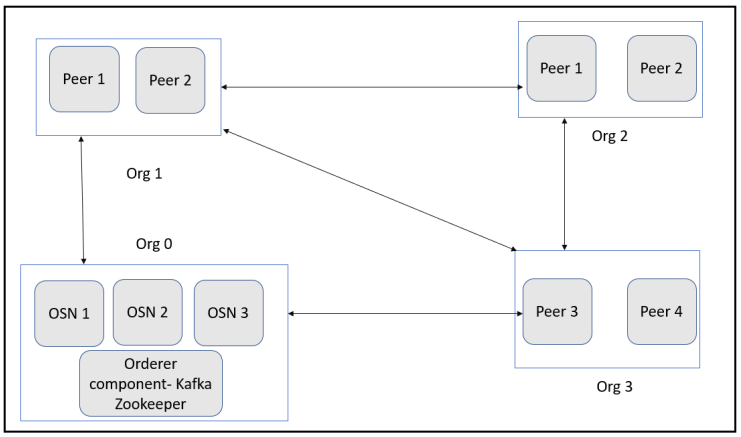


Figure 2: Orderer nodes in a separate organization

online, it is able to recover the lost data. Zookeeper is a distributed key-value store that is commonly used to store metadata. It allows the client of the Kafka service to subscribe and enables changes to be sent to them as they happen. The use of Kafka and Zookeeper makes the network crash tolerant. To ensure that no one organization has complete control over the ordering process, the orderer is not part of any of the organizations. Instead, the orderer organization is formed by representatives from the other organizations and is administered by a trusted third party. For scalability, incoming blocks from the orderer get relayed to a leader peer in each organization which then communicates the blocks to the other peer using gossip protocol. If at any time, the leader peer goes down, the other peer takes over, thus ensuring high availability. Each organization has an anchor peer which is discoverable by other anchor peers and the orderer. In our model, we keep the anchor peer and the leader peer the same in each organization for the sake of simplicity, but this is not necessary. Each organization has its own CA that issues certificates to components. These certificates are used by components to identify themselves and their organizations to each other. Transaction requests and responses also use certificates to digitally sign. We create two intermediate CAs in each organization- one for the client nodes and the other for the peer nodes so that the identity also specifies the type of node. Both CAs subscribe to the channels to which the organization subscribes,

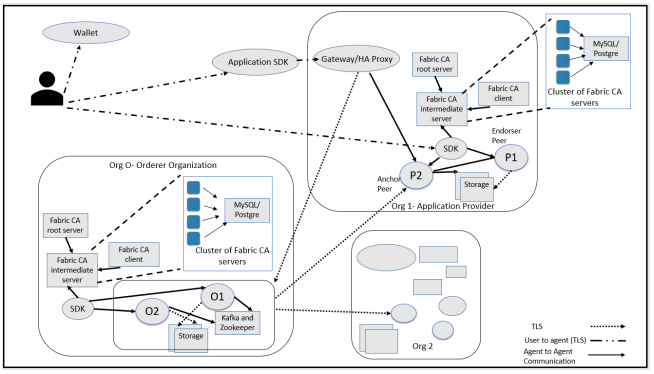


Figure 3: Orderer nodes in a separate organization- Transaction Flow

but endorsement policies would only refer to the CA certificates from the intermediate CA that refers to peers. In this way, endorsements are guaranteed to have come from the endorsing peers only and not from ordering peers or clients. At each peer we store the state in a CouchDB [16] database as recommended in the Hyperledger Documentation [17] as it supports rich querying and rich data types as required in many business use cases. As shown in Figure 3, the user begins by accessing their wallet which contains their identity. The identity used in conjunction with the CA to verify the users rights to access the ledger and other channel resources.

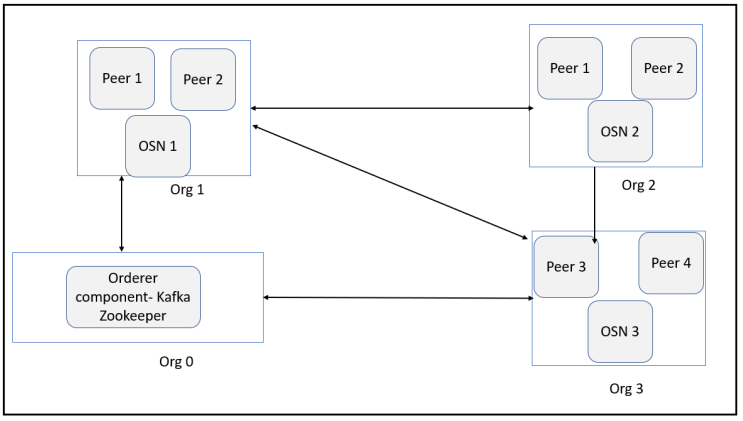


Figure 4: Orderer nodes in each organization

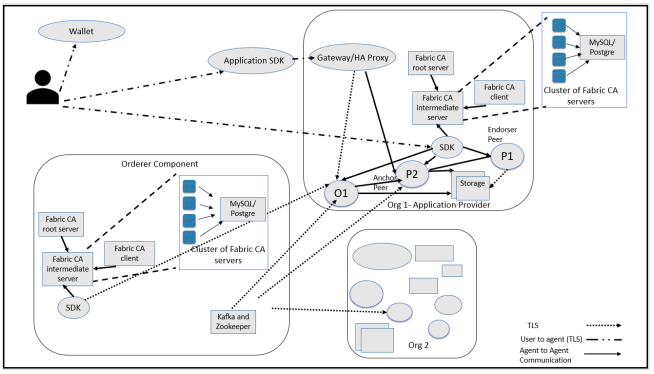


Figure 5: Orderer nodes in each organization- Transaction Flow

A transaction request is formed using the Application SDK and submitted to the anchor peers through the Gateway. A gateway is responsible for managing the network interactions on behalf of the application. A Hyperledger application can change due to planned outages or be scaled up or down depending on business need so the gateway is needed to handle the network topology changes for the application. The anchor peers send the request to other peers using gossip protocol. The peers execute the transaction locally and endorse it and send the response back to the Client application. The Client application then broadcasts the endorsed transaction. The orderer service nodes in the orderer organization then order the transaction request in the correct order and broadcast it to the anchor peers who communicate it to the other peers in the organization. The committing peers then validate the transaction to ensure the endorsement requirements were met and add this transaction to their ledgers. In the second architecture as shown in Figure 4, each organization has an OSN but the orderer component with the Kafka-Zookeeper implementation is still a separate component and is not controlled by any organization.

The transaction flow as shown in Figure 5 progresses in the same way as described for Architecture 1 with the salient difference that for intra organization transactions where the endorsement of other organizations is not required, the ordering can be done by the OSN local to that organization. So, in case of intra organization transactions,

this architecture may help reduce overhead.

4 Use cases

Introduction of smart grids into the energy infrastructure has a great potential to provide value to the consumer and the DSOs as well. Transacting the excess energy generated by the prosumers can lead to a variety of use cases. We explore some of them below:

4.1 Value To Prosumers

Prosumers that produce a surplus of energy at any given time could sell it to other members of the community. As shown in Figure 6, the seller can use the platform user interface (UI) to list the number of units of electricity available for sale and the price per unit. Similarly, the buyer could submit a request to the platform to buy electricity and input their requirements such as units of energy needed and price offered. The system can generate recommendations for the buyer who can choose a seller and place a bid. If the bid is accepted, the seller supplies energy to the buyer and the buyer pays for it with a token.

4.2 Value To DSO

Integration of EVs into the grid has increased the demand for electricity and with it, the frequency of peak periods [18]. Peaker plants [19] are run in order to balance the load on the grid. These plants, usually fossil fuel powered are only maintained for use in the peak periods in order to balance the load on the grid. This is not cost effective for DSOs to use and maintain, hence the interest in load balancing and demand response strategies. These strategies would help with peak shaving, reducing variance and flattening out of the usage curve by reducing peak usage and increasing off peak usage. For DSOs to effectively set the Time of Use (ToU) [19] pricing they need to be able to predict the load profile of the grid for the usage period. The transactive data generated on the microgrid is essential for them to

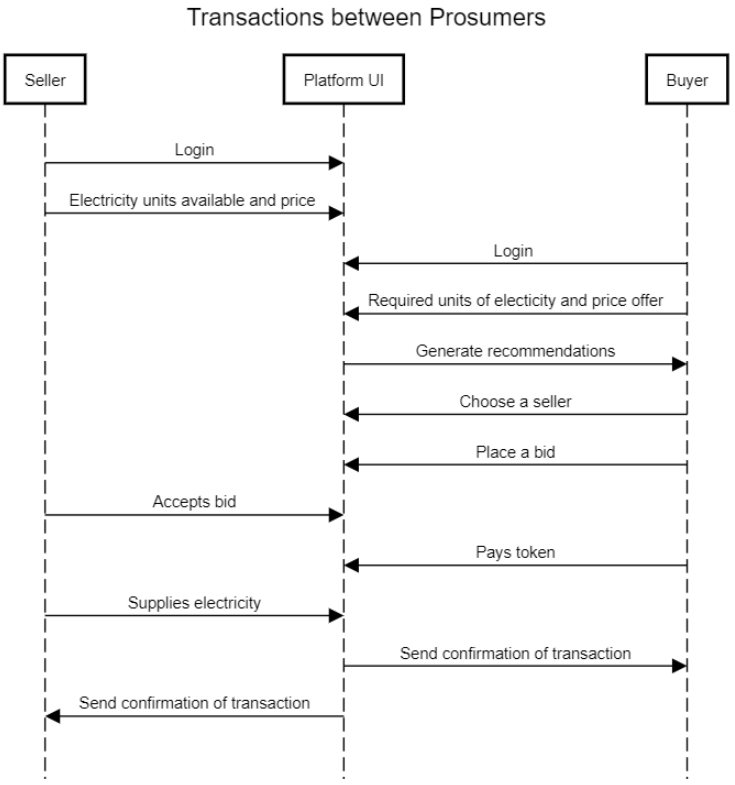


Figure 6: Transactions between Prosumers

make these predictions accurately. It is also in the interest of the consumer that the predictions are as accurate as possible for system reliability and reduction in cost. The goal of these strategies is to financially incentivize desired behavior and disincentivize undesired behavior. Figure 7 shows the sequence diagram for interactions between the Prosumer and the Platform UI for data collection for load balancing.

Aside from ToU pricing, other incentivization strategies can also be implemented to directly reward consumers for desired behavior. Gamification [20]- integration of game elements into non game scenarios can also be used to improve user engagement. The user can login to the platform and view the games available for the week, such as no heating on a sunny day or no running the washer or dryer between 5pm to 8pm. Each game will have a reward for participating and a

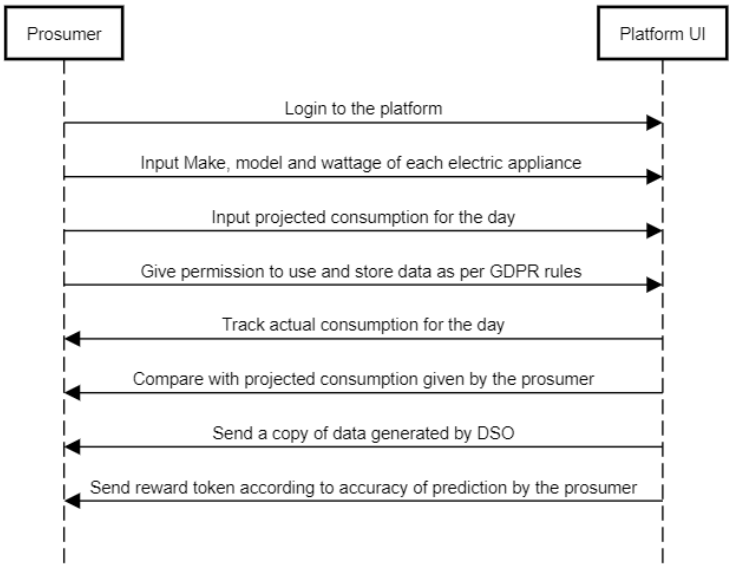


Figure 7: Data collection for Load Prediction

penalty for reneging on a commitment to play the game. At the end of the game playing window, the user is rewarded for games played as promised after deducting penalties for reneging as the case may be. Figure 8 shows the sequence diagram for the interaction between the Prosumer and Platform UI for gamification and incentivization for load balancing.

4.3 Value To prosumer community

Solar energy generation depends on the weather and is not constant. So, as energy storage solutions get cheaper and more efficient [21], prosumers who produce excess energy at a given time can store the surplus for later. Stored energy can also mitigate the effect of ToU pricing strategies implemented by DSOs to reduce load on the grid in peak times. Storage implemented on a community level would enable the prosumers in the community to pool their surplus energy and this storage can also act as charging stations for EVs for consumers that are not members of the community. Moreover, this stored energy can be directed to critical systems, such as hospitals in case of a power

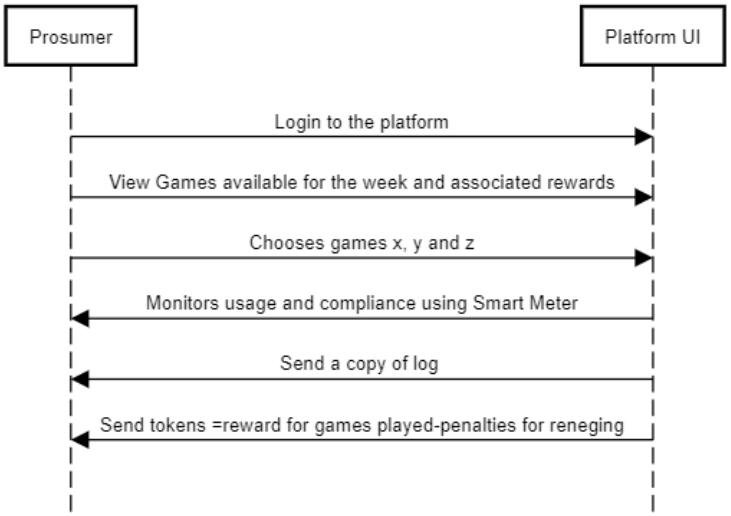


Figure 8: Gamification and Incentivization for Load Balancing

outage to support lifesaving efforts. Energy storage as a service is a burgeoning new utility business [22]. Instead of buying, housing and maintaining the energy storage facility, customers of this service can rent storage capacity for a period by signing a contract and paying a monthly or yearly fee that covers the aforementioned costs as well as guarantees 24/7 reliability for no asset investment. Companies that provide such a service include Constant Power Inc. [23] and GI Energy [24]. As this model progresses beyond the nascent stages, we envision a pricing model akin to many pay-as-you-go cloud based pricing [25] models. In the current scenario, as a subscription fee is required upfront, the platform can convert payment of this fee into tokens on the platform, such that storage tokens are given out to the users in proportion to the amount they paid for the subscription cost that month. These storage tokens would expire in a month and new ones would be introduced the following month. Prosumers can trade the storage tokens among themselves for use of the storage facility. Figure 9 shows the interactions between a user of this storage service and the Platform.

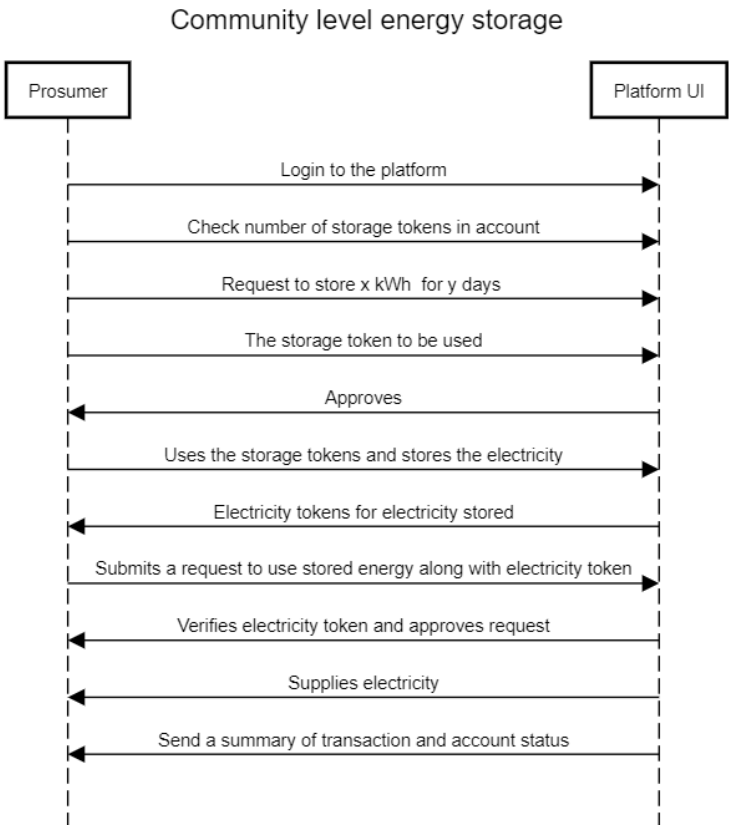


Figure 9: Community level energy Storage

4.4 Value To EVs

EV owners could capitalize on the EV batteries during periods they are not in use such as during vacations, office hours or at night. EV batteries can be integrated into the grid and can be rented out and be used to store or supply energy to a community. Owners can be incentivized in exchange for participation by giving them tokens which can be redeemed for free parking or a deduction in their monthly utility bills. Figure 10 shows the interaction between EV owners and the Platform for Battery sharing.

Community level energy storage or the prosumer‘s home batteries can serve as private EV charging stations. EV owners can login to the platform and place a request for the energy units needed, price

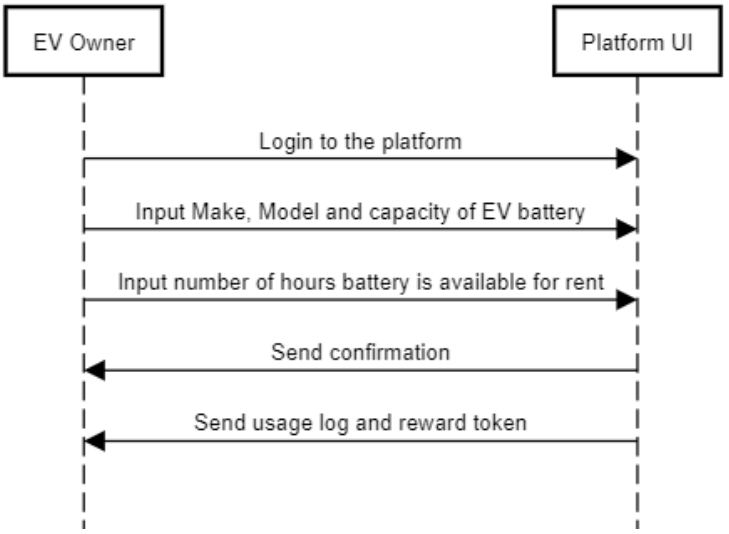


Figure 10: Battery Sharing

they would be willing to pay, along with their location and the radius they would be willing to drive to charge. The platform generates recommendations based on the requirements. The EV then drives to that location pays the token to the seller and charges the vehicle. The platform then sends both parties notification of a completed transaction. Figure 11 shows the interaction between the EV owner and the Platform UI for EV charging.

5 Related Works

Several works that discuss the use of blockchain technologies to build transactive microgrids were studied. Wang, et al. [26] have discussed transactive microgrids in their work which proposes a decentralized electricity transaction mode for microgrids based on blockchain and continuous double auction (CDA) mechanism in which the buyer and seller initially complete the transaction matching in the CDA market and an adaptive aggressiveness strategy is used to adjust the quotation according to market changes. Pop, et al. [27] proposed a demand response solution that uses a blockchain to store the prosumption

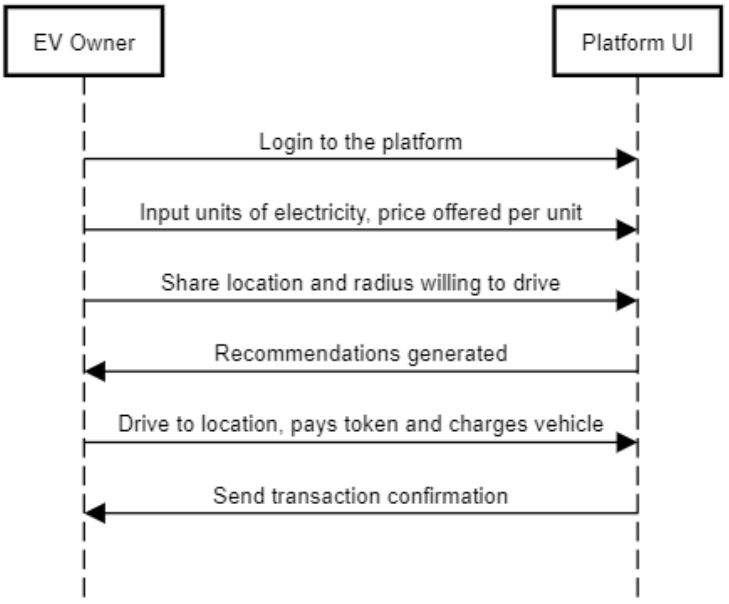


Figure 11: EV charging

information collected from Internet of Things (IoT) devices and using smart contracts to define the expected energy flexibility on the part of the prosumer and the rewards or penalties associated. Mengelkamp, et al. [28] in the Brooklyn microgrid project have conducted a pilot project that covers a 10 block radius in Brooklyn. Their work and implementation details however are not open access so it is difficult to comment on the specifics of their implementation or architecture. Sabounchi and Wei [29] use the public Ethereum blockchain platform to implement their peer to peer (P2P) electricity marketing mechanism, by defining a smart contract to create a well-defined auction. We have chosen the Hyperledger Fabric platform for reasons explained in the section 1.3 and 1.4 and our work does not investigate the setting up of the auction. Plaza, et al. [30], Pee, et al. [31] and Xue, et al. [32] discussed a high level design without going into the specifics of the architecture and the former two are focused on the Ethereum platform. Lavrijssen and Carrillo Parra [33] identify the innovations in the electricity market and analyze the legal and non-legal obstacles to prosumer and consumer empowerment.

They discuss the integration of demand response and P2P trading in the energy market. Jogunola, et al. [34] focusses on communication architectures for prosumer energy trading and contrasts a structured P2P protocol with an unstructured one. Bergquist, et al. [35] have focused on transactional anonymity in peer to peer transactions. Our work describes an in-depth architecture for setting up a Hyperledger Fabric based transactive energy platform for authenticated users which incorporates traceability and reduces operation costs due to the absence of costly mining operations.

6 Conclusion

In this work, we discussed the applicability of Blockchain to build a transactive microgrid and presented our rationale for our choice of blockchain platform. We then presented two architectures for building a Hyperledger Fabric based application and discussed how such an application used in conjunction with a smart grid has the potential to bring value to prosumers, DSO, EVs and prosumer communities.

References

[1] Allison Lantero. "How microgrids work." In: *US Department of Energy* 17 (2014).

[2] European Photovoltaic Industry Association et al. "Global market outlook for photovoltaics 2014-2018." In: *EPIA, Brussels* (2014).

[3] "United Nations Economic Commission for Europe Electricity System Development: A focus on Smart Grids." In: ().

[4] Roman Beck. "Beyond bitcoin: The rise of blockchain world." In: *Computer* 51.2 (2018), pp. 54–58.

[5] Satoshi Nakamoto et al. "Bitcoin: A peer-to-peer electronic cash system." In: (2008).

[6] Michael Crosby, Pradan Pattanayak, Sanjeev Verma, Vignesh Kalyanaraman, et al. "Blockchain technology: Beyond bitcoin." In: *Applied Innovation* 2.6-10 (2016), p. 71.

[7] Vitalik Buterin et al. "A next-generation smart contract and decentralized application platform." In: *white paper* (2014).

[8] Paul Makin and Consult Hyperion. "Regulatory Issues Around Mobile Banking." In: *The Development Dimension ICTs for Development Improving Policy Coherence: Improving Policy Coherence* 139 (2010).

[9] Konstantinos Christidis and Michael Devetsikiotis. "Blockchains and smart contracts for the internet of things." In: *Ieee Access* 4 (2016), pp. 2292–2303.

[10] "Hyperledger Fabric Documentation Release 1.4." In: *Hyperledger* (2019).

[11] A Nunez Mencias, D Dillenberger, P Novotny, F Toth, TE Morris, V Paprotski, J Dayka, T Visegrady, B O'Farrell, J Lang, et al. "An optimized blockchain solution for the IBM z14." In: *IBM Journal of Research and Development* 62.2/3 (2018), pp. 4–1.

[12] "Everledger Emerging technology solutions for real world challenges." In: *Everledger Ltd Documentation* (2018).

[13] "Apache Kafka- A distributed streaming platform." In: *Apache Kafka Documentation* (2017).

[14] "Apache Zookeeper." In: *Apache Zookeeper Documentation* (2018).

[15] "Hyperledger Fabric- Bringing up a Kafka Based Ordering Service." In: *Hyperledger* (2019).

[16] "CouchDB-Seamless multi-master sync, that scales from Big Data to Mobile,with an Intuitive HTTP/JSON API and designed for Reliability." In: *Apache CouchDb Documentation* (2019).

[17] "Hyperledger Fabric- CouchDB as a State Database." In: *Hyperledger* (2019).

[18] Kejun Qian, Chengke Zhou, Malcolm Allan, and Yue Yuan. "Modeling of load demand due to EV battery charging in distribution systems." In: *IEEE Transactions on Power Systems* 26.2 (2011), pp. 802–810.

[19] Melike Erol-Kantarci and Hussein T Mouftah. "TOU-aware energy management and wireless sensor networks for reducing peak load in smart grids." In: *2010 IEEE 72nd Vehicular Technology Conference-Fall*. IEEE. 2010, pp. 1–5.

[20] Sebastian Deterding, Miguel Sicart, Lennart Nacke, Kenton O'Hara, and Dan Dixon. "Gamification. using game-design elements in non-gaming contexts." In: *CHI'11 extended abstracts on human factors in computing systems*. ACM. 2011, pp. 2425–2428.

[21] Björn Nykvist and Måns Nilsson. "Rapidly falling costs of battery packs for electric vehicles." In: *Nature climate change* 5.4 (2015), p. 329.

[22] Fereidoon Sioshansi. "New trend: storage-as-a-service." In: *Energy Post* (2018).

[23] "Constant Power energy Storage as a Service (ESaaS)." In: *Constant Power Documentation* (2019).

[24] "GIEnergy: Reliable, Resilient, Clean and Cost-Effective Energy Infrastructure Solutions." In: *GI Energy Documentation* (2019).

[25] Shadi Ibrahim, Bingsheng He, and Hai Jin. "Towards pay-as-you-consume cloud computing." In: *2011 IEEE International Conference on Services Computing*. IEEE. 2011, pp. 370–377.

[26] Jian Wang, Qianggang Wang, Niancheng Zhou, and Yuan Chi. "A novel electricity transaction mode of microgrids based on blockchain and continuous double auction." In: *Energies* 10.12 (2017), p. 1971.

[27] Claudia Pop, Tudor Cioara, Marcel Antal, Ionut Anghel, Ioan Salomie, and Massimo Bertoncini. "Blockchain based decentralized management of demand response programs in smart energy grids." In: *Sensors* 18.1 (2018), p. 162.

[28] Esther Mengelkamp, Johannes Gärttner, Kerstin Rock, Scott Kessler, Lawrence Orsini, and Christof Weinhardt. "Designing microgrid energy markets: A case study: The Brooklyn Microgrid." In: *Applied Energy* 210 (2018), pp. 870–880.

[29] Moein Sabounchi and Jin Wei. "Towards resilient networked microgrids: Blockchain-enabled peer-to-peer electricity trading mechanism." In: *2017 IEEE Conference on Energy Internet and Energy System Integration (EI2)*. IEEE. 2017, pp. 1–5.

[30] Caroline Plaza, Julien Gil, Francois de Chezelles, and Karl Axel Strang. "Distributed Solar Self-Consumption and Blockchain Solar Energy Exchanges on the Public Grid Within an Energy Community." In: *2018 IEEE International Conference on Environment and Electrical Engineering and 2018 IEEE Industrial and Commercial Power Systems Europe (EEEIC/I&CPS Europe)*. IEEE. 2018, pp. 1–4.

[31] Seung Jae Pee, Eung Seon Kang, Jae Geun Song, and Ju Wook Jang. "Blockchain based smart energy trading platform using smart contract." In: *2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*. IEEE. 2019, pp. 322–325.

[32] Lei Xue, Yunlong Teng, Zhenyuan Zhang, Jian Li, Kunbing Wang, and Qi Huang. "Blockchain technology for electricity market in microgrid." In: *2017 2nd International Conference on Power and Renewable Energy (ICPRE)*. IEEE. 2017, pp. 704–708.

[33] Saskia Lavrijssen and Arturo Carrillo Parra. "Radical prosumer innovations in the electricity sector and the impact on prosumer regulation." In: *Sustainability* 9.7 (2017), p. 1207.

[34] Olamide Jogunola, Augustine Ikpehai, Kelvin Anoh, Bamidele Adebisi, Mohammad Hammoudeh, Haris Gacanin, and Georgina Harris. "Comparative analysis of P2P architectures for energy trading and sharing." In: *Energies* 11.1 (2017), p. 62.

[35] Jonatan Bergquist, Aron Laszka, Monika Sturm, and Abhishek Dubey. "On the design of communication and transaction anonymity in blockchain-based transactive microgrids." In: *Proceedings of the 1st Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers*. ACM. 2017, p. 3.

Paper II: RenewLedger : Renewable energy management powered by Hyperledger Fabric

RenewLedger : Renewable energy management powered by Hyperledger Fabric

N. Karandikar[1], **A. Chakravorty**[1], **C. Rong**[1]

[1] Department of Electrical Engineering and Computer Science, University of Stavanger

Abstract:
Trading and storage of renewable energy offers a way for the prosumer to extract value from the surplus energy that they produce, while also mitigating energy shortfall. Power companies can enlist prosumers in demand response strategies for grid stability and cost savings. We present RenewLedger, a blockchain-based framework for renewable energy transaction, storage management and direct-to-consumer demand response incentivization and gamification for peak shaving. We design and implement this system using Hyperledger Fabric and report on performance benchmarking experiments conducted using Hyperledger Caliper.

