2. *Automatically peer discovery and sending transactions*

A. Install node version manager. Please read all steps before running any command.

1. node version manager (nvm)
   Please go to this page and follow the steps: https://github.com/nvm-sh/nvm

2. If you do not want to read, in most of the cases, simply running the following should work:
   ```
   curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.37.2/install.sh | bash
   ```

3. Close your console/terminal and start a new one. In the new terminal, run
   ```
   command -v nvm
   ```
   If you see nvm, it means you have nvm installed successfully.

4. Run each of the following command
   ```
   nvm install 8.0.0
   nvm install 12.13.0
   nvm use 8.0.0
   node --version
   nvm use 12.13.0
   node --version
   ```

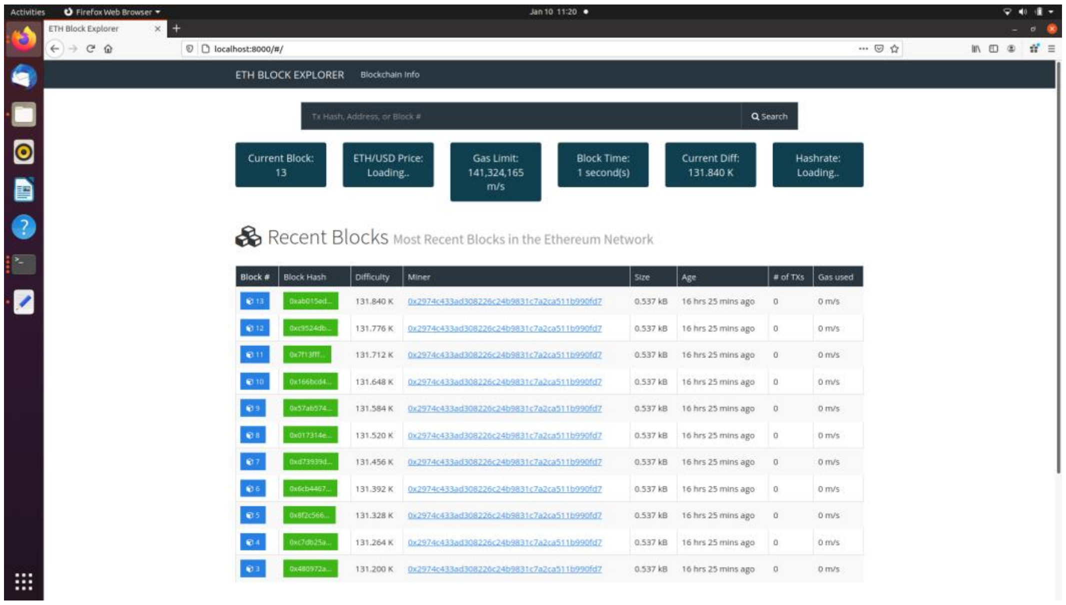
   Now, you should know how to install and switch between different versions of node.

B. Verification of the installation:

1. Run the following command to use node version 8.
   ```
   nvm use 8.0.0
   ```
2. In the home folder of your Ubuntu, create a folder called "Repo".
3. Run a terminal in Repo and run the following code to download the github repository
   ```
   git clone https://github.com/sthnaqvi/explorer
   ```
4. Continue to run
   ```
   cd explorer
   ```
5. In the terminal, run the following command (make sure you are using node version 8.0.0, otherwise the command will not run successfully)
   ```
   npm install
   ```
6. Open the file ~/Repo/explorer/app/app.js and change 8545 in line 9 to 9001, which is the value of `--http.port` of your running peer-1.
7. Make sure you have done the first tutorial and have several blocks mined already in your private blockchain. Make sure your peer-1 is also running in another terminal/console. Then, please run
   ```
   npm start
   ```
8. If the installation is successful, we can be the following output:

```
Starting up http-webnode, serving ./app
Available on:
  http://localhost:8000
Hit CTRL-C to stop the server
```

9. Open your web browser and go to http://localhost:8000 you can see the mined blocks like this.



Activities   Firefox Web Browser   Jan 10 11:20

ETH Block Explorer

localhost:8000/#/

ETH BLOCK EXPLORER   Blockchain Info

Tx Hash, Address, or Block #   Search

Current Block: 13

ETH/USD Price: Loading..

Gas Limit: 141,324,165 m/s

Block Time: 1 second(s)

Current Diff: 131.840 K

Hashrate: Loading..

