Experiment No. 5

Aim:

Performing a transaction using Geth

Theory:

Ethereum Node:

An Ethereum node is a computer that is running the software client. The blockchain network is made up of nodes, which are the only method to access it. Nodes communicate with one another in order to validate transactions and record data about the status of the blockchain.

Types of Node:

1. Mining Node : Nodes that belong to miners.These nodes are responsible for writing all the transactions that have occurred in the Ethereum network in theblock.

2. Ethereum Virtual Machine Node : These are the nodes in the Ethereum network in which SmartContracts are implemented.

By default, this node utilizes a 30303 port number for the purposeofcommunication among themselves.

Process of Mining:

A user writes and signs a transaction request with the private key of some account.

The user broadcasts the transaction request to the entire Ethereum network from some node.

Upon hearing about the new transaction request, each node in the Ethereum network adds the request to their local mempool, a list of all transaction requests they've heard about that have not yet been committedto the blockchain in a block.

At some point, a mining node aggregates several dozen or hundredtransaction requests into a potential block, in a way that maximizes the transaction fees they earn while still staying under the block gas limit.The mining node then:

Verifies the validity of each transaction request (i.e. no one is trying to transfer ether out of an account they haven't produced a signature for, therequest is not malformed, etc.), and then executes the code of the request, altering the state of their local copy of the EVM. The miner awards the transaction fee for each such transaction request to their own account.

Begins the process of producing the proof-of-work "certificate of legitimacy" for the potential block, once all transaction requests in the block have been verified and executed on the local EVM copy.
Eventually, a miner will finish producing a certificate for a block which includes our specific transaction request. The miner then broadcasts the completed block, which includes the certificate and a checksum of the claimed new EVM state.

Other nodes hear about the new block. They verify the certificate, executeall transactionson the block themselves (including the transaction originally broadcasted by our user), and verify that the checksum of their new EVM state after the execution of all transactions matches the checksum of the state claimed by the miner's block. Only then do these nodes append this block to the tail of their blockchain, and accept the newEVM state as the canonical state.

Each node removes all transactions in the new block from their local mempool of unfulfilled transaction requests.

New nodes joining the network download all blocks in sequence, including the block containing our transaction of interest. They initialize a local EVM copy (which starts as a blank-state EVM), and then go through the process of executing every transaction in every block on top of their local EVM copy, verifying state checksums at each block along the way.

What is Geth ?

Geth(Go Ethereum) is a command line interface for running Ethereum node implemented in Go Language. Using Geth you can join Ethereum network, transfer ether between accounts or even mine ethers.

Steps to create your own private Ethereum Blockchain:

1. Download Geth and install:
   https://geth.ethereum.org/docs/install-and-build/installing-geth#install-on-windows
2. Create a directory to hold your network filesmkdir eth-chain

   cd eth-chain
3. Create your genesis filetouchGenesis.json
4. Open your genesis file and paste the following

```
{
"nonce": "0x0000000000000042",
 "mixhash": "0x0000000000000000000000000000000000000000000000000000000000000000",
"difficulty": "0x20000","alloc": {},
"coinbase": "0x0000000000000000000000000000000000000000",
"timestamp": "0x0","parentHash":
"0x0000000000000000000000000000000000000000000000000000000000000000",
"extraData": "0x", "gasLimit":
"0xffffffff","config": {"chainId": 4224,
```

```
"homesteadBlock": 0,
"eip150Block": 0,
"eip155Block": 0,
"eip158Block": 0
}
}
```

5. Initial the genesis block
   Init our blockchain with the settings in the genesis file and define a folderfor storingchain data.
   > geth --datadir "./db" init genesis.json
   datadir : Data directory for the databases and keystoreinit : initialize a new genesis block

6. Start your Ethereum peer node.

   Networkid helps ensure the privacy of your network. You can use any number here(where we used "123456"), but other peers joining your network must use the same one.

   geth --datadir "./db" --networkid 123456 --http –http.port "8545" – http.corsdomain "*" --nodiscover – http.api="admin,db,eth,debug,miner,net,shh,txpool,personal,web3", -- allow-insecure-unlock

7. Attach your terminal to geth in order to interact with blockchain

```
geth attach ipc://./pipe/geth.ipc
```

8. Perform various commands. Some of them are given below

```
admin.nodeInfo
```

Creates a new account and prints the address. On the console, use:
personal.newAccount("123456")
Note: 123456 is the passphrase.