1 Introduction

The benefits of integrating renewable energy into our energy infrastructure are well established. Governments and individuals are showing an increased interest in renewable energy, especially solar energy, as awareness rises [1]. Prosumers are a new type of energy consumer that generate some of the energy they use by means of renewable sources like solar energy. Solar energy, however, is uncertain and depends on the weather. At times, the prosumer may have surplus energy they cannot use, other times, the prosumer may have a shortfall. Trading excess energy with other prosumers or storing it can help ease some of this uncertainty. This requires the formulation of a solution to facilitate transactions between prosumers and to manage energy storage.

Blockchain [2] was born out of one of the most successful attempts at building a cryptocurrency. Bitcoin, [2] which was introduced in 2008, used an append-only decentralized shared ledger called block chain which was composed of blocks of transactions linked together in a chain. As this was a public system allowing anyone to join and perform transactions, many features were implemented to make malicious activity difficult. Bitcoin ensured ordering of blocks by cryptographically linking each block to its predecessor by storing a hash of the previous block in each block. Moreover, the ledger existed on many nodes simultaneously, so any node unilaterally changing its own copy would not be accepted in the network. In order to get the privilege of adding transactions, nodes race to perform a computationally intensive operation called Proof of Work which requires time and extensive resources to calculate but is trivial to verify. Nodes are incentivized to contribute resources to the network. The ledger developed by the creators of Bitcoin has received interest in industry and academia as practitioners find it fulfils other use cases [3].

There are two broad types of blockchain networks: permissioned and permissionless. Permissionless networks such as Bitcoin are open to all and anyone is permitted to propose transactions on the system. Permissioned networks are formed of authenticated nodes with clearly defined access privileges in the network. Organizations that transact

with each other in a business network usually have their own disparate systems. While this centralized system gives them more control, it becomes a single point of failure. As the systems are not necessarily built to function seamlessly with each other, establishing provenance of an asset being traded becomes a laborious task. A permissoned blockchain network can provide a unified identity management and provenance tracing system. Consensus in a permissioned blockchain network is achieved based on the agreed upon endorsement policy, allowing organizations to simulate and endorse transactions concerning them before they are processed. This allows them to do away with energy intensive consensus mechanisms like Proof of Work and cryptocurrencies to incentivize nodes to contribute computational resources. Permissioned networks also help companies fulfil the Know your customer (KYC) and anti-money laundering (AML) obligations [4] imposed by various governments. Hyperledger Fabric [5] is one of the most widely used enterprise level permissioned blockchain platforms. It is an open source Linux Foundation project and has a development committee of over 200 developers and 35 organizations. It has a modular design allowing operators to tailor the implementation to the use case by supporting different implementations of consensus, membership management and transaction data format. Self enforcing smart contracts can be coded in popular languages such as Java, Go and Node.Js, thus eliminating the need for a developer to learn a new domain specific language. Moreover, developers are not restricted to predefined tokens for transactions and an asset can be defined as anything of value.

In our previous work [6], we presented a theoretical basis for a blockchain based energy trading and storage management system. In this work, we implement such a system and conduct performance benchmarking experiments for various scenarios. We use Hyperledger Caliper [7], which is another project under the Hyperledger umbrella, to conduct our benchmarking experiments. Caliper uses the Common Connection Profile (CCP) of the Fabric Software Development kit (SDK) that allows it to implement complex scenarios and take advantage of many Fabric features.

The rest of the paper is organized as follows. Section II presents an overview of system, in Section III we discuss the implementation and

in Section IV we present and analyse the results of our experiments. We present our work in the context of related works in Section V and we conclude in Section VI.

2 Overview of System

We consider the following capabilities of the system:
1. Transacting between Prosumers
2. Prosumer Incentivization
3. Prosumer gamification
4. Managing community level energy storage transactions
5. Managing use of Electric Vehicle (EV) battery as storage
6. Managing EV battery charging transactions

2.1 Application entities

2.1.1 Trader

The Trader is a user of the system who wishes to transact units and does so using the Trading Platform. The Trader can perform activities such as transacting surplus energy with other Traders and EVs, using storage facilities for excess energy by participating in a community level storage or storing energy in EV batteries rented out for this purpose. The Trader can also earn tokens by participating in demand response incentivization and gamification tasks offered by the Power Company.

2.1.2 Trading Platform

The Trading Platform lists available tokens for bidding on or transacting with. We consider bidding as expressing binding interest in a token without negotiating the price, but this can be tailored to the use case. The Trading Platform provides a client for Traders to interact with the network and facilitates transactions. The Trading Platform can include representatives of auditory bodies or government entities to monitor or administer the system.

2.1.3 Power Company

The Power Company can directly reach prosumers to aid its demand response strategies by incentivizing them for their participation. They can offer direct incentives to the prosumer for accomplishing tasks such offering up their projected consumption for the given time period, agreeing to being monitored by the Power Company for compliance and receiving reward tokens for tasks completed or having the agreed value taken away as penalty for reneging. Another way the Power Company can enlist the prosumer in their peak shaving efforts is to offer a list of games, which are tasks with associated rewards and penalties presented in a gamified context. Games can include tasks like not running the air conditioning on a particularly hot day or moving dishwasher or laundry machine use to off peak hours. Also, if the Power Company has access to the data about energy stored in the Community level storage, it could help inform their demand response strategy.

2.1.4 Electric Vehicle

The electric vehicles can buy surplus energy from the prosumers to charge their batteries. They can also earn reward tokens by renting out the EV battery to be used as storage devices when not in use.

2.1.5 Storage as a Service Provider

The Storage as a Service provider (hereafter Storage Provider) handles all the tasks involved with setting up and managing a community level storage for surplus energy and abstracts out the intricacies of these activities for the users.

2.2 Architecture

2.2.1 Organizations

Organizations in the Hyperledger Fabric architecture logically map the different organizations in our system. The Trading Platform, the Power Company and the Storage Provider are the three organizations

in our architecture. Each organization has two peers for redundancy. The leader peer is the peer that receives blocks from the orderer and then transmits them to the other peers in the organization using gossip protocol. The leader peer in each organization is chosen through election and peers are load balanced.

2.2.2 Clients

We setup one client for each organization. The Trading Platform client is used by the Traders to transact energy units, participate in gamification and incentivization tasks and for storing surplus energy. The Power Company uses its client to post gamification and incentivization tasks and to process the rewards. The Storage Provider organization processes storage requests and rewards using its client.

2.2.3 Orderer Organization

The ordering service nodes in the Hyperledger Fabric network are responsible for putting all the transactions in the correct order and creating blocks of transactions for the peers to validate and commit. As this is an important function, the ordering service nodes are not under the control of any one organization but are members of an orderer organization. This is a separate organization administered by certificate authorities from all other organizations.

2.2.4 Certificate Authorities

A certificate authority (CA) is present in each organization. The CA issues identity certificates to member components which components use to identify each other. The identity determines a user's access within the channel.

2.2.5 Chaincodes

The blockchain developer encodes the business logic of the application into the chaincode. A chaincode can have many smart contracts that cover the governance rules for different interactions between the

transacting parties. The chaincode must be installed and instantiated on the channel before it is called. When specifying a chaincode in the CCP, we specify target peers on which this chaincode will be installed. For instance, as the trading of energy units between prosumers should only involve the Traders and the Trading Platform, we specify the target peers of this chaincode to be members of the Trading Platform organization. Another option is to create separate channels for each set of trading relationships, but this was not explored in this work.

3 Implementation

3.1 Tokens

We consider six different types of tokens in our application reflecting the six use cases mentioned in Section II. We see the relationship between the different application entities and the tokens in Figure 1. Each token has a key field and value field containing four values. The values depend on the functionality for which the token is used. Updating and Querying the world state are time consuming operations and so to improve performance, we have chosen LevelDB which is the more performant choice [8]. LevelDB is a key value storage that does not provide rich querying so we use the key of the asset in order to store important information.

3.1.1 Token key

The token key is composed of the serial number of the token and the token type and has the format :serial number-token type. There are 6 types of tokens.

3.1.2 Types of Tokens

The following are the types of tokens
1. EUnit, EVUnit:
We use the EUnit token to represent energy units traded between two prosumers and the EVUnit token to represent energy units sold by a prosumer to an EV. The values in these tokens:

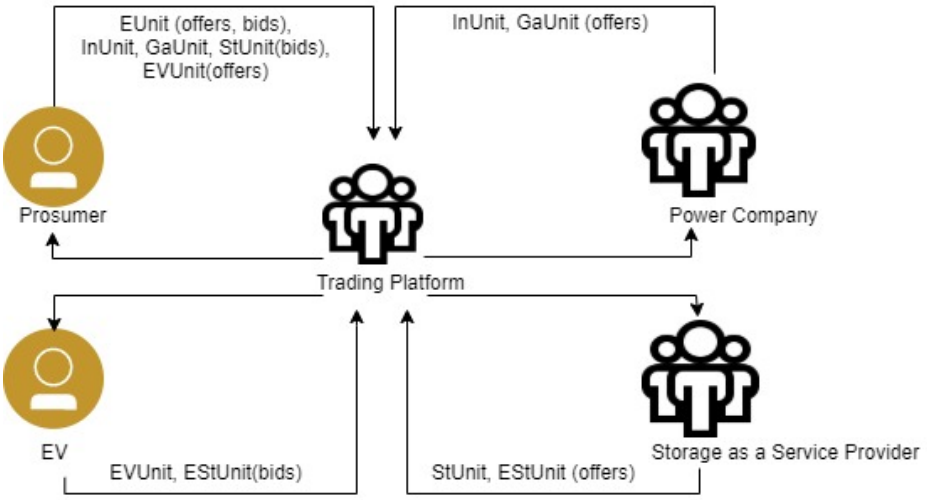


Figure 1: Relationship between Application Components

a)price- the price of the token
b)location- location of the seller
c)value- amount of electricity in the token
d)bid-true/false showing whether a user has bid on this unit
2. InUnit, GaUnit:
The InUnit and GaUnit tokens are created by the Power Company for Incentivization and Gamification respectively and are offered to the user. The values in these token:
a)reward- incentive for completing the prescribed task
b)penalty- for agreeing and then failing to complete the task
c)requirement- what the task entails
d)bid- true/false has a user chosen this particular token
3. StUnit, EStUnit :
The StUnit tokens are used by the user to access the community level storage facility in order to store their surplus energy. The community level storage facility may also include unused EV batteries. The values in these token:
a)price- to store a given amount of energy
b)value- amount of energy storage offered with this token
c)conditions - of storage

d)bid- true/false has a user chosen this particular token
The EStUnit tokens are reward tokens generated by the Storage Provider in order to supplement its storage capacity with EV batteries. The values in these token:
a)reward -associated with participating
b)amount- of energy that will be stored, or capacity required
c)conditions - of storage
d)bid- true/false has a user chosen this particular token

4 Experiments

4.1 Setup

Our experiments were conducted using Hyperledger Fabric V1.4.6 to build the system under test and Hyperledger Caliper V0.2 was the performance benchmarking tool. We used Ubuntu 16.04 on a machine with an Intel Core i7 processor, 4 cores CPU, 32 GB RAM, 256 GB SSD.

4.1.1 Experimental Parameters

Endorsement Policy: 1 member of each organization to endorse each transaction.
Consensus: RAFT with 3 Ordering Service Nodes [9]
We conducted two types of operation for each network configuration: read and write. We chose send rates in Transactions per second (TPS) based upon our infrastructure capabilities:
a) Read Operation:
Total transactions performed for each data point: 1000
Send Rate (in TPS): 25, 50, 100, 200, 400
b) Write Operation:
Total transactions performed for each data point: 500
Send Rate (in TPS): 5, 10, 20, 40, 80

4.2 Transaction flow and Performance

4.2.1 Tokens involving 1 Organization

Figure 2, shows the transaction flow for EUnit and EVUnit. Transacting electricity units between a buyer and a seller requires both users to be registered on the Trading Platform before submitting a transaction proposal. This transaction would involve a read from the world state by the buyer to see what Units are available for purchase and the associated price. An update transaction would then be initiated to bid on the Unit. The seller would read the state and initiate sale. This would involve two updates to the world state: the ownership of the token will be transferred from the seller to the buyer and a payment token to be transferred from the buyer to the seller. The only organization involved in this transaction is the Trading Platform. The user will initiate a transaction, whether read or write/update using a Client that uses an Application SDK to construct a well formed transaction proposal and signs it by generating a unique signature using its credentials stored in its wallet. This proposal is then sent to the appropriate number of endorsement peers as specified in the endorsement policy. In this case it only goes to the Trading Platform organization, which verifies that the proposal is well-formed and signed correctly by a user with the access to request this operation. It executes it against the world state and sends the generated read/write set back to the client with an endorsement. If the operation is a Read, then the transaction flow stops here. If the operation requested is a write/update, the client inspects the response to see if the endorsement policy was met and broadcasts it to the orderer which orders the transactions, creates blocks of transactions and sends them to the peer organizations. The peers verify the endorsements, that the read/write set has not been changed since creation and then update their world state and commit the block to their ledger.

Figure 3, shows the performance of a read operation involving a single organization. The throughput increases with increase in send rate and even at 400 TPS send rate, the throughput is close to the send rate. This shows that the system has the potential to process even higher send rates. The read latency stays well below 1 second for

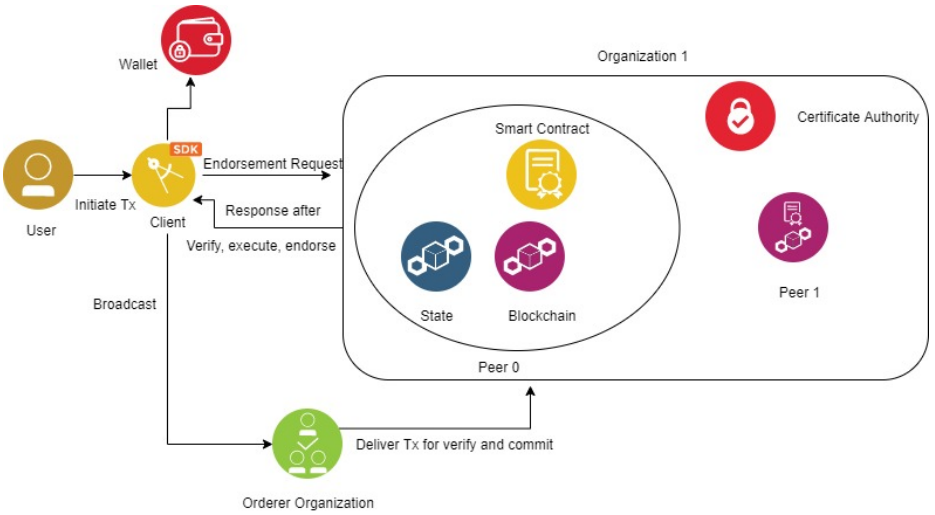


Figure 2: Transaction Flow involving 1 Organization: EUnit, EVUnit

all data points shown and interestingly, it reduces slightly for higher send rates. The Hyperledger Caliper tool automatically load balances between the available peers. The endorsement at the peers can run concurrently, allowing different peers to handle different transactions.

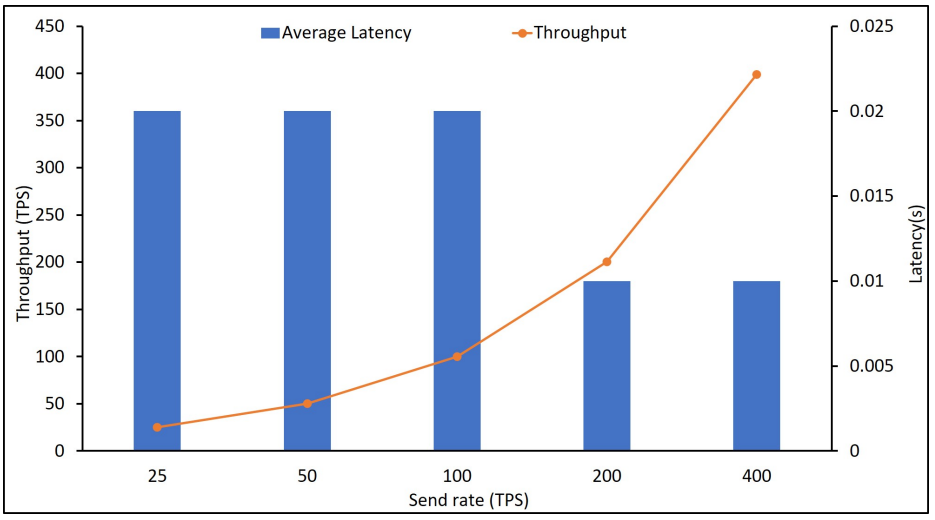


Figure 3: Read Operation involving 1 Organization: EUnit, EVUnit

Figure 4, shows the performance profile for a write operation

involving a single organization. The throughput of the write operation is lower that that of the read operation and the latency is higher. The write operation involves more steps as explained earlier in this subsection. The ordering operations and ledger update are time consuming operations leading to a higher latency for write operation as compared to read.

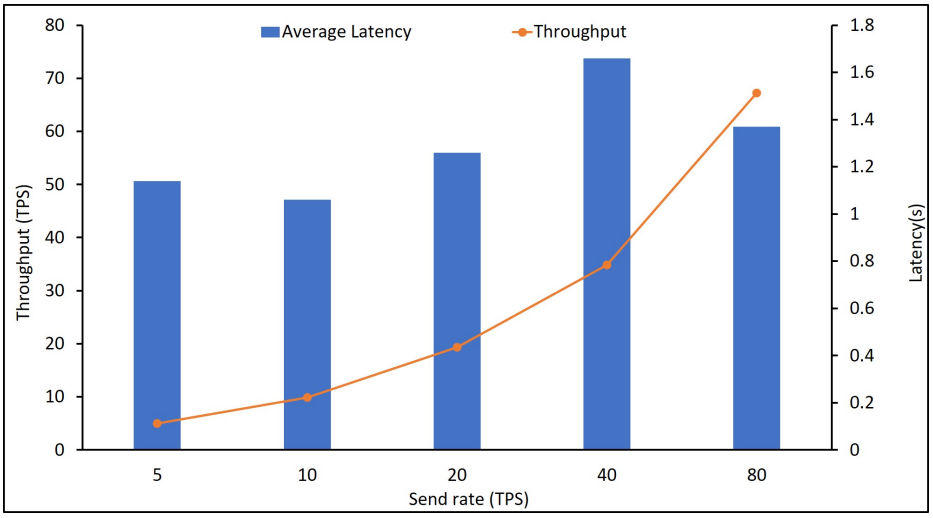


Figure 4: Write Operation involving 1 Organization: EUnit, EVUnit

4.2.2 Tokens involving 2 Organization

Figure 5, shows the transaction flow for InUnit and GaUnit. In this scenario, the Trading Platform and the Power Company are the two Organizations involved and one peer from each organization must approve each transaction. The client, must thus be authenticated by the Certificate Authorities of both organizations and both organizations will receive an endorsement request from the client. A user from the Power Company would access the network using the Client associated with that organization and perform write transactions, adding in different InUnit and GaUnit tokens as needed by the organization for its demand response strategy. The Trader or user will interact with the system using the Trading system Client and perform read requests to see what tokens are available and the associated rewards

for playing correctly, penalties for reneging and conditions. The user will then perform write/update requests bidding upon the InUnits and GaUnits, they are interested in. The Power Company reads these bids and monitors compliance and then initiates a change of ownership write request for the tokens. The business logic to ascertain whether a change of ownership will happen and the particulars of that transaction would be encoded in the chaincode and would depend on the specific implementation and their business needs and would impact performance based on their complexity. We focus on the actual interactions whether read or write with the blockchain network that these transactions would boil down to once these determinations are made.

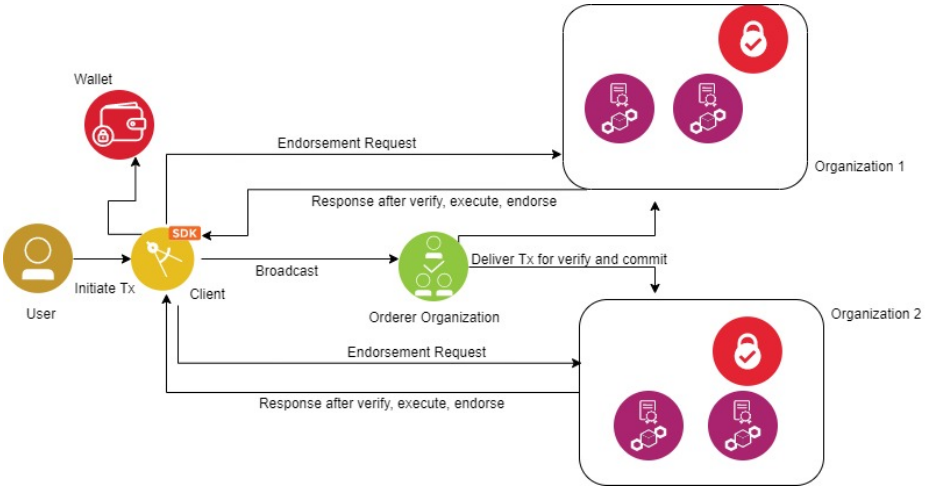


Figure 5: Transaction flow involving 2 Organizations: InUnit, GaUnit

Figure 6 shows the performance of Read operation involving two organizations. The throughput is close to the send rate in this case and the latency is very low, under 1 second in each case studied, showing that the system is not yet saturated. However, it does have a higher latency and slightly lower throughput as compared to the corresponding one organization values. As the number of Organizations increase, the number of endorsements required for each transaction increase, impacting the performance.

Figure 7, shows the performance of the Write operation involving

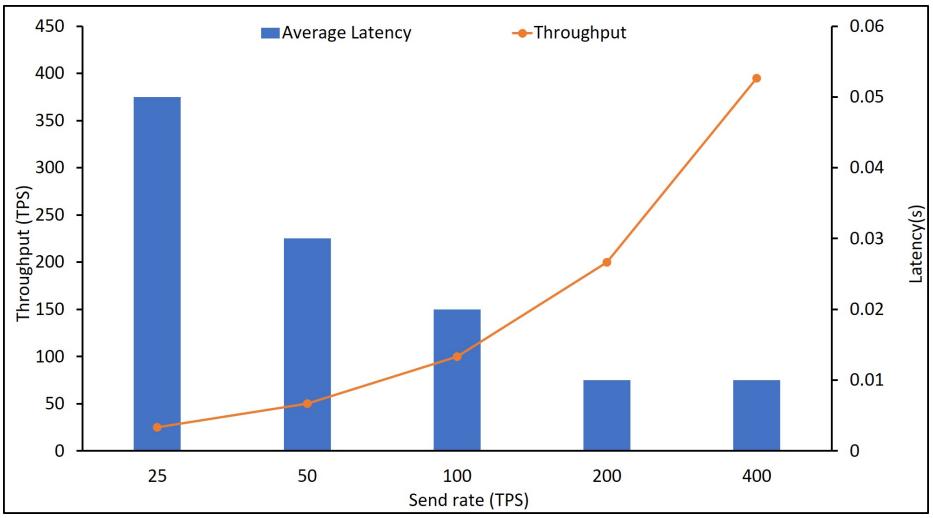


Figure 6: Read Operation involving 2 Organizations: InUnit, GaUnit

two organizations. The latency increases sharply at 40 TPS send rate from 1.44 s to 3.03 s. The throughput still increases with the increase in send rate. In this case, the number of endorsements required and also the number of peers that must validate and commit each transaction increases compared to the Write operation involving 1 Organization. Thus, this case has a higher latency and lower throughput as compared to corresponding the one organization write operations.

4.2.3 Tokens involving 3 Organization

Figure 8, shows the transaction flow for StUnit and ESTUnit tokens. Three Organizations are involved in this scenario- Trading Platform, Storage Provider and Power Company. The Storage Provider would subsume the EV battery in its community level storage facility and abstract this detail from the prosumer. The Prosumer is looking to store surplus units of electricity and conducts a read operation to check how many StUnits they have which can be used to rent storage capacity with the Storage as a Service Provider. The prosumer would then initiate a write operation to transfer the ownership of the StUnit to the Storage Provider. The Power Company could be an active

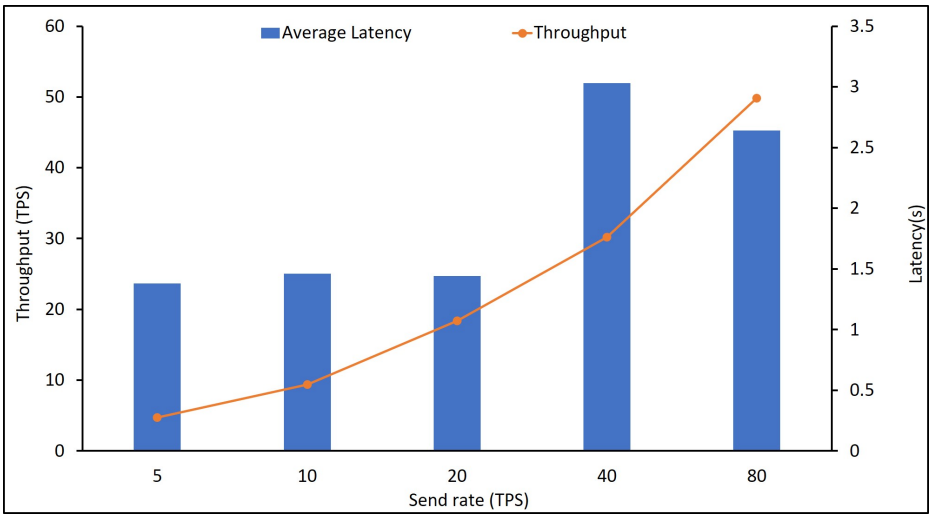


Figure 7: Write Operation involving 2 Organizations: InUnit, GaUnit

member of this scenario by providing endorsing peers or it could just have its peers be committing peers. The Power Company would then have data on not only the stored energy at all times (world state) but also the data for past storage (ledger data) and could factor it into their demand response calculations. This would be dictated by the exact business use case and relationship. In our experiments, we consider that all three organizations have endorsing peers and that the endorsement policy requires one peer of each organization to endorse each transaction.
The EV owner would earn ESTUnits from the Storage Provider for their participation in the community level storage facility. This transaction would start when the EV owner submits a write operation by bidding on one of the offered ESTUnits. The Trading Platform, the Storage Provider and if applicable, the Power Company must endorse this. When the task is successfully completed, all three organizations endorse a proposal to transfer ownership of the earned ESTUnit to the EV Owner.

Figure 9 shows the performance of Read Operation for a transaction involving three Organizations. Compared to the read operation for one and two organizations, the latency is higher and throughput is lower in each case. At 400 TPS send rate, the latency starts to rise

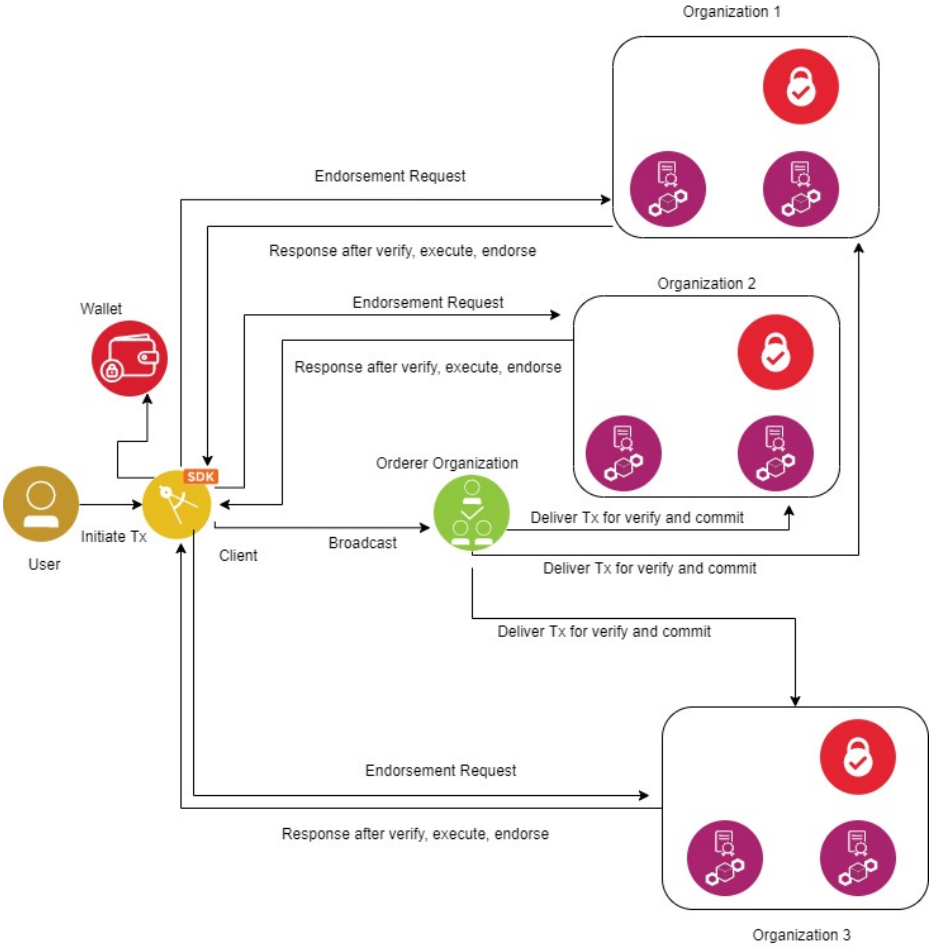


Figure 8: Transaction Flow involving 3 Organizations: StUnit,ESTUnit

again due to the bottleneck of the number of available endorsement peers. The latency while higher for this configuration was still well below 1 second and the effect of change in send rate was miniscule.

Figure 10 shows the performance of a Write Operation involving three organizations. The latency increases rapidly from 2.01 seconds at 20 TPS send rate to 4.37 seconds at 40 TPS send rate and the throughput while still increasing begins to level off. The write operation for three organizations had the highest latency and lowest throughput of all the configurations presented.

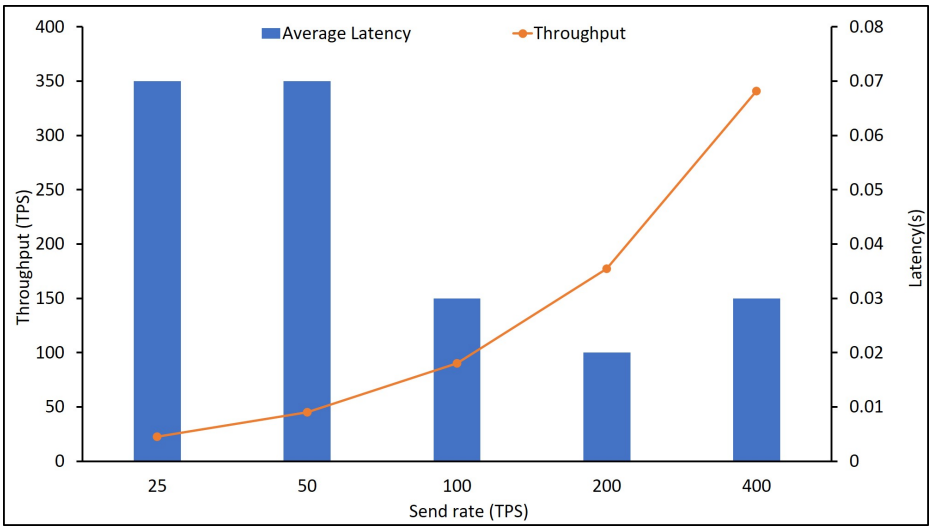


Figure 9: Read Operation involving 3 Organizations: StUnit, ESTUnit

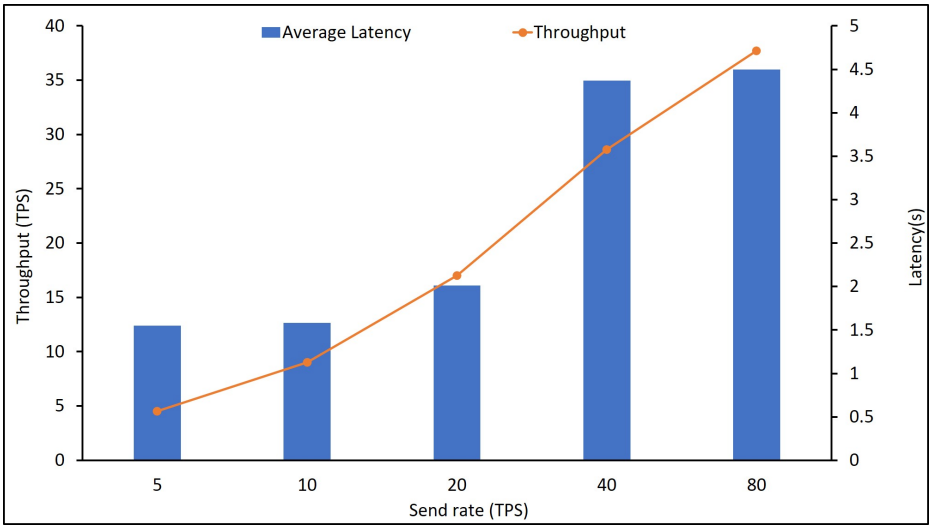


Figure 10: Write Operation involving 3 Organizations: StUnit, ESTUnit

5 Related Works

Pipattanasomporn et al. [10] and Jogunola et al. [11] have published their respective articles discussing energy trading on the blockchain. Their works target different use cases from ours and their implemen-

tations were built using Hyperledger Composer which has now been deprecated. Hyperledger Fabric provides a much more customizable platform for building blockchain solutions than Composer. Long et al. [12] investigated three market paradigms to reduce costs for customers trading energy in a community microgrid. Park et al. [13] developed their solution on the IBM Blockchain Platform which is a proprietary platform and is not free or open source. Also, they have targetted different use cases than we have. Saxena et al. [14] explore the impact of various bidding strategies used by energy consumers and present a market price clearing algorithm. Our work presents a framework for energy management implemented using Hyperledger Fabric.

6 Conclusion

In this paper, we presented RenewLedger, a Hyperledger Fabric based framework for renewable energy management. We extended our previous work 'Transactive energy on Hyperledger Fabric' and explained the design and implementation of this system in detail. We also conducted benchmarking experiments to evaluate the performance of the system with read and write operation for transactions involving one, two and three organizations. We found that, for our implementation, read operations were executed with latency under 1 second and throughput close to send rate for all three configurations for send rates up to 400 TPS. The write operations showed significant difference in latency based on the configuration and went from under 2 sec for 1 Organization to up to 4.5 for three organizations. The throughput also decreased from almost 70 TPS in 1 Organization for a 80 TPS send rate to under 40 TPS for 3 Organizations for the same send rate.

References

[1] European Photovoltaic Industry Association et al. "Global market outlook for photovoltaics 2014-2018." In: *EPIA, Brussels* (2014).

[2] Satoshi Nakamoto et al. "Bitcoin: A peer-to-peer electronic cash system." In: (2008).

[3] Michael Crosby, Pradan Pattanayak, Sanjeev Verma, Vignesh Kalyanaraman, et al. "Blockchain technology: Beyond bitcoin." In: *Applied Innovation* 2.6-10 (2016), p. 71.

[4] Paul Makin and Consult Hyperion. "Regulatory Issues Around Mobile Banking." In: *The Development Dimension ICTs for Development Improving Policy Coherence: Improving Policy Coherence* 139 (2010).

[5] "Hyperledger Fabric Documentation Release 1.4." In: *Hyperledger* (2019).

[6] Nikita Karandikar, Antorweep Chakravorty, and Chunming Rong. "Transactive Energy on Hyperledger Fabric." In: *2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*. IEEE. 2019, pp. 539–546.

[7] *Hyperledger Caliper Documentation*. 2020. URL: `2020://hyperledger.github.io/caliper/vLatest/getting-started/`.

[8] Parth Thakkar, Senthil Nathan, and Balaji Viswanathan. "Performance benchmarking and optimizing hyperledger fabric blockchain platform." In: *2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE. 2018, pp. 264–276.

[9] Diego Ongaro and John Ousterhout. "In search of an understandable consensus algorithm." In: *2014 USENIX Annual Technical Conference (USENIXATC 14)* (2014), pp. 305–319.

[10] Manisa Pipattanasomporn, Murat Kuzlu, and Saifur Rahman. "A blockchain-based platform for exchange of solar energy: Laboratory-scale implementation." In: *2018 International Conference and Utility Exhibition on Green Energy for Sustainable Development (ICUE)*. IEEE. 2018, pp. 1–9.

[11] Olamide Jogunola, Mohammad Hammoudeh, Bamidele Adebisi, and Kelvin Anoh. "Demonstrating Blockchain-Enabled Peer-to-Peer Energy Trading and Sharing." In: *2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE)*. IEEE. 2019, pp. 1–4.

[12] Chao Long, Jianzhong Wu, Chenghua Zhang, Lee Thomas, Meng Cheng, and Nick Jenkins. "Peer-to-peer energy trading in a community microgrid." In: *2017 IEEE power & energy society general meeting*. IEEE. 2017, pp. 1–5.

[13] In Hwan Park, Sung Jun Moon, Beom Suk Lee, and Ju Wook Jang. "A P2P Surplus Energy Trade Among Neighbors Based on Hyperledger Fabric Blockchain." In: *Information Science and Applications*. Springer, 2020, pp. 65–72.

[14] Shivam Saxena, Hany Farag, Aidan Brookson, Hjalmar Turesson, and Henry M Kim. "Design and Field Implementation of Blockchain Based Renewable Energy Trading in Residential Communities." In: *arXiv preprint arXiv:1907.12370* (2019).

Paper III: Blockchain-based prosumer incentivization for peak mitigation through temporal aggregation and contextual clustering.

Blockchain-based prosumer incentivization for peak mitigation through temporal aggregation and contextual clustering.

N. Karandikar[1], R. Abhishek[2], N. Saurabh[3], Z. Zhao[4], A. Lercher[3], N. Marina[5], R. Prodan[3], C. Rong[1], A. Chakravorty[1]

[1] Department of Electrical Engineering and Computer Science, University of Stavanger

[2] Department of Energy and Petroleum Engineering, University of Stavanger

[3] Institute of Information Technology, University of Klagenfurt

[4] Multiscale Networked Systems, University of Amsterdam

[5] University of Information Science and Technology, Ohrid, Macedonia

Abstract:
Peak mitigation is of interest to power companies as peak periods may require the operator to over provision supply in order to meet the peak demand. Flattening the usage curve can result in cost savings, both for the power companies and the end users. Integration of renewable energy into the energy infrastructure presents an opportunity to use excess renewable generation to supplement supply and alleviate peaks. In addition, demand side management can shift the usage from peak to off peak times and reduce the magnitude of peaks. In this work, we present a data driven approach for incentive based peak mitigation. Understanding user energy profiles is an essential step in this process. We begin by analysing a popular energy research dataset published by the Ausgrid corporation. Extracting aggregated user energy behavior in temporal contexts and semantic linking and contextual clustering give us insight into consumption and rooftop solar generation patterns. We implement, and performance test a blockchain based prosumer incentivization system. The smart contract logic is based on our analysis of the Ausgrid dataset. Our implementation is capable of supporting 792,540 customers with a reasonably low infrastructure footprint.

1 Introduction

Integration of renewable energy, especially solar energy into energy infrastructure is on the rise, driven in part by the economic benefits such as government incentives and money saved on energy bills and in part due to rising awareness of the environmental benefits [1]. Prosumers are a category of consumers who generate part of the energy they need through their on site micro generation devices and buy the remainder from the energy grid as needed [2]. Several prosumers living in close proximity to one another can form prosumer communities or microgrids [3]. Prosumers can use the generated solar energy for their own needs or if there is a surplus, sell it to the grid or other customers.

Peak periods [4] are periods when the demand of energy is the highest in a given time frame. Peak demand is rising as a result of an increasing number of retail users [5]. Maintaining grid stability in presence of variation in demand, especially during peak demand periods is an important task for the grid operator [6]. If peak demand approaches the available grid capacity, grid operators must take measures in order to maintain grid stability and reliable supply. This can be accomplished either by increasing available supply or reducing the peak load. Increasing available supply to match the projected peak usage value requires the operator to over-provision generation capacity, which can be expensive. This additional capacity is only used during peak periods and often takes the form of peaker plants [7] that are often coal or diesel powered and thus polluting. Moreover, the operation and maintenance costs of these peaker plants that are only used some of the time increase the price per kWh of energy, a cost that is usually passed on to the consumer.

The process of reducing the magnitude of the peak is called peak shaving. Peak shaving is of interest to grid operators and customers as it offers the potential for cost reduction by either deferring or avoiding investments in additional capacity. Surplus solar energy if sold back to the grid presents an opportunity to supplement energy supply during peak energy demand periods. Surplus solar energy that is generated during off peak periods, can be stored in a battery infrastructure to discharge as needed. Pilot studies have been conducted on using

grid connected battery systems [8] to reduce peaks. Local solar energy production has the advantage of being co located with the consumption sites, thus reducing transmission losses inherent in transporting electricity over large distances [9].

Another important peak shaving strategy is demand response or demand side management [10]. Traditionally, user demand was considered inelastic and supply was largely structured around demand. Now, while demand certainly continues to drive supply, there is value found in regulating demand in order to shift some of the peak time usage to off peak times, in order to flatten the usage curve and reduce the required capacity of energy infrastructure. Demand side response to peak consumption can take the form of increased prices to disincentivize consumers from running shiftable or non urgent appliances during peak times. Another approach is the use of incentivization tokens which is a scheme under which the prosumer can earn tangible benefits for reducing usage at peak times.

In this paper we present a data driven approach for incentive based peak shaving by using a two pronged strategy.

(1) Firstly, energy consumers who record consumption below a calculated threshold during identified peak consumption periods are awarded reward tokens.

(2) Secondly, the top surplus producing prosumers in the network are rewarded according to the amount of surplus they produce.

Net production or surplus production in a given time period is defined as:

$$Surplus = Production - Consumption \tag{1}$$

We chose for our analysis, the Ausgrid dataset [11] published by the Ausgrid corporation, which offers one year of energy generation and consumption data collected from smart meters installed on site for 300 random and anonymized customers in their network. The prosumer reward management system is implemented on a Hyperledger Fabric [12] blockchain in order to be transparent and decentralized. Further, the system is performance tested under varying loads using Hyperledger Caliper [13].

The work has the following structure:

(1) Section 2 presents the design rationale and the salient building blocks of the solution.

(2) An aggregation analysis of temporal energy behavior presented in section 3 : a) identifies periods of peak usage and high variation b) identifies thresholds for categorizing net producers based on surplus values.

(3) A semantic analysis of energy behaviour in order to identify thresholds for low, medium and high categories for energy consumption is discussed in section 4.

(4) The system is implemented on a blockchain and the sections 3 and 4 inform its logic which is encoded in smart contracts. The design, implementation and performance characterization of the incentivization system are discussed in section 5.

(5) Section 6 discusses how our solution builds upon the state of the art, while section 7 presents the salient conclusions of this study.

2 Proposed system

2.1 System participants and requirements

This system involves three distinct entities or organizations. First, the user platform is composed of prosumer representatives and perhaps a government agency to ensure legal compliance. The second organization is the power company that monitors the user generation and production and offers the rewards. The third organization is the grid battery, where the energy generated by the prosumers is stored and quality checked before sending it to its destination. Pilot studies [14] are currently underway to study peak shaving through the use of community batteries. Prosumers can connect through the grid network and store their surplus in the community battery owned and maintained by the Power company. By participating in a community battery infrastructure, prosumers have the opportunity to earn credits towards their electricity bills and thus get more value out of their

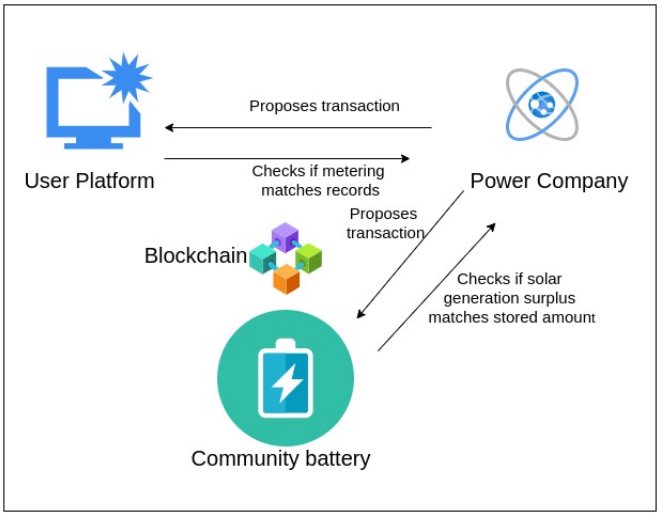


Figure 1: Participants of the system.

solar investment without needing to own and maintain individual battery systems. The power company and the user platform must be in agreement about the logic for calculation of rewards. As the user platform represents the prosumers, it must also have the opportunity to check and approve all the transactions against its own calculations as shown in figure 1. Moreover, the storage provider is responsible for storing the generated energy and hence must verify that the amount of energy generated shown by the smart meter is actually generated and available for use. Thus, all three organizations must agree upon the business logic and approve each transaction, and the reward system must be transparent to all parties involved.

As the ecosystem consists of a number of small actors with their own distributed generation devices, there is a push towards decentralized management in order to cut out intermediaries and prevent the management from being concentrated in the hands of a third party. We expect many small scale producers to join this system and we must evaluate how many users it can support.

Blockchain, a decentralized ledger fulfils the requirements outlined above, as it prevents the decision making from being concentrated in the hands of a single party. Moreover, its inherent features of

immutability and robustness due to its decentralized nature and the cryptographic linking of transaction blocks fit our use case well. Such a decentralized system must also be secure and only open to authenticated users. It is therefore necessary to restrict membership only to members of the community and to authenticate all users. This also reduces the operation cost and computational complexity inherent in an open decentalized ledger system, also known as a public blockchain system.

We built our implementation using the Hyperledger Fabric, which is one of the most popular enterprise grade permissioned blockchain platforms. It is open source, free to use and has a modular architecture, allowing the operator to tailor the implementation components to their needs. This blockchain implementation is the underlying transaction infrastructure of the reward system that processes and records the transactions. In order to performance benchmark our implementation, we ran benchmarks using varying loads to evaluate the implementation based on latency and throughput. We implemented our benchmarking experiments in Hyperledger Caliper which interacts with the Hyperledger Fabric implementation and submits transactions as per the configured parameters.

2.2 Data driven approach

The section 3 analyses and aggregates the user data and extracts user energy behavior based on seasons (winter, summer, autumn and spring), day of week (weekday or weekend) and time of day and uses this to identify peak consumption times. Also, the aggregation of surplus energy production gives thresholds for classifying prosumers according the the amount of surplus energy they produce.

Further in section 4 the dataset rows are semantically linked, clustered and labelled based on contexts such as Solar Production and User Demand. Based upon the clustering analysis, low, medium and high thresholds are identified for user demand and solar production contexts.

The blockchain system as described in section 5 encodes the business logic in smart contacts and implements the reward mechanism based on the peak consumption times, surplus production and user

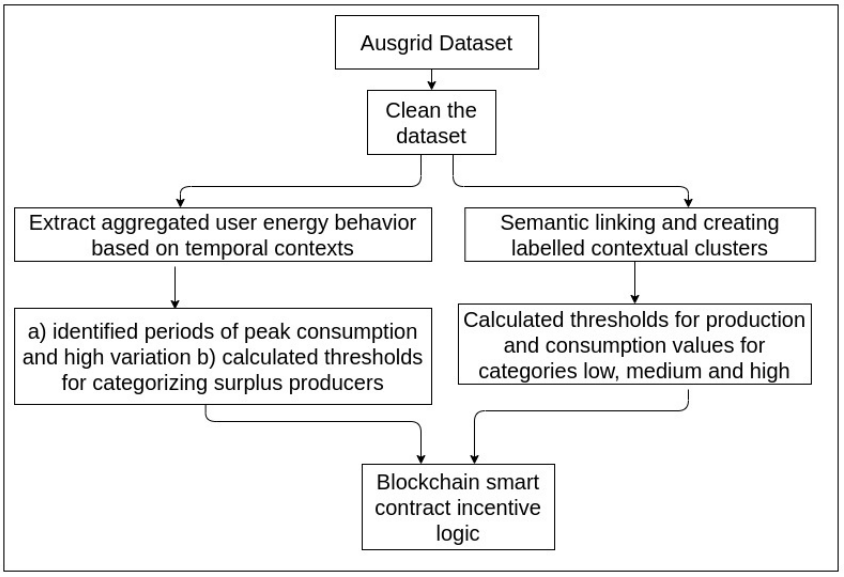


Figure 2: Schematic outline of the system.

demand thresholds identified in section 3 and section 4. This system is performance tested to identify the number of users that can be supported. The figure 2 presents the schematic outline of the proposed system and how the different modules of this work are linked.

3 Ausgrid Dataset and Analysis

The data used in this study is the solar home electricity dataset published by the Ausgrid Corporation in New South Wales (NSW), Australia [15]. We begin our analysis by presenting a brief overview of the dataset.

3.1 Ausgrid Dataset Overview

The Ausgrid dataset includes data collected from installed electricity meters for 300 random customers in the Ausgrid network during 1^{st} July 2012 to 30^{th} July 2013 recorded at half hour intervals ($\Delta t = 1/2\ hour$). The Ausgrid dataset is popular among researchers

Table 1: Customer IDs in the cleaned Dataset

13	14	20	33	35	38	39	56	69	73	74	75
82	87	88	101	104	106	109	110	119	124	130	137
141	144	152	153	157	161	169	176	184	188	189	193
201	202	204	206	207	210	211	212	214	218	244	246
253	256	273	276	297							

investigating grid-defection [16], home energy management [17] and load forecasting using historical smart meter data [18].

The dataset includes solar PV production from roof top solar panels and residential load data for individual households. The residential load includes two distinct categories: (1) Energy Consumption and (2) Heating Load. The first category records the general energy consumption in the household. The heating load refers to energy consumption for heating water in the household. The utility provider controls this load by operating electric water heating during specific periods of the day. This is done with the aim of reducing overall network load during peak times and providing financial incentives to the customer. This is an optional feature and in the Ausgrid [15] dataset, 137 out 300 have opted for this feature. The individual customers are only identified by a serial list (1 : 300) of Customer ID which serve as aliases and their geographical context location is identified by post code.

While, in principle the Ausgrid dataset consists of 1 year electricity production and consumption data for 300 consumers at a resolution to 30 minutes, in practice there are several inconsistencies and anomalies in the data. Ratnam et al. [11] performed a detailed study on the Ausgrid dataset to identify these inconsistencies and identify a subset of customers to be included in the clean dataset with complete records. The current study uses this subset of customers, listed in table 1. The geographic spread of these customers based on their postcodes is shown in figure 3. It can be seen in figure 3 that most of the customers in the clean dataset are located around Newcastle and Sydney metropolitan areas in NSW, Australia.

Figure 3: Location of the Customers

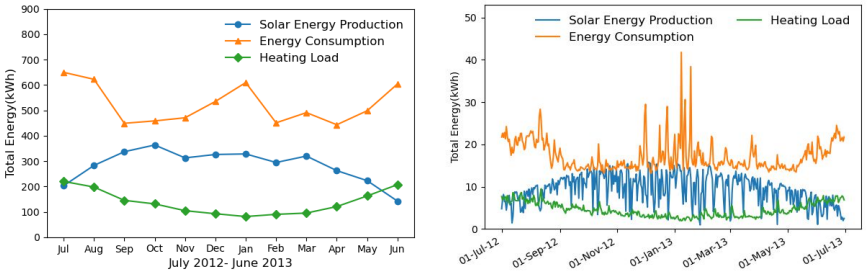
3.2 Aggregated Energy Profile for all customers

To gain an overview of the energy profiles of the customers as a group, we combined their energy profile into an aggregate consumer. The energy profile of this aggregated consumer was cumulatively re-sampled on a monthly (figure 4(a)) and daily basis (figure 4(b)). The seasonal characterization of the months of a calendar year for Australia is outlined in table 2. It can be seen from figure 4(a) and table 2 that the energy profile of the consumers as group shows significant seasonal variation. The peak in energy consumption corresponds to winter (June-August) and summer (December-February). This can be directly correlated with increased load due to climate control requirements during this period. While the overall monthly loads are similar for the summer and winter months, cumulative re-sampling on a daily basis shows higher peaks and greater variation during the summer months (December-February).

Table 2: Seasons in Australia

Season	Period	Representative month
Winter	June - July - August	July-2012
Spring	September - October - November	October-2012
Summer	December - January - February	January-2013
Autumn	March - April - May	April-2013

To better identify the days with high consumption during the summer months, we sorted aggregated household consumption in descending order, and the first 12 data-points were plotted chronologically along with the corresponding day of the week in figure 5. It can be seen in figure 5 that the highest aggregated daily consumption occurs on the weekends when the household members are home. However, there are two outliers of high consumption on weekdays: (1) Monday: 24-12-2012 and (2) Tuesday: 08-12-2013. The second outlier is especially interesting as it coincides with the peak summer temperatures and bush fire warning across most of Australia and especially NSW [19]. It can be seen in figure 4 that solar energy production is highest in the summer months due to higher solar radiation after which it tapers off. Conversely the heating load for electric water heating is highest during the winter months and it tapers off during the summer months.



(a) Aggregated monthly energy profile. (b) Aggregated daily energy profile.

Figure 4: Energy profile aggregated across all customers.

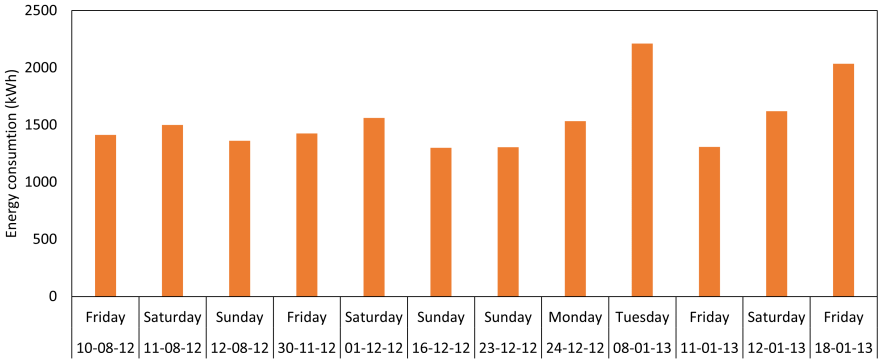
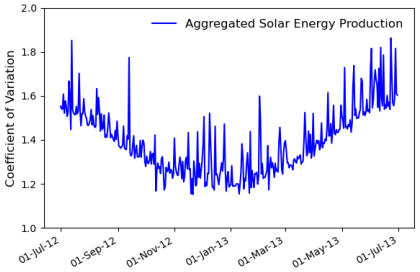


Figure 5: Peak daily energy consumption in summer.

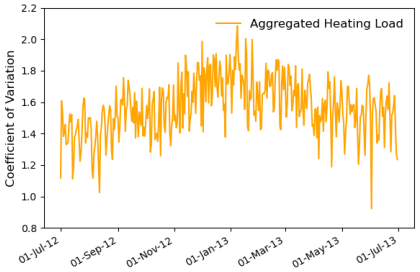
In addition to the solar home electricity dataset [15], we obtained the historical Recommended Retail Price (RRP) for electricity in NSW and the total regional demand data for NSW, Australia between 1-July-2012 to 30-June 2013 at 30 minute interval [20] corresponding to the the timestamps in the Ausgrid Dataset. We computed the coefficient of variation for all the fields in the cleaned Ausgrid dataset (aggregated across all customer) and the regional demand and RRP and shown in figure 6. It can be seen that both Solar energy and heating load show high variation in the range of 1.2 to 1.8. It can be observed that variation in solar energy production reduces during summer whereas the variation in heating load peaks during summer. Energy consumption shows much lower variation. However we can see comparatively higher variation in energy consumption during the summer. This agrees with the observations made earlier in figure 4. Figure 6(d) shows that the regional demand remains quite stable with low variation except during the summer. Finally, figure 6(e) shows that the energy price in NSW shows low variation throughout the year with occasional sharp spikes throughout the year.

3.3 Seasonal Energy Profile for all customers

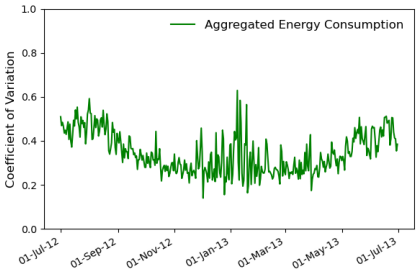
To better understand the seasonal variation in the energy profiles of the customers, we focused on the energy data for all the customers over a 1 month period in the middle of each season. The representative



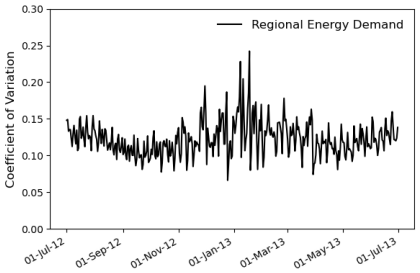
(a) Solar energy



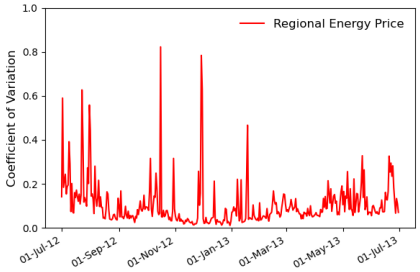
(b) Heating consumption



(c) Energy consumption

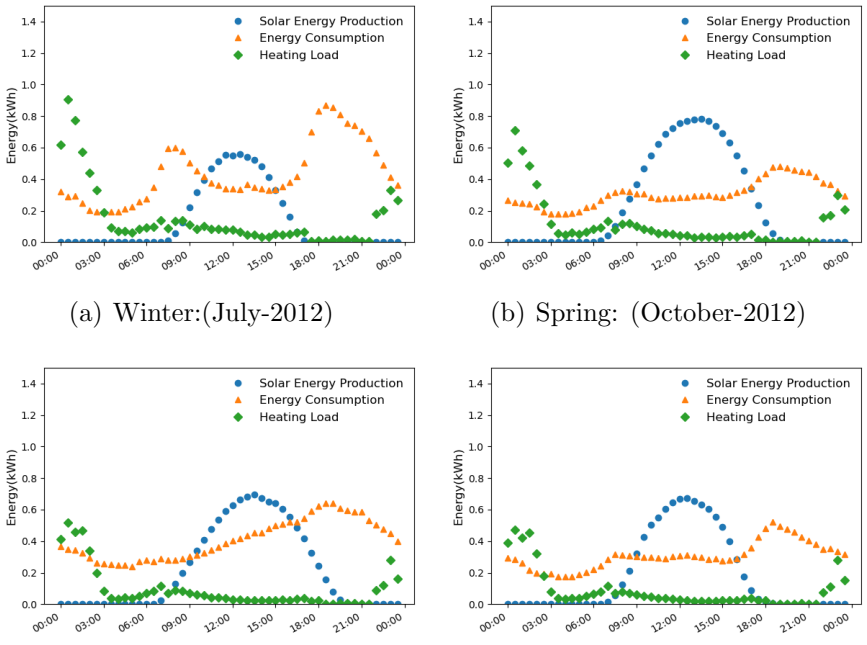


(d) Regional energy demand



(e) Recommended Retail Price (RRP)

Figure 6: Coefficient of Variation on a daily basis for fields in the Ausgrid Dataset, Regional Energy Demand and Recommended Retail Price.



(a) Winter:(July-2012) (b) Spring: (October-2012)

(c) Summer: (January-2013) (d) Autumn: (April-2013)

Figure 7: Seasonal variation in daily energy profile aggregated across all customers.

months for each season are listed in table 2. There after we averaged the energy profile across all the customers for the 1 month period. This allowed us to compute a representative daily energy profile for an averaged customer during the four seasons. The results are shown in figure 7. It can be observed in figure 7 that the heating load is concentrated around midnight and early mornings i.e., during periods of low consumption. The solar energy production follows a clear bell curve. Energy production starts at around 8 am followed by a gradual rise and a peak at around 2 pm. Thereafter it declines, finally stopping around 7 pm. As expected, the highest overall production is observed in spring and summer followed by autumn and least in winter. The daily energy consumption profile shows significant seasonal variation. In the winter (figure 7 (a)) we observe two peaks: one in the morning before the start of the working day and the second one in the evening

Table 3: Summary statistics during periods of surplus energy production

Metric	**Surplus Energy (kWh)**
Min	0.001000
Mean	0.397672
Max	4.095000
25 %	0.134000
50 %	0.284000
75 %	0.477000

at the end of the working day. This second peak can be observed persistently across all seasons in figure 7. This indicates consistent user demand in the evenings. In all the seasons except summer we observe either a flat load curve between morning and evening or a inverse plateau in winter. However, in the summer season, we can observe an almost linear rise in load between morning and evening. This may be related to the rising electricity consumption associated with increased climate control as the the temperature increases during the day. As the temperatures fall in the evening and the night, the load tapers off.

It can also be seen in figure 7 that between 8 am and 7 pm there are periods where solar production is higher than the consumption in the household leading to surplus energy production which can be fed back into the grid or be utilized for charging the community level battery storage. The surplus energy produced by the individual customers at each half hour interval was calculated by subtracting the energy consumption and controlled heating load from the solar production. The resulting column was used to filter the data set by dropping the rows with negative surplus as they represent the intervals where the the total consummation in the household exceeds the solar production. The summary statistics for the resulting dataset are listed in table 3.

4 Smart energy transaction analytic

The semi structured dataset includes information about timestamp, involved actor identifiers etc. Analysis of this dataset can help to understand the behavior of actors and their associated contextual activities. For example, in the case of smart energy systems, the energy dataset can be studied in depth to understand the energy demand and solar production. To achieve this, we extract information from the dataset rows and transform it into a more expressive structured representation. Each row of the dataset is considered to be a transaction. If the proposed blockchain based incentivization system is implemented, each dataset transaction along with additional reward fields would be added to the blockchain through a blockchain transaction as explained in section 5. The procedural workflow for analyzing Ausgrid Dataset follows four steps:

(1) Split the dataset transactions into data nodes based on context, where a context represents an event category such as heating consumption, solar production etc;

(2) Semantically link data nodes occurring within multiple context;

(3) Cluster data nodes for each context based on their contextual similarity, such as similar energy consumption value ranges. The identified value ranges are used as labels for each unique cluster;

(4) Compute cross-contextual similarity to understand the energy demand and solar production.

Figure 8 shows the workflow for smart energy transaction analytics. Initially, we take semi-structured and smart energy dataset transactions and split them into unique contexts (e.g. solar production, heating consumption, energy consumption, total demand per region, price etc), and semantically link each transaction through diverse contexts. Next, we cluster data nodes for each individual context, where each cluster represents data nodes with similar properties. Further, we label each cluster with unique properties. Afterwards, we compute the pairwise similarity between clusters inside a single layer

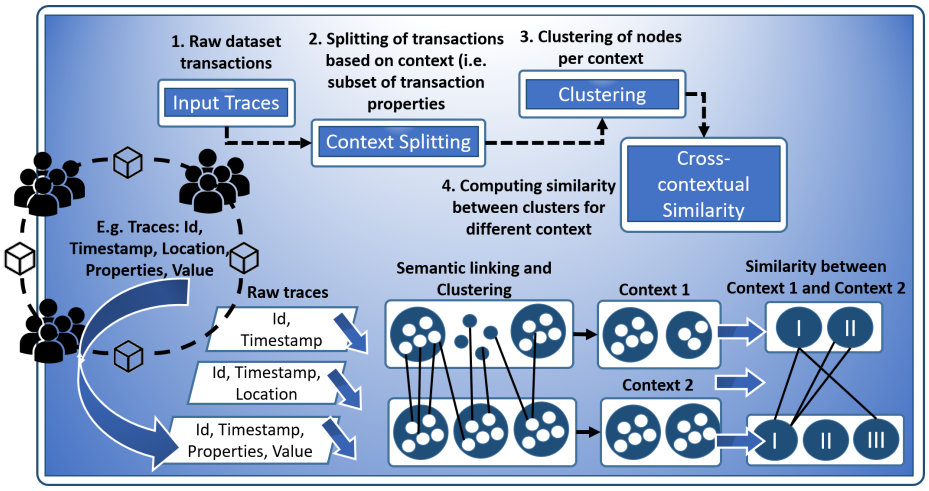


Figure 8: Transaction analytic workflow.

by calculating their Euclidean distance related to clusters of all other layers, where a lower distance means a higher similarity. This process allows us to not only understand the changing behavioural patterns of transactions in a single context but also across contexts. In this paper, we primarily analyse similarity for cluster of transactions within user demand layer with respect to other contextualised layers.

Based on the presented system design in section 4, we implement smart energy transaction analytic in the following stages.

4.1 Semantic linking and contextualisation

Semantic linking and contextualization in complex datasets is required to analyse actors' transactions. In general, datasets are represented as monolayer graphs to visualise the relationships between the actors, where graph nodes denote different actors and edges represent the interactions between actors. However, monolayer graphs often fail to capture the dynamically changing structural contexts of actors and their corresponding evolving relationships. Hence, we adopt a multilayer graph theory that defines a set of context-based layers.

Figure 9 shows an example of such a multilayer representation of smart energy transactions arranged in five layers, where each layer denotes a context. In this example, a multilayer network has a set of

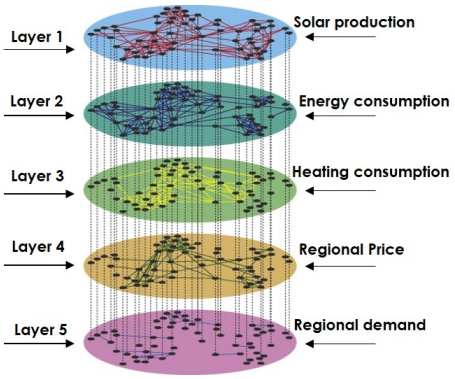


Figure 9: Multilayered contextualisation of smart energy transactions.

nodes like a normal network (i.e. a monolayer graph) but a distinct set of layers. Each layer in the multi-layered representation represents a diverse context (e.g. solar production, heating consumption, energy consumption, regional price, regional demand) and various relationships of entities [21]. A multilayer network representation allows to link edges between same entities across multiple layers and provides a cross-context view of smart energy transactions. This improves the understanding of different interactions among the entities within complex systems across multiple and cross-contextual viewpoints.

To provide contextualised semantic linking and exploit semantic enrichment of complex datasets, we adopt a multilayer approach based on transaction attributes and topological structure. For this purpose, we define a set of layers showing different contexts. Additionally, we jointly consider the context of all networked entities and their network similarity strength. To define a semantic link, we consider different semantic labels for edges across cross-contextual layers, while similar semantic labels for edges stay within a single layer. Our multi-layered approach provides a fully interconnected network where all layers contain all nodes and follows a diagonal coupling model in which inter-layer edges only exist between nodes and their counterparts. Our model also adopts a categorical coupling model in which inter-layer edges are present between any pair of layers and links between pairs in each layer describe similarity strengths. We implement three steps for constructing a semantic multi-layered network.

Step 1 takes dataset transactions as input and extracts each row of data schema as context (e.g. energy consumption, price ranges, location), facts, attributes, concepts and events to enable the accurate analysis of unstructured data.

Step 2 creates an attributed multi-layered network embedding M using the extracted activities pertaining to different transactions. Contrary to monolayer networks, a multilayer network $M = (V_M, E_M, V, L)$ has an underlying set V of N physical nodes that manifests on layers in L constructed from elementary layer sets (i.e L_1,L_2,$\cdots$, L_d, where d is number of contexts). The set of node-layer tuples in M is $V_M \subseteq V \times L_1 \times \cdots \times L_d$, and the set of multilayer edges is $E_M \subseteq V_M \times V_M$. The edge $((i, \alpha), (j, \beta)) \in E_M$ indicates that there is an edge from node i on layer α to node j on layer β (and vice versa, if M is undirected).

Step 3 creates a semantic link between layers in the multilayer network M by utilising attribute similarity between nodes in the same layer. We link nodes with similar attributes in each layer and measure the similarity of nodes using the Euclidean distance as a measure of similarity to compare the pairwise affinity of nodes. The higher the Euclidean distance score, the lower is the similarity score and vice-versa. We further construct a weighted similarity graph in each layer by measuring the similarity between nodes based on their contextual activity in each specific layer. If the similarity score of two nodes is higher than a threshold, it creates a link between them across layers. The threshold is use case dependent and determined based on the input transaction dataset schema. Finally, we assign a weight to each link between two similar nodes based on their similarity score, where higher weight corresponds to links between more similar nodes.

Based on the smart energy dataset presented in section 3.1, we construct a multilayered network with six identified layers.

- *L1-Solar production* represents the amount of energy produced by different residents;
- *L2-Energy consumption* represents the amount of energy consumption pertaining to each resident;

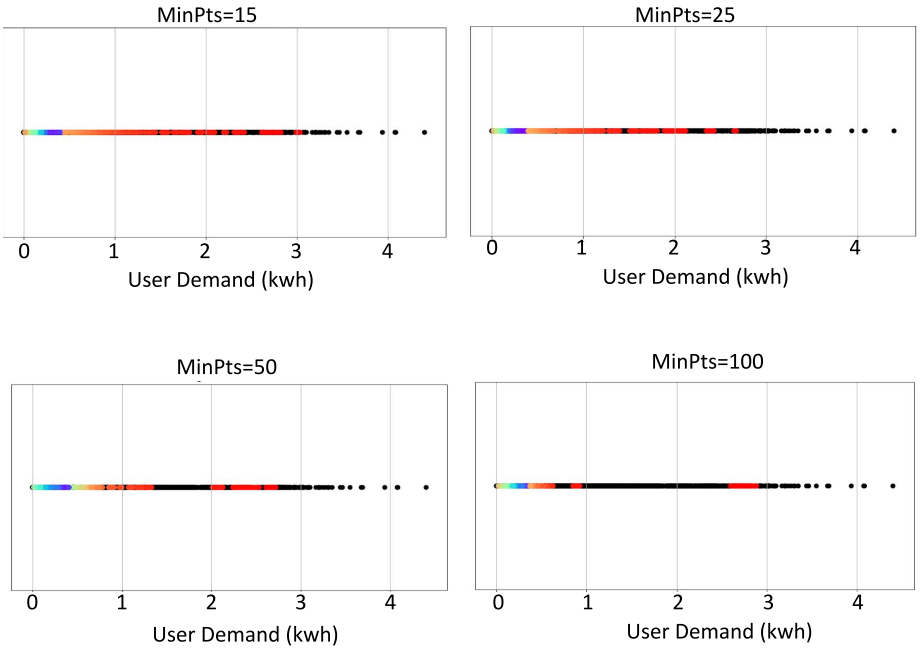


Figure 10: User demand clusters for all customers.

- *L3-Heating consumption* is the heating load across diverse residents;

- *L4-Regional price* shows the price paid by each resident for their energy usage at the given time across a geographical location;

- *L5-Regional demand* represents the overall energy demand across a specific geographical region;

- *L6-User demand* is the overall demand per resident. While, user demand is not represented in dataset, we represent it as an additional layer, where user demand per resident is the sum of energy consumption and heating consumption per resident.

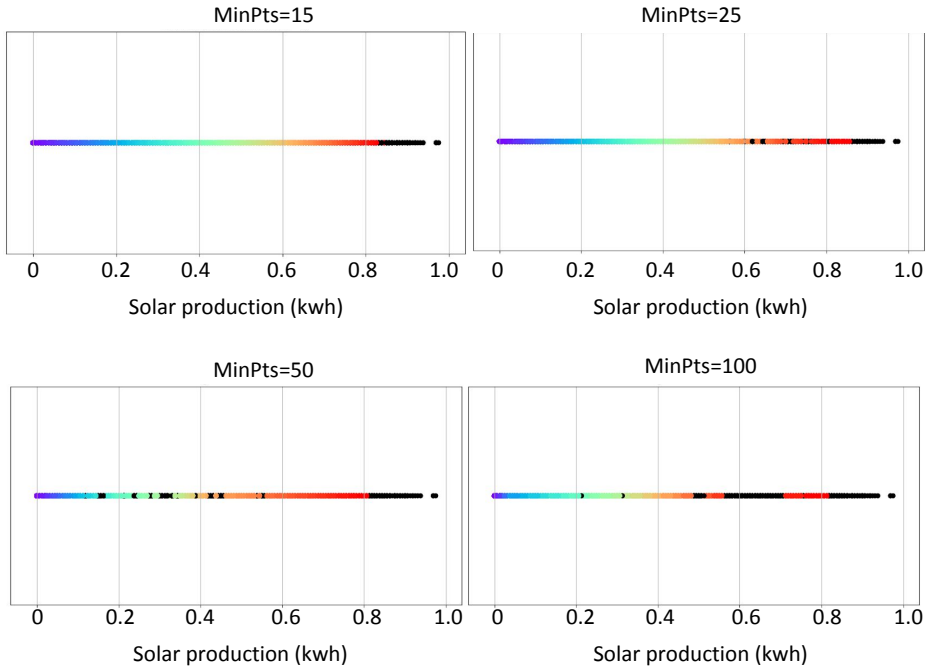


Figure 11: Solar production for all customers.

4.2 Contextual clustering and labelling

The multi-layer network constructed using steps in section 4.1 contains raw, but contextualised information. Hence, the next step is to find similar nodes based on the characteristics of each layer. The smart energy transaction analytics model does not consider links or edges for clustering. Instead, we utilize feature values of each row of dataset, where a feature is represented by the column values for each transaction. To achieve this, we applied a clustering technique that tags similar nodes based on their feature values with the same arbitrary but fixed label. Initially, we fetch the multilayered contextualized graphs across temporal stages as input for clustering and labelling. Further, we utilize OPTICS [22], an augmented cluster-ordering algorithm to find similarities among different nodes in each layered context through a distance function and a minimum number of neighbors required as a unique cluster.

We implemented OPTICS clustering technique in python using

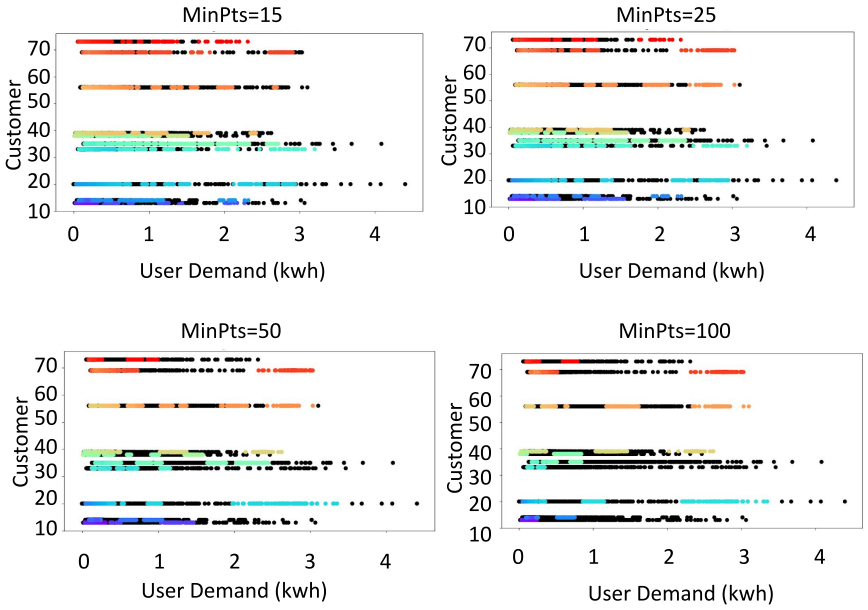


Figure 12: User demand cluster distribution per customer.

scikit-learn[1] library. In the current implementation, we configured clustering hyper-parameter *MinPts*=15 , and *Eps* is set to infinity. The hyper-parameter *Eps* is the radius of clustering neighborhood around a node point, while *MinPts* represents minimum number of neighbors around a radius *Eps*. We used Minkowski [23] distance metric to compute the Euclidean distance between different node points with unique feature values. Figure 10 shows the clusters obtained for `L6-User demand` layer, while figure 11 shows the clusters obtained for `L1-Solar production` layer for all customers. Each point in figure 10 and 11 with similar color represent the customers belonging to same cluster with similar transactions, while black colored points represent the noise. In both cases, we compute clusters with four different values of *MinPts*= 15, 25, 50, 100 respectively. We observe that with *MinPts*=15, our clustering approach find clusters closest to the original dataset, but with low density. In general,

[1] `https://scikit-learn.org/`

increasing *MinPts* value increases the density of clusters. Hence, in the current implementation, we fixed the value of *MinPts*=15. However, for future works, we will consider dynamically estimating the value of *MinPts* depending upon the individual layer size and structure. Figure 12 shows clustering for a subset of customers within *L6-User demand* layer. Similarly as in figure 10 and 11, figure 12 shows clustering with four *MinPts* values, but represent cluster for a subset of customers within *L6-User demand* layer. We observe that same customers in different transactions can correspond to varied demand values represented in different clusters.

After identifying the clusters for each layer, we implicitly know the transaction activity of different customers represented as node within a cluster. Hence, the next step is to label the clusters. In this work, we primarily focus on analysing energy production and consumption per transaction, hence we label only the clusters in *L1-solar production* and *L6-User demand* layers. To this end, we encapsulate each cluster with either of k label categories. In this paper we define $k = 3$ namely: `High`, `Medium`, and `Low` label categories. Initially, we define a feature associated to each cluster $\in C$, where a feature is a tuple of length n=2 with the highest and lowest values of all nodes in the same cluster, and C is the number of clusters obtained in a specific layer. Further, we compute the average $MEAN$ of these feature values of each cluster. Henceforth, we identify the lowest (X) and the highest (Y) $MEAN$ value of all clusters $\in C$ and compute the distance between the lowest and the highest mean X and Y as $Z = Y - X$. Next, we map k ranges to k labels using Z/k representing the value range for the labels. The value range enables us to define the two global thresholds t_1 and t_2, where $t_1 = X + Z/k$ and $t_2 = Y - Z/k$. Based on the thresholds, finally we assign labels to each cluster. A cluster is labelled `High`: if $MEAN > t_2$, while we label a cluster `Medium`: if $t_1 < MEAN \leq t_2$. Finally, a cluster is labelled `Low`: if $MEAN \leq t_1$. In this work, we primarily label clusters obtained in solar production and user demand layers. For solar production layer, the computed thresholds t_1 and t_2 are 1.43016 and 2.8603 respectively. While for user demand layer, the thresholds t_1 and t_2 are 1.8306 and 3.6608.

4.3 Cross-contextual similarity

After obtaining the clusters and corresponding cluster labels in each layer, the next step is to analyze the similarity of clusters in one layer with respect to all clusters of another layer. This allows us to understand the changes in behavioral activity of transactions in one context with regards to another context. To achieve this, we compute pairwise similarity of two clusters in one layer (e.g. cluster (C_i, C_j) in *L6-User demand* layer with all clusters of *L1-solar production* layer). Henceforth, we compute the similarity of cluster pair (C_i, C_j) $\in$ *L6-User demand* with cluster $C_k \in$ *L1-solar production* layer using Euclidean distance E as follows:

$$E = \sqrt{\sum_{k=1}^{N} (D_i^k - D_j^k)^2} \tag{2}$$

where, N is the total number of clusters in *L1-solar production* layer, D_i^k and D_j^k are the number of edges from cluster C_k to clusters C_i and C_j respectively. We perform similarity computation for all the possible combinations of clusters in *L6-User demand* with all the clusters of other layers. The final output of this operation is a collection of all suitable pairs of clusters with their associated similarity regarding each layer. The key of the dictionary is a pair of clusters labels and the value is another dictionary of the computed similarity in regard to each layer, where the key is the name of the layer and the value is the computed Euclidean Distance. As a future work, we also plan to use this pairwise similarity computation for proactive prediction of cluster size and structure.

5 Blockchain based reward system

In this section we use the insights garnered from section 3 and section 4 to design the smart contract that will process prosumer rewards. Each row from the dataset (1 transaction) is modelled as a key value pair token on the blockchain, created through a blockchain transaction, where the token's value field is composed of the values from the row being processed. In addition, we add six new values

in the value field of each token, RSeasonal, RWeekend, RPeakTime, RMiniProducer, RProducer and RMegaProducer which are boolean fields, which will be set to true if the conditions for a reward of that type are met for a particular transaction. A token is created for each customer at each half hour interval and consists of customer data from the dataset as well as the reward fields mentioned.

Section 3, identified several energy generation and usage patterns. Of these, we focus on the observations listed below for creating our smart contract. Alongside we mention, the reward field that will be updated based on that observation. The smart contract is written in Golang v1.16 [24] and creates the token shown in figure 13 for each transaction.

(1) Highest consumption was observed in summer and winter seasons, which can be attributed to climate control needs. Moreover, highest variation in energy consumption was observed in summer. So, for low consumption during summer and winter the customer can win a reward associated with the field RSeasonal.

(2) Peak loads were seen on weekends when household members are home. If a customer's consumption is low on the weekend, they will gain a reward associated with the field RWeekend.

(3) Peaks were twice a day, once in the morning before the start of workday and once in the evening at the end of workday. So, in case of low consumption during identified peak times the customer can get a reward associated with the field RPeakTime.

(4) The identified thresholds for 25th percentile, 50th percentile and 75th percentile surplus production are t_a, t_b and t_c. If surplus solar production for any customer is greater than or equal to t_a, then the field set to true is RMiniProducer. Similarly, the fields RProducer and RMegaProducer are set to true for transactions when surplus generation is greater than or equal to t_b and t_c respectively. Thus, a transaction with surplus production in the 75th percentile will have RMiniProducer, RProducer and RMegaProducer, all set to true.

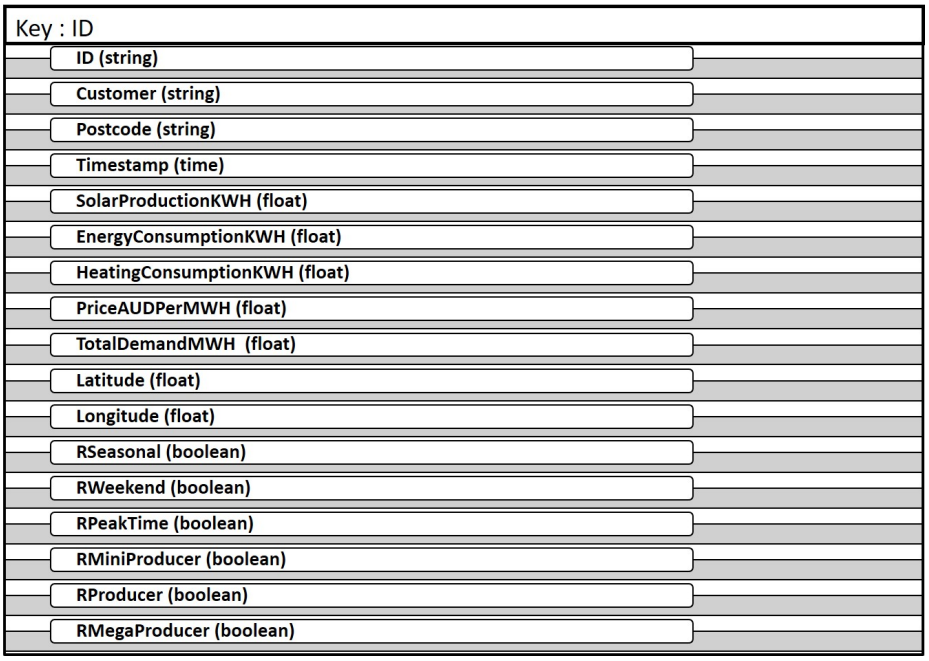


Figure 13: Structure of transaction token

Moreover, in section 4, of the several observations made on user generation and usage behavior, we focus on the thresholds identified for user demand. The thresholds identified are used for labelling clusters. In case of user demand, if the demand in kWh associated with a given transaction is less than t_1 then it is labelled as low, while if it is more than t_2, it is labelled as high. Demand between t_1 and t_2 is labelled as medium.

(5) For user demand, the thresholds $t_1 = 1.8306$ kWh and $t_2 = 3.6608$ kWh were used to create labelled clusters of low, medium and high consumption levels.

5.1 Smart contract

The smart contract encodes the business logic of the reward system. When this smart contract is deployed, we set an endorsement policy that stipulates that all organizations in the network must endorse each transaction. This is due to the reasons identified in section 2.2.

The algorithm of interest is Algorithm 2 , while Algorithm 1 is a helper algorithm.

Algorithm 1 TokenExists

```
1: function TokenExists(idval string)
2:     token ← worldstate(idval)
3:     if token=nil then
4:         return false
5:         else return true
6:     end if
7: end function
```

In Algorithm 2, first the identity of the invoker organization is found from the identity of the invoker client. If the client is not a member of the Power Company this attempt to create a transaction is rejected. As the Power company is the one that awards the tokens and has access to both creation and generation data for all customers, it is the only organization with the privilege of initiating transactions. However, it cannot actually process transactions without the endorsement of the other two organizations. Then, the provided timestamp string is converted into an integer array by removing special characters and by type conversion. This is done in order to work with the individual parts of the timestamp such as month and date. The key of the token to be created is checked against the world state to make sure that there is no collision between the new key and an existing key. This can only happen if a transaction token for a given customer and timestamp has already been created and is now being recreated. Each customer will have transaction tokens with timestamps corresponding to each half hour period of the day and thus will have only one token per half hour. If the token for a given half hour period for a customer exists already, the transaction will be rejected. Now, the conditions for the various rewards are checked and the relevant field is updated if the customer will get that particular reward. Thus, if the customer's consumption is low and the season calculated from the month part of the timestamp is summer or winter, which are known to be high consumption periods, then the RSeasonal field gets updated.

Algorithm 2 Create transaction token

function CREATETOKEN(Customer string, Postcode string, TimeStamp string, SolarProductionKWH float, EnergyConsumptionKWH float, PriceAUDPerMWH float, TotalDemandMWH float, Latitude float, Longitude float)
    $invokerorg \leftarrow$ organization of invoker client
    if invokerorg is not Power Company **then**
        Invalid invoker
    end if ▷ Only the Power company can initiate the transaction

    Create integer array from string timestamp
    $idval := customer + \text{“\_”} + timestamp$

    $exists \leftarrow$ TokenExists(token idval)
    if $exists$ = true **then** return error
    end if
    if $energyconsumptionKWH <= t1$ and timestamp month in (summer, winter) **then** RSeasonal =true
    end if
    if $energyconsumptionKWH <= t1$ and timestamp date is weekend **then** RWeekend =true
    end if
    if $energyconsumptionKWH <= t1$ and timestamp hour in (morning peak, evening peak) **then** RPeakTime =true
    end if
    if $surplusproductionKWH >= ta$ **then** RMiniProducer =true
    end if
    if $surplusproductionKWH >= tb$ **then** RProducer =true
    end if
    if $surplusproductionKWH >= tc$ **then** RMegaProducer =true
    end if
    $Putinworldstate(idval, token)$ ▷ Saving new token
end function

Similarly, for low consumption on the weekend or during identified morning and evening peak times, the fields RWeekend and RPeakTime respectively will be updated. Also, if the transaction shows a net

production, the value of the surplus will be checked against the percentile thresholds identified in table 3 in order to update the fields RMiniProducer, RProducer and RMegaProducer. After updating the rewards, the token will be saved to the world state with the key customer_timestamp.

5.2 Implementation

Our test infrastructure included 5 Virtual Machines (VM) on a Cloud environment as shown in figure 14. Each VM has Ubuntu 20.04 installed and features 32 GB RAM, 4 dedicated virtual CPUs and a 100 GB SSD disk. Each VM uses Docker version 19.03, Docker Compose version 1.26, Hyperledger Caliper version 0.4.2 and Hyperledger Fabric 2.3.0, which are all the latest stable versions available at the time of writing. The Ordering service uses RAFT [25] consensus algorithm with 3 Ordering Service Nodes (OSN) as recommended in the Hyperledger Fabric documentation. We chose LevelDB as the state database as it is the more performant choice [26]. Each node of our network runs as a Docker container and is connected in a Docker Swarm to ensure high availability. Our experiments feature an architecture of 3 Organizations. We run the load generator Hyperledger Caliper on a separate VM as it is resource intensive, in this case VM1. Each organization, including the orderer organization uses a separate VM to run its Docker containers. Our reason for this setup is twofold. Firstly, the orderer organization, or any of its OSN must not be in control of any of the member organizations as it performs a vital function. Moreover, putting the orderer organization on the same VM as an organization will consume resources of that VM and will skew the results for that organization negatively due to resource contention and perhaps positively due to proximity to the orderer. In a real world implementation, each organization would have its own infrastructure and therefore, we put each organization on a separate VM. Thus, we avoid resource contention arising from an increasing number of containers on the same infrastructure.

