Blockchain and Smart Contract Payment for Electric Vehicle Charging

Natthanan Chanthong
Department of Computer Engineering
Faculty of Engineering
King Mongkut's Institute of Technology Ladkrabang
Bangkok, Thailand
e.natthanan@outlook.co.th

Thitiwat Ruangsakorn
Department of Computer Engineering
Faculty of Engineering
King Mongkut's Institute of Technology Ladkrabang
Bangkok, Thailand
thitiwataeim@gmail.com

Sorayut Glomglome
Department of Computer Engineering
Faculty of Engineering
King Mongkut's Institute of Technology Ladkrabang
Bangkok, Thailand
sorayut.gl@kmitl.ac.th

***Abstract*—Our research aims to study, design and build the electronic payment system for electric vehicle (EV) charging using blockchain and smart contract technologies. Both technologies are used to control and manage payments and to decentralize the payment system, so the devices can automatically pay each other. In addition, our system lessens the inadequate of charging station for EV when travel in the long distance because our proposed let EV owners who also have charging facilities at their homes share their facilities and automatically get paid. Blockchain node is set up in charging station and is controlled by the application for EV owners. The application can show the charging information and control the payment and charging process automatically. For EV part, we simulated the data transmission between charging station and EV.**

Keywords—Blockchain; Smart contract; Electric vehicle charging; Electronic payment

I. Introduction

In the present, the rate of electric vehicle (EV) usage is gradually increasing globally. One of importance problems that people are reluctant to change to EV is probably the inadequate of charging stations. Therefore, a trip of a long distance such as vacation trip or a trip to visit relatives seems to be impossible because there might not have charging station established throughout the trip.

Assuming that the most of the EV owners have charging facilities in their homes. If they share their charging stations to other EV owners to charge their vehicles and also receive payment, this will increase the number of charging stations and reduce the anxiety of EV owners for their long-distance trips. This scenario will result in the increasing rate of EV usage.

The case study of this solution is Share&Charge[1]. Share&Charge built the blockchain platform to share personal EV charging stations to other EV owners. Share&Charge combines the blockchain and EV charging together by using the blockchain to manage the charging process. Share&Charge has a blockchain base application that allows EV owners getting access to personal charging station and then making payments for electric charging in peer-to-peer manner.

We apply the concept of this case study to our research and develop to our purposed system. We purpose to build the electronic payment system for EV charging using blockchain and smart contract[2] to manage transactions of payments such as, validating and automatic payment. In addition, we use smart contract to control the charging process in charging station. The blockchain can help getting rid of central payment system because every transaction is open to public, so EV can charge and pay in peer-to-peer manner.

Blockchain platform that we used is Ethereum[3] because it has a smart contract to create program to manage and control hardware for charging process and Ethereum has a small transaction time around 15 - 20 seconds.

We separate blockchain nodes into two types which are the application node and the station node. The application node is a node for customer to order to charge the electric. The station node is a node for charging station to manage charging process.

II. Related Works

A. Managing IoT Devices using Blockchain Platform [4]

This research presents the method to manage hardware devices such as, Raspberry Pi (RPi) using Ethereum blockchain by programming smart contract to control the privacy of electronic equipment and deploy smart contract on Ethereum blockchain then set up Ethereum node on RPi.

The result is that RPi can control electronic equipment policy as programmed in smart contract. So, we apply this solution to our research to use smart contract to manage control charging station with RPi.

B. Thing-to-Thing Electricity Micro Payments Using Blockchain Technology [5]

This research presents micropayments between devices without human interaction using blockchain technology by using RPi with blockchain node to manage payment and control electricity supply with relay.

We apply concept of this research to make automatic payment system to EV charging.

978-1-7281-6486-1/20/$31.00 ©2020 IEEE

III. Related Knowledge

A. *Blockchain*

A blockchain is a decentralize database that stores data in several local nodes. Each local node has the same data called "Blockchain". Each block is referred to previous block. Inside each block is a set of transactions that were created in each blockchain node.

B. *Go-ethereum*

Go-ethereum [6] is the command line interface for running a full Ethereum node implemented in Go. It has many functions such as, mining real ether, transferring funds between addresses, creating contracts and sending transactions, explore block history and much more

Go-ethereum has 3 interfaces:

- Javascript Console including web3 javascript API
- JSON-RPC server
- Command line options

C. *Smart Contract*

Nick Szabo defined a smart contract as is a protocol that computerizes transactions such as payment terms, liens, confidentiality, and even enforcement. The objective of smart contract is to manage a common contractual condition and to minimize the need for trusted intermediaries.

