Dynamic Verifiable Frequency Based Billing Using Blockchain

Project Report
for
Summer Internship
C3i Center, IIT Kanpur
May 2019 - July 2019

Authors

Md. Azmal

Institute of Technical Education and Research, Bhubaneshwar

Bachelors of Technology
Department of Computer Science & Engineering

Shubham Singh

Indian Institute of Technology, Kanpur

Bachelors of Science
Department of Mathematics

Under the guidance of
Prof. Sandeep Kumar Shukla
Department of Computer Science & Engineering
Indian Institute of Technology, Kanpur
Uttar Pradesh, India

CERTIFICATE

This is to certify that the project **Dynamic Verifiable Frequency Based Billing Using Blockchain** submitted by Md. Azmal and Shubham Singh as a part of Undergraduate Research and Graduate Excellence 2019 offered by the Indian Institute of Technology, Kanpur is a bonafide record of the work done by them under my guidance and supervision at the Indian Institute of Technology, Kanpur from 13th May 2019 to 12th July 2019.

Prof. Sandeep Kumar Shukla
Head of Department (H.O.D)
Department of Computer Science & Engineering
IIT, Kanpur

ACKNOWLEDGEMENTS

- We would like to express our gratitude towards **Prof. Sandeep Kumar Shukla** for giving us this opportunity and our mentor **Mr. Ras Dwivedi** for providing guidance and support throughout the course of project.
- We would like to thank Nilesh Vasita Final year, MTI-Dual for his help and constructive criticism during our project work.
- We would like to thank all our friends who have provided insights and ideas for the project.
- Finally, we would like to express our gratitude to our parents for their constant support and motivation.

ABSTRACT

"The grid," refers to the electric grid, a network of transmission lines, substations, transformers and more that deliver electricity from the power plant to your home or business. Although the electric grid is considered an engineering marvel, we are stretching its patchwork nature to its capacity. The digital technology of future will allow for two-way communication between the utility and its customers, and the sensing along the transmission lines is what makes the grid smart. It helps the consumers to become producers as it is able to channel electricity produced by households mostly via solar energy to fill the power gap. The traditional billing done on the basis of units consumed is however not the most intuitive. For this a new concept of billing arises known as frequency based billing, where cost is lower when the frequency of grid is high and vice versa. There is a lot of settlement issue when it comes to agreement upon the price at which the distributer should buy electricity from grid, and also the consumer has to be sure that the bill recieved is genuine. Blockchain technology is used to fulfil this trust gap. Each household have a smart meter that is able to fetch the frequency of grid for the current block and charged the consumer accordingly. This transaction record is maintained in the private Blockchain. The transaction can be two way as well, in which the household can provide their produced electricity to the grid in low frequency scenario. The use of frequency based billing will also promote power saving as the houeseholds will be inclined to use their solar power or battery inverter when the grid is running on low frequency to avoid high charge upon electricity. Households joining hands with grid to sell the energy produced by them not only reduces the power gap faced by our country but also promote the use of modern method of eco-friendly electricity generation like solar energy.

Contents

List of Figures

Chapter 1

Introduction

A Blockchain[1] is a growing list of records, called blocks, which are linked using cryptography. Each block contains a cryptographic hash of the previous block a timestamp, and transaction data.

Blockchain technology functions as a public record of transactions between users agreeing upon a consenus. Although it was at first seen as part of the mechanism that makes cryptocurrencies work, it could be applied in a great many other ways: a few of which provide interesting possibilities. It will have a profound influence on the many ways that people and markets work and interact in the foreseeable future; and because of this there are likely to be profound implications for the nature of capitalism. Although this reality may be somewhat remote, we could still enrich our knowledge-base and be ready for the future by delving into a few of the implications now.

One of such used cases of Blockchain can be explored in power grids. an intelligent grid offering two-way communication between the consumers and the grid. This enables the gird to become a distributed generation system unlike the traditional ones that are centralized generation system by default. As a result more customer choices are provided where one can decide whether to take power from grid(buying) or provide power to the grid(selling), which can maitain the grid's frequency[2] and save it from "take down"[2].

1.1 Blockchain

The blockchain[1] is a distributed database of records of all transactions or digital event that have been executed and shared among participating parties often termed as nodes. Each transaction verified by the majority of participants of the system through a shared consenus which is mostly Byzantine Fault Tolerance[3].One of the famous use of Blockchain is Bitcoin[1]. The bitcoin is a cryptocurrency and is used to exchange digital assets online. Bitcoin uses cryptographic proof known as proof of work(PoW)[3] instead of third-party trust for two parties to execute transactions over the internet. Each transaction protects through digital signature[1].

There is no Central Server or System which keeps the data of Blockchain. The data is distributed over Millions of Computers around the world which are connected with the Blockchain. This system allows Notarization of Data as it is present on every Node and is publicly verifiable. Node gets connected with Blockchain using the client. Client helps in validating and propagates transaction on to the Blockchain. When a computer connects to the Blockchain, a copy of the Blockchain data gets downloaded into the system and the node comes in sync with the latest block of data on Blockchain. The Node connected to the Blockchain which helps in the execution of a Transaction in return for an incentive is called Miner.

On the basis of access control, storage, consenus blockchains are broadly classified into two categories.

1.2 Public Blockchain

Public blockchains are permissionless blockchains where anyone can join the network and read and write to the ledger. Such blockchains allow users from anywhere in the world to interact with the blockchain and submit or read transactions as far as they are connected with the blockchain network. In a permissionless blockchain, any user can develop and add smart contracts to blockchain without any intervention forced by developers. Permissionless blockchains bring complete decentralization , which means that there exists no central authority to edit the ledger state or make any kind of modifications to the network protocols. This makes the system robust against a single point of failure with distributed trust. Bitcoin blockchain is one of the examples of public blockchain.

The transaction flow of public blockchain is as follows:

- Transaction broadcasted to all nodes.
- Node collects new transaction and finds Proof of Work for its block which is also known as mining.
- After finding PoW it broadcasts the block to all nodes.
- Nodes accept the block if all transactions in it are valid.
- After acceptance a new block is created, which is cryptographically linked with previous block using previous block hash.

Public blockchains require incentives[1] to function, which in most casesd a cryptocurrency.

1.3 Private Blockchain

Permissioned blockchains do not allow any user to freely join the network and read or write to the ledger. It maintains an access control mechanism between the list of users connected to the network. Such blockchains are widely adopted by centralized organizations where the company consortiums are likely to securely record transactions, and interchange important information between one another. Such blockchains can be partly decentralized or fully centralized as the users have the right to do negotiation and reach a consensus with desired varying decentralization.

In private blockchains, users' identity is known to everyone but transactions are only visible to those who have appropriate permission. Also, since all the users are not involved in the consensus process, such blockchains have higher throughput compared to public blockchains. Hyperledger[4] is one of such examples of private blockchain. These blockchains are highly modular and flexible and can be customized depending upon the used cases.

Chapter 2

Hyperledger Family

Hyperledger is an open source collaborative effort created to advance cross-industry blockchain technologies. It is a global collaboration, hosted by The Linux Foundation, including leaders in finance, banking, Internet of Things, supply chains, manufacturing and Technology.

Hyperledger launched in 2016 with a technical and organizational governance structure and 30 founding corporate members. Initially, the Hyperledger Technical Steering Committee welcomed two business blockchain framework codebases into incubation: Hyperledger Fabric, a codebase combining work by Digital Asset, libconsensus from Blockstream and OpenBlockchain from IBM; and Hyperledger Sawtooth, developed at Intel's incubation group.

Some of the Hyperledger frameworks are listed below:

- **Hyperledger Fabric** : Intended as a foundation for developing applications or solutions with a modular architecture, Hyperledger Fabric allows components, such as consensus and membership services, to be plug-and-play.
- **Hyperledger Sawtooth** : Hyperledger Sawtooth is a modular platform for building, deploying, and running distributed ledgers. Hyperledger Sawtooth includes a novel consensus algorithm, Proof of Elapsed Time (PoET), which targets large distributed validator populations with minimal resource consumption.
- **Hyperledger Indy** : Hyperledger Indy is a distributed ledger, purpose-built for decentralized identity. It provides tools, libraries, and reusable components for creating and using independent digital identities rooted on blockchains or other distributed ledgers for interoperability.
- **Hyperledger Grid** : Hyperledger Grid is a WebAssembly-based project for building supply chain solutions. It includes a set of libraries, data models, and SDK to accelerate development for supply chain smart contracts and client interfaces.
- **Hyperledger Burrow** : Hyperledger Burrow is a permissionable smart contract machine. The first of its kind when released in December, 2014, Burrow provides a modular blockchain client with a permissioned smart contract interpreter built in part to the specification of the Ethereum Virtual Machine (EVM).

- **Hyperledger Iroha** : Hyperledger Iroha is an easy to use, modular distributed blockchain platform with its own unique consensus and ordering service algorithms, rich role-based permission model and multi-signature support.

2.1 Hyperledger Fabric

Hyperledger Fabric is a blockchain framework implementation and one of the Hyperledger projects hosted by The Linux Foundation. Intended as a foundation for developing applications or solutions with a modular architecture, Hyperledger Fabric allows components, such as consensus and membership services, to be plug-and-play. Hyperledger Fabric leverages container technology to host smart contracts called "chaincode" that comprise the application logic of the system.

Where Hyperledger Fabric breaks from some other blockchain systems is that it is private and permissioned. Rather than an open permissionless system that allows unknown identities to participate in the network (requiring protocols like "proof of work" to validate transactions and secure the network), the members of a Hyperledger Fabric network enroll through a trusted Membership Service Provider (MSP)[4].

Hyperledger Fabric also offers the ability to create channels, allowing a group of participants to create a separate ledger of transactions. This is an especially important option for networks where some participants might be competitors and not want every transaction they make — a special price they're offering to some participants and not others, for example — known to every participant. If two participants form a channel, then those participants — and no others — have copies of the ledger for that channel.

Hyperledger Fabric has a ledger subsystem comprising two components: the world state and the transaction log. Each participant has a copy of the ledger to every Hyperledger Fabric network they belong to. The world state component describes the state of the ledger at a given point in time. It's the database of the ledger. The transaction log component records all transactions which have resulted in the current value of the world state; it's the update history for the world state. The ledger, then, is a combination of the world state database and the transaction log history.