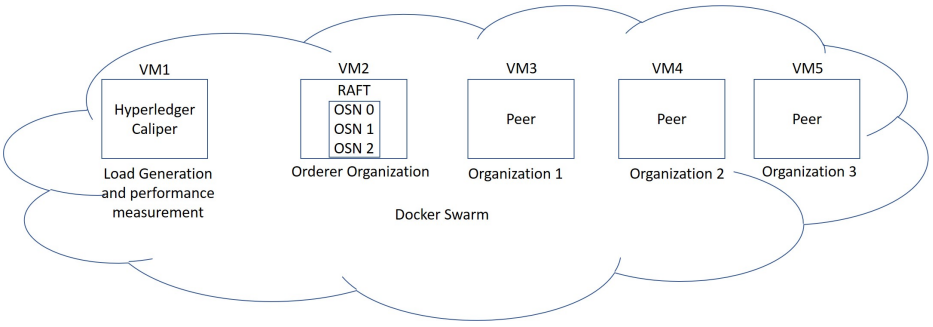


Figure 14: Experimentation testbed

5.3 Results

As mentioned in the section 5.2, we use Hyperledger Caliper to create and send transaction requests to the implemented blockchain network. Each transaction in our experiments runs the smart contract for creating a token which was described in Algorithm 2. Each transaction thus consists of one query and one create transaction. We ran 10000 transactions for each data point shown in the graphs. Four worker processes were created to drive the load and we configured the fixed load rate control mechanism in our benchmark. The fixed load rate controller in Hyperledger Caliper starts with a configured send rate in transactions per second (TPS) and maintains a defined backlog of transactions in the network by modifying the send rate. We configured the starting send rate to 1000 TPS and varied the maximum limit of unfinished transactions to observe the effects on the request send rate as shown in figure 15. We found that the send rate achieved rose with the increase in the maximum permissible number of unfinished transactions configured. As shown in figure 16, with the rise in send rate, the throughput of the system increased. However, average latency per transaction and maximum latency per transaction rose as well. The value of average latency remained under 1 second even at throughput of over 440.3 TPS which was achieved at a send rate of 443 TPS. The throughput could be increased further by increasing request send rates, but with diminishing returns due to the increase in transaction latency. The current dataset contains data for 300 customers updated every half an hour, which

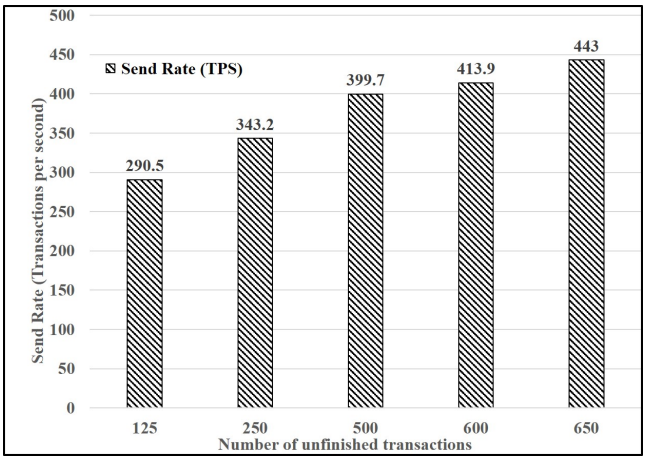


Figure 15: Request send rate at varying limits of maximum unfinished transactions

corresponds to a throughput of 0.167 transactions per second. So, extrapolating, we can say that the current implementation is able to support 792,540 customers with a reasonably low infrastructure footprint. This implementation can be scaled further by providing it with more resources through horizontal and vertical scaling as described in Thakkar and Nathan [27].

6 Related Works

Several studies [28], [29], [30], [31] focus on different aspects of blockchain based energy trading among prosumers in a microgrid. Our work presents an approach for incentive based peak shaving and does not discuss prosumer to prosumer trading.

Pop et al. [32] present an architecture for implementing demand response programs in microgrids. Their solution, implemented in Ethereum takes the baseline values for each prosumer which is the average of their past values and then calculates the required flexibility per prosumer. Their results do not include an analysis of the performance of their blockchain implementation itself.

Di Silvestre et al. [33] also present a framework for demand side response by calculating baseline usage per customer, publishing the

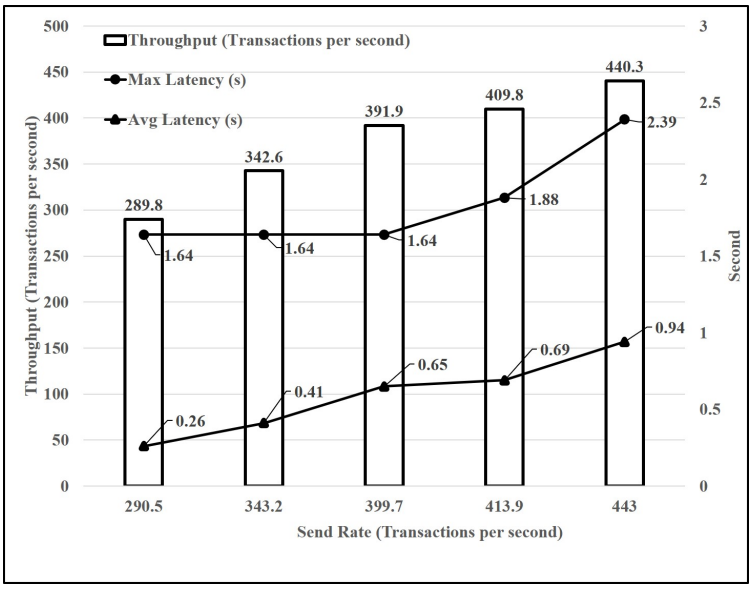


Figure 16: Transaction throughput and latency at varying send rates

required reduction in usage with the reduction amount and time window, and monitoring compliance. Their system architecture considers two organizations, one composed of grid and market operators and the other composed of market operators and customers. This architecture reduces the agency that the users have in the implemented business logic. Also, the article does not characterize the performance of the blockchain implementation.

Guo et al. [34], Wang et al. [35] and Afzal et al. [36] also propose an individualized incentivization model. These articles also do not discuss the performance of their blockchain implementation. Moreover, Afzal et al. [36] implement their solution using Ethereum. In addition, Di Silvestre et al. [33], Guo et al. [34], Wang et al. [35] and Afzal et al. [36] do not use real world data to inform their incentivization logic, but Guo et al. [34] and Wang et al. [35] do use real world data to validate their model.

In our work, the logic of our incentivization system is informed by an in depth analysis of the Ausgrid dataset. The logic encoded in the smart contracts is based on an aggregation analysis and semantic clustering of all transactions so that all customers are subject to

the same rules. Thus, it does not penalize good performers. Our system was implemented in Hyperledger Fabric, which is formed of authenticated nodes with clearly defined privileges in the network and provides identity management and provenance tracing. Hyperledger Fabric authentication can also help companies fulfil their legal obligations like Know your customer and Anti money laundering which are imposed by governments of several countries [37]. Also, consensus in Hyperledger Fabric is achieved based on an agreed endorsement policy and thus it does not rely on computation intensive and thus energy intensive mechanisms like Proof of Work to reach consensus. In our architecture, the user platform is a separate organization that gives the user representatives the opportunity to review and approve the business logic before it is updated on the platform. Further, we performance test our implementation under varying loads and present our findings in terms of transaction throughput and latency.

7 Conclusion

This work presents a data-centric approach for incentive based peak shaving and demonstrates the implementation of a blockchain based reward platform. First, we extracted from the Ausgrid dataset, aggregated user energy behavior in different temporal contexts such as seasons (summer, winter, spring, autumn), days of the week (weekday, weekend) and time of the day. Analysing the aggregated user profile gave us peak consumption times in seasonal, weekly and daily contexts, as well as thresholds to categorize surplus production. Semantic linking and contextual clustering and labelling of this data gave us thresholds to categorize user demand. This analysis informed the logic of the smart contract in our blockchain implementation. Based on this study, we present the following conclusions.

(1) Seasonal peak consumption was observed in the summer and winter seasons. On a weekly basis, the highest consumption was seen on weekends. Moreover, there are two peaks observed daily, one before the start of the work day and one in the evening at the end of the workday. Highest variation in consumption was also seen in summer. Additionally, outliers associated with

high consumption were linked to increased demand during heat waves in the region.

(2) Aggregation analysis of surplus production gave us thresholds t_a, t_b and t_c for categorizing transactions into Mini Producer, Producer and Mega producer categories respectively. The values for t_a, t_b and t_c were 0.134000 kWh, 0.284000 kWh and 0.477000 kWh respectively.

(3) Contextual clustering identified two thresholds for production t_1 and t_2 with values 1.43016 and 2.8603 respectively. Similarly, for user demand layer, the thresholds t_1 and t_2 with values 1.8306 and 3.6608 were identified. These threshold values enable us to characterize production and demand in low, medium and high categories.

(4) The implementation of the blockchain based reward system encoded a smart contract with the business logic for earning rewards based on our analysis. If the conditions for a given reward are met, the smart contract processes the reward for the transaction. A given transaction can acquire multiple rewards.

(5) While the Power company is the only organization allowed to initiate transactions, all three organizations must endorse each transaction in order to perform various checks and prevent the management from being concentrated with a particular entity. This implementation with a reasonably low infrastructure footprint was shown to be able to support 792,540 customers.

References

[1] Xiaolei Yang, Lingyun He, Yufei Xia, and Yufeng Chen. "Effect of government subsidies on renewable energy investments: The threshold effect." In: *Energy Policy* 132 (2019), pp. 156–166.

[2] Laura Olkkonen, Kristiina Korjonen-Kuusipuro, and Iiro Grönberg. "Redefining a stakeholder relation: Finnish energy "prosumers" as co-producers." In: *Environmental Innovation and Societal Transitions* 24 (2017), pp. 57–66.

[3] Allison Lantero. "How microgrids work." In: *US Department of Energy* 17 (2014).

[4] Kejun Qian, Chengke Zhou, Malcolm Allan, and Yue Yuan. "Modeling of load demand due to EV battery charging in distribution systems." In: *IEEE Transactions on Power Systems* 26.2 (2011), pp. 802–810.

[5] Moslem Uddin, Mohd Fakhizan Romlie, Mohd Faris Abdullah, Syahirah Abd Halim, Tan Chia Kwang, et al. "A review on peak load shaving strategies." In: *Renewable and Sustainable Energy Reviews* 82 (2018), pp. 3323–3332.

[6] VK Mehta and Rohit Mehta. *Principles of power system: including generation, transmission, distribution, switchgear and protection*. S. Chand, 2005.

[7] Melike Erol-Kantarci and Hussein T Mouftah. "TOU-aware energy management and wireless sensor networks for reducing peak load in smart grids." In: *2010 IEEE 72nd Vehicular Technology Conference-Fall*. IEEE. 2010, pp. 1–5.

[8] Ausgrid. *Ausgrid demand management Newington grid battery trial*. https://www.ausgrid.com.au/-/media/Documents/Demand-Mgmt/DMIA-research/. Accessed: 2021-02-17. 2016.

[9] Stefan Ivanov Sulakov. "The cross-border trade impact on the transmission losses." In: *2017 15th International Conference on Electrical Machines, Drives and Power Systems (ELMA)*. IEEE. 2017, pp. 115–118.

[10] Wujing Huang, Ning Zhang, Chongqing Kang, Mingxuan Li, and Molin Huo. "From demand response to integrated demand response: Review and prospect of research and application." In: *Protection and Control of Modern Power Systems* 4.1 (2019), pp. 1–13.

[11] Elizabeth L Ratnam, Steven R Weller, Christopher M Kellett, and Alan T Murray. "Residential load and rooftop PV generation: an Australian distribution network dataset." In: *International Journal of Sustainable Energy* 36.8 (2017), pp. 787–806.

[12] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. "Hyperledger fabric: a distributed operating system for permissioned blockchains." In: *Proceedings of the thirteenth EuroSys conference*. 2018, pp. 1–15.

[13] Hyperledger Caliper. "Hyperledger Caliper Architecture." In: *Electronic Article. url: https://hyperledger. github. io/caliper/-docs/2_Architecture. html (visited on 04/04/2021)* (2021).

[14] Ausgrid. *Community Batteries*. https://www.ausgrid.com.au/In-your-community/Community-Batteries. Accessed: 2021-02-17. 2021.

[15] Ausgrid. *Solar home half-hour data - 1 July 2012 to 30 June 2013*. 2013. DOI: https://www.ausgrid.com.au/-/media/Documents/Data-to-share/Solar-home-electricity-data/Solar-home-half-hour-data---1-July-2012-to-30-June-2013.zip.

[16] Hangyue Liu, Donald Azuatalam, Archie C. Chapman, and Gregor Verbič. "Techno-economic feasibility assessment of grid-defection." In: *International Journal of Electrical Power Energy Systems* 109 (2019), pp. 403–412. ISSN: 0142-0615. DOI: https://doi.org/10.1016/j.ijepes.2019.01.045. URL: http://www.sciencedirect.com/science/article/pii/S0142061518321136.

[17] B. Jeddi, Y. Mishra, and G. Ledwich. "Dynamic programming based home energy management unit incorporating PVs and batteries." In: *2017 IEEE Power Energy Society General Meeting*. 2017, pp. 1–5. DOI: 10.1109/PESGM.2017.8273925.

[18] B. Yildiz, J. I. Bilbao, J. Dore, and A. Sproul. "Household electricity load forecasting using historical smart meter data with clustering and classification techniques." In: *2018 IEEE Innovative Smart Grid Technologies - Asia (ISGT Asia)*. 2018, pp. 873–879. DOI: 10.1109/ISGT-Asia.2018.8467837.

[19] Australian Government Bureau of Meteorology (2013). *Special Climate Statement 43—Extreme heat in January 2013*. 2013. URL: http://www.bom.gov.au/climate/current/statements/scs43.pdf.

[20] Australian Energy Market Operator AEMO. *Aggregated Price and Demand Data - Historical*. 2013. DOI: https://aemo.com.au/energy-systems/electricity/national-electricity-market-nem/data-nem/aggregated-data.

[21] Mikko Kivelä, Alex Arenas, Marc Barthelemy, James P Gleeson, Yamir Moreno, and Mason A Porter. "Multilayer networks." In: *Journal of complex networks* 2.3 (2014), pp. 203–271.

[22] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. "OPTICS: ordering points to identify the clustering structure." In: *ACM Sigmod Record* 28.2 (1999), pp. 49–60.

[23] Graziano Crasta and Annalisa Malusa. "The distance function from the boundary in a Minkowski space." In: *Transactions of the American Mathematical Society* 359.12 (2007), pp. 5725–5759.

[24] Golang. *The Go programming language*. https://golang.org/. Accessed: 2021-02-17. 2021.

[25] Diego Ongaro and John Ousterhout. "In search of an understandable consensus algorithm." In: *2014 USENIX Annual Technical Conference (USENIXATC 14)* (2014), pp. 305–319.

[26] Parth Thakkar, Senthil Nathan, and Balaji Viswanathan. "Performance benchmarking and optimizing hyperledger fabric blockchain platform." In: *2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE. 2018, pp. 264–276.

[27] Parth Thakkar and Senthil Nathan. "Scaling Hyperledger Fabric Using Pipelined Execution and Sparse Peers." In: *arXiv preprint arXiv:2003.05113* (2020).

[28] Nikita Karandikar, Antorweep Chakravorty, and Chunming Rong. "Transactive Energy on Hyperledger Fabric." In: *2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*. IEEE. 2019, pp. 539–546.

[29] Gijs van Leeuwen, Tarek AlSkaif, Madeleine Gibescu, and Wilfried van Sark. "An integrated blockchain-based energy management platform with bilateral trading for microgrid communities." In: *Applied Energy* 263 (2020), p. 114613.

[30] Aparna Kumari, Rajesh Gupta, Sudeep Tanwar, Sudhanshu Tyagi, and Neeraj Kumar. "When blockchain meets smart grid: Secure energy trading in demand response management." In: *IEEE Network* 34.5 (2020), pp. 299–305.

[31] Nikita Karandikar, Antorweep Chakravorty, and Chunming Rong. "RenewLedger: Renewable energy management powered by Hyperledger Fabric." In: *2020 IEEE Symposium on Computers and Communications (ISCC)*. IEEE. 2020, pp. 1–6.

[32] Claudia Pop, Tudor Cioara, Marcel Antal, Ionut Anghel, Ioan Salomie, and Massimo Bertoncini. "Blockchain based decentralized management of demand response programs in smart energy grids." In: *Sensors* 18.1 (2018), p. 162.

[33] M. L. Di Silvestre, P. Gallo, E. R. Sanseverino, G. Sciumè, and G. Zizzo. "Aggregation and Remuneration in Demand Response With a Blockchain-Based Framework." In: *IEEE Transactions on Industry Applications* 56.4 (2020), pp. 4248–4257. DOI: 10.1109/TIA.2020.2992958.

[34] Zishan Guo, Zhenya Ji, and Qi Wang. "Blockchain-Enabled Demand Response Scheme with Individualized Incentive Pricing Mode." In: *Energies* 13.19 (2020), p. 5213.

[35] Jianxiao Wang, Haiwang Zhong, Chenye Wu, Ershun Du, Qing Xia, and Chongqing Kang. "Incentivizing distributed energy resource aggregation in energy and capacity markets: An energy sharing scheme and mechanism design." In: *Applied Energy* 252 (2019), p. 113471.

[36] M. Afzal, Q. Huang, W. Amin, K. Umer, A. Raza, and M. Naeem. "Blockchain Enabled Distributed Demand Side Management in Community Energy System With Smart Homes." In: *IEEE Access* 8 (2020), pp. 37428–37439. DOI: 10.1109/ACCESS.2020.2975233.

[37] Christian Bührer, Ivo Hubli, and Eliane Marti. "The regulatory burden in the Swiss wealth management industry." In: *Financial Markets and Portfolio Management* 19.1 (2005), pp. 99–108.

Paper IV:
Blockchain based energy management system with fungible and non-fungible tokens.

Blockchain based energy management system with fungible and non-fungible tokens.

N. Karandikar[1], **A. Chakravorty**[1], **C. Rong**[1]

[1] Department of Electrical Engineering and Computer Science, University of Stavanger

Abstract:
Renewable energy microgeneration is rising leading to creation of prosumer communities making it possible to extract value from surplus energy and usage flexibility. Such a peer-to-peer energy trading community requires a decentralized, immutable and access-controlled transaction system for tokenized energy assets. In this study we present a unified blockchain-based system for energy asset transactions among prosumers, electric vehicles, power companies and storage providers. Two versions of the system were implemented on Hyperledger Fabric. Assets encapsulating an identifier or unique information along with value are modelled as non-fungible tokens (NFT), while those representing value only are modelled as fungible tokens (FT). We developed the associated algorithms for token lifecycle management, analyzed their complexities and encoded them in smart contracts for performance testing. The results show that performance of both implementations are comparable for most major operations. Further, we presented a detailed comparison of FT and NFT implementations based on use-case, design, performance, advantages and disadvantages. Our implementation achieved a throughput of 448.3 transactions per second for the slowest operation (transfer) with a reasonably low infrastructure.

1 Introduction

Renewable energy, especially solar energy is being increasingly integrated into the energy grid as photo voltaic installations continue to mushroom in residential contexts. This rise in adoption is fuelled partly by financial incentives like government programs and monetary benefits of local energy production [1] and in part by rising environmental awareness [2] . A new category of energy users called the prosumers [3] has been created, who generate a portion of the energy they consume through their local microgeneration devices. Several prosumers, when collocated give rise to prosumer communities or microgrids [4] and can create a local market for sale and purchase of surplus renewable energy. This creates a scenario for peer to peer energy transactions. Pilot studies in peer to peer prosumer trading [5], have been conducted with different business models [6] and have shown its importance in facilitating local power and energy balance [7]. Prosumer communities can also choose to store surplus energy for later use by exploring energy storage at the community level. Energy Storage as a service [8] is a burgeoning new business that takes over the logistics of setting up and maintaining a large scale storage facility and offers storage credits to the users for purchase. Such an arrangement can offer significant economic benefits [9]. Community level energy storage can facilitate energy transactions as energy can be supplied from the seller to the buyer via this storage, thus reducing the need for wired connections between all the members of the community. In addition, surplus stored energy can be used for powering Electric Vehicle (EV) charging stations, thus offering more opportunities for monetizing the surplus energy. EV batteries, when not in use can be rented out to the community level storage provider to add to the storage capacity. Mahmud et al. [10] proposed a system for using community level storage for charging EVs and presented a decision tree based algorithm for peak load reduction through coordinated management of EV, photovoltaics and community level storage.

Peak demand periods [11] present a challenge to the grid operator as they may require them to over provision grid capacity in order to maintain grid stability, thus increasing the marginal cost of electricity.

Peak shaving strategies such as demand side response [12] are thus of particular importance to grid operators. An incentive based, direct to consumer demand response mechanism can be considered, allowing the power company to offer reward tokens to their customers in exchange for energy flexibility. As the prosumers are located in close proximity to each other, considering them as a prosumer community will allow the grid operator or power company to consider the required energy flexibility for the community as a whole [13]. In addition, the community structure can be used to increase interest and engagement by offering game based energy flexibility tasks [14].

Prosumers, EV owners, Power Companies and Storage Providers, by virtue of their relationships are stakeholders in a business network for transacting energy assets such as energy units, storage credits and reward tokens. An integrated solution is required to on-board the stakeholders and encapsulate the business network and relationships. As several small scale producers and consumers will constitute the network, a decentralized system is necessary in order to prevent the management from being concentrated in the hands of a single central entity. Moreover, all stakeholders must agree on the business logic and the transactions of energy assets must be immutable, transparent and verifiable to all. Provenance tracing of assets should also be enabled in order to make the system more trustworthy.

Blockchain [15] is a distributed shared ledger that fulfils these requirements [16, 17]. Due to its characteristics of decentralization and immutability it prevents any member from unilaterally making decisions on the network [18, 19]. Members are required to seek consensus before adding any transactions to the ledger and transactions are transparent to all members. Transaction history of assets on the blockchain can be readily traced for establishing provenance.

Blockchain implementations can be broadly categorized as permissioned or permissionless. Blockchain platforms such as Bitcoin [15] and Ethereum [20] are permissionless and allow anyone to join the network and perform transactions. Due to the anonymity inherent in these systems, computationally expensive consensus mechanisms such as Proof of Work are used due to the lack of any trust between transacting parties. Permissioned blockchain platforms such as Hyperledger Fabric [21] only permit authenticated parties to join

the network and can have defined access permissions to dictate the privileges of each network member. As transactions are traceable to the invoking member, dependence on resource intensive consensus mechanisms is eliminated, thus reducing the operating cost.

Authentication of clients also helps the network satisfy know your customer and anti money laundering requirements [22] imposed by many countries. Moreover, Hyperledger Fabric supports the encoding of business logic into smart contracts using popular general purpose programming languages such as Golang [23] to create and modify assets or tokens on the network where a token is defined as anything of value. The algorithms defining the business logic encoded as smart contracts in this work have been presented in Algorithms 1–11 .

Hyperledger Fabric's modular architecture allows the operator to tailor implementation of trust models, transaction format and consensus mechanisms to the use case. Gur et al. [24] used Hyperledger Fabric to implement an energy metering system with privacy protection for smart grids. Che et al. [25] implemented a prosumer transaction system using Hyperledger Fabric with a focus on transaction authentication mechanisms. These features of Hyperledger Fabric thus, make it particularly suited for developing a peer to peer energy transaction system.

Such a system would involve three main actors/organizations. The Transaction Platform would be the first organization and it would include all the small scale energy prosumers and EV owners. The Power Company that supplies electricity to the community would be the second organization. Finally, the Storage Provider which stores the renewable energy generated by the prosumers would be the third organization in this network. In our previous work [26] we highlighted the need for an energy transaction system for prosumer communities and discussed the applicability of blockchain to build such a transaction system. Subsequently [27], we identified the potential stakeholders in this organization and proposed and defined tokens to encapsulate different types of energy assets.

Assets on the blockchain are represented by tokens in order to facilitate transactions. Tokens are broadly categorized into two categories fungible tokens (FT) and non fungible tokens (NFT), based on whether they are identical and interchangeable or not. FT are

interchangeable and identical in all respects and are divisible. On the other hand NFT cannot be substituted for other tokens of the same type and are indivisible. In an energy transaction system, energy assets with an attached Guarantee of Origin [28] have a unique identifier and are not interchangeable and can be implemented as NFT. Conversely, if energy assets are considered interchangeable, then tokens representing them can be broken up and traded in parts and can be implemented as FT. Both implementations are relevant in energy transaction systems.

Implementation of NFT and FT requires defining the lifecycle of the tokens from being issued to being redeemed. Methods and algorithms to take the tokens through the lifecycle need to be created and implemented and any challenges that arise need to be identified and suitably addressed. Comparative analysis of both types of tokens in terms of design, implementation, performance and limitations can provide a guide for future implementations of a transaction system. Mezquita et al. [29] proposed an architecture for transaction of fungible energy transactions on the Ethereum blockchain. However, this work did not feature an implementation of their proposal. Pop et al. [30] proposed a blockchain based peer to peer energy market for NFT focusing on aspects such as prosumer access control, automation of bids and offers matching. However, to the best of the our knowledge, a comparative analysis of the design, performance and limitations of FT and NFT based on an implementation has not been studied.

The motivation of this study is to design, implement and performance test a unified transaction system for energy assets represented as Fungible and Non Fungible Tokens that incorporates all the stakeholders and business relationships into a transparent and decentralized solution in order to address the identified gap in literature. Such a system would enable transactions of energy assets over a decentralized peer to peer network. A transparent system with inherent characteristics of immutability, authentication, access control and provenance tracing as well as the ability to encode business logic into smart contracts would enable automation of transactions and can be adapted for future implementations of microgrid transaction systems. Finally, performance is an essential aspect of a transaction system,

which can be characterized based on metrics such as transaction throughput and latency.

Based on the aspects discussed in this section and the motivations outlined, the main contributions of this work include the following:

(1) A Transaction Platform that includes the Prosumers and the EV owners, the Power Company and the Storage Provider were identified as stakeholders and represented as the three organizations in this business network. The trading relationships and energy assets that are transacted in this system were outlined. The energy assets were encapsulated in a blockchain token structure with token level consensus policies reflecting the identified stakeholders.

(2) The main stages in the lifecycle of all tokens were defined as Create, Bid, Transfer and Redeem. The methods and algorithms in order to take the token through the lifecycle were separately developed for NFT and FT due to the different characteristics of these token implementations.

(3) A proof of concept was implemented on the Hyperledger Fabric blockchain platform and the developed algorithms were encoded as smart contracts. Experiments were designed and executed in order to performance test the implementation based on transaction throughput and transaction latency metrics.

(4) The limitations of the study, proposed solutions and future avenues for investigation were identified.

(5) Based on the experiments and analysis, NFT and FT implementations were critically compared in terms of design, performance and limitations.

The remainder of the work has the following structure: Section 2 presents an overview of the system, describing the design rationale, the system participants and tokens and the relations between them. In Section 3, we delve deeper into the design and implementation of the tokenized energy assets, describe the design requirements and address those requirements by presenting methods and algorithms

that will take the token through the lifecycle for both NFT and FT versions. Section 4 describes the experimental infrastructure and presents the execution details of our experiments and the results obtained, as well as a comparative analysis of NFT and FT based on performance and limitations. In Section 5 we present the contributions of our work in the context of related works. Section 6 presents the salient conclusions from this study.

2 System Overview

The blockchain network consists of three organizations- the Transaction Platform, the Power Company and the Storage Provider. Each organization can have several client identities. A client identity is used by a registered user of the organization in order to identify themselves and access the network resources. The identity determines a user's access within the system.

2.1 System Participants and Tokens

Prosumers who want to sell their excess energy or buy energy from other prosumers, as well as EV owners who want to charge their batteries can be registered users of the Transaction Platform organization. As members of the network, prosumers can earn reward tokens for participating in demand response tasks, while EV owners can earn reward tokens for renting out their batteries as temporary storage devices. Moreover, prosumers can also use the Transaction Platform to access the Storage Provider and store or withdraw their excess stored energy.

Power Companies can directly enlist customers in their demand response efforts by offering reward tokens. Incentivization tokens can be offered to the prosumers for accomplishing tasks such as estimating their own consumption accurately in advance. Presenting tasks such as not running the air conditioner on a hot day in a gamified context can increase engagement and be rewarded by Gamification tokens. Representatives of power companies can create and offer Incentivization and Gamification tokens, with associated penalties for

reneging, for the prosumers to bid on. We consider bidding upon a token as registering binding interest in the token without negotiating the price or value. If the Power Company has access to the energy stored with the Storage Provider, this may further aid their demand response efforts.

The Storage Provider organization handles all the complexities and tasks of setting up a community level energy storage and abstracts away the details from the prosumers. The Storage Provider can be a separate business or can be owned by the Power Company. Pilot studies on the use of community level batteries for peak shaving are currently underway and in this case, often the community battery is also owned and operated by the Power Company [31].

In this system we consider, six different types of tokens in the system in order to fulfil six different functionalities.

(1) EUnit energy tokens for transacting renewable energy between prosumers

(2) EVUnit energy tokens for battery charging transactions between prosumers and EV owners.

(3) InUnit reward tokens offered to prosumers by the Power Company for tasks such as consumption estimation.

(4) GaUnit reward tokens offered to prosumers by the Power Company for energy flexibility in a gamified context.

(5) StUnit energy storage tokens for managing the community level energy
storage transactions.

(6) EStUnit reward tokens for managing the use of EV batteries as energy storage.

Figure 1 presents an overview of the system participants and the relationships between them.