Check accounts using eth.accounts
Accounts are in an array so you can search account by index.
`Use eth.accounts[0]`

Check balance of account:
`eth.getBalance(eth.accounts[0])`
Check balance by using web3

```
web3.fromWei(eth.getBalance(eth.accounts[0]),"ether")
```

9. Mining

a. Set Default Account

Check your default account,type

```
eth.coinbase
```

To set your default account, type

```
miner.setEtherbase(web3.eth.accounts[0])
```

b. Start mining Check your balance with

eth.getBalance(eth.coinbase)
Run Miner.start(1)

Note: 1 refer to the number of threads
Look at your other terminal window, you should see some mining action in the logs. Check yourbalance again and it should be higher.

To end mining, typeMiner.stop()

10. Transfer

a. Check your balance
eth.getBalance(eth.coinbase)

b. Unlock account

```
personal.unlockAccount("eth.accounts[0]")
```

c. Transfer Ether

```
eth.sendTransaction({from: eth.accounts[0], to:
eth.accounts[1], value: web3.toWei(1, "ether")})
```

Output:

Fig 1: Initializing genesis.json

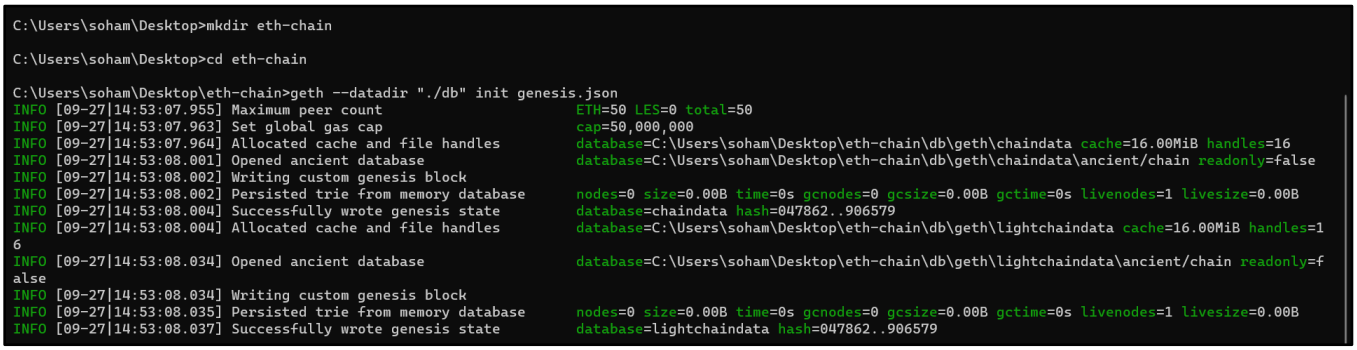

Fig 2: Staring Ethereum peer node

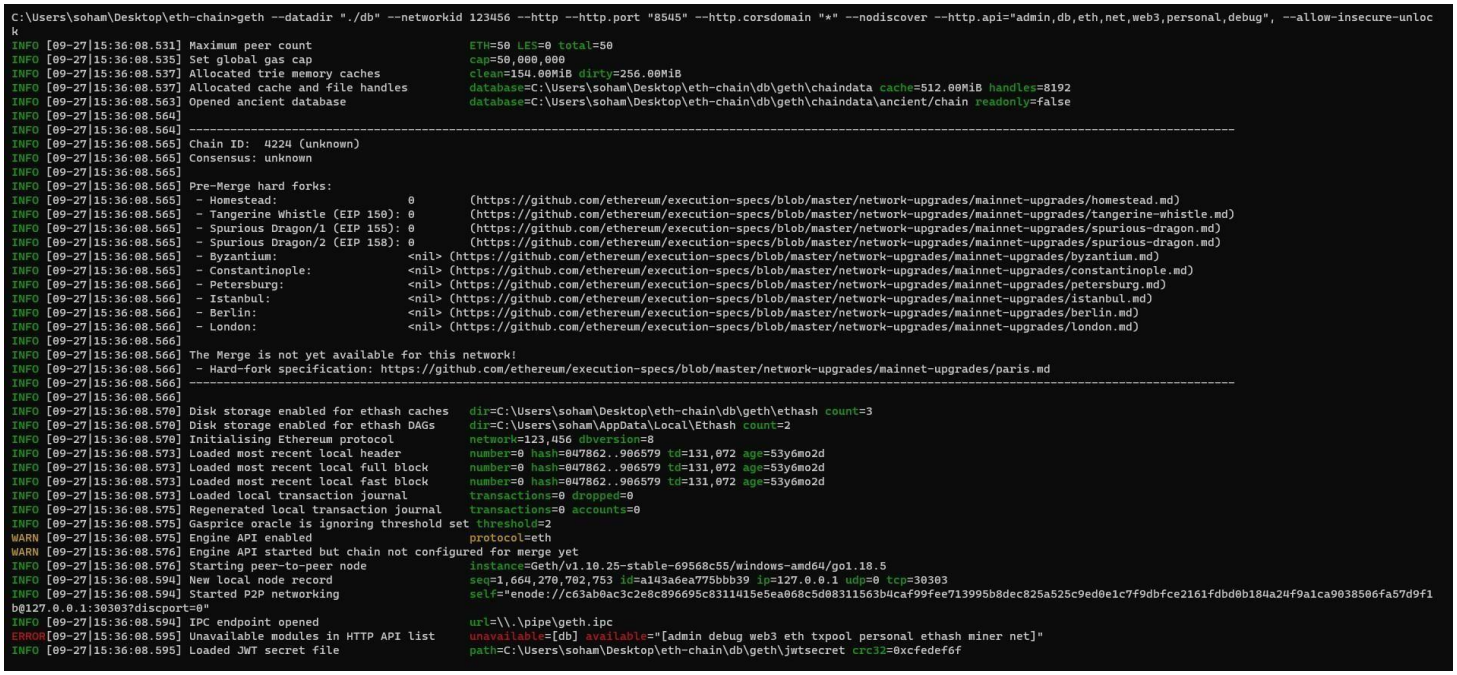

Fig 3: Attaching terminal to geth in order to interact with blockchain using ipcgeth attach ipc://./pipe/geth.ipc

Fig 4: Checking accounts, creating new accounts, starting mining process,checking account balance.

```
To exit, press ctrl-d or type exit
> eth.accounts
[]
> personal.newAccount()
Passphrase:
Repeat passphrase:
"0x3933105f51a6b7744a26fee147b06c62ea38dfa1"
> eth.accounts
["0x3933105f51a6b7744a26fee147b06c62ea38dfa1"]
> miner.setEtherbase(eth.accounts[0])
true
> miner.start(1)
null
> personal.newAccount("12345")
"0x805cb9f0962c47bc34d6d56c6b3819a2a011fb75"
> eth.accounts
["0x3933105f51a6b7744a26fee147b06c62ea38dfa1", "0x805cb9f0962c47bc34d6d56c6b3819a2a011fb75"]
> eth.getBalance(eth.accounts[0])
98000000000000000000
> web3.fromWei(eth.getBalance(eth.accounts[0]),"ether")
1090
> eth.coinbase
"0x3933105f51a6b7744a26fee147b06c62ea38dfa1"
> miner.stop()
null
> eth.getBalance(eth.accounts[1])
0
> web3.fromWei(eth.getBalance(eth.accounts[1]),"ether")
0
```

