Simulating SIGChain: Microgrid Agent transactions with a Lightweight Blockchain

Abstract. Solar production is becoming a growing source of new energy worldwide. More and more consumers and companies are interested in investing in photovoltaic (PV) panels and battery storage for different reasons: ecologic interest in green and renewable energy, economic interest in energy trading. This paper presents simulation results between neighbor customers (agent nodes in a Microgrid) trading their solar photovoltaic production overhead. The simulation is run with AnyLogic. Different scenarios with different types of nodes have been implemented such as: agents using storage (battery) versus agent without storage, agents with different trading motivations (green agents versus greedy agents. Also, each scenario is played at different seasons of the year taking into account the different sunshine levels. A "lightweight" blockchain has been implemented and added to the simulation tool to handle the transactions of surplus energy among customers in each Microgrid: 100 nodes split in 16 Microgrids according to the neighborhood are running so far. Different statistics have been collected and averaged per year such as the average energy savings per year and the average customer benefits per year. The simulation is currently tuned with real consumption and production data produced by the SIG (Services Industriels de Genève) in order to incorporate it to the existing SIG smart meters and discover the price interval at which trading customers can achieve benefits while maintaining the SIG infrastructure cost-effective.

Keywords: Agent-based Simulation, Microgrid, and Blockchain

1 INTRODUCTION

Solar production is becoming a growing source of new energy worldwide. More and more consumers and companies are interested in investing in photovoltaic (PV) panels and battery storage for different reasons: ecologic interest in green and renewable energy, economic interest in energy trading. This paper presents simulation results between neighbor customers (agent nodes in a Microgrid) trading their solar photovoltaic production overhead. The simulation is run with AnyLogic[1]. In order to support the inter-agent electricity trading a SIGChain[2] (lightweight blockchain) has been implemented. The remaining of the paper is organized as follows: next chapter describes recent related works, chapter 3 details the different agent types. Chapter 4 explains the simulation context and the different scenarios. Chapter 5 provides the SIGChain implementation. The conclusion is provided in the last section.

2 RECENT RELATED WORKS

This section summarizes the use of blockchain technology in recent energy market. In 2014, [2] first uses blockchain technology in energy markets. The authors propose a virtual currency (Nrgcoin) for trading renewable energy in smart grids. However, the system is not fully distributed since the market model keeps depending on the central network operator. A recent paper

[1] AnyLogic is an agent-based simulation tool at: https://www.anylogic.com/.

[2] SIG is the main company providing the public services at Geneva (Switzerland) such as: gas, electricity, and water. at: https://ww2.sig-ge.ch/en/home-en.

[1] proposes the concept of a blockchain-based microgrid energy market without central energy management. The Brooklyn microgrid is used as a case study and the authors show that three of the seven identified requirements are fully satisfied. In [3] and [4] the authors focus on maintaining the privacy aspects when performing energy trading in a local environment with anonymous participants. An issue with classical blockchain mining paradigms [5] is their greedy aspect in terms of power consumption, which is particularly undesired in a microgrid environment. Therefore research works to implement lightweight mining paradigm in a local grid are desired. We use the point of view of implementing such a paradigm for our project with the SIG Company. Our work is inspired by the lightweight blockchain model proposed by [6][7].

3 AGENT DESCRIPTION

Entities used in AnyLogic simulation tool are called agents. There are of two types: Producer Agents (PAs) and the Power Plant Agent (PPA).

3.1 Power Plant Agent (PPA)

There is just one PPA. It has two parameters that are collected for statistics purpose: a) the number of received transactions that have been carried out, b) the total bought energy amount expressed in kilowatts per hour. This amount is re-initialized every day in the simulation.

3.2 Producer Agents

PAs play both roles of purchasers and buyers. They produce electricity for their own use according to their consumption. The overhead is either stored in a local battery or directly sent to the network. When sent to the network, the corresponding amount (expressed in kilowatts per hour) is saved by the PA. At the time the user needs to buy electricity from the power station, this amount is deducted by the PPA and provided for free to the user PA. Only the network usage fee will be charged to the user according to the SIG photovoltaic network usage policy.

4 SIMULATION

4.1 Initial Conditions

At the beginning of the simulation, the PAs are randomly dispatched. Then, they are gathered in 16 identical sectors called districts. PAs of the same district are considered as neighbors. Only neighbors of the same district (neighborhood) can execute power transactions between each other. In the current simulation, different agents and parameters can be configured which are:

- finalProducers: The table, which contains all the producers. It defines desired agent number as well as their initial parameters.
- powerPlant: The SIG power network.

4.2 State Machine Explanation

- When the simulation starts, all the agents are in “statechart” state. This state will define each agent neighborhood from the beginning.
- Then the agent moves to the “production” state. This state will attribute each agent its electricity balance.

