CS 601.441/641 – Blockchains and Cryptocurrencies

Assignment 2: Programming Portion

Instructor: Abhishek Jain

Deadline: March 23, 2021, 11:59 pm

Introduction In the last lecture we discussed a number of mining strategies. In this project you will implement three strategies discussed in class.

Getting started

1. Download the startup code and import it into your favorite IDE. You can use maven to download the required dependencies.
2. Familiarize yourself with the starter code. You should especially look at the `Miner` interface. All your miners should implement this interface. Look at the JavaDoc that explains what each function does. Additionally we provide you with an implementation of a `CompliantMiner`
3. Your goal in all parts is to maximize your profit relative to the other miners.
4. The class `MiningSimulation` provides a set of tests that your Miners should pass. The revenue goals should be achievable with standard implementations but feel free to try to surpass them. Especially if you plan on entering the class competition.
5. To run the tests, you can run the class `MiningSimulation` as a JUnit test.
6. The `BitcoinNetwork` class maybe useful to understand how the mining process is simulated.

Simulation Model The simulation uses a simplified model of the real Bitcoin network world to simulate the mining and propagation of blocks. A fixed number of blocks will be simulated. You do not need to understand all the details of the simulation but the most important points are

1. The simulation proceeds in discrete iterations where each iteration consists of a mining round and propagation round.
2. Each miner draws a creation time from $Exp[hashRate]$ and the miner with the lowest creation time mines a block. The probability of mining a block in each iteration for a miner i is, thus:
$$\frac{hashRate_i}{\sum_{j=1}^{n} hashRate_j}$$
3. With small probability a second block gets mined at the same time (by the miner who drew the second smallest number). Both blocks will be propagated at the same time. With even smaller probability a third block is mined and so on.
4. If a miner s wants to broadcast a block he draws transmission times for each other miner r from $Exp[connectivity_s * connectivity_r]$. If two miners s_1 and s_2 broadcast a block to r at the same time then the probability of s_1s block arriving first is:
$$\frac{connectivity_{s_1}}{connectivity_{s_1} + connectivity_{s_2}}$$

5. If a miner upon receiving a block wants to broadcast a new block, they draw a set of transmission times for the other agents as in 4. but delayed by the current time. Consequently there exists a first mover advantage in block propagation.

6. The simulation ends at randomized times, using an exponential distribution.

Submission Please create a .tar or .zip file of your submission. The tar or zip file should contain only the code for your miners: `MajorityMiner.java`, `SelfishMiner.java` and `FeeSnipingMiner.java`. Submit your code via Gradescope.

We will be testing your miners against the unit tests in the original starter code and some additional unit tests. If you modified any of the starter code, be sure that the miners still work in the original environment.

Exercises

1. Create a "majority miner" called `MajorityMiner` (by extending the `Miner` interface) that performs a 51% attack if it is capable. A 51% in this context means extracting as much relative profit as possible. The network may have some natural churn, so your status as a majority miner may be changing.

2. Create a "selfish miner" called `SelfishMiner` that performs a temporary block withholding attack if profitable. This strategy is outlined in Chapter 5 (Section 5.5) of the NBFMG textbook.

3. Create a "fee sniping miner" called `FeeSnipingMiner` that forks to try stealing unusually valuable blocks when profitable. That is, when a block with an unusually large transaction fee is mined by a competitor, your miner should temporarily reject that block and try to re-mine a longer fork where it keeps the large transaction fee for itself.