2.2 Hyperledger Fabric Architecture

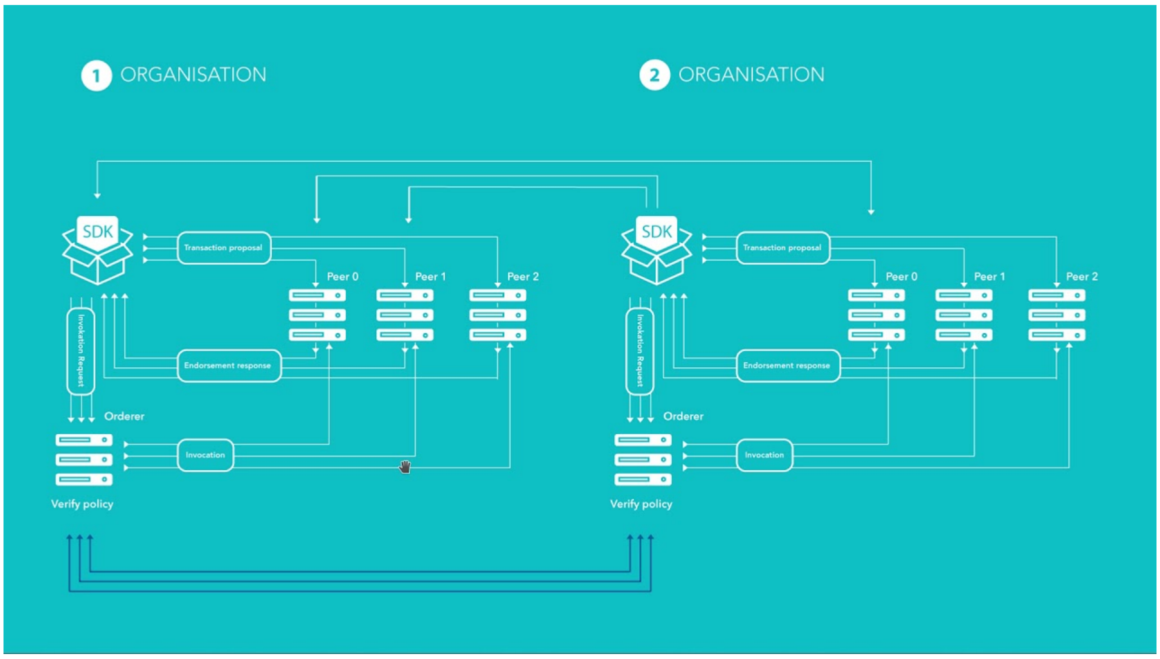


Figure 2.1: simple fabric architecture

Fig:2.1 shows a basic network scenario which will help in understanding the working of the fabric netwrok. This network consists of two Organisations[4] with three peers each named peer0,peer1,peer2. Peers of Organisation 1 are identifiable by Org1.peer#.example.com, similarly peers of Organisation 2 are identifiable as Org2.peer#.example.com . Each organisation has its own SDK[5] to interact with the fabric, and its own Orderer for Ordering service[4]. Both the organisations are on a channel and have their chaincodes installed.

The SDK prepares transaction proposal for peers. This is sent to one or more peers. Every peer have a different physical copy of ledger in the blockchain, they are all synchronized. The peer then accepts the proposal sent from SDK and simulate it, i.e.; execute the transaction with their current version of ledger. The results of each peer is signed cryptographically and sent back as an endorsement response to SDK. SDK then collects the responses and packs them in an invocation request and signs them with all the endorsements and keys, which is then sent to ordering service. The orderer verifies all cryptographic materials and policies. Once all the policies and other informations are verified, an invocation request is to the peers for updating their physical copy of ledger. One key thing to note here is that, even if the transaction is invalid it is still recorded in the blockchain as an invalid transaction but the ledger is not updated. This helps in future reference and inspection.

Chapter 3

Frequency Based Billing

Presently the electricity billing is based upon the units consumed. There is no involvement of frequency of grid or any other factors in calculating the bill. However, this model is going to cause issues when we have grids which allow the consumers to be the producers. The whole power generated or conserved by the households be it in form of solar energy or anything else acts as a decentralized power grid. When the consumers are able to communicate with the grid many factors come into picture one of such factor being the frequency of the grid. All of such factors must be taken into account when calculating the bill[2].

3.1 Implementing frequency based bill generation

In order to have a fully functioning two-way power transfer it is important to inculcate right incentives among the consumers. For an instance, if the grid is running on frequency which is above 50Hz, it is producing more than what is required as a result there is no neccessity for the households to share power to the grid. Similarly, when the grid is running at a lower frequency means it is producing less than the demand, houeseholds must supply their generated power to the grid or lower their consumption to maintain the grids frequency, which ultimately results in maintaining the equilibrium between demand and supply of power.

Frequency plays the most vital role in the above mentioned case, thus having a frequency based billing platform is required to drive the incentive of when to supply power to the grid and when to consume power from the grid. Note that distribution centers have no role when households supply power to grid. When the frequency of the grid is high it is producing more than what is consumed, thus the cost per unit is less when the frequency is high, which motivates consumer to use the power from grid and not from other sources such as solar energy etc, as consuming from the grid will be cheaper which will result in balancing out the production of grid and consumption of households. Also the selling price of power from the household to grid is cheaper at high frequency which creates an incentive of not supplying power to grid as it won't be profitable. When the grid is running at a lower frequency it is evident that it is unable to fulfil the power requirement, thus the cost per unit is more when the frequency is less, and the price of selling power to the grid is high. This creates an incentives for the households to shift using solar or any other source of power other than grid so they can be profitable, also it provides a good opportunity for the households to sell their generated power to the grid for earning more. Consumers are going to be charged depending upon the power consumed on different frequency slots which will dynamically get accumulated. The dynamic billing will give consumers the freedom to clear their dues anytime they want and the consumers will be charged with fine if the accumulated price rises above a certain threshold which may be decided by the government. This bill will be inclusive of the power sold by households to the grid.

3.2 Blockchain based monitoring

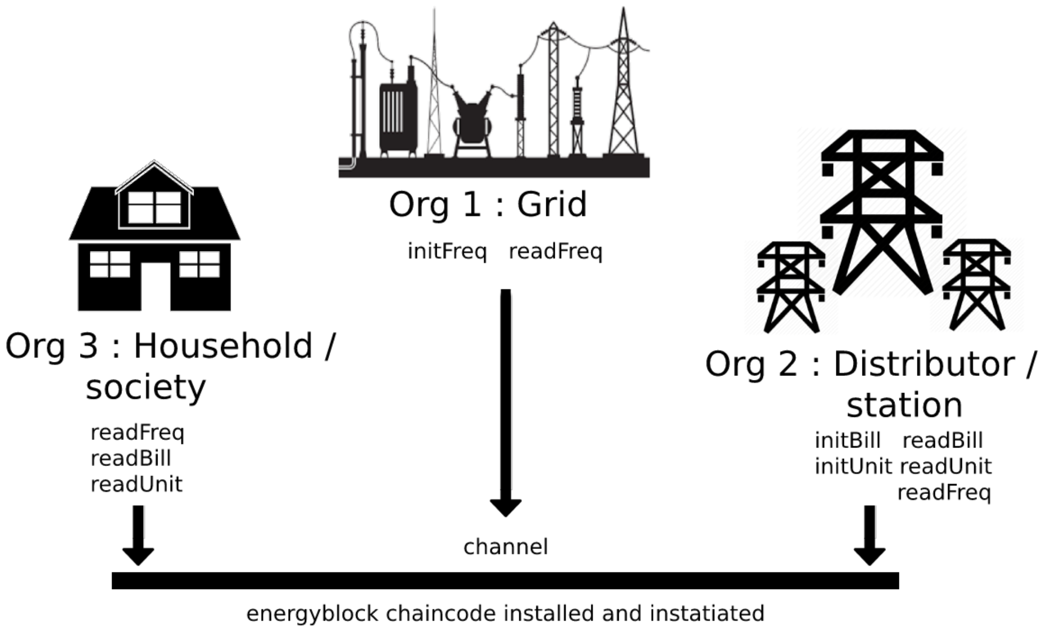
Consumers must be assured that the amount they have been charged with is legitimate. This is where Blockchain comes into play being an immutable ledger it provides complete transperancy and legitimacy even when the players involved do not know each other. A private blockchain is more suitable in our case as it supports higher transactions per second, higher security, as private blokchain do not require mining they are less resource intensive and can be used in IoT platforms as well. For monitoring the frequency and calculating bill a smart meter is required which is capable of interacting the blockchain. In this project we have simulated a smart meter which interacts with the blockchain. We have developed an application on Hyperledger fabric to implement frequency based billing. Hyperledger fabric provides organizational oriended modular framework for blockchain[4]. It provides features for establishing policies, abstraction and ownership among the organisations.

Chapter 4

Energy Blocks : Frequency based billing platform

Energy block is an application that implements dynamic frequency based billing on Blockchain. It is built upon Hyperledger Fabric and provides a REST server along with a web interface for interacting with the fabric. It calculates the electricity based on different frequency and has an option to simulate the transactions as well.

4.1 Architecture



Org 1 : Grid
initFreq readFreq
Org 3 : Household / society
readFreq
readBill
readUnit
Org 2 : Distributor / station
initBill readBill
initUnit readUnit
readFreq
channel
energyblock chaincode installed and instatiated

Figure 4.1: Energy Blocks network architecture

The network consists of three organistations i.e.; Grid, Distributor and Households. Grid is resposnible for producing most of the power that is being consumed. This power then goes to the distributor who then distributes this power to households. On the other hand households can sell their generated power directly to grid without any intermediary.

All the three organisations are connected in a channel. Each organisation has two peers among which one peer is anchor peer. Although one peer could have been sufficient in this case but an extra peer is kept to test if the different physical copies of ledger synchronize or not. The sample network uses solo ordering service which can be changed to kafka in production.

Fig:4.1 the permissions that each organisations have. Grid is resposnible for executing initFreq method which provides the frequency for each slot of time and readFreq to read the frequency from the ledger. Similarly Distributor has permissions to execute initBill and initUnit method that generates the cost of consumption of power in that timeslot and get the units consumed in a particular timeslot respectively. Distributor can also execute readFreq method and readBill and readUnit method to get the bill and units from the ledger respectively.

4.2 Chaincode Methods

Before moving towards discussing what are the chaincode methods for interacting with leadger it is important to get familiar with the following terminologies.

- **Station ID** : Unique identifiaction key for a particular distribution station.
- **Society ID** : Unique identifiaction key for households.
- **Start time** : The start time of timeslot.
- **End time** : End time of a timeslot. The size of each slot is 15 minutes.
- **Average Frequency** : The frequency of the grid.
- **Unit** : Amount of power consumed in KWh.