Fig 5: Mining process started (Terminal 1)

```
INFO [10-03|15:04:55.110] Updated mining threads                   threads=1
INFO [10-03|15:04:55.116] Transaction pool price threshold updated price=1,000,000,000
INFO [10-03|15:04:55.120] Commit new sealing work                  number=1 sealhash=e6d27b..912b0c uncles=0 txs=0 gas=0
 fees=0 elapsed=3.405ms
INFO [10-03|15:04:55.122] Commit new sealing work                  number=1 sealhash=e6d27b..912b0c uncles=0 txs=0 gas=0
 fees=0 elapsed=5.485ms
INFO [10-03|15:04:57.879] Generating DAG in progress               epoch=0 percentage=0 elapsed=863.991ms
INFO [10-03|15:04:58.724] Generating DAG in progress               epoch=0 percentage=1 elapsed=1.708s
INFO [10-03|15:04:59.595] Generating DAG in progress               epoch=0 percentage=2 elapsed=2.579s
INFO [10-03|15:05:00.426] Generating DAG in progress               epoch=0 percentage=3 elapsed=3.411s
INFO [10-03|15:05:01.259] Generating DAG in progress               epoch=0 percentage=4 elapsed=4.243s
INFO [10-03|15:05:02.071] Generating DAG in progress               epoch=0 percentage=5 elapsed=5.056s
INFO [10-03|15:05:02.900] Generating DAG in progress               epoch=0 percentage=6 elapsed=5.885s
INFO [10-03|15:05:03.771] Generating DAG in progress               epoch=0 percentage=7 elapsed=6.755s
INFO [10-03|15:05:04.591] Generating DAG in progress               epoch=0 percentage=8 elapsed=7.575s
INFO [10-03|15:05:05.417] Generating DAG in progress               epoch=0 percentage=9 elapsed=8.401s
INFO [10-03|15:05:06.246] Generating DAG in progress               epoch=0 percentage=10 elapsed=9.231s
INFO [10-03|15:05:07.081] Generating DAG in progress               epoch=0 percentage=11 elapsed=10.065s
INFO [10-03|15:05:07.913] Generating DAG in progress               epoch=0 percentage=12 elapsed=10.897s
INFO [10-03|15:05:08.751] Generating DAG in progress               epoch=0 percentage=13 elapsed=11.735s
INFO [10-03|15:05:09.570] Generating DAG in progress               epoch=0 percentage=14 elapsed=12.554s
INFO [10-03|15:05:10.423] Generating DAG in progress               epoch=0 percentage=15 elapsed=13.407s
INFO [10-03|15:05:11.243] Generating DAG in progress               epoch=0 percentage=16 elapsed=14.227s
INFO [10-03|15:05:12.069] Generating DAG in progress               epoch=0 percentage=17 elapsed=15.054s
INFO [10-03|15:05:12.898] Generating DAG in progress               epoch=0 percentage=18 elapsed=15.882s
INFO [10-03|15:05:13.741] Generating DAG in progress               epoch=0 percentage=19 elapsed=16.725s
INFO [10-03|15:05:14.589] Generating DAG in progress               epoch=0 percentage=20 elapsed=17.573s
INFO [10-03|15:05:15.429] Generating DAG in progress               epoch=0 percentage=21 elapsed=18.413s
```

