Cryptocurrencies (WS2024/25) Dr. Zeta Avarikioti, Prof. Matteo Maffei

Kerma - Task 5

1 General Notes

In your group of up to three students, you may use any programming language of your choice to implement the following task. We provide skeleton projects in Python and TypeScript (see `python-skeleton-for-task-5.tgz` and `typescript-skeleton-for-task-5.tgz` on TUWEL). You can use them to build upon. After the deadline, these skeleton projects will be updated and act as sample solutions for the previous tasks. These are guaranteed to pass all test cases for the previous tasks.

2 High level Overview

After completing this task, your node should be able to:

- maintain a mempool.

Please also be sure to read the respective sections in the Protocol Description. At the end of this task, your node should implement the complete protocol.

3 Mempool UTXO

In this exercise, you will maintain a mempool and update it based on new transactions and blocks.

1. Implement a data structure for the mempool. You should maintain a list of transaction ids in the mempool and also maintain the required state that allows you to update your mempool when you receive new transactions and blocks.

2. Initialize the mempool state by setting the mempool UTXO to the UTXO after your chain-tip block.

3. Send a `getmempool` message immediately after sending the hello message.

4. On receiving a `mempool` message, request from peers the transactions corresponding to the `txids` in the `mempool` message using `getobject` messages if they are unknown to you.

5. Listen for transactions as they are gossiped on the network. If a valid transaction can be added to your mempool, add it to your mempool and update the mempool state.

6. When a new block arrives that is added to your longest chain, update your mempool by removing transactions that are already included in the block, or are now invalid. Update your mempool state.

7. Deal with mempool updates when your longest chain reorgs. More information can be found in the Protocol Description.

4 Sample Test Cases

IMPORTANT: Make sure that your node is running at all times! Therefore, make sure that there are no bugs that crash your node. If our automatic grading script can not connect to your node, you will not receive any credit. Taking enough time to test your node will help you ensure this. Consider two nodes Grader 1 and Grader2.

- Reset your database before submitting for grading. With the exception of the Genesis block, your node should not know any objects.
- Grader 1 sends a number of blocks and transaction objects.
- Grader 1 sends a `getmempool` and `getchaintip` message to obtain your mempool and longest chain.
  - The mempool must be valid with respect to the UTXO state after the chain.
  - Grader 1 sends a transaction that is valid with respect to the mempool state. Grader 1 again sends a `getmempool` message, and this time the mempool should contain the sent transaction.
  - Grader 1 sends a transaction that is **invalid** with respect to the mempool state. Grader 1 again sends a `getmempool` message, and this time the mempool **should not** contain the sent transaction.
  - Grader 1 sends a coinbase transaction. Grader 1 again sends a `getmempool` message and this time the mempool **should not** contain the sent transaction.
  - Grader 1 will send a longer chain (causing a reorg) and then send a `getmempool` message. The received mempool must be consistent with the new chain:
    - It must not contain any transactions that are already in the new chain or are invalid with respect to the new chain UTXO state.
    - It must also contain transactions that were in the old chain but are not in the new chain and valid with respect to the new chain UTXO state.

How to Submit your Solutions We will grade your solution locally. Please upload your submission as a tar archive. When grading your solution, we will perform the following steps:

```
tar -xf <your submission file> -C <grading directory>
cd <grading directory>
docker-compose build
docker-compose up -d
```

```
# grader connects to localhost port 18018 and runs test cases
docker-compose down -v
```

When started in this way, there should be neither blocks (except the genesis block) nor transactions in the node's storage.
If you use the skeleton templates, you can use the provided Makefile targets to build and check your solution. Note however, that this will only check if the container can be built, started and a connection can be established to localhost:18018. We highly recommend that you write your own test cases.
The deadline for this task is 14th January, 11.59pm. We will not accept any submissions after that. Each group member has to submit the solution individually. Plagiarism is unacceptable. If you are caught handing in someone else's code, you will receive zero points.