The above method terms are frequently used as arguments for the chaincode Methods. There are three types of transactions taking place in this blockchain network i.e.; transactions inolving frequency, unit and bill each having init, read and delete methods. Chincode methods are listed below.

- **initFreq :** Parameters[StartTime,EndTime,AvgFreq] This method initialises the given frequency for a particular timeslot and writes it to the ledger.
- **readFreq :** Parameters[StartTime] This method returns the Frequency of a timeslot from the ledger given its start time.
- **deleteFreq :** Parameters[StartTime] This method deletes the frequency of a timeslot given its start time.
- **initUnit :** Parameters[SocietyID,StationID,StartTime,EndTime,Unit] This method writes the units consumed in a timeslot to the grid, provided the start and end time of the timeslot, It also takes StationID and SocietyID to specify who has consumed the power and who has distributed it to them.

- **readUnit :** Parameters[StationID,SocietyID,StartTime] This method returns the units consumed by the household provided its provider and start time of the timeslot.
- **deleteUnit :** Parameters[StationID,SocietyID,StartTime] Delete the unit for a given household provided the start time of timeslot.
- **initBill :** Parameters[SocietyID,StationID,StartTime,EndTime] This method calculates the bill for a given Society/household. It invokes the ledger with the given parameters to get the units consumed and calculates the price based on Deviation Settlement Mechanism. After calculating the bill it writes it to the ledger.
- **readBill :** Parameters[StationID,SocietyID,StartTime] This method reads the bill from the ledger for a particular household provided its StationID and start time of the timeslot.
- **deleteBill :** Parameters[StationID,SocietyID,StartTime] Delete the bill for a given household provided the start time of timeslot.

4.3 Transaction Flow

The goal of the application is to provide users the report of their consumption and shared power over different timeslots, each time slot has a different frequency which is used for calculating the bill for the household in that timestamp. Both consumption and sharing is tracked simultaneously as if the consumer has sold the power it's price will be negative as households are at the producer end, similarly if the household is consuming the power from the grid they are at the reciever end and the price will be positive. Hence, they add up to provide the price that household has to pay on a given timeslot.

Firstly, the SDK sends a transaction proposal to the gird to execute initFreq method, after the method is successfully executed the endorsing peer send the endorsement response to the SDK which is then packed into invocation request to the ordering service. Once the verification is done the orderer invokes the peers to update the ledger and the frequency of a particular timeslot is updated in the ledger. It can be verified by executing readFreq transaction. Once the frequency is available the SDK then sends transaction proposal to the distributor to execute initUnit method to calculate the units consumed by a particular households, then the endorsement follows as above, the orderer invokes the peers to update the ledger with the units of the given timeslot. After the frequency and units consumed in a timeslot are available the SDK then sends a transaction proposal to distributer to execute initBill method to calculate the bill of the given timeslot. The initBill then reads the frequency and units consumed in a particular timeslot by a given households and calculates the bill and returns an endorsement respose to the SDK which again sends the invocation request to the orderer for adding the bill into the ledger. Orderer verifies the signatures and policies and then invokes the peer to update their ledger with the bill.

The above cycle calculates the complete bill for one household in one timeslot. This cycle repeats for all the households in all the available timeslots for the day to calculate the bill per day for each and every household. Each timeslot is about 15 minutes long, thus there are all together $\frac{1440}{15} = 96$ timeslots a day. The application comes with a run simulation method as well to simulate one day by scaling down the timeslot duration to 15 seconds, and execute all the above mentioned methods in sequence to calculate the bill for a day.

Chapter 5

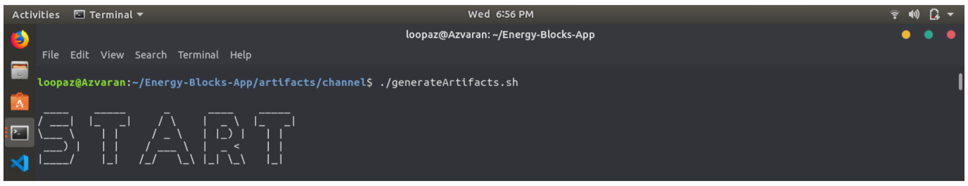
Energy-Blocks Manual

Follow this guide to setup an run the Energy Blocks application. For further details and insights go to **Appendix**.

5.1 Generating cryptographic materials

Before running the application it is important to generate the neccesary crypto files and signature that are required for verification of transactions. Once in the directory of Energy-Blocks-App run the following commands to generate the crypto files as shown in Fig:5.1.

```
$ cd artifacts/channel
$ chmod +x generateArtifacts.sh
$ ./generateArtifacts.sh
```



```
loopaz@Azvaran:~/Energy-Blocks-App/artifacts/channel$ ./generateArtifacts.sh
 ____  _____  _    ____  _____
/ ___||_   _|/ \  |  _ \|_   _|
\___ \  | | / _ \ | |_) | | |
 ___) | | |/ ___ \|  _ <  | |
|____/  |_/_/   \_\_| \_\ |_|
```