In blockchain, smart contract is a program that was deployed into blockchain and mined as transaction so, it can't be edited or deleted. This program will be invoked from each node in blockchain by receiving input from node then executing the code and sending the output result to store in blockchain. We use solidity [7] to write the smart contract to manage and control charging process.

IV. Proposed System

A. *System Architecture*

The proposed system is shown in Fig. 1, we separate blockchain nodes into two types, the application node and station node. Every node is set up in private blockchain system.

Application node is a node that was set up on server. This node is accessed by the user web application to order the charging and this node is used as mining node to mine all transactions in system network. Transactions created in this node are verified the owner of each account wallet, transfer ETH token from EV owner to station owner, validate the ETH transfer and command to start or stop charging.

Station node is a node that was set up in charging station using RPi to control charging process. Transactions that is created in this node are calculating amount of ETH token from charging data that read from EV, confirm charging and control charging station to start and stop charging.

In web3 library, it has many necessary functions including account management, ETH transferring, transaction reading etc. Moreover, we create our own smart contract and deploy into blockchain to be invoked by application node and station node to execute function inside smart contract. Inside smart contract has important variables that are electric power, ETH to pay, price rate, status, account name and account address etc. Smart contract has functions to get and set variables, payment validation and confirm charging.

Simulation of EV, we created a program to simulate communication between charging station and EV by using RPi communicate with UART protocol.

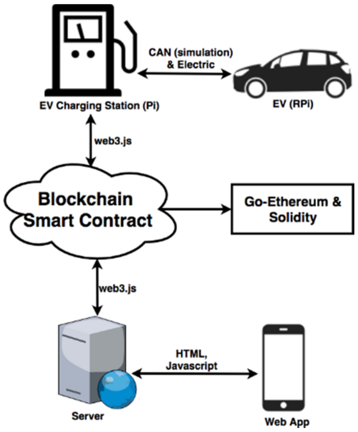


Fig. 1. System Diagram

B. *Activity Diagram*

Fig. 2 shows the activity diagram of the system. The workflow of the system depends on the "status" value. In idle state, the status is "available" that means there is no transaction in application node and station node.

From activity diagram in Fig. 2, Step 1) and 2) When EV connects to charging station. EV will send charging data to charging station then station node create transaction as follows:

status = "wait for paid"

- power = amount of power to charge
- balance = power * price rate

All variables that were set by station node will be read by application node and show in web application.

Step 3) and 4) When user orders to charge. Application node will create a transaction as follows:

- transfer ETH from user account to station account
- status = "validation"

Step 5) When a status is equal to "validation". Application node will create a transaction to call smart contract function to validate payment. If validation success, the status will be changed to "charging"

Step 6) When the status equals to "charging". Station node will start charging. If charging station charge power to 1 kilowatt, station node will read battery value and create a transaction to call smart contract function to confirm charging. If confirmation success, the status will be changed to "payment" and set variables as follows:

- power = power – 1
- balance = power * price rate

Step 7) If the power variable is more than 0, application node will continue to payment state.

Step 8) if the power variable equals to 0, application node will create a transaction as follows:

- status = “fully charged”

Step 9) and 10) When the status equals to “fully charged”, Charging station will stop charging. Then charging station disconnects from EV. Station node will create a transaction as follows:

- status = “available”

In this state, if user orders to stop charging, application node will create a transaction as follows:

- status = “stopping”

And then charging station will stop charging and station node will create a transaction as follows:

- status = “stopped”