Fig 6: Mining process details: Mined potential block and sealing of blocks(Terminal 1)

```
INFO [10-03|15:06:28.748] Successfully sealed new block           number=1 sealhash=e6d27b..912b0c hash=25cf94..706e8b
elapsed=1m33.629s
INFO [10-03|15:06:28.749] 🔨 mined potential block                 number=1 hash=25cf94..706e8b
INFO [10-03|15:06:28.753] Commit new sealing work                 number=2 sealhash=f1be22..4da5a8 uncles=0 txs=0 gas=0
 fees=0 elapsed=0s
INFO [10-03|15:06:28.753] Commit new sealing work                 number=2 sealhash=f1be22..4da5a8 uncles=0 txs=0 gas=0
 fees=0 elapsed=0s
INFO [10-03|15:06:29.301] Generating DAG in progress              epoch=1 percentage=3  elapsed=3.621s
INFO [10-03|15:06:30.186] Generating DAG in progress              epoch=1 percentage=4  elapsed=4.506s
INFO [10-03|15:06:30.358] Successfully sealed new block           number=2 sealhash=f1be22..4da5a8 hash=719e0d..dbf04f
elapsed=1.605s
INFO [10-03|15:06:30.359] 🔨 mined potential block                 number=2 hash=719e0d..dbf04f
INFO [10-03|15:06:30.360] Commit new sealing work                 number=3 sealhash=64539a..503a81 uncles=0 txs=0 gas=0
 fees=0 elapsed="143.7µs"
INFO [10-03|15:06:30.362] Commit new sealing work                 number=3 sealhash=64539a..503a81 uncles=0 txs=0 gas=0
 fees=0 elapsed=1.681ms
INFO [10-03|15:06:31.094] Generating DAG in progress              epoch=1 percentage=5  elapsed=5.414s
INFO [10-03|15:06:31.979] Generating DAG in progress              epoch=1 percentage=6  elapsed=6.299s
INFO [10-03|15:06:32.884] Generating DAG in progress              epoch=1 percentage=7  elapsed=7.204s
INFO [10-03|15:06:33.781] Generating DAG in progress              epoch=1 percentage=8  elapsed=8.101s
INFO [10-03|15:06:34.671] Generating DAG in progress              epoch=1 percentage=9  elapsed=8.991s
INFO [10-03|15:06:35.583] Generating DAG in progress              epoch=1 percentage=10 elapsed=9.903s
INFO [10-03|15:06:36.478] Generating DAG in progress              epoch=1 percentage=11 elapsed=10.798s
INFO [10-03|15:06:36.687] Successfully sealed new block           number=3 sealhash=64539a..503a81 hash=e5537b..fabcaf
elapsed=6.326s
INFO [10-03|15:06:36.688] 🔨 mined potential block                 number=3 hash=e5537b..fabcaf
INFO [10-03|15:06:36.689] Commit new sealing work                 number=4 sealhash=95ad18..5f9f1d uncles=0 txs=0 gas=0
 fees=0 elapsed=1.001ms
INFO [10-03|15:06:36.689] Commit new sealing work                 number=4 sealhash=95ad18..5f9f1d uncles=0 txs=0 gas=0
```

Fig 7: Transferring balance

```
PS C:\Users\soham> geth attach ipc://./pipe/geth.ipc
Welcome to the Geth JavaScript console!

instance: Geth/v1.10.25-stable-69568c55/windows-amd64/go1.18.5
coinbase: 0x3933105f51a6b7744a26fee147b06c62ea38dfa1
at block: 225 (Mon Oct 03 2022 15:20:44 GMT+0530 (IST))
 datadir: C:\Users\soham\Desktop\PrivateNetwork\Eth
 modules: admin:1.0 debug:1.0 engine:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 txpool:1.0 web3:1.0

To exit, press ctrl-d or type exit
> eth.accounts
["0x3933105f51a6b7744a26fee147b06c62ea38dfa1", "0x805cb9f0962c47bc34d6d56c6b3819a2a011fb75"]
> eth.getBalance(eth.accounts[0])
1.125e+21
> eth.getBalance(eth.accounts[1])
0
> web3.fromWei(eth.getBalance(eth.accounts[0]),"ether")
1125
> web3.fromWei(eth.getBalance(eth.accounts[1]),"ether")
0
> eth.sendTransaction({from: eth.accounts[0], to: eth.accounts[1], value: web3.toWei(1, "ether")})
Error: authentication needed: password or unlock
        at web3.js:6365:9(45)
        at send (web3.js:5099:62(34))
        at <eval>:1:20(21)
```

Note: Authentication is needed to transfer balance. Use
`personal.unlockAccount(eth.accounts[0])`

Observations and Findings: From this we came to know that an Ethereum is a decentralized, open-source blockchain with smart contract functionality. We have also learned about the types of nodes, process of mining and Geth.

Conclusion:

Q. How can you perform a transaction using Geth (the Go Ethereum client) from the command line, and what are the essential steps to send Ether from one account to another?