COP5615: Dist Oper Sys Princ, Fall 2018

Project 4.1

Date - 12/2/2018

**Team Members**

Sourav Dutta (9863 3443)

----------------------------

**Coding environment**

Erlang/OTP 21

IEx 1.7.2

**Linux Ubuntu 18.04** machine with 6 cores

-------------------------------------

**Part 1**

**Instructions**

The code is in directory project4 under proj4.1

cd project4

mix clean && mix compile

mix test --trace

**Functionalities implemented**

The main functions needed to mine bitcoin, maintain a wallet and perform transactions are added in this project. The details are mentioned below.

**Public ledger:** The ledger contains the blockchain, the list of transactions, the unverified transactions (all transactions since the last block was mined are added here), and the outputs which forms as inputs and outputs for the transactions. At the beginning, the ledger contains a default or Genesis block in blockchain as it is required by next blocks to get the previous hash. It also contains a default coin-base transaction which is the first transaction added to the blockchain by any miner.

**Miner:**  Each miner is a GenServer with a public address (hash of the PID). All interactions are done using the public address. The difficulty is set to get hashes with equal or more than 4 leading zeros. The function is present in lib/types/blockchain.ex with name get\_target\_bits.

**Adding blocks:** The miner takes all the unverified transactions from the ledger and starts mining a new block. It also adds a transaction which rewards the miner for the new block. The reward is set at 15 bitcoins. Integer values are used just to make it easier for human to verify. The blockchain can be verified easily to check that the hashes are correct.

**Transactions:** When a transaction is initialed, it is checked if there are unspent outputs for the sender which add up to greater than the value needed to transfer. If not, then the transaction is declined. Otherwise the list of unspent outputs for the user are calculated which add up to the amount required to transfer or is larger by a small amount. New outputs are created, one for the receiver and one for the sender if it gets back any change back from the transaction. The transaction is added as unverified transaction and waits to be verified and added to the blockchain.

**Bonus**

**Combine change:** Apart from the main functionalities, a combine change operation was added to be used by the miners. As users receive and send bitcoins it accumulates a large number of unspent outputs with small value from previous transactions. This forces new transactions to go through a large number of outputs to accumulate the amount required for transaction. This operation combines all unspent outputs of a user which are of value less than 3 bitcoins to one output.

**Test cases**

1. **test "create the shared ledger":**

Steps:

1. The test creates a common ledger.

Tests:

1. The test checks if the ledger process is alive and if the genesis block and coin-base transaction is present.
2. **test "create a miner" do**

Steps:

1. The test creates a miner process.

Tests:

1. Checks if it is alive.
2. **test “start the miner” do**

Steps:

1. The test creates a both ledger and miner client.
2. Starts the mining process for the client. The client picks up the unverified transactions and starts mining a block. Since there is no blocks present, the miner takes the coin-base transaction and adds the first block using the genesis block as the previous block.

Tests:

1. The test waits for 3 seconds and checks if there are 2 blocks present (genesis block and the new block) in the ledger.
2. It also checks if the generated blockchain is valid.
3. **test "test unspent outputs" do**

Steps:

1. The test creates follows the same steps as the previous test.

Tests:

1. The test waits for 3 seconds and checks if there are 2 blocks present (genesis block and the new block) in the ledger.
2. It checks if the balance of the miner is 15 which is the reward given to the miner who adds a new block.
3. **test "test two blockchain updates" do**

Steps:

1. The test starts two miners, and both start mining the first block.
2. The first one to mine the block gets the reward and the other miner is unsuccessful.

Tests:

1. The test asserts the blockchain length and unverified transactions.
2. It checks and asserts the balance of both the miners.
3. **test "test transaction failure and success" do**

Steps:

1. The test creates follows the same steps as the previous test.
2. Then it transfers an amount from one user to other. It tries two scenarios, first when the amount to be transferred is more that the unspent amount that the sender has and second when the amount is less that the unspent amount present with the user.

Tests:

1. The test checks the final balance of both the miners before and after the successful and failed transactions.
2. **test "test combine change" do**

Steps:

1. The test creates two miners and mines the starting block. One of them gets a reward.
2. User 1 sends multiple transactions of values 5, 2, 1, 1, 1. The total amount that User 2 has is 10 but has 5 outputs associated for each of the amounts.
3. After this the combine change operation is called the output values with amount less than 3 are combined. So now User 2 has tow outputs of values 5 and 5.

Tests:

1. The test checks the final balance of both the miners before and after the set of transactions.
2. It also checks the balance before and after the combine change operation.
3. The test also verifies the number of unspent outputs associated with the User 2 after the combine change.