Recent Blocks Most Recent Blocks in the Ethereum Network

Block #	Block Hash	Difficulty	Miner	Size	Age	# of TXs	Gas used
13	0xab015ed...	131.840 K	0x2974c433ad308226c24b9831c7a2ca511b990fd7	0.537 kB	16 hrs 25 mins ago	0	0 m/s
12	0xc9524db...	131.776 K	0x2974c433ad308226c24b9831c7a2ca511b990fd7	0.537 kB	16 hrs 25 mins ago	0	0 m/s
11	0x7f13fff...	131.712 K	0x2974c433ad308226c24b9831c7a2ca511b990fd7	0.537 kB	16 hrs 25 mins ago	0	0 m/s
10	0x166bcd4...	131.648 K	0x2974c433ad308226c24b9831c7a2ca511b990fd7	0.537 kB	16 hrs 25 mins ago	0	0 m/s
9	0x57ab574...	131.584 K	0x2974c433ad308226c24b9831c7a2ca511b990fd7	0.537 kB	16 hrs 25 mins ago	0	0 m/s
8	0x017314e...	131.520 K	0x2974c433ad308226c24b9831c7a2ca511b990fd7	0.537 kB	16 hrs 25 mins ago	0	0 m/s
7	0xd73939d...	131.456 K	0x2974c433ad308226c24b9831c7a2ca511b990fd7	0.537 kB	16 hrs 25 mins ago	0	0 m/s
6	0x6cb4467...	131.392 K	0x2974c433ad308226c24b9831c7a2ca511b990fd7	0.537 kB	16 hrs 25 mins ago	0	0 m/s
5	0x8f2c566...	131.328 K	0x2974c433ad308226c24b9831c7a2ca511b990fd7	0.537 kB	16 hrs 25 mins ago	0	0 m/s
4	0xc7db25a...	131.264 K	0x2974c433ad308226c24b9831c7a2ca511b990fd7	0.537 kB	16 hrs 25 mins ago	0	0 m/s
3	0x480972a...	131.200 K	0x2974c433ad308226c24b9831c7a2ca511b990fd7	0.537 kB	16 hrs 25 mins ago	0	0 m/s

10.

C. Usage of `bootnode`

1. Assume we have created x peers in the private network. If we want all of them know each other, we need to call function `admin.addPeer()` for $x\,(x-1)/2$ times. It is definitely not a good solution to add them one by one, so we have the `bootnode` provided with `geth` to automatically add new peers in the network and let peers know each other.
2. There are three steps to use `bootnode`: **create it**, **run it**, and **use it**.
   a) **Create it**. Use command

   ```
   bootnode -genkey ~/ethereum/bootnode.key
   ```

   to create a bootnode.key file.
   The utility program `bootnode` should be in the same directory of `geth`. Option `-genkey` is to create an ID for a new bootnode. You can pick a different location and choose a different file name if you want by changing the last argument in the above example. A new file `bootnode.key` will be generated after running this command.
   b) **Run it**. In a new terminal, run command

```
bootnode -nodekey ~/ethereum/bootnode.key -addr :39999
```

to run the bootnode.
We need to specify the nodekey by using option `-nodekey` follows the location of the file that you have just created. The `bootnode` program has a port number for communication purposes. The port number is specified by using `-addr` option. In this example, the port number is 39999. Please note that we used a pair of double-quotes outside the port number, we also used a colon at the beginning. Running this command will create a process and we will see the following output in the terminal:

```
enode://bootnode-id@ip-address:0?discport=39999
Note: you're using cmd/bootnode, a developer tool.
We recommend using a regular node as bootstrap node for production deployments.
```

It is basically an enode with an address. In the output, the IP address of the bootnode. A running bootnode is just like a server. A new peer will communicate with the server and say “Hi, I am a new peer with ID: Blah-blah”. The server then broadcast the information of the new peer to all the existing peers in the network.

c) **Use it**. Please make sure that you have already terminated the first peer running by the previous tutorial, otherwise it would not work (as the port number is being used). The `geth` program has another option that we have not used in the previous tutorial, the `--bootnode`. To use it, we run the following command in a new terminal:

```
geth --datadir "~/ethereum/peer1" --networkid chainId --
port 12341 --http --http.port 9001 --authrpc.port 8551 --
http.corsdomain "*" --ipcpath "~/ethereum/peer1/geth1.ipc"
--bootnodes "enode://bootnode-id@ip-
address:0?discport=39999" console
```

It is the same command that we used in the previous tutorial but with a new option `--bootnodes "enode://bootnode-id@ip-address:0?discport=39999"` and without the –`nodiscover` option.

d) Then, run a second peer by using the following command in a new terminal:

```
geth --datadir "~/ethereum/peer2" --networkid chainId --
port 12342 --http --http.port 9002 --authrpc.port 8552 --
http.corsdomain "*" --ipcpath "~/ethereum/peer2/geth2.ipc"
--bootnodes "enode://bootnode-id@ip-
address:0?discport=39999" console
```

It should be noticed that both peers are using the same bootnode, so the second peer to be run can be automatically known by the first peer.

e) Now, in either of your two terminals, run command:

```
admin.peers
```

the other peer should be listed.

D. Tips: In some cases, you may see a lot of warnings WARN in the console. It is a little bit annoying to have so many warnings. So, you can open another terminal and connect to the console of the running peer-1 by using

```
geth attach ~/ethereum/peer1/geth1.ipc
```

or

```
geth attach ~/ethereum/peer2/geth2.ipc
```

for your peer-2.

