Decentralized Energy Utility

Problem

In an isolated community there are two households with solar panel arrays: of Angel and of Bennie, they are producers of electrical energy. There's a consumer of electricity, a household of Carlos. All three of them are connected into an isolated electrical grid.

When Carlos burns electricity, his meter turns and counts kWh consumed. Some of this power came from Angel and some from Bennie; neither of them want to give out free energy and would like to be compensated. How does Carlos know how much to compensate each? Does he divide equally? But Bennie may be farther away and most of the electricity has flown from Angel; does he calculate the distance to his neighbors? But there could be hundreds of producers in the community and they may not know or trust each other.

Solution

Smart meters as nodes on a blockchain

We don't need to derive every producer's contribution as long as we count the amount of energy outflow by smart meters at the producers' connection to the grid; while every consumer's meter measures energy consumed. Neither of the parties should know or trust each other.

Each resident of the grid has an account on the blockchain with an amount of abstract electricity units: coins k. The balance of the account is changed by a smart contract according to the amount of energy consumed or produced.

Every hour an embedded code in the meters sends the readings to a smart contract on a blockchain to settle their accounts. Consumers' accounts are debited and producers' credited. Some of the coins are set aside into the utility's account to pay for the maintenance of the grid.

The contract does not move coins from consumers' accounts to the producers' directly as it is much simpler to first move the compensation from each consumer's account into a common account then move coins from it into each producer's account proportionate to their energy contribution.

Maintenance fees

The grid connecting producers and consumers needs to be maintained so some of the coins Carlos contributes are set aside into the utility's account for operational expenses. It could be a membership fee or a percentage of use. For simplicity let's use a small hourly fee of 0.01 k.

Energy lost in the grid

Energy produced and consumed is measured in kWh – a measure of work. Energy produced is spent on running consumer devices and unfortunately, on heating the grid wires. Thus producers' meters will always count more work done than consumers' meters count work consumed, so producers cannot be compensated one for one kWh.

If Angel's meter shows produced work of 1 kWh, Bennie's 2.3 kWh and Carlos' meter shows 3 kWh consumed it is clear 0.3 kWh went to heating the grid. But we don't need to calculate this and simply divide Carlos' contribution proportionately between the producers.

Settlement

Periodically a smart contract settles the accounts according to the readings all meters on the grid reported within the last hour.

1. Consumers report usage:
    1. Carlos' meter reports 3 kWh consumed: contract moves 3 coins k into the common account.
    2. Another consumer Danny’s meter reports 2 kWh consumed: contract moves 2 coins k into the common account.
2. Utility gets the hourly fee: contract moves 0.01 k from the

common account into the utility's account.

3. Producers are compensated:
    1. Angel’s meter reports 3 kWh produced.
       Angel gets (3 k + 2 k – 0.01 k fee) * 3 kWh/(3 kWh + 2.5 kWh) = 2.721 k: contract moves 2.721 from the common into the Angel's account
    2. Bennie’s meter reports 2.5 kWh produced.
       Bennie gets (3 k + 2 k – 0.01 k fee) * 2.5 kWh/(3 kWh + 2.5 kWh) = 2.286 k: contract moves 2.084 from the common into the Angel's account

Compensation in fiat money

Energy producers have paid for their solar arrays in fiat money and would like to be compensated in fiat not coins. Thus we need to introduce a payment system that lets convert fiat into coins and vise versa. The consumers need to go outside of the blockchain to the payment system to buy coins k for dollars; the funds are deposited into the utility’s bank account. The producers go to the payment system to sell their coins and the dollars are deposited into their personal bank accounts.

Prepaid electricity

In order to be able to receive electricity a consumer needs to first fill in his account with coins. An hourly settlement moves coins from his account into the common pool. Once the coins in the consumer's account are depleted the meter disconnects the consumer from the grid. Once the account is replenished the smart contract notifies the meter to reconnect the consumer.

Thus the hourly communication between the meters and the contract is two way: the meter reports the usage and gets back a decision to keep the connection open or not.

Funding arbitrary consumers

In the above examples a single resident pays for his own household: Carlos has his house's meter and pays for his own usage. But with the decentralized utility it does not have to be the only use case. A homeowners association may pool money to light their street; a condominium may pay for usage in their building hallways; a baseball team may pay to light a stadium at night. Each consumer of electricity: a household with a smart house meter or a lighting pole with its own meter may be funded with electricity coins from anybody's account. A municipality does not have to get involved. If a resident wants a lighting pole illuminating his drive way he finds the lighting pole 's id displayed and funds it with the coins from his account. This introduces transparency and promotes responsible spending of electricity.

Fiat payment system

A payment system such as a credit card processor acts as a trusted oracle with the knowledge about transfers of fiat money between the actors accounts. Upon a successful payment in fiat the payment system notifies affected chaincodes.

Use case: resident buys coins for fiat

Interface

Resident goes to the decentralized utility's website and enters his id and the amount in dollars to be added to his account with the utility. He is transferred to the payment system's page where he enters his credit card's details and confirms the payment. Once payment is successful, the payment system calls a webhook on the utility's website which in turn invokes the chaincode with the resident's id and the amount in dollars.

Chaincode

`addFiat(residentId, amount)`

Divides the amount in dollars by the preset dollar price of a coin k and adds the result to the resident's account.

Use case: producer sells coins for fiat

Interface

Producer goes to the decentralized utility's website and enters his id in and the amount in dollars to be withdrawn into his bank account. The payment system executes the withdrawal. Upon completion it calls a webhook on the utility's website which in turn invokes the chaincode with the producer's id and the amount in dollars, negative.

Chaincode

addFiat(residentId, amount)

Same as the above but the negative amount in dollars subtracts coins from the producer's account.

Hourly reporting and settlement

Use case: energy produced and consumed reported hourly

Interface

A page with a map of producers depicted as solar panels and consumers depicted as a house or a lighting pole. Both producers and consumers have meters attached with visible count of energy produced or consumed.

A clock on a page can be clicked to move the time one hour ahead. On every change the numbers on the meters change illustrating hourly report to the blockchain. The numbers can be random but arbitrary: the sum of readings of the producers' meters should be 10% higher than the sum of the readings of the consumers. This simulates the 10% loss to the resistance of the grid.

Chaincode

report(meterId, k)

Records into a map of keys of meterId, a temporary storage on the blockchain. The amount of kilowatt is positive for the consumers and negative for the producers.

Chaincode

settle()

Executed periodically. Goes over the map of readings collected over the past hour. Moves coins from the consumer's accounts into the common pool. Calculates total consumed and produced. Moves coins from the common pool into the producers' accounts according the formula shown above. Moves coins into the utility's account as hourly fees.

Funding devices

Use case: a resident funds his household's electricity with coins from his account

Interface

A resident has one or more meters he needs to fund: his house and a rental property. He goes to his personal page on the decentralized utility's website, views total coins in his account, views current balance in coins per meter, may need to go to the payment system to replenish total amount of coins.

Selects a device denoted by its meterId, enters amount of coins to fund it. The backend is called and in turn invokes a smart contract.

Chaincode

```
fund(residentId, meterId, amount)
```

Moves amount of coins from the resident's account to the device's account.