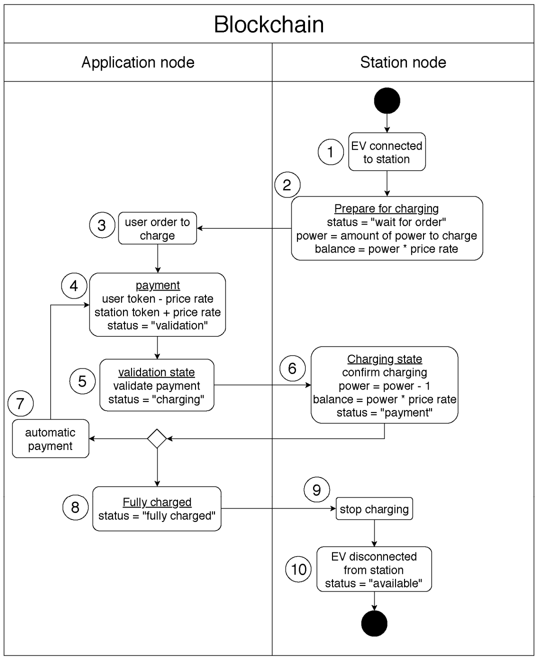


Fig. 2. Activity Diagram

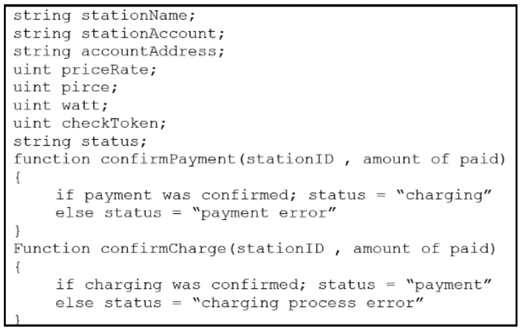
C. Smart Contract

Fig. 3 shows the pseudo code of smart contract that was deploy on blockchain. This smart contract will be called by application node and station node.

Fig. 4 shows the description of a function in smart contract including confirmPayment function, confirmCharge function and all get and set functions for each variable.

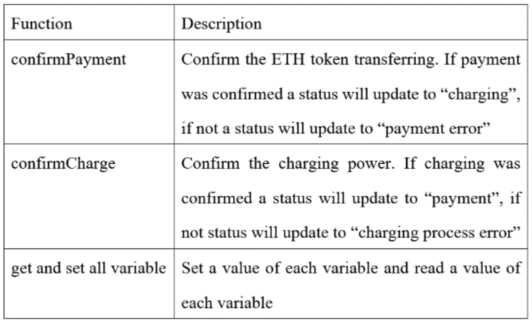
D. Behavior Diagram

Fig. 5 shows the behavior of the system. First step, the station node sets a power and status variable in blockchain then blockchain will calculate a price depend on amount of power and price rate. Second step, the application node gets a power, status and balance variables from blockchain then pays the ETH token and sets a status in blockchain. Third step, the application node calls a confirmPayment function in blockchain then blockchain validates a payment and set a status variable. Fourth step, the station node gets a status variable from blockchain and continues a charging process. After finish a charging process, the station node calls a confirmCharging function in blockchain then blockchain confirms a charging and sets a status then continues to next cycle of the system.



```
string stationName;
string stationAccount;
string accountAddress;
uint priceRate;
uint pirce;
uint watt;
uint checkToken;
string status;
function confirmPayment(stationID , amount of paid)
{
    if payment was confirmed; status = “charging”
    else status = “payment error”
}
Function confirmCharge(stationID , amount of paid)
{
    if charging was confirmed; status = “payment”
    else status = “charging process error”
}
```

Fig. 3. Smart Contract Code



Function	Description
confirmPayment	Confirm the ETH token transferring. If payment was confirmed a status will update to “charging”, if not a status will update to “payment error”
confirmCharge	Confirm the charging power. If charging was confirmed a status will update to “payment”, if not status will update to “charging process error”
get and set all variable	Set a value of each variable and read a value of each variable

Fig. 4. Smart Contract Description

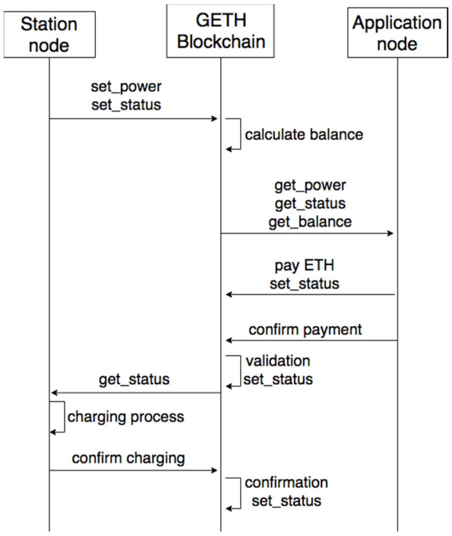


Fig. 5. Behavior Diagram

V. EXPERIMENTS

A. Set up blockchain node and connet node together

To create a blockchain network to manage a transaction of the system.

1) Experimental procedure

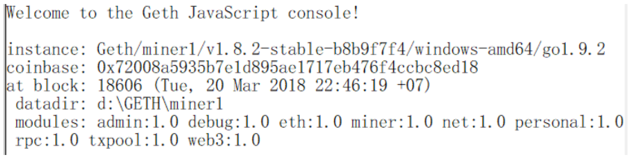
a) Set up go-ethereum blockchian on computer.

b) Set up go-ethereum blockchian on RPi.

c) Connect all node via WiFi.