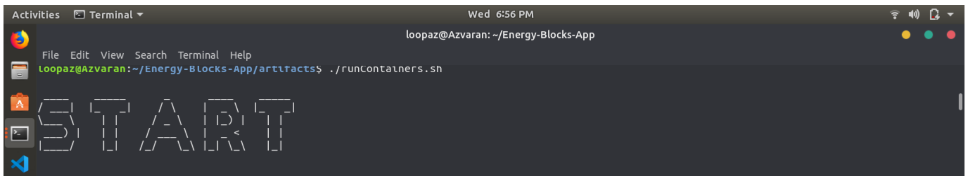
Figure 5.1: Generate artifacts

The above commands make generateArtifacts.sh as an executable file. This script generates the cryptographic materials like channel configurations, genesis block configurations, organisations configurations, generating certificates for organisations and orderers etc. This also modifies the docker-compose.yaml and network-config.yaml in artifacts directory to include latest signatures. Note that generateArtifacts uses cryptogen.yaml and configtx.yaml which are resposnible for creating certificates. Also the fabric tools used for generation of crypto materials and interacting with chaincode must be added to the path variables before running any of shell scripts in Energy-Blocks App's directory. Check the terminal to see if any errors were found or not to ensure that the script ran successfully.

5.2 Starting the network

After running the above command run the following commands to start the network. It should look like Fig:5.2

```
$ cd ..
$ chmod +x runContainers.sh
$ ./runContainers.sh
```


Activities
Terminal
Wed 6:56 PM
loopaz@Azvaran: ~/Energy-Blocks-App
File Edit View Search Terminal Help

```
loopaz@Azvaran:~/Energy-Blocks-App/artifacts$ ./runContainers.sh
 ____  _____  _    ____  _____
/ ___||_   _|/ \  |  _ \|_   _|
\___ \  | | / _ \ | |_) | | |
 ___) | | |/ ___ \|  _ <  | |
|____/  |_/_/   \_\_| \_\ |_|
```

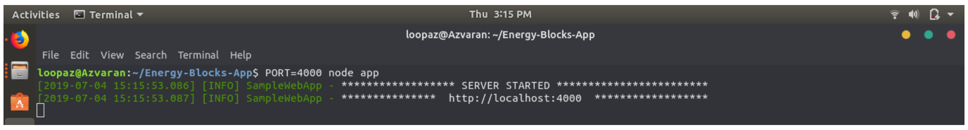
Figure 5.2: Run the containers

This script removes the prerunning docker containers and previous user credentials if any. It then takes the latest docker-compose.yaml file to turn the network up. The network consists of all total 3 organisations with 2 peers each, 3 certificate authourities for each organisation and an ordering service. You can use the following command in a new terminal to see if the containers are up or not.

```
$ docker ps
```

After the above script runs successfully. Run the following command to start the REST Api.

```
$ cd ..
$ npm install
$ PORT=4000 node app
```


Activities
Terminal
Thu 3:15 PM
loopaz@Azvaran: ~/Energy-Blocks-App
File Edit View Search Terminal Help

```
loopaz@Azvaran:~/Energy-Blocks-App$ PORT=4000 node app
[2019-07-04 15:15:53.086] [INFO] SampleWebApp - ***************** SERVER STARTED ***********************
[2019-07-04 15:15:53.087] [INFO] SampleWebApp - **************  http://localhost:4000  *****************
```

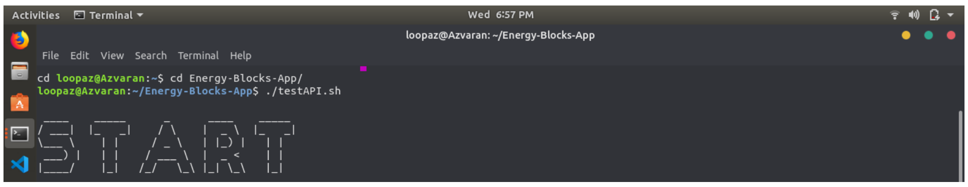
Figure 5.3: Run the SDK

Your nodejs server should run successfully as shown in Fig:5.3.

5.3 Create users channels and invoke chaincode

Once the REST API is running without any issue, run the following command to enroll users in different organisations, create channel, join organisatons to channel, update anchoer peers, install chaincode in each organisation, instantiate chaincode on the channel. Run the following command in a new terminal, change the location to the root directory of Energy-Blocks App.

```
$ chmod +x testApi.sh
$ ./testApi.sh
```



```
cd loopaz@Azvaran:~$ cd Energy-Blocks-App/
loopaz@Azvaran:~/Energy-Blocks-App$ ./testAPI.sh

  _____ _______       _____ _______
 / ____|__   __|/\   |  __ \__   __|
| (___    | |  /  \  | |__) | | |
 \___ \   | | / /\ \ |  _  /  | |
 ____) |  | |/ ____ \| | \ \  | |
|_____/   |_/_/    \_\_|  \_\ |_|
```

Activities
Terminal
Wed 6:57 PM
loopaz@Azvaran: ~/Energy-Blocks-App
File Edit View Search Terminal Help

Figure 5.4: Run the testApi script

The Fig:5.4 shows the execution of testApi. After successfull execution of API launch the react app for the interface.

5.4 Running the Frontend Web Interface

Go to frontend root directory and execute the following commands in the terminal.

```
$ npm run start
```

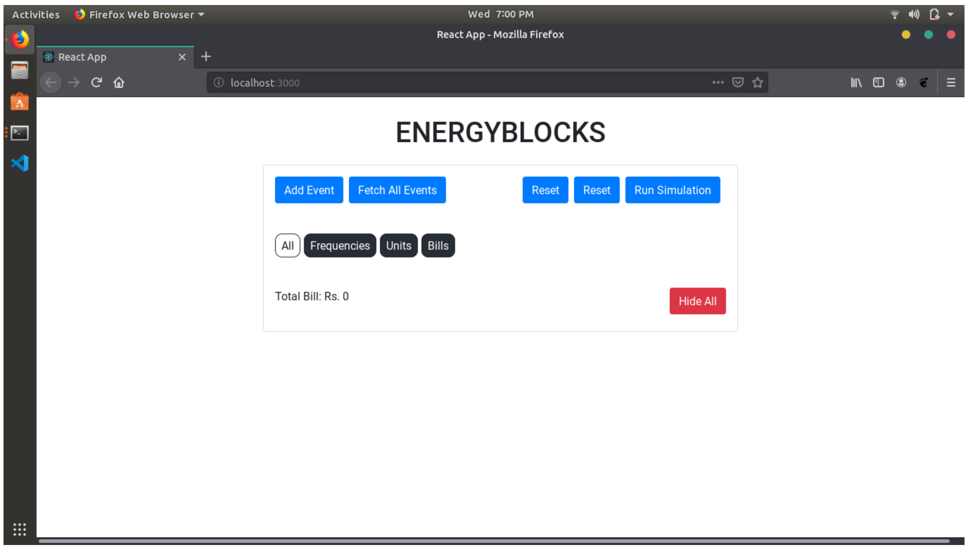


Figure 5.5: Web interface

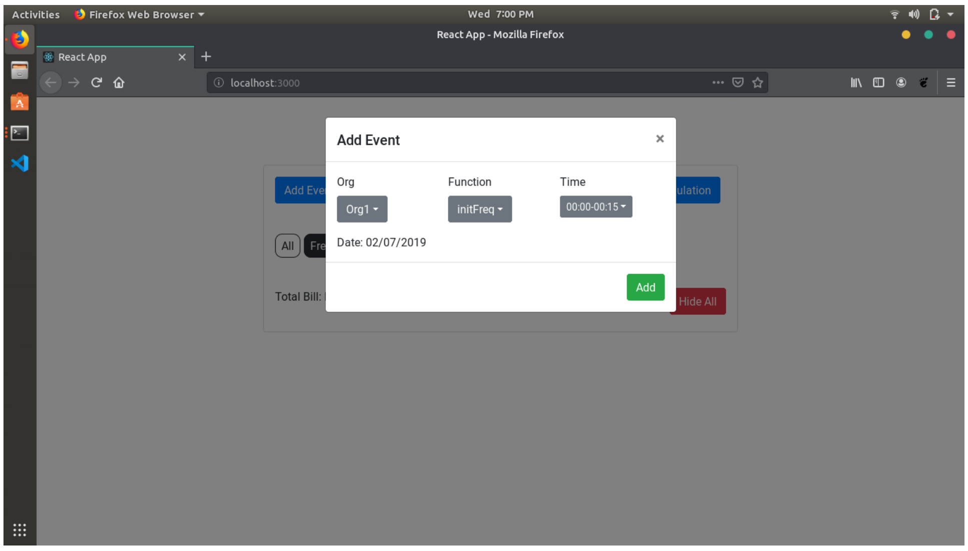
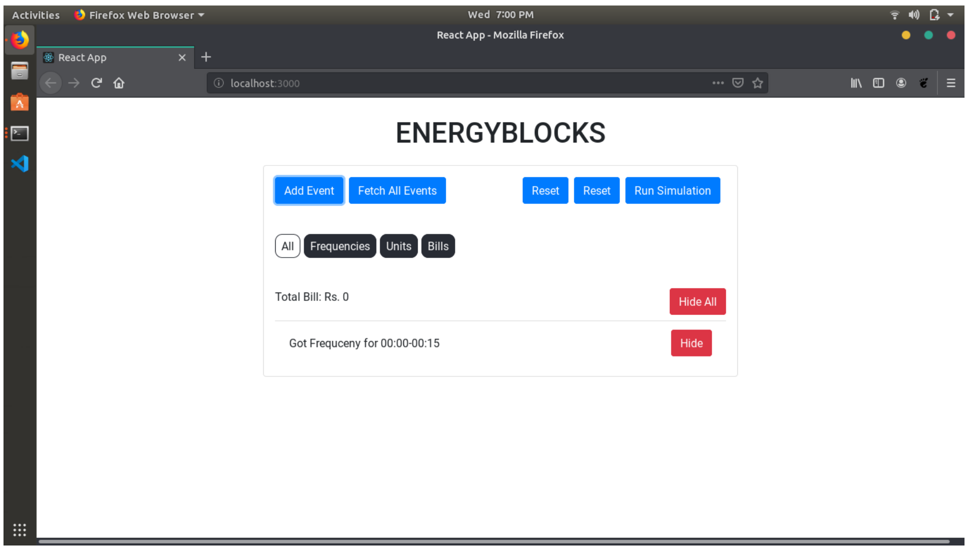


Figure 5.6: init Frequecy Methods

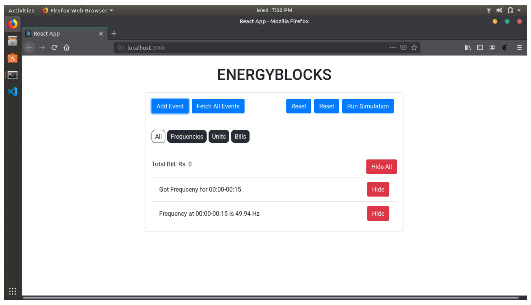


Total Bill: Rs. 0

Got Frequceny for 00:00-00:15

Figure 5.7: Read Frequecy Methods

Visit http://localhost:3000 to access the web interface as shown in Fig:5.5 . Run initFreq, then initUnit and then initBill for generating bill for a particular timeslot. Some of the test run pictures are provided.