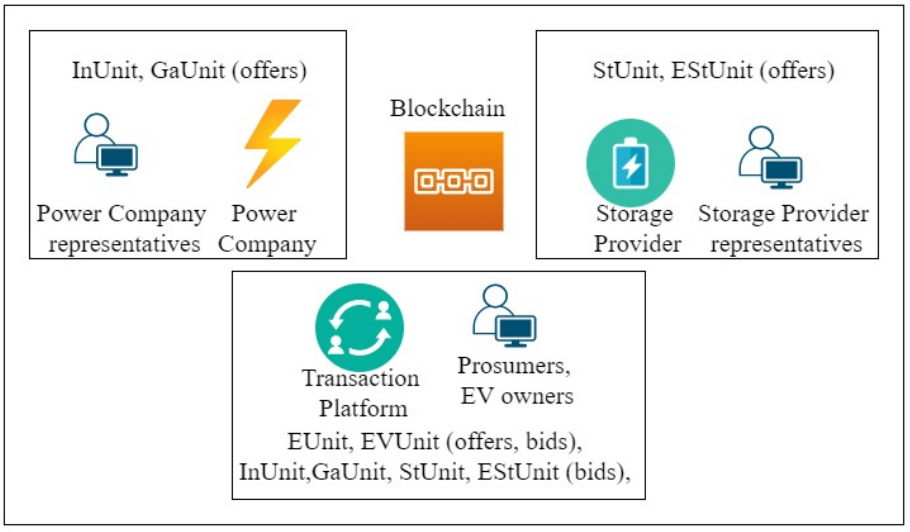


Figure 1: System Participants.

The implementation of the tokens as FT and NFT is done in two separate chaincodes, written in Golang v1.16 [23], each with smart contracts for functionalities that correspond to the lifecycle of the token, which are create, bid, transfer and redeem. Endorsement policies in Hyperledger Fabric stipulate how many or which organizations must endorse transactions. The chaincode level endorsement policies override the network level endorsement policies. The chaincode level endorsement policies for both of the chaincodes is that two out of the three organizations must endorse each transaction. This chaincode level policy will apply at token creation time for FT as well as NFT implementations. After the token is created, i.e., the key-value pair corresponding to the token is created in the world state, key level endorsement policies for the given token can be configured which will override the chaincode level endorsement policy. For each token, we configure the issuer organization as the organization of the client that invokes the token creation. We also configure a endorser organization for each token that is required to endorse all transactions involving that token. We add both these organizations as required endorsers to each token's key level endorsement policy. In Table 1, we present a mapping for token types and required issuer and endorsing organi-

zations. We ensure by encoding into the smart contract, that only a client of the stipulated issuer organization is permitted to create a token of the corresponding token type. This mapping is the same for FT and NFT implementations.

As shown in Table 1, for EUnit and EVUnit, the Storage Provider along with the Transaction Platform must endorse each transaction. We envision that the prosumers do not have any local storage and use the community storage to store their energy, and it is from here that the energy is supplied to a buyer. Thus, before being allowed to transact a token, the Storage Provider must check if the energy is actually physically stored for the mentioned amount. Also, the Storage Provider must make sure that the change of ownership occurs successfully at their end and then approve the transaction. Moreover, for StUnit and EStUnit, the Storage Provider as the issuer and the Transaction Platform as the place where the user interacts with these units, must endorse. Similarly, for all transactions involving InUnit and GaUnit, the Power Company and the Transaction Platform must endorse. The Power Company is the issuer and the Transaction Platform is where the prosumers access the system, so both must approve.

3 Design and Implementation

A token is acted upon by four operations in its lifecycle as shown in Figure 2. First the token is created by a seller. A buyer then bids upon the token, whereupon the seller transfers the value of token to the buyer. The owner of a token can redeem the value of the token.

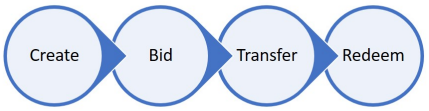


Figure 2: Operations of the token.

Table 1: Token issuers, endorsers and usage.

Token Type	**Issuer**	**Endorser**	**Use**
EUnit	Transaction Platform	Storage Provider	Prosumers
EVnit	Transaction Platform	Storage Provider	Prosumer to EVs
InUnit	Power Company	Transaction Platform	Prosumer Incentivization
GaUnit	Power Company	Transaction Platform	Prosumers Gamification
StUnit	Storage Provider	Transaction Platform	Storage token
EStUnit	Storage Provider	Transaction Platform	EV Reward token

3.1 Structure of the Token

Each token, whether FT or NFT in our implementation consists of a key value pair where the key is the ID of the token and the value consists of fields with information about the token as shown in Figure 3. However, there are differences in the the design and implementation of the contents of the token fields for FT and NFT and is described in Sections 3.3 and 3.4. The fields in the token are:

- *ID* uniquely identifies the token in the network.
- *AvailableAssets* lists the count of energy assets included in this token
- *Notforsale* is a Boolean flag that is set to TRUE if the token is not for sale.
- *Owner* stores the identity of the owner of the token.

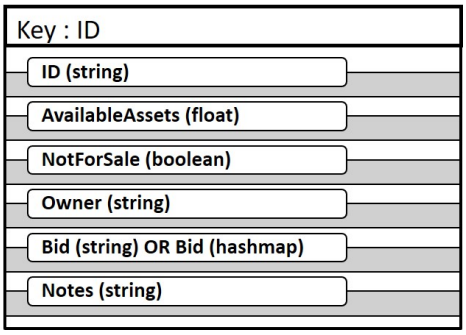


Figure 3: Structure of token.

- *Bid* field accepts a bid on the token. This field is implemented as a string in the case of NFT in order to receive a single bid and as a hashmap in case of the FT implementation in order to accept multiple bids. A bid in this system registers a binding interest in a token.
- *Notes* field is a placeholder to store other details of the token such as price or terms and conditions that may be needed in the exact use case but are not general enough for our work.

3.2 Design Requirements

The following are the requirements of such a system:

- Methods and algorithms must be provided to take the token through its entire lifecycle with four main operations: create, bid, transfer and redeem.
- The user must be able to query the entire list of all tokens of each type owned by them without having to remember the token ID.
- Only a member of the designated issuer organization of each type of token as defined in Table 1 must be permitted to initiate token creation. Moreover, token level endorsement must be configured on each created token so that the designated endorser organizations of each type of token are required to endorse transactions for that token.

- After the token is created, a designated owner is defined which can be either the issuer or the recipient of the token and only they must be allowed to initiate any transactions to the token. This is with the exception of placing a bid upon the token. For this purpose, another user must be able to access the token and register their bid.

- A user must be able to designate any of their tokens as not for sale and others should not be able to bid upon it.

- When a buyer bids upon a token, their client identity must be recorded, so that a transfer to them can be processed.

- If the token is NFT, then only one bid is permitted on a given token for the entire value of the token.

- For FT tokens, multiple bids are permitted and each must be recorded with the bid amount of energy assets and bidder identity. After each bid on a FT, the value of the available tokens should be updated and bidding should be allowed until all the energy assets in the token have been bid upon. If the same buyer bids multiple times on the same token, the older bid amount must not be lost. The new bid amount must be added to the buyer's bid.

- In FT tokens, the token must be able to accept further bids on the remaining value, without requiring the owner to transfer the value right away. Moreover, the owner should not have to wait for the entire value of the token to be bid upon before initiating transfer.

- The owner of the token must not be able to spend the token value that has been bid upon. For NFT tokens, the owner of a token with a bid on it must not be able to redeem that token at all. Similarly, for FT tokens, the owner must only be able to redeem value that is still available, i.e has not been bid.

3.3 Implementing Energy Assets as NFT

When a NFT is put up for sale, or to be won as a reward, the bidder must bid upon the token in its entirety. This allows the sellers to set different prices for different tokens of the same token type. Also, it allows the Power Company to set up different conditions for tokens of the same token type as per their business need.

LevelDB is used as the state database in this work as it offers better performance than CouchDB which is the other supported state database in Hyperledger Fabric [32]. As mentioned before, the ID field should be unique and each client must be able to query for a list of tokens they own, without having to remember token IDs. However, LevelDB does not support rich querying. So, we accomplish this with the use of range queries in Hyperledger Fabric which takes a start and end value of the ID and returns the list of tokens that have IDs that fall between the values or are equal to the provided values. This means that the client would be expected not only to create IDs having a lexically serialized order, but also remember the first and last token IDs to provide to the query function. If a client were to forget the range to use they would lose access to tokens outside that range. This would be untenable, so our smart contract takes over the task of generating IDs. When creating the the ID of any token, the client identifier and token type must be included in the ID so as to enable range queries allowing the user to query for all tokens they own of a particular type.

In case of NFT, this poses a challenge as multiple tokens can have the same owner and token type, so we need to add in a unique identifier to the ID to distinguish between different tokens each having the same client and token type. We could have saved the maximum value ID used in the system, or per client as a token and refer to that each time a token is created, ensuring that the number is unique across the entire system. Setting a centralized maximum value would need that token to be referred in every single transaction involving that client, creating a bottleneck and slowing down the process. Instead, the transaction ID which uniquely identifies the transaction within the scope of the channel was used and the ID for NFT of is the form:

ID: clientid_tokentype_transactionID

3.3.1 Creating a NFT

Algorithm 3 describes the creation of NFTs. The algorithm starts by checking the client's identity and the identity of the organization of which the client is a part. It then conducts a check to make sure that the invoker organization is permitted to create the requested token type and sets value of the additional organization apart from the invoker that must be added to the endorsement policy. The ID and Owner of the token will depend on whether the token is being created as part of a transfer to a buyer or is being created for the invoker. Algorithm 5 uses Algorithm 3 in order to create a token for a buyer, as will be described in Section 3.3.3. After creating and committing the new token, Algorithm 1 is called to set the key level endorsement policy to override the chaincode level policy. This algorithm accepts the ID of a token in the world state and the organizations that will be included in the new endorsement policy. It instantiates a new policy, adds the requested organizations and commits it.

Algorithm 1 SetTokenStateBasedEndorsement

1: **function** SetTokenStateBasedEndorsement(id string, issuerorg string, endorserorg string)
2: $endorsementPolicy \leftarrow statebased.NewStateEP()$
3: $endorsementPolicy.AddOrgs(statebased.RoleTypePeer, issuerorg, endorserorg)$
4: $policy \leftarrow endorsementPolicy.Policy()$
5: $SetStateValidationParameter(id, policy)$
6: **end function**

Algorithm 2 ReadToken

1: **function** ReadToken(id string)
2: $tokenJSON \leftarrow GetState(id)$
3: $token \leftarrow json.Unmarshal(tokenJSON)$
4: return token
5: **end function**

Algorithm 3 Create a NFT

1: **function** CREATENFT(tokentype string, creatingfortransfer bool, buyer string, availableassets int)
2: $invokerorg \leftarrow GetClientIdentity().GetMSPID()$
3: $invokerclient \leftarrow GetClientIdentity().GetID()$
4: $test \leftarrow tokentype + \text{“\_”} + invokerorg$
5: **switch test**
6: $\text{“}EUnit\_Org1MSP\text{”} or \text{“}EVUnit\_Org1MSP\text{”}$
7: $endorserorg = \text{“}Org3MSP\text{”}$
8: $\text{“}InUnit\_Org2MSP\text{”} or \text{“}GaUnit\_Org2MSP\text{”}$
9: $endorserorg = \text{“}Org1MSP\text{”}$
10: $\text{“}StUnit\_Org3MSP\text{”} or \text{“}EStUnit\_Org3MSP\text{”}$
11: $endorserorg = \text{“}Org1MSP\text{”}$
12: **Default: “Invoker Organization and Token Type combination invalid”**
13: **if** creatingfortransfer == true **then**
14: $idval \leftarrow buyer + \text{“\_”} + tokentype + \text{“\_”} + GetTxID()$
15: $owner \leftarrow buyer$
16: **else**
17: $idval \leftarrow invokerclient + \text{“\_”} + tokentype + \text{“\_”} + GetTxID()$
18: $owner \leftarrow invokerclient$
19: **end if**
20: $tokennew \leftarrow NewToken(ID \leftarrow idval, TokenType \leftarrow tokentype$, AvailableAssets $\leftarrow availableassets, Owner \leftarrow owner)$
21: $tokenJSON \leftarrow json.Marshal(tokennew)$
22: $PutState(idval, tokenJSON)$
▷ Saving new token
23: $SetTokenStateBasedEndorsement(tokennew.ID, invokerorg, endorserorg)$
▷ Calling Algorithm 1
24: **end function**

3.3.2 Bidding on a NFT

Algorithm 4 illustrates how a buyer can bid upon a token of their choosing. The requested token is read using Algorithm 2 and checked if it is for sale and that there is no bid on this token already. If there

is no bid, then the algorithm updates the bid field with the client identity of the buyer and commits it to state. As the token must be purchased in its entirety, there is no mention of the amount of energy assets in the bid.

Algorithm 4 Bid on a NFT

```
1: function BidNFT(id string)
2:     token ← ReadToken(id)                    ▷ Calling Algorithm 2
3:     if token.NotForSale == true then
4:         return "Token ID not for sale"
5:     end if
6:     bidderclient ← GetClientIdentity().GetID()
7:     if token.Bid == "" then
8:         token.Bid = bidderclient
9:     else
10:        return "There is already a bid on this token by token.Bid"
11:    end if
12:    tokenJSON ← json.Marshal(token)
13:    PutState(id, tokenJSON)
                                          ▷ Saving updated token
14: end function
```

3.3.3 Transferring a NFT

Using Algorithm 5 the owner of a token can transfer the token to the buyer who has bid upon it. The token is read using Algorithm 2 and it is checked that the invoker client is in fact the owner of the token and there exists a bid on the token. If the token exists, is bid upon and the transfer is requested by the owner, the transfer can take place. We will need the newly transferred token to be reflected when the new owner checks their list of tokens using Algorithm 7 which will use the ID of the token stored on the world state. However, we cannot actually edit the ID of the token on the world state, so we create a new token owned by the buyer for the same value using Algorithm 3 and destroy the original token owned by the seller.

Algorithm 5 Transfer a NFT

```
1: function TransferNFT(id string)
2:     token ← ReadToken(id)                    ▷ Calling Algorithm 2
3:     invokerclient ← GetClientIdentity().GetID()
4:     if invokerclient != token.Owner then
5:         return "The client invokerclient is not authorized to
   transfer token owned by token.Owner"
6:     end if
7:     if token.Bid == "" then
8:         return "No bid yet "
9:     end if
10:    CreateNFT(token.TokenType, TRUE, token.Bid,
   token.availableassets)
       ▷ Calling Algorithm 3, to create new token with buyer's id
11:    DelState(id)                    ▷ Delete the token with seller's id
12: end function
```

3.3.4 Redeeming a NFT

When the token is redeemed it is deleted from the world state, but the record of transactions remain in the ledger. The Algorithm 6 for redeeming a token, starts by getting the invoker client's identity to make sure they are the owner of the token being redeemed. Also, a owner should not be able to redeem a token that already has a bid, so the algorithm checks to make sure there is no bid on the token being redeemed. If a token with no bid is being redeemed by the owner, the redeem operation goes through as requested.

3.3.5 Querying for a list of owned NFTs

As mentioned in Section 3.2, an important design consideration is that the user must not be required to remember the IDs of all the tokens they own and the system must provide an easy way to query that. Algorithm 7 uses a range query to return a list of all tokens of the specified type owned by the invoking client.

Algorithm 6 Redeem a NFT

```
1: function REDEEMNFT(id string)
2:     token ← ReadToken(id)                          ▷ Calling Algorithm 2
3:     invokerclient ← GetClientIdentity().GetID()
4:     if invokerclient != token.Owner then
5:         return "The client invokerclient cannot redeem token owned by token.Owner"
6:     end if
7:     if token.Bid != nil then
8:         return "The client invokerclient cannot redeem token as it has a bid"
9:     end if
10:    DelState(id)
                                                      ▷ Deleting token
11: end function
```

Algorithm 7 Get my NFT

```
1: function GETMYNFT(tokentype string)
2:     ownerclient ← GetClientIdentity().GetID()
3:     checkstr ← ownerclient + "_" + tokentype + "_"
4:     resultsIterator ← GetStateByRange(checkstr + pad(0, 64), checkstr + pad(z, 64)
5:     defer resultsIterator.Close()
6:     var tokens []*Token
7:     for resultsIterator.HasNext() do
8:         queryResponse ← resultsIterator.Next()
9:         token ← json.Unmarshal(queryResponse.Value)
10:        tokens ← append(tokens, token)
11:    end for
12:    return tokens ▷ Returns all tokens of tokentype for requesting client
13: end function
```

The algorithm takes the token type queried for as input and gets the invoker client's ID to create start and end values for the range query by padding to the right to create alphanumeric strings of the

same length as the transaction ID in order to get the smallest and largest possible transaction IDs. This returns an iterator which loops through to produce the list of tokens to be returned to the user.

3.4 Implementing Energy Assets as FT

FT are those that are for all intents and purposes identical. Thus they can be broken up and traded in parts, and added up like currency. Each client will have at most six tokens, one of each type of tokens EUnit, EVUnit, InUnit, GaUnit, StUnit and EStUnit. Additional tokens, whether created or purchased will be added to this token value. Tokens redeemed or sold will be reduced from the token value. Thus tokens for a given client in the FT implementation act as accounts, where each client has at most six accounts. The ID of the token in the FT implementation is of the form:

ID: clientid_tokentype

Also, as mentioned in Section 3.1, in the FT implementation a token can have multiple bids from the same or different buyers. In order to accomplish this we implemented the Bid field as a hashmap. The hashmap stores a list of key value pairs where key stores the identity of the bidder and the value stores the bid amount. The Bid field thus takes bids on the token and keeps adding bids to the hashmap. If the same bidder bids multiple times on a token, the value for that client key is updated with the total of the bids.

3.4.1 Create a FT

Algorithm 8 for creating a FT first verifies that the invoking client is from an organization that is allowed to create the requested type of token.

Additionally, it sets the appropriate additional organization needed to endorse this token as described in Table 1. The token ID depends on whether the token is being created for the invoker or for transfer to a buyer. Next, the algorithm will check if the token exists by trying to read it. As this is a FT implementation, if the token exists already, a new token will not be created but the requested energy asset count will be added to the existing token.

Algorithm 8 Create a FT

```
1: function CreateFT(tokentype string, creatingfortransfer bool,
   buyer string, availableassets int)
2:     invokerorg ← GetClientIdentity().GetMSPID()
3:     invokerclient ← GetClientIdentity().GetID()
4:     test ← tokentype + "_" + ownerorg
5:     switch test
6:         "EUnit_Org1MSP" or "EVUnit_Org1MSP"
7:         endorserorg = "Org3MSP"
8:         "InUnit_Org2MSP" or "GaUnit_Org2MSP"
9:         endorserorg = "Org1MSP"
10:        "StUnit_Org3MSP" or "EStUnit_Org3MSP"
11:        endorserorg = "Org1MSP"
12:    Default: "Invoker Organization and Token Type
   combination invalid"
13:    if creatingfortransfer == true then
14:        id ← buyer + "_" + tokentype
15:        owner ← buyer
16:    else
17:        id ← invokerclient + "_" + tokentype
18:        owner ← invokerclient
19:    end if
20:    token ← ReadToken(id)                    ▷ Calling Algorithm 2
21:    if token! = nil then
22:        token.AvailableAssets ← token.AvailableAssets +
   availableassets
23:        tokenJSON ← json.Marshal(token)
24:        PutState(id, tokenJSON)
25:    else
26:        tokennew ← NewToken(ID ← id, TokenType ←
   tokentype, AvailableAssets ← availableassets, Owner ← owner)
27:        tokenJSON ← json.Marshal(tokennew)
28:        PutState(id, tokenJSON)
29:        SetTokenStateBasedEndorsement(tokennew.ID,
   ownerorg, endorserorg)
                                              ▷ Calling Algorithm 1
30:    end if
31: end function
```

If it is not found, this means that the invoker (if creating for self) or buyer (if creating for transfer) does not already own a token of this type. If so, a new token is created with the appropriate ID, supplied token type and energy asset count and added to the world state. Moreover, the applicable key level endorsement policy is set using Algorithm 1.

Algorithm 9 Bid on a FT

```
1: function BidFT(id string)
2:     token ← ReadToken(id)                    ▷ Calling Algorithm 2
3:     if token.NotForSale == true then
4:         return "Token ID is not for sale"
5:     end if
6:     if token.AvailableAssets < bidvalue then
7:         Return "Available value token.AvailableAssets is less than
   bid bidvalue
8:     end if
9:     bidderclient ← GetClientIdentity().GetID()
10:    if token.Bidmap[bidderclient]! = nil then
11:        token.Bidmap[bidderclient] ← existingval + bidvalue
12:    else
13:        token.Bidmap[bidderclient] = bidvalue
14:    end if
15:    token.AvailableAssets = token.AvailableAssets − bidvalue
16:    tokenJSON ← json.Marshal(token)
17:    PutState(id, tokenJSON)                  ▷ Saving updated token
18: end function
```

3.4.2 Bid on a FT

Algorithm 9 for bidding on a FT accepts the ID of the token as well as the number of energy assets in the bid. It reads the token using Algorithm 2 and checks if the token is available for sale and that the number of energy assets in the bid do not exceed the value of the token. If the same buyer has bid on the token before, the new bid value is added to the old bid value in the existing key value pair of the hashmap. If this is a new buyer, a new key value pair is created in the

hashmap to accept the bid, where the key is the client identity and value is the amount of the bid. Finally, the count of available assets in the token is reduced by the amount of the bid and the updated token is committed to the state.

Algorithm 10 Transfer a FT

```
1: function TransferFT(id string)
2:     token ← ReadToken(id)                    ▷ Calling Algorithm 2
3:     invokerclient ← GetClientIdentity().GetID()
4:     if invokerclient != token.Owner then
5:         return "The client invokerclient is not authorized to
   transfer token owned by token.Owner"
6:     end if
7:     for key, value ← range(token.Bidmap) do
8:         CreateToken(token.TokenType, TRUE, key, value)
   ▷ Calling Algorithm 8
9:         delete(token.Bidmap, key)
10:    end for
11:    tokenJSON ← json.Marshal(token)
12:    PutState(id, tokenJSON)          ▷ Save updated token after
   transfers
13: end function
```

3.4.3 Transfer a FT

Algorithm 10 for transferring a FT, first reads the token using Algorithm 2 and verifies that the invoker client is the owner of the token being transferred. Then for each bid in the bid hashmap, it processes the value transfer to the buyer, using Algorithm 8. If the token for the buyer exists already, it is updated with the transferred value. Otherwise a new token is created for the buyer and a key level endorsement policy is set. The processed bid is now removed from the map.

When all the bids have been processed and deleted, finally, it updates the token that was being transferred with the newly empty bid field into the state. As the value of available energy assets on the token is reduced by the value of the bid whenever a bid is placed,

as explained in Section 3.4.2, the count of available assets on the token already reflects the value after deducting the bid amounts.

3.4.4 Redeem a FT

Tokens in the FT implementation work like accounts, so when a FT is redeemed, some or all of the available energy assets may be redeemed, but the token itself not deleted as is the case for the NFT implementation described in Section 3.3.4. The Algorithm 11 for redeeming FT begins by reading the token using Algorithm 2 and checking that the invoker client owns the token and that the requested redeem amount is not greater than the total available energy assets in the token. As the available assets count is reduced by the bid amount for every bid on the token as described in Section 3.4.2, it reflects the true count of the assets available to the token owner for redeeming. Next, the available energy assets on the token is reduced by the redeem value and the updated token is then committed to the world state.

Algorithm 11 Redeem a FT

1: **function** REDEEMFT(id string, redeemcount int)
2: $token \leftarrow ReadToken(id)$ ▷ Calling Algorithm 2
3: $invokerclient \leftarrow GetClientIdentity().GetID()$
4: **if** $invokerclient! = token.Owner$ **then**
5: return "The client $invokerclient$ is not authorized to redeem token $token.ID$ owned by another"
6: **end if**
7: **if** $redeemcount > token.AvailableAssets$ **then**
8: return "The token $token.ID$ has $token.AvailableAssets$ assets, which is less than requested redeem value $redeemcount$
9: **end if**
10: $token.AvailableAssets \leftarrow token.AvailableAssets - redeemcount$
11: $tokenJSON \leftarrow json.Marshal(token)$
12: $PutState(id, tokenJSON)$
13: **end function**

3.4.5 Querying for a List of Owned FT

For any client there will be six FT at most, one corresponding to each of the types of tokens in the system which are EUnit, EVUnit, GaUnit, InUnit, StUnit and EStUnit. The Algorithm 2 provides the list of FT of the specified token type owned by the invoking client.

3.5 Complexity of the Algorithms

The complexity of an algorithm describes its efficiency in terms of the size of the input. The two main complexity measures of the efficiency of an algorithm are time and space complexity. Time complexity describes the time taken to execute an algorithm independently of factors that are unrelated to the algorithm. Factors such as programming language, memory cache, type of processing capacity and compiler optimization are not related to the algorithm itself but can affect the actual time taken to execute the algorithm. Similarly, space complexity describes the amount of memory space needed to execute an algorithm independently of the actual hardware used.

NFT has operations, NFT Create (Algorithm 3), NFT Bid (Algorithm 4), NFT Transfer (Algorithm 5), NFT Redeem (Algorithm 6) and NFT BulkRead (Algorithm 7). Similarly, FT has the operations FT Create (Algorithm 8), FT Bid (Algorithm 9), FT Transfer (Algorithm 10) and FT Redeem (Algorithm 11).

In Algorithm 8 for FT, if a token exists already, then the token will simply be updated and the key level endorsement policy will not be set again. In order to distinguish between the two types of create operations in FT, we call the operation FT ReCreate if the token is updated and the endorsement policy is not set again. If the token is created and the key level endorsement policy is configured, we call it FT Create. Similarly, for the Algorithm 10, we have two operations FT Transfer, if the token is created and FT ReTransfer if the token is updated with the transferred value.

Table 2 shows the time and space complexities of all the algorithms described in Sections 3.3 and 3.4.

Table 2: Time and space complexities of algorithms

Token Type	Operation	Time Complexity	Space Complexity
NFT	Create	O(1)	O(1)
NFT	Bid	O(1)	O(1)
NFT	Transfer	O(1)	O(1)
NFT	Redeem	O(1)	O(1)
NFT	Read	O(1)	O(1)
NFT	BulkRead	O(n)	O(n)
FT	Create	O(1)	O(1)
FT	ReCreate	O(1)	O(1)
FT	Bid	O(1)	O(1)
FT	Transfer	O(n)	O(n)
FT	ReTransfer	O(n)	O(n)
FT	Redeem	O(1)	O(1)
FT	Read	O(1)	O(1)

In Hyperledger Fabric, create, update, endorsement policy configurations and deletes are all processed as writes to the state. Thus, NFT Create involves no read operations and two write operations, one to create the token and one to configure the endorsement policy. Also, this operation only uses space to store that one token. The time and space complexities for this algorithm are O(1), as for any invocation, only one token will be created using the mentioned operations. Similarly, NFT Bid (one read, one write), NFT Transfer (one read, three writes) NFT Redeem (one read, one write), NFT Read (one read), FT Create (one read, two writes), FT ReCreate (one read, one write), FT Bid (one read, one write), FT Redeem (one read, one write) and FT Read (one read) will have a constant number of operations as well as space usage for any invocations and thus have both time and space complexities of O(1).

NFT BulkRead will execute read operations, which will depend on

and grow with the number of tokens owned by the invoking client. Here, each token returned by the range query is considered to be a separate read. Similarly, the space used will also depend on the number of tokens returned. Thus, the time and space complexities of NFT BulkRead are both O(n).

FT Transfer and FT ReTransfer will execute one read in order to read the seller's token. Then, both operations will execute a loop based on the number of bids present. For each bid, they will execute one read to check if the buyer's token exists and one write to either create (FT Transfer) or update (FT ReTransfer) the buyer's token. FT Transfer will execute an additional write in order to set the key level endorsement policy for the buyer's token. In each execution of the loop, both operations will use the space required to operate upon two tokens, the buyer's token and the seller's token. As the number of times this loop is run will depend on the number of bids present, the time and space complexities of both FT Transfer and FT ReTransfer are O(n). Finally, at the end of the loop, the seller's token will be updated with a single write, which is constant for any invocation.

Multiple transfers that originate from a single FT perform transfer of value to multiple distinct recipients in a FT Transfer operation. In order to complete comparable transfers in NFT, we would need to perform multiple separate transactions, one for each bid. Thus, even though the time and space complexities of the presented algorithms for FT Transfer and ReTransfer are O(n), we do not consider them to be a bottleneck as they cannot be considered equivalent to a single NFT transfer. A fair comparison, between the value transfer operations Transfer and ReTransfer for FT and Transfer for NFT, should thus consider a single bid per token. We performance test and make a comparison between these three algorithms in Section 4.2.

For NFT, retrieving all tokens of a type owned by a client is expected to be a bottleneck if executed on chain. We experimentally investigate this in Section 4.2 and the results of our experiments are shown in Figure 10.

