Assignment 3

Due: 11:55 pm 31 August 2023
Total Mark: 17 (17% of Final Mark)

General Instructions: Please read the following instructions carefully.

Create a simple **blockchain system** simulating proof-of-work between two parties, Alice and Bob. Use the blockchain system given in Lab 4 and create blocks by competing with each other using proof-of-work.

1. Both players have an initial block and series of transactions in a genesis block given in Lab 4 to start their play.
2. When the system is initialized, both Alice and Bob compete with each other to add a new block in the system using proof-of-work. Therefore, they compute <Nounce> of the new block which satisfies the following condition:

$$SHA256(<New\ block>) < 2^{236}$$

3. To make Alice and Bob's mining time different. Let Alice and Bob search a valid <Nonce> value in a new block from 0 and 1,000,000,000, respectively and increase it by 1.
4. If one finds a valid <Nonce> value for a new block, it broadcasts the block (or necessary information to form the block) using the PubNub system. The other party then verifies the received block in a way discussed in Task 8 of Lab 4 Note. If the received block is valid, it stops the current mining process and starts mining of the next block.

[Alternative] Stopping a process in the middle computation needs some advanced coding skills to implement and manage threads. If you are not familiar with thread computing, you can implement that the receiver waits until the current mining is finished and send a completion message to the sender. Therefore, both parties start the next block generation if the mining of the current block for both parties finishes. It should be noted that only the block generated first is considered a valid block. The block created later (from the receiver) will be discarded.

[Alternative2] The above two methods require multiprocessing implementation using threads computing. The second alternative makes the programming a bit easier. In each block generation, only one party generates a new block and the other party will just wait until it received the information of the new block. Therefore, they do not compete with each other.

1. Both players have an initial block and series of transactions in a genesis block given in Lab 4 to start their play.
2. Both Alice and Bob add a new block in the system using proof-of-work. Therefore, they compute <Nounce> of the new block which satisfies the following condition:

$$SHA256(<New\ block>) < 2^{236}$$

3. *There are 11 transaction records in lab4_tr.py. Alice generates the 1^{st}, 3^{rd}, 5^{th}, 7^{th}, 9^{th} and 11^{th} blocks. Bob generates the 2^{nd}, 4^{th}, 6^{th}, 8^{th} and 10^{th} blocks. So, they take a turn to generate a new block rather than compete with each other. Alice starts generating the first block using the genesis block once the system is initialized.*
4. If one finds a valid <Nonce> value for a new block, it broadcasts the block (or necessary information to form the block) using the PubNub system. The other party then verifies the received block in a way discussed in Task 8 of Lab 4 Note. If the received block is valid, it starts mining the next block.
5. Let Alice and Bob search a valid <Nonce> value in a new block from 0 and 1,000,000,000, respectively and increase it by 1.

*The total mark you can get from this alternative 2 is 15. 2 marks will be deducted in "*The other basic requirements of the system (e.g., block generation)" as it is relatively easy compared to the first two methods in this specification.

Marks

- Implementation of the proof-of-work process. (3 marks)
- Block validation process implementation. (2 marks)
- The other basic requirements of the system (e.g., block generation) (5 marks)
- Implementation of the blocks exchange protocol (PubNub) (5 marks)
- Report (2 marks)

Submission

Make a folder named `Assignment3 and` include

- Programs for Alice and Bob.
- The history of one of the completed games (e.g., block0.txt, block1.txt, …, and block10.txt)
- Report explaining the requirements to execute your program and the expected outcome of your program.

Compress the `Assignment3` folder using a zip program to create `yourSurname_Assignment3.zip`.

Use Moodle to upload your zip file.