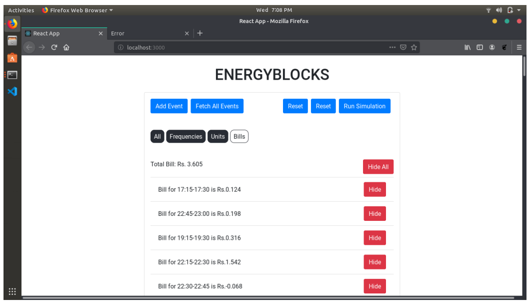


ENERGYBLOCKS

Total Bill: Rs. 0

Got Frequceny for 00:00-00:15

Frequency at 00:00-00:15 is 49.94 Hz



ENERGYBLOCKS

Total Bill: Rs. 3.605

Bill for 17:15-17:30 is Rs.0.124

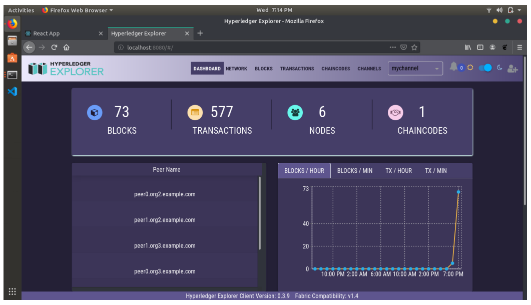
Bill for 22:45-23:00 is Rs.0.198

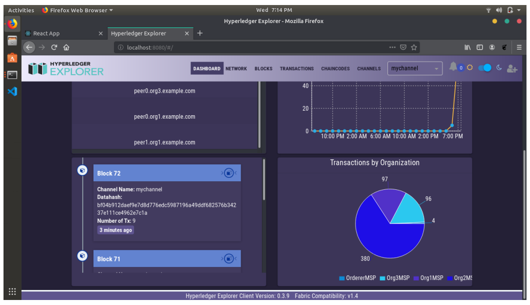
Bill for 19:15-19:30 is Rs.0.316

Bill for 22:15-22:30 is Rs.1.542

Bill for 22:30-22:45 is Rs.-0.068

Figure 5.8: Simulation test cases





Channel Name: mychannel
Datahash:
bf04b912daef9e7d8d776edc5987196a49ddf682576b342
37e111ce4962e7c1a
Number of Tx: 9
3 minutes ago

Figure 5.9: Hyperledger Explorer

Chapter 6

Conclusion

Through this report we see that the power gap between the grid and household is soon going to be reduced significantly in the near future. With the evolution of grids with two-way communication allowning the consumers to be a local power producer is going to fulfil this massive gap. Such services won't be feasible in the current billing mechanism as these mechanisms lack incentives for the consumers to sell what they produced. Hence, the concept of frequency based billing comes into picture. This implementation is seamlessly handled by our blockchain platform. Chapter 3 describes why and how frequency based billing is applicable. Chapter 4 introduces our Energy-Blocks app which is built using Hyperledger Fabric to solve the issue of frequency based billing. Chapter 5 provides the manual on how to use the app to interact with the blockchain. The results of this application have shown its uncanny precission to handle different factors involved in such billing scenarios.

6.1 Future Work

- Making the current architecture and interface to support multiple households and distributors organistaions.
- Adding feature to monitor the past consumption and production of power by the households.
- Adding feature to fine the households if they do not pay the bill until a given threshold is reached.

Bibliography

[1] S. Nakamoto *et al.*, “Bitcoin: A peer-to-peer electronic cash system,” 2008.

[2] K. Bhattacharya, “Frequency based pricing as an alternative to frequency regulation ancillary service.” `http://www.iitk.ac.in/npsc/Papers/NPSC2000/p35.pdf`. Accessed : 28 June, 2019.

[3] M. Andoni, V. Robu, D. Flynn, S. Abram, D. Geach, D. Jenkins, P. McCallum, and A. Peacock, “Blockchain technology in the energy sector: A systematic review of challenges and opportunities,” *Renewable and Sustainable Energy Reviews*, vol. 100, pp. 143–174, 2019.

[4] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, *et al.*, “Hyperledger fabric: a distributed operating system for permissioned blockchains,” in *Proceedings of the Thirteenth EuroSys Conference*, p. 30, ACM, 2018.

[5] C. Cachin, “Architecture of the hyperledger blockchain fabric,” 2016.

Appendix

The source code for the above application can be found on github : `https://github.com/MD-AZMAL/Energy-Blocks-App`

The frontend for the app can be found here : `https://github.com/shubhsherl/frontend`

Test run video : `https://www.youtube.com/watch?v=BOYDjUOLbOI`

Follow the readme in Energy-Blocks-App to setup your environment and install the pre-requisites. Refer to manual section of this report to understand and run the application without any errors. The readme provides a section to run Hyperledger Exploerer for the network to monitor the transactions, channels, chaincode and more.