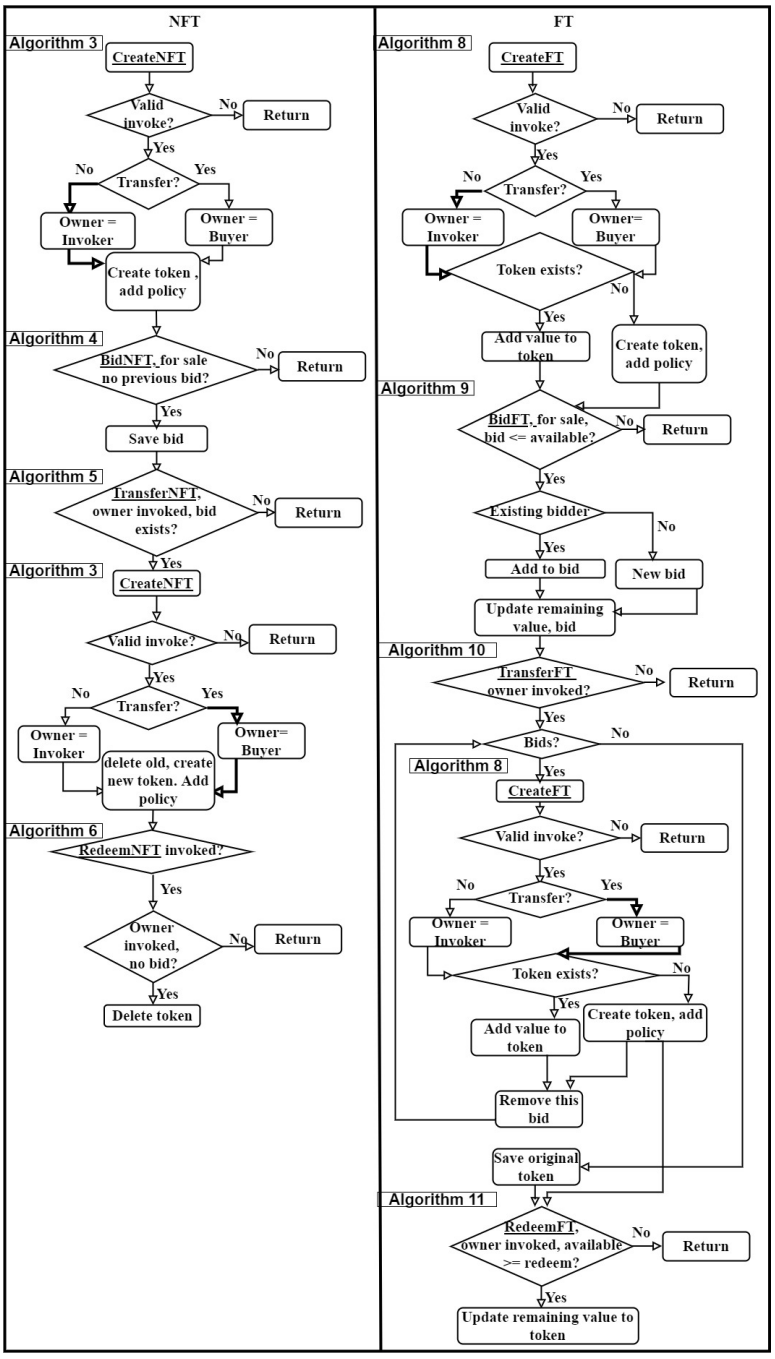


Figure 4: Lifecycle for Fungible Tokens (FT) and Non Fungible Tokens (NFT).

3.6 Token lifecycle

The algorithms developed for managing token lifecycles for both NFT and FT, have been described in detail in Sections 3.3 and 3.4 respectively. The overall token lifecycle and the associated algorithms called at different stages from create to redeem have been summarized as flowcharts in Figure 4. The sequence of experiments shown later on in Figure 5 were developed based on this workflow in order to performance test and compare the implementations of the specific algorithms.

The left pane in Figure 4 shows the lifecycle of a NFT. First, the seller invokes Algorithm 3 (Section 3.3.1) to create the token. If the invoke is valid, a check is performed to see if the token is being created for a transfer to a buyer. At this stage in the lifecycle, the token is being created for the invoker so the execution path highlighted in the flowchart applies and the invoker is the owner of the token. The token is then created and the token level endorsement policy is set. The token is now ready to accept a bid. The buyer calls Algorithm 4 (Section 3.3.2) and if the token is for sale and no previous bid is present, the bid is accepted. The seller then calls Algorithm 5 (Section 3.3.3), to transfer the token to the buyer who has bid upon it. In order to do so, a new token is created for the buyer by calling Algorithm 3 and executing the path highlighted in the flowchart. This new token has no bid on it. The original token owned by the seller is deleted. The current owner of the token can redeem the token using Algorithm 6 (Section 3.3.4) if there is no existing bid on it. A redeemed NFT is deleted from the state.

The right pane of Figure 4 summarizes the lifecycle of a FT. First, Algorithm 8 (Section 3.4.1) is invoked by the seller. At this stage of the lifecycle, the token is being created for the invoker. If the invoke is valid, a check is performed to see if a token for the same token type and owner combination exists. If such a token exists, the value of the token to be created is added to the value of the existing token and no new token is created. Otherwise, a new token with a token level endorsement policy is created. This token is now ready to accept bids. A buyer uses Algorithm 9 (Section 3.4.2) in order to bid upon the token. If the same bidder has already bid upon

NFT experiment sequence		FT experiment sequence	
	Balance		**Balance**
Alice: Creates 10,000 NFT of 10 assets each	Alice: 10,000 NFT *10	Alice: Creates 10,000 FT of 10 assets each	Alice: 10,000 FT *10
Bob: Bids on 10,000 NFT for full value	Alice: 10,000 NFT *10	Bob: Bids on 10,000 FT, 1 bid per token, 5 assets per bid	Alice: 10,000 FT *10
Alice: Creates 10,000 NFT of 10 assets each for Bob, Deletes 10,000 NFT for Alice	Alice: 0 NFT Bob: 10,000 NFT *10	Alice: Creates 10,000 FT for Bob with 5 assets each. Reduces Alice's 10,000 FT by 5 assets each	Alice: 10,000 FT *5 Bob: 10,000 FT *5
Bob: Redeems 10,000 NFT	Alice: 0 NFT Bob: 0 NFT	Bob: Redeems 10,000 FT for 2 assets each	Alice: 10,000 FT *5 Bob: 10,000 FT *3
		Alice: Adds 10 assets each to 10,000 FT accounts	Alice: 10,000 FT *15 Bob: 10,000 FT *3
		Bob: Bids on 10,000 FT, 1 bid per token, 5 assets per bid	Alice: 10,000 FT *15 Bob: 10,000 FT *3
		Alice: Adds 5 assets each to Bob's 10,000 FT. Reduces Alice's 10,000 FT by 5 assets each	Alice: 10,000 FT*10 Bob: 10,000 FT *8
		Bob: Redeems 10,000 FT for 2 assets each	Alice: 10,000 FT *10 Bob: 10,000 FT *6

Figure 5: Sequence of experiments.

the token, the value of the new bid is added to the existing bid, else a new bid is accepted on the token. In contrast to a NFT, a FT can accept multiple bids on the available value, and after each bid the value of the token is reduced by the bid amount. The seller, using Algorithm 10 (Section 3.4.3), for each bid will create a new token for the buyer using Algorithm 8 and remove the bid from the original token. This will be done until all the bids are removed from the token. A token can be redeemed by its owner for the value available in the token. Thus, the buyer can redeem the newly purchased token and the seller can redeem the value left on their token using Algorithm 11 (Section 3.4.4). A FT that is redeemed is not deleted, but the value of the token is reduced by the redeem amount.

4 Experimental Setup, Results and Discussion

The experimental infrastructure included 5 Virtual Machines (VM) created on a cloud environment. Each VM used Ubuntu 20.04 and had 32 GB RAM, 4 dedicated virtual CPUs and a 100 GB SSD. The nodes of the network were implemented as Docker containers with Docker version 19.03 and Docker Compose version 1.26 connected in a Docker Swarm for availability. Hyperledger Fabric v2.3.0 was the blockchain platform and Hyperledger Caliper v 0.4.2 was used to generate the load and measure the performance. Both are the latest stable versions at the time of writing. The Ordering service was implemented using the RAFT [33] consensus algorithm and had 3 Ordering Service Nodes (OSN) as recommended in the Hyperledger Fabric official documentation [34]. LevelDB was the state database due its performance advantage. The three organizations in our network are implemented with one peer node each and run on separate VMs. One VM is dedicated to running the Hyperledger Caliper and another VM runs the Ordering Service which is implemented as a separate Orderer Organization. In a production implementation, cloud security issues and mitigation strategies would need special consideration. Singh et al. [35] conducted an extensive survey of specific threats and solutions to be considered for better security

management for a cloud based service.

4.1 Experimental Setup

In order to performance test the implementation of each algorithm, the sequence of experiments shown in Figure 5 were designed based on the token lifecycle shown in Figure 4 and explained in Section 3.6, for NFT and FT algorithms.

A client, Bob was created for the Transaction Platform and another client Alice was the created for Storage Provider. The left pane on Figure 5 shows the sequence of operations for experimental evaluation of the NFT implementation. Using Algorithm 3 Alice creates 10,000 NFT each with value of 10 assets. Bob bids upon each of these tokens using Algorithm 4 and as this is a NFT implementation, the bids are for the whole value of the asset. Alice then initiates transfer to Bob using Algorithm 5, which involves creating 10,000 NFT owned by Bob each with the value of 10 assets and deleting all 10,000 NFT owned by Alice. Finally, using Algorithm 6, Bob redeems the complete value of all 10,000 NFT.

Similarly in the FT implementation, clients Alice and Bob are created for the Storage Provider and the Transaction Platform respectively. The sequence of operations for experimental evaluation of the FT implementation is shown in the right pane of Figure 5. In order to have a fair comparison between FT and NFT implementations, and because it was not feasible to implement thousands of distinct clients in order to create 10,000 tokens of the predefined token types, Alice creates 10,000 separate FT with 10 assets each using Algorithm 8, effectively creating 10,000 accounts owned by Alice. Bob places one bid per FT with 5 assets per bid on all 10,000 FT using Algorithm 9. Alice initiates transfer using Algorithm 10 which creates 10,000 FT for Bob, each with value 5 assets and reduces the value of Alice's FT by the bid amount, in this case 5 assets. Bob redeems 10,000 FT for 2 assets each using Algorithm 11 leaving a balance of 10,000 FT with 3 assets each for Bob. Alice's balance at this point is 10,000 FT with 5 assets each.

Alice again initiates creation of 10,000 FT with the same IDs as before with 10 assets each using Algorithm 8. This time, however,

as the FT already exist, the created value is added to the existing FT. In order to distinguish this operation from the create operation executed before, we call this ReCreate. Bob again bids on all 10,000 FT with 1 bid per FT and 5 assets per bid using Algorithm 9. Alice initiates transfer using Algorithm 10. Bob's FT exist already, as they were created before. So, the algorithm adds the value of the bid to Bob's FT making Bob's balance 10,000 FT with 8 assets each. Alice's balance after transfer is 10,000 FT with 10 assets each. In order to distinguish this from the transfer done before, we call this ReTransfer. Finally, Bob redeems all 10,000 FT for the value of 2 assets each leaving a balance of 10,000 FT with 6 assets each.

In case of FT, the number of key operations in each operation depend on the number of bids received on each token as explained in Section 3.5. However, for a fair comparison, for both NFT and FT, we place only one bid on each token as described above and in the right pane of Figure 5. Hyperledger Caliper was used to drive the load and measure performance as mentioned before. The load was driven with four worker processes and the fixed load rate control mechanism was configured. The fixed load rate controller starts with a configured transaction send rate in transactions per second (TPS) and maintains a defined queue length of unfinished transactions in the network by modifying the send rate. The starting send rate was set to 1000 TPS for all our experiments, while the queue length was varied. The sequence of operations between Alice and Bob described above was conducted 10 times each for NFT and FT implementations by varying the queue length from 100 unfinished transactions to 1000 unfinished transactions with equal increments of 100 TPS. In addition, experiments were conducted on the Read operation described in Algorithm 2 for both NFT and FT with 1 read per query and 10,000 queries in one iteration, with a total of 4 iterations with queue lengths varying from 100 unfinished transactions to 400 unfinished transactions with a step size of 100. Increasing the queue length beyond this point did not show any increase in throughput. Moreover, in case of NFTs, each client can have several tokens, which can be retrieved using Algorithm 7. In order to test this algorithm, experiments were conducted for the bulk read operation by running 4 iterations with 10,000 bulk read queries in one iteration

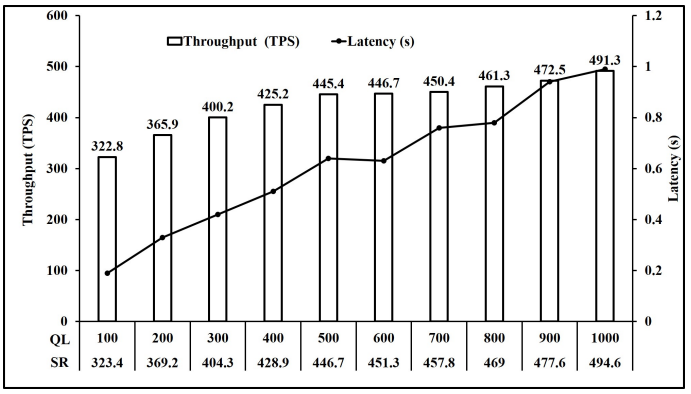
with each bulk read query returning 10,000 NFT. The queue lengths were varied from 100 to 400 over the 4 iterations with a step size of 100. Queue lengths of more than 400 unfinished transactions did not result in any improvement in throughput. Table 3 shows a summary of key transactions such as read, write and key level endorsement policy configuration that were involved in each operation conducted in the experimental evaluation.

Table 3: Summary of key operations involved in each operation for experimental evaluations

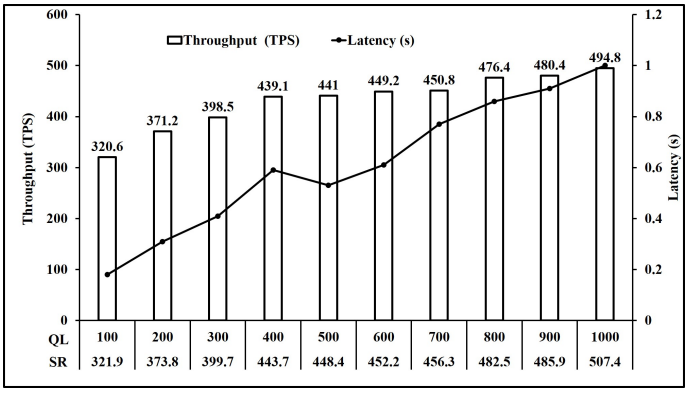
Token Type	Operation	Write	Set Policy	Read
NFT	Create	1	1	0
NFT	Bid	1	0	1
NFT	Transfer	2	1	1
NFT	Redeem	1	0	1
NFT	Read	0	0	1
NFT	BulkRead	0	0	10,000
FT	Create	1	1	1
FT	ReCreate	1	0	1
FT	Bid	1	0	1
FT	Transfer	2	1	2
FT	ReTransfer	2	0	2
FT	Redeem	1	0	1
FT	Read	0	0	1

4.2 Results and Discussion

An endorsing peer simulates each transaction before endorsing and creates a read-write set. Deletes are processed by setting a delete marker in the write set. Similarly, key level endorsements, creation and updating of key value pairs are also processed as writes to the state in the read-write set. Thus for performance analysis, we consider them all as writes to the state.

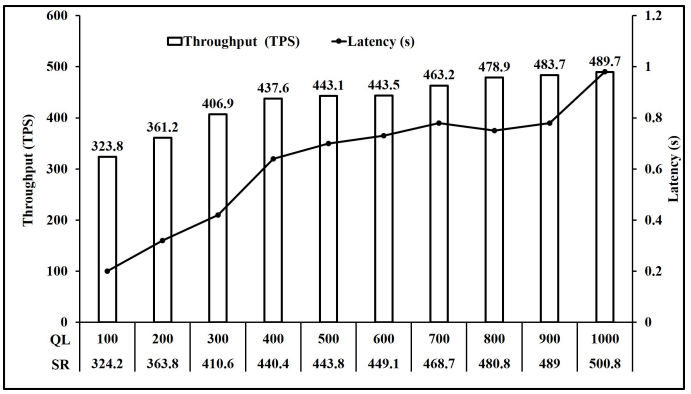


(a) NFT Bid- Peak throughput = 491.3 TPS

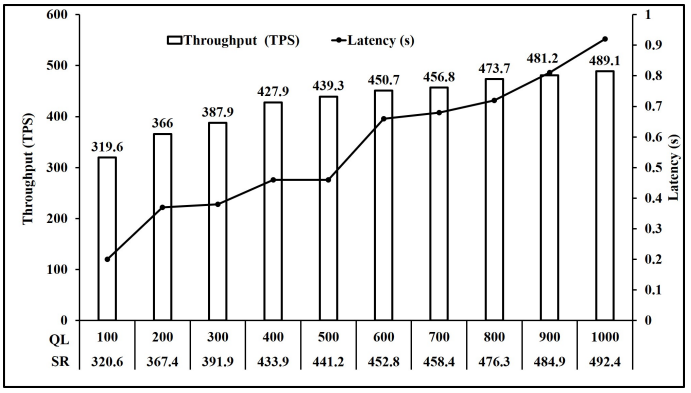


(b) NFT Redeem- Peak throughput = 494.8 TPS

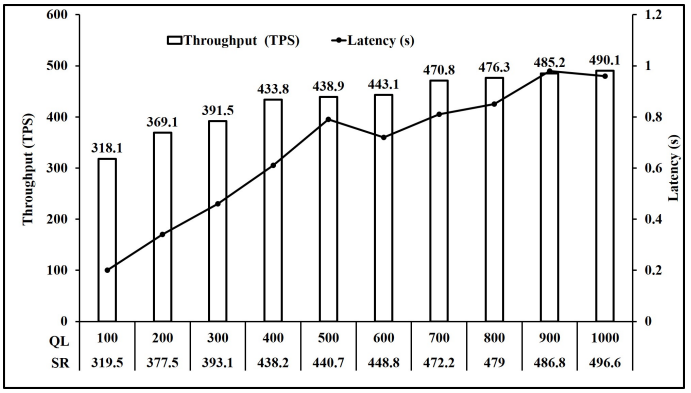
Figure 6: *Cont.*



(c) FT ReCreate- Peak throughput = 489.7 TPS



(d) FT Bid- Peak throughput = 489.1 TPS

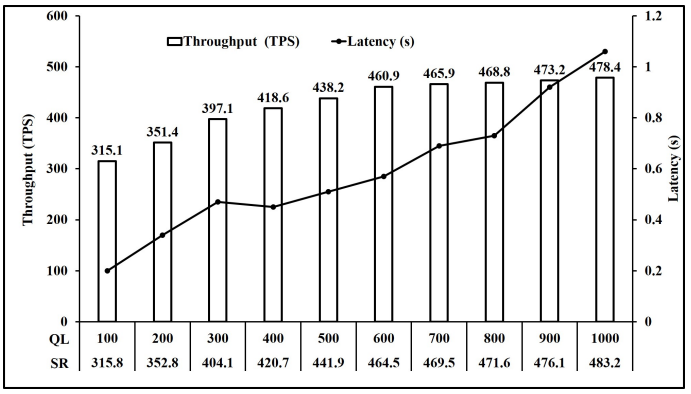


(e) FT Redeem- Peak throughput = 490.1 TPS

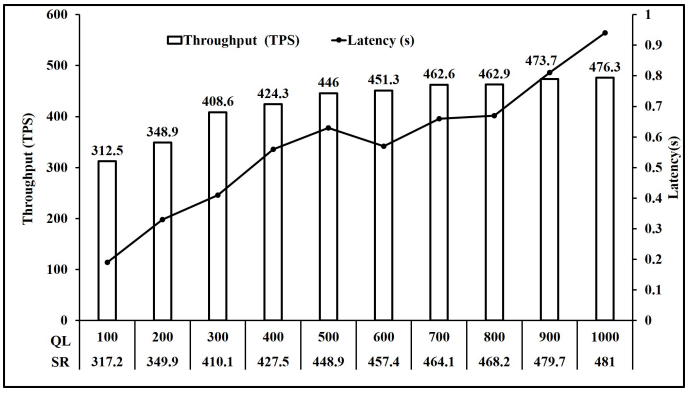
Figure 6: Operations with one write key operation (Queue Length: QL, Send Rate (TPS): SR).

For our experiments, we define transaction complexity of an operation as the number of writes to the state executed in that operation. Based on Table 3, we can group the operations based on transaction complexity. Figure 6 shows the performance of operations involving one write key operation. Here, we observe that all five operations NFT Bid, NFT Redeem, FT ReCreate, FT Bid and FT Redeem have comparable peak throughput of 491 TPS on average. Similarly, Figures 7 and 8 present the performance of operations involving two and three write key operations respectively. It is seen in Figure 7 that all three operations NFT Create, FT Create and FT ReTransfer have comparable peak throughput of 475.6 TPS on average. Also, Figure 8 shows that operations NFT Transfer and FT transfer also have comparable peak throughput of 449.55 TPS on average. Performance of read operations for FT and NFT shown in Figure 9 also show a comparable peak throughput of 845.95 TPS on average for both NFT and FT implementations. As mentioned in Section 4.1, we used the fixed load rate controller in Hyperledger Caliper which maintains the configured queue length by modifying the send rate. Figure 6a shows that when the queue length is increased from 100 unfinished transactions to 1000 unfinished transactions, the request send rate achieved rose from 323.4 to 494.6. Moreover, the throughput achieved rose from 322.8 TPS to 491.3 TPS at a cost to the per transaction latency which also rose from 0.19 s to 0.99 s. Similar observations can be made for all operations in Figures 6 and 7–10.

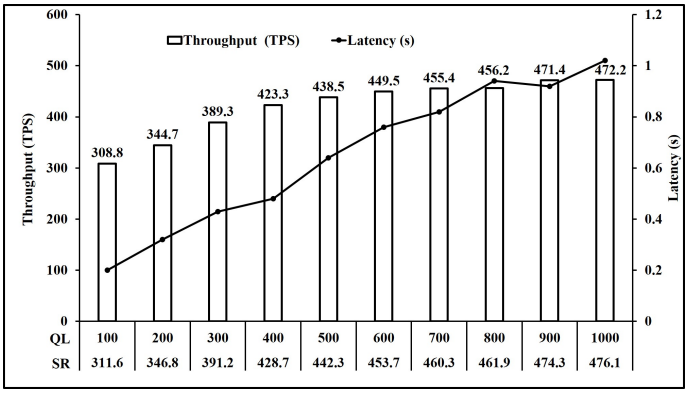
Comparing Figures 6 and 7, we see that the throughput achieved reduces from 491 TPS on average to 475.6 TPS on average when the number of writes per transaction increased from 1 to 2. Similarly, comparing Figures 7 and 8, we see that the throughput achieved further reduces to 449.55 TPS on average when number of writes per transaction increased from 2 to 3.



(a) NFT Create- Peak throughput = 478.4 TPS



(b) FT Create- Peak throughput = 476.3 TPS



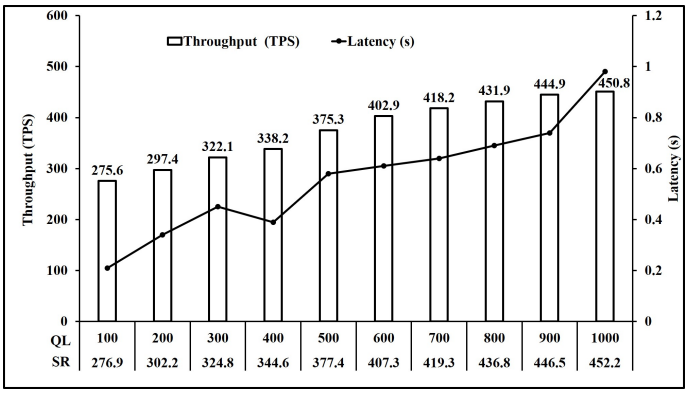
(c) FT ReTransfer- Peak throughput = 472.2 TPS

Figure 7: Operations with two write key operations (Queue Length: QL, Send Rate (TPS): SR).

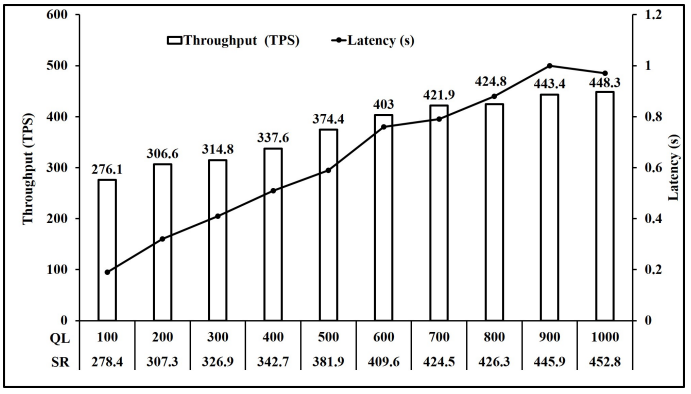
Analysis of Figure 7a,b shows that a similar performance is achieved for Create operation FT and NFT, as the throughput achieved is 476.3 TPS and 478.4 TPS respectively. Moreover, FT Bid in Figure 6d and NFT bid in Figure 6a are 489.1 TPS and 491.3 TPS respectively. Similarly, Transfer operation throughput for FT shown in Figure 8b and NFT shown in Figure 8a are 448.3 TPS and 450.8 TPS respectively. Finally Redeem operation throughput for FT shown in Figure 6e and NFT shown in Figure 6b are similar at 490.1 TPS and 494.8 TPS respectively. Read operation throughput for FT and NFT both shown in Figure 9 are 846.7 TPS and 845.2 TPS respectively. Thus, for the Create, Bid, Transfer, Redeem and Read, FT and NFT show similar performance for the same number of tokens. However, FT operations ReCreate and ReTransfer, do not set the key level endorsement policy, as tokens with their associated key level endorsement policy already exist as compared to Create and Transfer for FT and NFT. This is reflected in the performance as ReCreate as shown in Figure 6c achieved throughput of 489.7 TPS compared to 476.3 TPS for FT Create as shown in Figure 7b and 478.4 TPS for NFT Create as shown in Figure 7a. Similarly, ReTransfer throughput shown in Figure 7c at 472.2 TPS was higher than 448.3 TPS for FT Transfer shown in Figure 8b and 450.8 TPS for NFT transfer shown in Figure 8a.

For NFT, Bid (Figure 6a) and Redeem operations (Figure 6b) were the fastest at 491.3 TPS and 494.8 TPS respectively, followed by Create (Figure 7a) at 478.4 TPS and then by Transfer (Figure 8a) at 450.8 TPS. For FT Bid (Figure 6d), Redeem (Figure 6e) and Recreate (Figure 6c) were the fastest at 489.1 TPS, 490.1 TPS and 489.7 TPS respectivly. Create (Figure 7b) and ReTransfer (Figure 7c) were slower at 476.3 TPS and 472.2 TPS respectively, while Transfer (Figure 8b) at 448.3 TPS was the slowest operation for FT.

Finally, Figure 10 shows the results of performance testing the implementation of algorithm 7 for retrieving all NFT owned by a client by performing bulk read operations. As mentioned in Table 3, each operation involves bulk reads of 10,000 NFT. As expected the performance achieved is poor. The peak throughput achieved in our experiments was 13 TPS for a latency of 18.44 s at the queue length of 400 unfinished transactions. Thus, the execution of Algorithm 7



(a) NFT Transfer- Peak throughput = 450.8 TPS



(b) FT Transfer- Peak throughput = 448.3 TPS

Figure 8: Operations with three write key operations (Queue Length: QL, Send Rate (TPS): SR).

should be considered to be handled off-chain as is mentioned in the Hyperledger Fabric official documentation [36].

4.3 Comparison of Non Fungible Tokens and Fungible Tokens

An advantage of the NFT implementation is that it allows the seller of energy assets to set different prices for different tokens of the same token type. Similarly, this implementation permits the actor offering rewards to set different conditions for rewards of the same type.

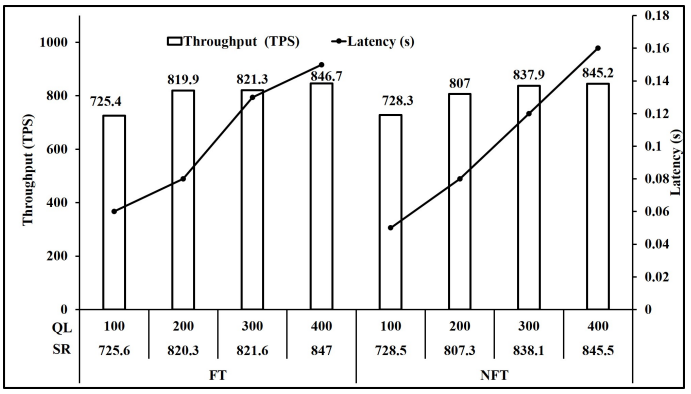


Figure 9: Read operation- Peak Throughput: FT- 846.7 TPS, NFT- 845.2 TPS (Queue Length: QL, Send Rate (TPS): SR).

However, in cases where standardization of assets is a requirement, the FT implementation offers an advantage as it ensures uniformity between tokens of the same token type.

FT maintain an updated count of tokens for each token type for each client in a single token, so that each client has at most 6 tokens or accounts. Thus, the total count of all energy assets of a particular type owned by a client, can be retrieved in a single read operation using Algorithm 2. This algorithm has time and space complexities of O(1) as explained in Section 3.5. In our implementation, the peak throughput for read operation was over 845 TPS with sub second latency as shown in Figure 9.

In the NFT implementation, a client can have several tokens of the same token type. Thus, in order to retrieve a list of all energy assets of a type owned by a client, a bulk read operation needs to be performed using Algorithm 7. As explained in Section 3.5, this algorithm has time and space complexities of O(n) as the time and space required to run this algorithm depends on the number of tokens to be retrieved. In our implementation, we tested the performance for a bulk read operation for 10,000 transactions in each iteration, where each transaction retrieves 10,000 NFT. As shown in Figure 10, the peak throughput was 13 TPS, while average latency was over 18 seconds.

Due to this, while getting all FT for a client can be executed on

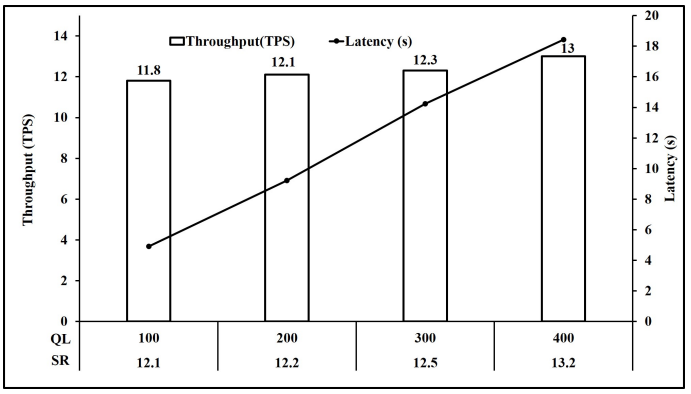


Figure 10: NFT Bulk Read Operation. Peak Throughput 13 TPS (Queue Length: QL, Send Rate (TPS): SR).

chain, getting all NFT may need to be executed off-chain. Moreover, Baliga et al. [37] have shown that with the increase in number of total tokens on the state, the read performance suffers marginally, which could be an issue for NFT implementations, for a very large number of tokens in state. Thus, FT have a clear advantage in terms of retrieving all assets of a given type owned by a client.