E. Sending "money" from one account to another

1. Prerequisites
   - **a)** You should have at least two accounts, either on the same peer, or on different peers.
   - **b)** The sender must have more than the balance for the transaction fee.
2. From the previous tutorial, you should have an account in the first peer. This account should have sufficient amounts of "money" to send to another account, as long as the peer has mined some blocks. This is because the reward of mining a block is 2 `Ether`, which is 2e+18 `Wei`. `Wei` is the smallest unit of "money" in Ethereum. Basically, `1 Ether=1e+18 Wei`.
3. Now, you can create another account for the second peer in the console where the second peer is running by using

   ```
   personal.newAccount()
   ```

   command. Typing in your password twice and do not forget it. Also, EECE571G is suggested to be your password. The new account has a 160 bit hexadecimal value in a pair of quotation marks, e.g., `"0x0123456789abcdef0123456789abcdef01234567"`
4. Assume we want to transfer "money" from the first account on the first peer to the account on the second peer. For ease of presentation, we use HASH_ACNT_1 to represent the address of the sender account with double quotes included, and use HASH_ACNT_2 to represent the address of the receiver account with double quotes included. Note that, in the console of the first peer, we actually have `HASH_ACNT_1=eth.accounts[0]`.
5. We now use the following command to send money from HASH_ACNT_1 to HASH_ACNT_2

   ```
   eth.sendTransaction({from : HASH_ACNT_1, to : HASH_ACNT_2,
   value : 1e+18})
   ```

next

The parameter in the function `sendTransaction()` is given in JavaScript Object Notation (JSON) with three fields. The order of these three values in JSON can be shuffled.

6. Running the command probably will give you the following error.
   ```
   Error: authentication needed: password or unlock
     at web3.js:6347:37(47)
     at web3.js:5081:62(37)
     at <eval>:1:20(10)
   ```
   This is because you have to first unlock the account HASH_ACNT_1 before sending “money” out from it.
7. To unlock the account, you can run either of the two following commands in the console of first peer:

   ```
   personal.unlockAccount(HASH_ACNT_1)
   ```

   The terminal will ask you to enter the password.
   But, eventually, it says:

   ```
   GoError: Error: account unlock with HTTP access is forbidden at web3.js:6347:37(47)
     at native
     at <eval>:1:24(3)
   ```

8. In order to make the above steps work, we need to add `--allow-insecure-unlock` option when running the peer1. Now, terminate the running peer-1 and rerun it using this
   ```
   geth --datadir "~/ethereum/peer1" --networkid chainId --
   port 12341 --http --http.port 9001 --authrpc.port 8551 --
   http.corsdomain "*" --ipcpath "~/ethereum/peer1/geth1.ipc"
   --bootnodes "enode://bootnode-id@ip-
   address:0?discport=39999" --allow-insecure-unlock console
   ```

9. Please make sure there is no mining in the network. Then, repeat step 7 and step 5, you will have the follows:

   ```
   > personal.unlockAccount(HASH_ACNT_1)
   Unlock account 0x2974c433ad308226c24b9831c7a2ca511b990fd7
   Passphrase:
   true
   > eth.sendTransaction({from : HASH_ACNT_1, to : HASH_ACNT_2,
   value : 1e+18})
   "0x25e3befee9603c57d657e1541063f9787b372e82262273b36fa375d261318b
   b8"
   ```

10. The returned hash code is the transaction ID, we use HASH_TRANS_1 to represent it with both double quotes included.

11. Run command

```
eth.pendingTransactions
```

in the console of the first peer (i.e., the same console that you sent the transaction). It will list you the information about the transaction that you have just created. The transaction will not disappear from `eth.pendingTransactions` until a new block is mined and the transaction is inserted into it.

12. Now, you can start a miner by calling

```
miner.start(1)
```

in either one of the two consoles. As long as a new block is mined, the transaction information will disappear when you run `eth.pendingTransactions` again.

13. After the transaction has been mined into a block, we can run command

```
eth.getTransaction(HASH_TRANS_1)
```

to see where the transaction is logged.

14. Please also try to play with the block explorer in the webpage at http://localhost:8000/. On the page, you can go through each of the mined blocks and locate the transaction. Or, you can search HASH_TRANS_1 in the search box. You can also try to search HASH_ACNT_1 or block number (e.g., 0, 1, 12, etc).

15. Also using

```
eth.getBalance(HASH_ACNT_2)
```

command in the console of the peer-1 where the receiver's account exists to see that there should be 1e+18 Wei (which is 1 Ether) has been received by the receiver account.

F. In this tutorial, we learnt how to run blockchain explorers for your private blockchain and how to create transactions from your local wallet. In the next tutorial we will learn how to write the deploy a very simple smart contract.

G. If you have any problems when you try to reproduce the experiment, please contact me for help. Email: zwang@ece.ubc.ca

H. In the next tutorial, we will learn how to deploy a smart contact in the blockchain and call the functions defined in the smart contract.