2) Result

Every node can connect and communicate with each other as shown in Fig. 6



```
Welcome to the Geth JavaScript console!

instance: Geth/miner1/v1.8.2-stable-b8b9f7f4/windows-amd64/go1.9.2
coinbase: 0x72008a5935b7e1d895ae1717eb476f4ccbc8ed18
at block: 18606 (Tue, 20 Mar 2018 22:46:19 +07)
 datadir: d:\GETH\miner1
 modules: admin:1.0 debug:1.0 eth:1.0 miner:1.0 net:1.0 personal:1.0
 rpc:1.0 txpool:1.0 web3:1.0
```

Fig. 6. Javascript console

B. Access web application

To use smart contract to manage and control work process of EV charging system and monitor work state of the system

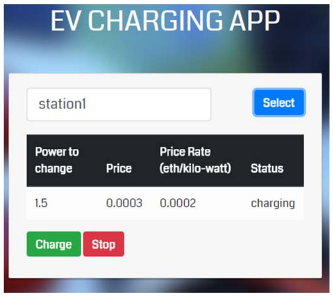
1) Experimental procedure

a) Create web application to access application node and call smart contract function.

b) Create the application that runs on station node to call smart contract function and communicates with another RPi that is simulated as EV charging system.

2) Result

Web application can order to charge, and display charging status as shown in Fig.7. The charging process is working correctly as programmed in smart contract as shown in TABLE I. "Power to charge" shows a remaining power that charging station has to charge. "Price to pay" shows a remaining ETH token that application node has to pay for the charging. "Price rate" shows a price rate of the charging station. "Status" shows current status of the charging process.



EV CHARGING APP

station1

Power to change	Price	Price Rate (eth/kilo-watt)	Status
1.5	0.0003	0.0002	charging

Fig. 7. Web Application

All results show that we can use blockchain smart contract to control the charging process. Fig.8 shows the transaction time in every 1 kilowatt and the average time is 33.46 seconds not including a charging time. The hardware specification of mining node is intel core i7 4710HQ CPU, GTX850M VGA and 16 GB of RAM. We can reduce a transaction time by increase a mining thread in GETH command or use the better hardware to perform mining. We use 2 mining threads in our experiments.

VI. CONCLUSION

Electronic payment system using private blockchain and smart contract has application node and station node. Application node can order to charge by creating the transaction to pay ETH token and validate payment in smart contract. Station node control charging process in charging station by creating transaction to update charging value and confirm the charging. The workflow in the proposed system is controlled by a status variable.

Although, our system is not directly apply to a real charging station and EV charging system but the experiment shows that we can control devices to make payments with each other automatically and control a work process in each state of charging station using smart contract.

The first limitation of the system is that the transaction time is quite long in each working state. Secondly, the storage of blockchain will increase if the system kept running continuously.

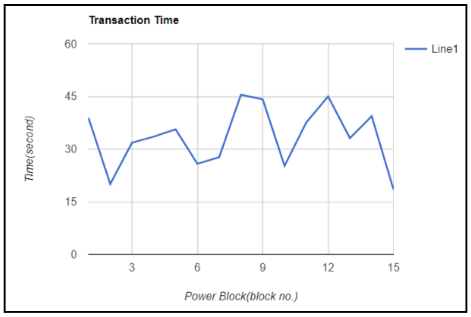


Fig. 8. Transaction time in each power block

TABLE I. SYSTEM STATE FLOW

Power to charge (kilowatt)	Price to pay (ETH)	Price rate (ETH/kilowatt)	Status
0	0	0.0002	available
1.5	0.0003	0.0002	wait for payment
1.5	0.0003	0.0002	validation
1.5	0.0003	0.0002	charging
0.5	0.0001	0.0002	payment
0.5	0.0001	0.0002	validation
0.5	0.0001	0.0002	charging
0	0	0.0002	payment
0	0	0.0002	fully charged

REFERENCES

[1] Share&Charge. 2017. What is Share&Charge?. [Online]. Available : https://shareandcharge.com/.

[2] K. Hobbs, L. Cheng, J. Walsh and B. Onn, "User-Friendly Smart Contract Compiler", Etherparty

[3] Narayan Prusty. 2017. Building Blockchain Projects. Birmingham : Packt Publishing Ltd.

[4] Seyoung Huh, Sangrae Cho and Soohyung Kim. 2017. "Managing IoT Devices using Blockchain Platform." Thesis of Electronics and Telecommunications Research Institute.

[5] Thomas Lundqvist, Andreas de Blanche, H. Robert H. Andersson. "Thing-to-Thing Electricity Micro Payments Using Blockchain Technology." Department of Engineering Science University West, Trollhättan, Sweden.

[6] GitHub. 2017. Go-ethereum. [Online]. Available : https://github.com/ethereum/go-ethereum.

[7] Solidity. 2017. Solidity 0.4.21 documentation. [Online]. Available : http://solidity.readthedocs.io/en/v0.4.21/