FT implementations, however, have limitations with concurrent execution of transactions. When two operations try to update the same token, for instance when two buyers try to bid on the same token or when two sellers try to transfer value to the same token, at most one operation will succeed. NFT implementations avoid this problem. When creating or transferring a NFT, a new token is created each time with a distinct key which avoids the problem of contention. Moreover, the issue of accepting multiple bids on the same token does not arise for NFT. Thus, in terms of concurrent execution of transactions, NFT implementations have a clear advantage.

Another implementation specific consideration is the size of the token. As a single FT token accepts multiple bids, the size of the token increases, which can degrade the performance of the network [37, 38]. So, specific implementations could consider setting a maximum number of unprocessed bids a token can have at any given time.

Additionally, we have shown that while Create, Bid, Transfer and Redeem operations are similar in performance for FT and NFT,

ReCreate and ReTransfer in FT are faster than Create and Transfer respectively, for both FT and NFT. This is due to the fact that the tokens with associated endorsement policies already exist in case of ReCreate and Retransfer operations of FT as discussed in Section 4.2. Thus, neither of the implementations, FT or NFT have a complete advantage over the other in all respects and the choice of implementation will depend on the specific use case.

4.4 Limitations and Future Work

In any transaction system, prevention of double spending is an important consideration. Hyperledger Fabric implements Multiversion concurrency control (MVCC) in order to prevent double spending. This means that when multiple operations try to update the same key at the same time, all but one update will fail. FT tokens function like accounts where each client has at most six accounts in our implementation. For instance if two sellers attempt to transfer value to the same token belonging to a buyer, only one of those transfers will complete successfully. The NFT implementation when creating tokens uses the transaction ID and creates new NFT each time with no duplicated keys. Similarly, transfers also create new NFT for the recipient each time, thus preventing contention. Contention is a consideration whenever two processes seek to update a common resource and thus inherent in any account based implementation.

This limitation can be addressed by queuing of transactions. For EUnit and EVUnit, both the buyer and seller are clients of the Transaction Platform. The Transaction Platform application could thus queue the sending of transactions for endorsement and ordering and postpone sending of other transactions that update the same key until the first one has completed or failed. For InUnit, GaUnit, StUnit and EStUnit, the create operation originates from the Power Company and Storage Provider respectively. These could thus be queued by their respective applications.

Alternatively, as tokens are created for each client, for each token type, newer types of tokens or extra clients can be added to create more tokens that can then be simultaneously updated. For tokens that receive a high number of transactions, a running total approach

can be considered where several FT of the same token type are created, that are consolidated periodically. The bidding operation on InUnit, GaUnit, StUnit and EStUnit is invoked by clients of the Transaction Platform which can be queued by its application. Similarly, queuing of Transfers can be handled by the Power Company or the Storage Provider as the case may be and queuing of Redeems can be handled by the Transaction Platform.

Thakkar et al. [39] proposed the use of a dependency graph for transactions in order to increase parallel execution in blockchain transactions. Li et al. [40] presented priority based queuing model to reduce the waiting time of blockchain transactions. Investigation of application based contention management through queuing or other methods can be considered for a future work.

5 Related Works

Blockchain continues to gain considerable research interest, as the number of publications in the field of blockchain and its applications show a clear upward trend [41]. Features of blockchain such as decentralization, immutability, provenance tracing and self enforcing smart contracts make blockchain suitable to a variety of applications involving information and value exchange, transparency, access control and encoding of business logic for automatic execution. Blockchain was developed as an enabling technology for Bitcoin, a cryptocurrency and thus has several applications in the banking and financial sectors [42]. Blockchain implementations can encompass the business network and provide provenance tracing and can thus facilitate invoice reconciliation and dispute resolution [43]. The immutability of blockchain can increase trust and transparency in transactions and smart contracts can be used to automate transaction flows and processes.

Health care data management is another application area for blockchain. Hasselgren et al. [44] identified access control, data provenance and data integrity as crucial in order to maintain the patient's privacy during data exchanges between institutions in the healthcare ecosystem. Jiang et al. [45] proposed BlocHIE, a blockchain based platform for privacy and authenticability preserving exchange

of medical data. Their solution used two loosely coupled blockchains for storing two different kinds of medical data and proposed two transaction packing algorithms for block creation in order to enhance throughput and fairness. Zhuang et al. [46] proposed the integration of blockchain in order to improve the workflow of health information exchange. Access control was implemented using smart contracts in their solution, which helped them address privacy and data integrity concerns and provide permitted clinicians access to records across multiple medical facilities. Additionally, in light of the global Coronavirus pandemic, blockchain has also been proposed as a solution to address issues such as robust privacy management [47] and provenance based supply chain management [48] for vaccines and contact tracing of affected patients.

Blockchain can also be integrated into industrial internet of things (IIoT) applications due to many of the same reasons that make it a good fit for healthcare and financial domains. Wang et al. [49] analysed the security risks associated with data storage in the IIoT and proposed the use of blockchain in order ensure secure data storage. Wu et al. [50] proposed the integration of blockchain into the supply chain management workflow and implemented a proof of concept on the Hyperledger Fabric for a food traceability system. Chen et al. [51] proposed and implemented a blockchain based access control framework for IoT systems. Jiang et al. [52] proposed the use of blockchain for IIoT data management and presented Fair-Pack, a transaction packing algorithm which succeeded in improving the average response time and fairness of the blockchain as compared to existing algorithms. Bordel et al. [53] presented a theoretical framework to investigate the applicability of blockchain for implementing a privacy and trust preserving solution for storage of data generated by Internet of Things implementations. Dai et al. [54] proposed the integration of blockchain technology to create a privacy preserving platform for Internet of Things. Blockchain mining tasks were delegated to edge nodes through a process of offloading and caching in order to maximize profit and reduce time based on game theory and auction theory. Their approach was shown to perform better than both, a centralized mode as well as a completely decentralized mode where all mining is done by edge nodes, on both profit maximization

and time reduction.

Blockchain applications to energy grids have also been extensively studied. Andoni et al. [55] identified that blockchain can add value to the energy grids in the broad areas of billing, sales and marketing, transactions, automation in decentralized applications, smart grid integration, secure information exchange, grid stabilization through usage flexibility, privacy and security, sharing of common resources, competition and transparency. Blockchain is intuitively suited for implementing peer to peer decentralized transaction systems. Li et al. [56] proposed a blockchain based energy trading system that used a credit-based payment scheme for faster transaction confirmation. They presented an optimal pricing strategy based on Stackelberg game for credit-based loans. Additionally, they evaluated their proposal in terms of security and performance to show the efficiency of their solution. Gai et al. [57] proposed a noise-based privacy preserving energy transaction approach by using account mapping to hide user information like location and energy usage. The presented algorithm for noise creation showed an improvement over existing differential privacy approaches in masking for privacy. Aitzhan et al. [58] also focused on implementing transaction security in decentralized smart grid trading. They implemented a token based transaction system to enable users to anonymously negotiate and perform transactions. Paudel et al. [59] proposed a model with competitions between buyers modelled using evolutionary game theory and competition among sellers modelled as a non-cooperative game. The interactions between buyers and sellers were modelled as a M-leader and N-follower Stackelberg game. The evaluation of their approach using simulation showed the convergence of the model and significant financial benefits to the community.

Blockchain has also been used for implementing usage flexibility programs for grid stability. Pop et al. [60] investigated the use of blockchain to store the consumption and production data collected from smart meters. Smart contracts were used to define the usage flexibility expected from each prosumer, with the corresponding incentives and penalties and the rules for maintaining grid stability. Their evaluation of the proof of concept showed that their approach followed the demand signal with high accuracy and reduced the

amount of energy flexibility needed. Jindal et al. [61] focused on the security aspects of implementing a demand response mechanism by selecting miner nodes to validate the blocks of energy transactions. The results show that the approach has a low communication and computation overhead. Noor et al. [62] proposed a game theoretic approach to reduce the Peak-to-Average ratio and smooth the load profile. They evaluated their approach using a case study using synthetic data of 15 consumers with multiple appliances and storage capacity. Silvestre et al. [63] proposed using blockchain to record the production and consumption data, calculated the baseline and expected usage flexibility and evaluated their implementation to show the efficacy of their approach.

Tokenization in blockchain is an important topic of investigation as assets on the blockchain are represented in the form of tokens in order to facilitate transactions. Chirtoaca et al. [64] reviewed the applicable metrics for extensibility and reusability of NFT on the Ethereum blockchain and identified the most insightful metrics for these features. Borkowski et al. [65] proposed DeXTT, a protocol for blockchain interoperability for token transactions and showed the logarithmic scalability of their solution with respect to the number of participating nodes. Barreiro-Gomez et. al [66] presented a study of blockchain tokens based on mean-field-type game theory in order to establish a relationship between network characteristics, count of token holders, token price and token supply. Based on their findings, they proposed that the number of tokens in circulation be adjusted in order to capture the risk-awareness and self-regulatory behavior in blockchain economics. Bal et al. [67] proposed NFTTracer, a framework for tracking NFT through modifications. They presented their architecture and algorithms and built a proof of concept but did not present a quantitative analysis of the efficacy of their approach.

Devine et al. [68] proposed a mechanism for renewable energy providers to sell customers future power output in form of NFT on the blockchain. They proposed two ways of structuring these power delivery instruments and evaluated their proposal using a notional market simulation. Regner et al. [69] proposed and described the use of NFT for an event ticketing application. However, a quantitative evaluation of their proposal was not included in their work.

Cioara et al. [70] presented an architecture of a blockchain-based smart grid platform and described challenges in implementation of future grid management scenarios such as energy trading, energy flexibility management and virtual power plants. Another work by some of the same authors [30] proposed a blockchain based energy market and described the theoretical framework for implementing the envisioned energy market for NFT assets with features such as registration, bids automation and offers matching.

However, to our knowledge this is the first work that implements a unified energy transaction system in FT as well as NFT versions and draws a comparison between the two implementations in terms of design, algorithmic complexity, limitations and performance.

6 Conclusions

In this study we presented a unified blockchain-based system for energy asset transactions among prosumers, EVs, Power Companies and Storage Providers. We implemented and performance tested this system in two versions, one with the energy assets as fungible tokens (FT) and another with non fungible tokens (NFT). We defined operations Create, Bid, Transfer and Redeem for NFT and FT. Additionally, we defined operations ReCreate and ReTransfer for FT, as outlined by the algorithms and token lifecycle presented in Section 3. Based on our analysis, we have the following conclusions:

(1) The time and space complexities for Create, Bid, Transfer, Redeem and Read algorithms for NFT are both O(1). The time and space complexities for BulkRead for NFT are both O(n).

(2) The time and space complexities for Create, ReCreate, Bid, Redeem and Read algorithms for FT are both O(1), while those for Transfer and ReTransfer algorithms for FT are both O(n).

(3) Increasing the permitted queue of unfinished transactions increases the request send rate for all operations. Due to the increase in request send rate, the throughput as well as latency increases for all operations.

(4) Increasing the transaction complexity i.e., the number of writes per operation decreases the send rate and throughput achieved for the same queue length and comparable latency.

(5) The performance of operations for FT and NFT is similar for Create, Bid, Transfer and Redeem. However, FT operations ReCreate and Retransfer are faster than Create and Transfer for both NFT and FT due to lower number of write operations.

(6) For the NFT implementation, Bid and Redeem operations are the fastest, followed by Create and then by Transfer. In the FT implementation, the fastest operations are Bid, Redeem and ReCreate, followed by Create and ReTransfer, followed by Transfer operation.

(7) The FT implementation stores the total count of energy assets of a particular type in a single token, while NFT can have multiple tokens with energy assets of the same type. So, performance of retrieving all the energy assets of a particular type was observed to be vastly faster for FT (845 TPS, sub second latency) than for NFT (13 TPS, Latency over 18 s).

(8) The NFT implementation avoids contention by creating new tokens with distinct keys whenever Create and Transfer operations are called. Moreover, as two buyers cannot bid on the same token by design, contention is avoided in the NFT implementation. FT tokens function like accounts so, contention is a consideration when two operations try to update the same account.

The implementation and results from the performance testing of the presented energy transaction system with fungible and non fungible tokens provide a proof of concept and show the applicability of blockchain for transacting energy assets in a community based energy infrastructure. Both implementations have comparable performance for all major operations. However, querying for all energy assets owned by a client is a bottleneck for the NFT implementation and could be addressed by moving this operation off-chain. Contention between operations trying to update the same key is a limitation

for FT and could be addressed by application based queuing of transactions based on dependency.

No absolute performance related reasons for choosing one implementation over the other were found, and the choice of implementation will thus depend on the specific use case.

References

[1] Xiaolei Yang, Lingyun He, Yufei Xia, and Yufeng Chen. "Effect of government subsidies on renewable energy investments: The threshold effect." In: *Energy Policy* 132 (2019), pp. 156–166.

[2] Weihua Su, Mengling Liu, Shouzhen Zeng, Dalia Štreimikienė, Tomas Baležentis, and Ilona Ališauskaitė-Šeškienė. "Valuating renewable microgeneration technologies in Lithuanian households: A study on willingness to pay." In: *Journal of Cleaner Production* 191 (2018), pp. 318–329.

[3] Axel Gautier, Julien Jacqmin, and Jean-Christophe Poudou. "The prosumers and the grid." In: *Journal of Regulatory Economics* 53.1 (2018), pp. 100–126.

[4] Allison Lantero. "How microgrids work." In: *US Department of Energy* 17 (2014).

[5] Chenghua Zhang, Jianzhong Wu, Chao Long, and Meng Cheng. "Review of existing peer-to-peer energy trading projects." In: *Energy Procedia* 105 (2017), pp. 2563–2568.

[6] Yifei Wei, Yu Gong, Qiao Li, Mei Song, and Xiaojun Wang. "Energy Efficient Resource Allocation Approach for Renewable Energy Powered Heterogeneous Cellular Networks." In: *CMC-COMPUTERS MATERIALS & CONTINUA* 64.1 (2020), pp. 501–514.

[7] Chenghua Zhang, Jianzhong Wu, Yue Zhou, Meng Cheng, and Chao Long. "Peer-to-Peer energy trading in a Microgrid." In: *Applied Energy* 220 (2018), pp. 1–12.

[8] Forbes. *The Rising Popularity of Energy Storage as a Service*. https://www.forbes.com/sites/pikeresearch/2019/12/06/the-rising-popularity-of-energy-storage-as-a-service/?sh=223abe19a3a7. Accessed: 2021-04-23. 2021.

[9] Jura Arkhangelski, Pierluigi Siano, Abdou-Tankari Mahamadou, and Gilles Lefebvre. "Evaluating the economic benefits of a smart-community microgrid with centralized electrical storage and photovoltaic systems." In: *Energies* 13.7 (2020), p. 1764.

[10] Khizir Mahmud, M Jahangir Hossain, and Graham E Town. "Peak-load reduction by coordinated response of photovoltaics, battery storage, and electric vehicles." In: *IEEE Access* 6 (2018), pp. 29353–29365.

[11] Moslem Uddin, Mohd Fakhizan Romlie, Mohd Faris Abdullah, Syahirah Abd Halim, Tan Chia Kwang, et al. "A review on peak load shaving strategies." In: *Renewable and Sustainable Energy Reviews* 82 (2018), pp. 3323–3332.

[12] Wujing Huang, Ning Zhang, Chongqing Kang, Mingxuan Li, and Molin Huo. "From demand response to integrated demand response: Review and prospect of research and application." In: *Protection and Control of Modern Power Systems* 4.1 (2019), pp. 1–13.

[13] Kaveh Paridari, Alessandra Parisio, Henrik Sandberg, and Karl Henrik Johansson. "Demand response for aggregated residential consumers with energy storage sharing." In: *2015 54th IEEE conference on decision and control (CDC)*. IEEE. 2015, pp. 2024–2030.

[14] Tarek AlSkaif, Ioannis Lampropoulos, Machteld Van Den Broek, and Wilfried Van Sark. "Gamification-based framework for engagement of residential customers in energy applications." In: *Energy Research & Social Science* 44 (2018), pp. 187–195.

[15] Satoshi Nakamoto et al. "Bitcoin: A peer-to-peer electronic cash system." In: (2008).

[16] David Vangulick, Bertrand Cornélusse, and Damien Ernst. "Blockchain for peer-to-peer energy exchanges: design and recommendations." In: *2018 Power Systems Computation Conference (PSCC)*. IEEE. 2018, pp. 1–7.

[17] Naiyu Wang, Xiao Zhou, Xin Lu, Zhitao Guan, Longfei Wu, Xiaojiang Du, and Mohsen Guizani. "When energy trading meets blockchain in electrical power system: The state of the art." In: *Applied Sciences* 9.8 (2019), p. 1561.

[18] Subhasis Thakur and John G Breslin. "Peer to peer energy trade among microgrids using blockchain based distributed coalition formation method." In: *Technology and Economics of Smart Grids and Sustainable Energy* 3.1 (2018), pp. 1–17.

[19] Ayman Esmat, Martijn de Vos, Yashar Ghiassi-Farrokhfal, Peter Palensky, and Dick Epema. "A novel decentralized platform for peer-to-peer energy trading market with blockchain technology." In: *Applied Energy* 282 (2021), p. 116123.

[20] Vitalik Buterin et al. "A next-generation smart contract and decentralized application platform." In: *white paper* (2014).

[21] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. "Hyperledger fabric: a distributed operating system for permissioned blockchains." In: *Proceedings of the thirteenth EuroSys conference*. 2018, pp. 1–15.

[22] Christian Bührer, Ivo Hubli, and Eliane Marti. "The regulatory burden in the Swiss wealth management industry." In: *Financial Markets and Portfolio Management* 19.1 (2005), pp. 99–108.

[23] Golang. *The Go programming language*. `https://golang.org/`. Accessed: 2021-02-17. 2021.

[24] Ahmet Önder Gür, Şafak Öksüzer, and Enis Karaarslan. "Blockchain based metering and billing system proposal with privacy protection for the electric network." In: *2019 7th International Istanbul Smart Grids and Cities Congress and Fair (ICSG)*. Ieee. 2019, pp. 204–208.

[25] Zheng Che, Yu Wang, Juanjuan Zhao, Yan Qiang, Yue Ma, and Jihua Liu. "A distributed energy trading authentication mechanism based on a consortium blockchain." In: *Energies* 12.15 (2019), p. 2878.

[26] Nikita Karandikar, Antorweep Chakravorty, and Chunming Rong. "Transactive Energy on Hyperledger Fabric." In: *2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*. IEEE. 2019, pp. 539–546.

[27] Nikita Karandikar, Antorweep Chakravorty, and Chunming Rong. "RenewLedger: Renewable energy management powered by Hyperledger Fabric." In: *2020 IEEE Symposium on Computers and Communications (ISCC)*. IEEE. 2020, pp. 1–6.

[28] Statnett. *Elcertificates and guarantees of origin*. https://www.statnett.no/en/for-stakeholders-in-the-power-industry/system-operation/the-power-market/elcertificates-and-guarantees-of-origin/. Accessed: 2021-04-23. 2021.

[29] Yeray Mezquita, Amin Shokri Gazafroudi, JM Corchado, Miadreza Shafie-Khah, Hannu Laaksonen, and Aida Kamišalić. "Multi-agent architecture for peer-to-peer electricity trading based on blockchain technology." In: *2019 XXVII International Conference on Information, Communication and Automation Technologies (ICAT)*. IEEE. 2019, pp. 1–6.

[30] Claudia Pop, Marcel Antal, Tudor Cioara, and Ionut Anghel. "TRADING ENERGY AS A DIGITAL ASSET: A BLOCKCHAIN-BASED ENERGY MARKET." In: *Cryptocurrencies and Blockchain Technology Applications* (2020), pp. 261–279.

[31] Ausgrid. *Community Batteries*. https://www.ausgrid.com.au/In-your-community/Community-Batteries. Accessed: 2021-02-17. 2021.

[32] Parth Thakkar, Senthil Nathan, and Balaji Viswanathan. "Performance benchmarking and optimizing hyperledger fabric blockchain platform." In: *2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE. 2018, pp. 264–276.

[33] Diego Ongaro and John Ousterhout. "In search of an understandable consensus algorithm." In: *2014 USENIX Annual Technical Conference (USENIXATC 14)* (2014), pp. 305–319.

[34] Hyperledger Fabric. *The Ordering Service*. https://hyperledger-fabric.readthedocs.io/en/release-2.3/orderer. Accessed: 2021-04-23. 2021.

[35] Saurabh Singh, Young-Sik Jeong, and Jong Hyuk Park. "A survey on cloud computing security: Issues, threats, and solutions." In: *Journal of Network and Computer Applications* 75 (2016), pp. 200–222.

[36] Hyperledger Performance and Scale Working Group. *Hyperledger Blockchain Performance Metrics White Paper*. https://www.hyperledger.org/learn. Accessed: 2021-04-23. 2021.

[37] Arati Baliga, Nitesh Solanki, Shubham Verekar, Amol Pednekar, Pandurang Kamat, and Siddhartha Chatterjee. "Performance characterization of hyperledger fabric." In: *2018 Crypto Valley conference on blockchain technology (CVCBT)*. IEEE. 2018, pp. 65–74.

[38] Fridtjof Nystrøm. "Network Performance in Hyperledger Fabric-Investigating the network resource consumption of transactions in a Distributed Ledger Technology system." MA thesis. 2019.

[39] Parth Thakkar and Senthil Nathan. "Scaling Hyperledger Fabric Using Pipelined Execution and Sparse Peers." In: *arXiv preprint arXiv:2003.05113* (2020).

[40] Tianmu Li, Yongjun Ren, and Jinyue Xia. "Blockchain Queuing Model with Non-Preemptive Limited-Priority." In: *INTELLIGENT AUTOMATION AND SOFT COMPUTING* 26.5 (2020), pp. 1111–1122.

[41] Sherali Zeadally and Jacques Bou Abdo. "Blockchain: Trends and future opportunities." In: *Internet Technology Letters* 2.6 (2019), e130.

[42] Alex Tapscott and Don Tapscott. "How blockchain is changing finance." In: *Harvard Business Review* 1.9 (2017), pp. 2–5.

[43] Krishnasuri Narayanam, Seep Goel, Abhishek Singh, Yedendra Shrinivasan, Shreya Chakraborty, Parameswaran Selvam, Vishnu Choudhary, and Mudit Verma. "Blockchain Based e-Invoicing Platform for Global Trade." In: *2020 IEEE International Conference on Blockchain (Blockchain)*. IEEE. 2020, pp. 385–392.

[44] Anton Hasselgren, Katina Kralevska, Danilo Gligoroski, Sindre A Pedersen, and Arild Faxvaag. "Blockchain in healthcare and health sciences—A scoping review." In: *International Journal of Medical Informatics* 134 (2020), p. 104040.

[45] Shan Jiang, Jiannong Cao, Hanqing Wu, Yanni Yang, Mingyu Ma, and Jianfei He. "Blochie: a blockchain-based platform for healthcare information exchange." In: *2018 ieee international conference on smart computing (smartcomp)*. IEEE. 2018, pp. 49–56.

[46] Yan Zhuang, Lincoln R Sheets, Yin-Wu Chen, Zon-Yin Shae, Jeffrey JP Tsai, and Chi-Ren Shyu. "A patient-centric health information exchange framework using blockchain technology." In: *IEEE journal of biomedical and health informatics* 24.8 (2020), pp. 2169–2176.

[47] Laura Ricci, Damiano Di Francesco Maesa, Alfredo Favenza, and Enrico Ferro. "Blockchains for covid-19 contact tracing and vaccine support: a systematic review." In: *IEEE Access* 9 (2021), pp. 37936–37950.

[48] Claudia Daniela Antal, Tudor Cioara, Marcel Antal, and Ionut Anghel. "Blockchain platform for COVID-19 vaccine supply management." In: *arXiv preprint arXiv:2101.00983* (2021).

[49] Jin Wang, Wencheng Chen, Lei Wang, Yongjun Ren, and R Simon Sherratt. "Blockchain-based data storage mechanism for industrial internet of things." In: *Intelligent Automation and Soft Computing* 26.5 (2020), pp. 1157–1172.

[50] Hanqing Wu, Jiannong Cao, Yanni Yang, Cheung Leong Tung, Shan Jiang, Bin Tang, Yang Liu, Xiaoqing Wang, and Yuming Deng. "Data management in supply chain using blockchain:

Challenges and a case study." In: *2019 28th International Conference on Computer Communication and Networks (ICCCN)*. IEEE. 2019, pp. 1–8.

[51] Hao Chen, Wunan Wan, Jinyue Xia, Shibin Zhang, Jinquan Zhang, Xizi Peng, and Xingjie Fan. "Task-Attribute-Based Access Control Scheme for IoT via Blockchain." In: *CMC-COMPUTERS MATERIALS & CONTINUA* 65.3 (2020), pp. 2441–2453.

[52] Shan Jiang, Jiannong Cao, Hanqing Wu, and Yanni Yang. "Fairness-based Packing of Industrial IoT Data in Permissioned Blockchains." In: *IEEE Transactions on Industrial Informatics* (2020).

[53] Borja Bordel, Ramon Alcarria, Diego Martin, and Alvaro Sanchez-Picot. "Trust provision in the internet of things using transversal blockchain networks." In: *INTELLIGENT AUTOMATION AND SOFT COMPUTING* 25.1 (2019), pp. 155–170.

[54] Yao Dai. "Edge computing-based tasks offloading and block caching for mobile blockchain." In: *Computers, Materials & Continua* 62.2 (2020), pp. 905–915.

[55] Merlinda Andoni, Valentin Robu, David Flynn, Simone Abram, Dale Geach, David Jenkins, Peter McCallum, and Andrew Peacock. "Blockchain technology in the energy sector: A systematic review of challenges and opportunities." In: *Renewable and Sustainable Energy Reviews* 100 (2019), pp. 143–174.

[56] Zhetao Li, Jiawen Kang, Rong Yu, Dongdong Ye, Qingyong Deng, and Yan Zhang. "Consortium blockchain for secure energy trading in industrial internet of things." In: *IEEE transactions on industrial informatics* 14.8 (2017), pp. 3690–3700.

[57] Keke Gai, Yulu Wu, Liehuang Zhu, Meikang Qiu, and Meng Shen. "Privacy-preserving energy trading using consortium blockchain in smart grid." In: *IEEE Transactions on Industrial Informatics* 15.6 (2019), pp. 3548–3558.

[58] Nurzhan Zhumabekuly Aitzhan and Davor Svetinovic. "Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams." In: *IEEE Transactions on Dependable and Secure Computing* 15.5 (2016), pp. 840–852.

[59] Amrit Paudel, Kalpesh Chaudhari, Chao Long, and Hoay Beng Gooi. "Peer-to-peer energy trading in a prosumer-based community microgrid: A game-theoretic model." In: *IEEE Transactions on Industrial Electronics* 66.8 (2018), pp. 6087–6097.

[60] Claudia Pop, Tudor Cioara, Marcel Antal, Ionut Anghel, Ioan Salomie, and Massimo Bertoncini. "Blockchain based decentralized management of demand response programs in smart energy grids." In: *Sensors* 18.1 (2018), p. 162.

[61] Anish Jindal, Gagangeet Singh Aujla, Neeraj Kumar, and Massimo Villari. "GUARDIAN: Blockchain-based secure demand response management in smart grid system." In: *IEEE Transactions on Services Computing* 13.4 (2019), pp. 613–624.

[62] Sana Noor, Wentao Yang, Miao Guo, Koen H van Dam, and Xiaonan Wang. "Energy Demand Side Management within micro-grid networks enhanced by blockchain." In: *Applied energy* 228 (2018), pp. 1385–1398.

[63] M. L. Di Silvestre, P. Gallo, E. R. Sanseverino, G. Sciumè, and G. Zizzo. "Aggregation and Remuneration in Demand Response With a Blockchain-Based Framework." In: *IEEE Transactions on Industry Applications* 56.4 (2020), pp. 4248–4257. DOI: 10.1109/TIA.2020.2992958.

[64] Dan Chirtoaca, Joshua Ellul, and George Azzopardi. "A framework for creating deployable smart contracts for non-fungible tokens on the Ethereum blockchain." In: *2020 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*. IEEE. 2020, pp. 100–105.

[65] Michael Borkowski, Marten Sigwart, Philipp Frauenthaler, Taneli Hukkinen, and Stefan Schulte. "DeXTT: deterministic cross-blockchain token transfers." In: *IEEE Access* 7 (2019), pp. 111030–111042.

[66] Julian Barreiro-Gomez and Hamidou Tembine. "Blockchain token economics: A mean-field-type game perspective." In: *IEEE Access* 7 (2019), pp. 64603–64613.

[67] Mustafa Bal and Caitlin Ner. "NFTracer: a Non-Fungible token tracking proof-of-concept using Hyperledger Fabric." In: *arXiv preprint arXiv:1905.04795* (2019).

[68] Mel T Devine, Marianna Russo, and Paul Cuffe. "Blockchain electricity trading using tokenised power delivery contracts. ESRI Working Paper No. 649 December 2019." In: (2019).

[69] Ferdinand Regner, Nils Urbach, and André Schweizer. "NFTs in Practice–Non-Fungible Tokens as Core Component of a Blockchain-based Event Ticketing Application." In: (2019).

[70] Tudor Cioara, Claudia Pop, Razvan Zanc, Ionut Anghel, Marcel Antal, and Ioan Salomie. "Smart Grid Management using Blockchain: Future Scenarios and Challenges." In: *arXiv preprint arXiv:2012.06256* (2020).