Assignment 2 - Report

May 18, 2022

1 P2P Blockchain System

A P2P Blockchain System consists of three core components: server, client and miner. All three components form a peer, which interacts with other peers in a fully connected network (in this implementation). That is, all peers have knowledge of every other peer in the network due to configuration files which list all peers in the network topology.

1.1 Server

A server in a P2P Blockchain System stores a copy of a blockchain. It synchronizes with other peers by listening over a socket for when a peer receives a new transaction. When receiving a new transaction, the server first validates the transaction, and if the transaction is valid it then adds the transaction to its transaction pool. Since a server contains a copy of a blockchain, it contains a pool of transactions which could be included in a block, given that all transactions are valid. Usually, a pool has a maximum capacity, after which is reached a server will not add any transactions to. Furthermore, a server recieves a proof from a miner; and if the proof is valid, the server creates a current block using the hash of the previous, the proof and all transactions in the transaction pool.

1.2 Client

A client in a P2P Blockchain System receives input from a user and relays such input to its associated server. Some examples of user input are for a user to attempt to commit a transaction to a blockchain; or a users client to print the contents of a blockchain.

1.3 Miner

A miner in a P2P Blockchain System continuously runs a mining algorithm. There are is a variety of mining algorithms such as Ethash, Equihash, and Scrypt which all use the SHA256 cryptographic hashing function. Usually, the aim of a miner is to guess a number within a space which is below a threshold value and for the network to define a threshold such that a correct guess is computationally expensive. Some more usual characteristics of a miner is to adjust the difficulty of guessing a correct number by adjusting the difficulty of guessing correct according to a moving average of the total hashing power across all peers. That is, as the aggregate computational power of the network increases or decreases, the size of the space decreases or increases; respectively.

2 Techniques

To implement the given specification, VSCode and WSL were used simultaneously. Each time a function was implemented, the function would be tested, then in a logical sense, the next function would be implemented. The system was built-out by reading the assignment specification, then implementing all classes (Blockchain.py, BlockchainServer.py, ... etc). Such building was haphazard and could have been executed in any order up to getting peers to exchange blocks and transactions. Lastly, this implementation makes use of Python threads, mutexes and sockets.

3 Methodology

The assignment specification was read, with key deliverables kept in mind, then the system was built-out from starting at peer to peer communication. Each time there was any ambiguity between what came to my mind while implementing and any targets to execute, the assignment specification was consulted.

4 Simulations of Testing Criteria

To minimize the time to find a valid proof in this blockchain system, lines 38 and 39 in BlockchainMiner.py were changed to

```
hexHash = self.calculateHash(_input)[:6]
    if(hexHash == "000000"):
```

to slow down the computation of proofs which allows easier testing.

The steps executed to simulate testing of all criteria were as follows:

1. Start peer A, add tx|Bob|100 and let peer A mine second block
2. Start peer B, print blockchain at startup then wait for it to sync to peer A
3. Execute 'pb' in peer B and observe that tx|Bob|100 now appears in block with index 2
4. Add tx|Alice|100 in peer B
5. Observe that peer A adds tx|Alice|100 to its pool, then finds proof before peer B
6. Start peer C and execute 'pb'
7. Wait for peer C to synchronize, then execute 'pb' again in peer C

After executing all steps, peers A, B and C have the same blockchain.

An accompanying screenshot after executing all of the above steps is given below where the top-left quadrant corresponds to peer A and the rest flows from there: