Blockchain and Cryptocurrency

Assignment 5: Introduction to Ethereum & geth

This assignment can help you understand Ethereum network. To do this, you should install and run the go-ethereum (**geth**). **geth** is one of the clients with full functionality of Ethereum, providing multiple interfaces. It enables interactive JavaScript consoles and JSON-RPC server features. Geth is an official golang implementation of the Ethereum protocol. You will perform this assignment in a similar way to the third one.

Description of this assignment:

- This task consists of a series of steps, like a tutorial.
- You must submit a report containing the results obtained as you go through each step.
  - ⇨ **Submit a report containing all results obtained from Steps 1 to 8.**

Minimum Requirements:

- You can use your laptop or desktop running Windows, Mac OS X, Ubuntu or other linux.
- You need at least 130 GB of hard disk space with a read/write speed of 100 MB/s.
  - ■ If you do not have required disk space, you can use the --fast option to reduce the storage requirement
- 4 gigabytes of memory (RAM).

<Step. 1> Install Go language

The Go language must be locally installed to build the source code because **geth** is written in go language.

Go to the following URL for installing Go ☞ https://golang.org/dl/

<Step. 2> Install go-ethereum (geth)

Go to https://github.com/ethereum/go-ethereum, download the source code from git, and compile it on your PC.

※ After installing Go and **geth**, check the version and completion of installation.

<Step. 3> Check out the command line and management APIs.

Search for useful commands and management APIs. Include them in your report.

<Step. 4> Building Private Network

4-1) Generate a genesis file

To build a private network, you have to generate a json-type genesis file and name it something like genesis.json

4-2) Create a data directory to use in the private network

4-3) Make genesis block by using a genesis file

4-4) Run **geth** engine with console

<Step. 5> Generate addresses / Mine Ether / Check the balance

5-1) Create two addresses which can be used for sender and receiver in transmission testing.

5-2) Look up your account list.

5-3) Set the etherbase address. Before you transfer Ether, you must mine Ether to verify the transaction. So, you have to specify the etherbase to be rewarded after mining.

※ Basically, the address generated first is assigned as an etherbase address. You can check it first with a certain API.

5-4) Start mining.

※ To get some Ether to be used for transmission in the private network, you have to start to mine Ether.

5-5) After mining, check the balance of all accounts.

<Step. 6> Unlock the account / Send the transaction.

6-1) Check the status of sender's account

6-2) Before sending Ether, you have to unlock the sender's account.

6-3) Send 20 Ethers to the receiver's address

※ Keep in mind that the minimum gas for transfer is 21,000.

6-4) Check the pending transactions.

※ To validate the transaction, block mining is required.

6-5) Check the pending transactions again after mining.

6-6) look up the balance of receiver's account to see if sender successfully transfer 20 Ethers.

<Step. 7> Search for blocks

Try to search for blocks contained to the private chain.

1) Search the total number of blocks connected to the main chain.
2) Height of block: 0
3) Height of block: 14460
4) The block including the transaction which is used in transferring your Ether.

<Step. 8> Join other's private network

※ You can join other's private network by connecting a node. To connect a node, each node has the same network id and genesis file. By using attached genesis file, enode and IP address, join the private network which is already built before.

8-1) Connect the node from your PC

8-2) Print out the information of connected peers

8-3) Try to mine new block

8-4) Send Ether as much as you want to 0xfa20ec148217d1091194adae3c9f90558ca471c4

<TA>
Email: kkc90@postech.ac.kr

Genesis file:

```
{
      "config": {
            "chainId": 8070,
            "homesteadBlock": 0,
            "eip155Block": 0,
            "eip158Block": 0
      },
      "difficulty" : "0x10000",
      "gasLimit"      : "0x10000000",
      "alloc"         : {
            "0x517d601e0b20483732d04c22246dfec575e1e6ad": { "balance": "300000000000000000000"},
            "0x93a1dd1d03e157a48e573da5b598dcdb3e8a205d": { "balance": "500000000000000000000"}
      },
      "nonce"         : "0x0000000000000000"
}
```

enode:
9157807b41da7be331120e8bd94afabae22d99b8c312c80ed1223fde71cbe33a304e8b0e3a9ed8f0a35
51e4ec38ad6225ab5ecb7393e4e6765a53bb75de3ce9e
ip address/port: 141.223.82.142:30303