- Third, the state machine will move the agent in the “NoBattery” (agent does not have a battery) or “HaveBattery” (agent owns a battery) state.
- These two states have a reflexive (self) transition, which is used when the agent electricity balance is positive. An agent without battery will send its overproduction to the network. Otherwise, the agent saves its surplus directly in its battery. When the battery is full, the surplus is sent to the network. As long as an agent is in a negative balance it needs to buy electricity and it stays in the “Buying” state.

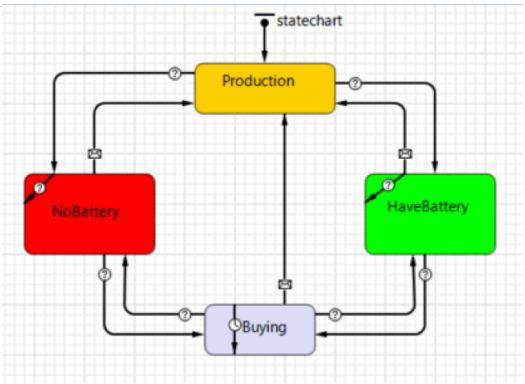

Figure 1: PA state machine during the simulation

When the simulation is started, the interface in Figure 2 is displayed. All running agents are on the left side with different colors corresponding to different neighborhood (community of self consumers) that agents belong to. On the right, appear the several useful statistics.

- The two pie charts correspond current SIG tariff distribution and battery distribution.
- Below the pie charts, 1) the two savings blue chart: one is the average savings made by agents when by buying electricity to their neighbors. The other, is the average savings made when re-buying produced electricity previously sold to the SIG. This corresponds to the same purchase price exempt from the network usage tax. For example with a purchase price of 24 cents: 24 - 10 (network usage tax) = 14 cents of savings.
- In the right, the graph containing bars is the production and consumption in one of the self-consumer communities (district in the figure).
- The bottom graph is the purchase of electricity from the SIG, which is simply the electricity that the community did not produce and asked the SIG for. The chart on the right is the profits made by selling electricity to its neighbors.

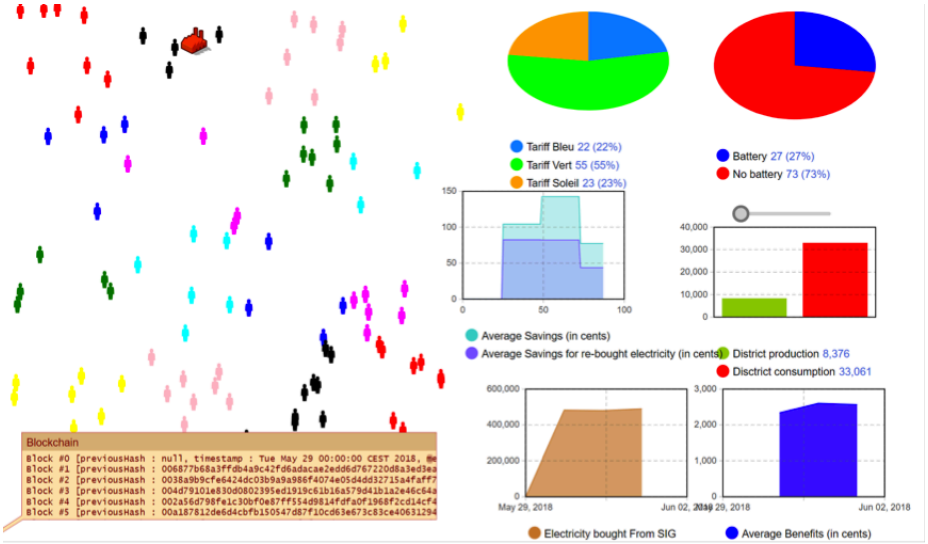

Figure 2: Simulating 16 Microgrids with AnyLogic

4.3 Different scenarios

For the moment, there are two different scenarios in the simulation.

- Scenario 1: First of all, the user who absolutely wants to buy his electricity at the cheapest price possible. This type of buyer will analyze its neighbors to see if there is not a way to have electricity cheaper than its current SIG agreement.
- Scenario 2: In the second scenario, a user absolutely wants to buy green electricity regardless of the price. He is willing to pay more to get neighbors' photovoltaic electricity. He also distinguishes the type of electricity in his neighborhood; for example, he refuses to buy electricity from a neighbor at the blue tariff.

5 LIGHTWEIGHT BLOCKCHAIN

The blockchain (called SIGChain) that is used in the simulation appears as a standard blockchain where blocks containing electrical power transactions are linked between each other. Each neighborhood constitutes a consortium where the mining node is randomly designated every hour in order to mine the next block. The blockchain has been implemented in java.

5.1 Block description

Blocks are the crucial part of the blockchain. They contain all the transactions and ensure that the blockchain is secured and has not been forged (blockchain integrity). The "Block" class implements the blocks with the following parameters (Figure 3):

- Index or : Block number in the blockchain
- PreviousHas: Previous block hash
- Timestamp: Mining date (when the block has been mined)
- merkleRoot: Root value of the transaction Merkle tree
- transactionsID: Table with transaction identifiers of the current block
- Data: Optional field used to store information. Notably used to announce the first

block.

- idMiner: Identifier of the block miner
- Hash: Block hash
- Nonce: This value is incremented until the bloc hash fits to the blockchain difficulty

Figure 3: A SIGchain example

5.2 Building the blocks

The block class owns two constructors:

- constructor 1: is used for the first block only. It assigns the values corresponding to index, timestamp, previousHash, and data.
- constructor 2: is used for the other blocks. It assigns more values: index, timestamp, previousHash, and idMiner. Besides, it computes the Merkle.

5.3 Root Tree

The root tree calculation function works as follows:
(1) Verification that there are transactions to work on. In this case, their respective identifiers are saved in a temporary list for future calculation.
(2) Verification that the number of transactions is even. Otherwise, the last transaction is duplicated.
(3) Transactions are paired in a loop to compute their common (intermediary) hash. This operation creates the upper nodes of the tree. They will not be kept in the root tree process.
(4) If there is more than one intermediary node created, the function returns to step 2 with these nodes. Otherwise, the function goes to step 5.
(5) There is one last node, the function keeps it as of the tree root.

When the block has been successfully created, its hash has not yet been determined. The agent

creating the block must then call the "mineBlock" function. This function computes the hash by incrementing the nonce so that the created hash respects the nonce difficulty linked to the nonce. This difficulty is the number of zeros that the hash must contain at its beginning. For example, if the difficulty is 4 zeros, the hash must start with 0000. All blocks must then be stored in the blockchain.

5.2 SIGChain Operation

The blockchain is created from the start of the simulation. It has two properties:

- Difficulty: The difficulty of the blockchain corresponding to the number of zeros at the beginning of the hash.
- Blocks: The blocks that make up the blockchain.

Operation:
The Blockchain class constructor deals with assigning the value of the difficulty as a parameter and initializing the block list. The first created block of the list is mined and added.
For this purpose, the Blockchain class calls the "newBlock" method that creates a new block. It is available with two different settings options. After a block has been created, the agent that created it must add it to the blockchain. For this purpose, the "addBlock" method of the Blockchain class is called. The method uses the "mineBlock" method of the class Block to correctly mine the block. Once it has been mined, the block is added to the chain.

Validity verification methods are available:
The "IsBlockchainValid" method checks the first blockchain block and the 24 next created blocks created. This method calls the "isValidNewBlock" method that checks the links between the blocks and the new block hash value.

To summarize, a randomly selected agent adds a block in the blockchain by following the logic below:
(1) Check the blockchain validity with "isBlockchainValid" method.
(2) Add this block to the blockchain with "addBlock" method.

6 CONCLUSION AND FUTUREWORK

In these scenarios, only the first is economically viable. In the second scenario the user is losing a significant amount of money because he finds himself paying more than SIG prices. The next step of this work is to include the real data and up to date production and consumption data from the SIG Company and integrate our SIGChain to the SIG smart meters.

References

1. E. Mengelkampa, J. Gärttnera, K. Rockb, S. Kesslerb, L. Orsinib, C. WeinhardtaDesigning microgrid energy markets: A case study: The Brooklyn Microgrid , Elsevier Applied Energy Volume 210, 15 January 2018, Pages 870-880
2. M. Mihaylov, S. Jurado, N. Avellana, K. Van Moffaert, I.M. de Abril, A. Nowe Nrgcoin: virtual currency for trading of renewable energy in smart grids 2014 11th International conference on the European energy market (EEM), IEEE (2014), pp. 1-6
3. E. Al Kawasmi, E. Arnautovic, D. Svetinovic Bitcoin-based decentralized carbon emissions trading infra-

structure model Syst Eng, 18 (2) (2015), pp. 115-130
4. N.Z. Aitzhan, D. Svetinovic Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams IEEE Trans. Depend. Sec. Comput., PP (99) (2016),
5. Wenbo Wang, Dinh Thai Hoang, Zehui Xiong, Dusit Niyato, Ping Wang, Peizhao Hu, Yonggang Wen, A Survey on Consensus Mechanisms and Mining Management in Blockchain Networks, November 2018 at https://arxiv.org/pdf/1805.02707.pdf
6. C. Worley et al., "Scrybe: A 2nd-Generation Blockchain Technology with Lightweight Mining for Secure Provenance and Related Applications," 2018.
7. R. R. Brooks et al., "Scrybe: A Blockchain Ledger for Clinical Trials," IEEE, Blockchain Clin. Trails Forum, 2018.