CS765 Project Part-2 (HW2)

Simulating a double selfish mining attack using the P2P Cryptocurrency Network developed in the last assignment

Due Date: 23:59 hrs, March 22, 2024

Note: Marks will be awarded for the points mentioned within the text. The marks have been indicated in square brackets (e.g. [1]).

In this assignment you will have to simulate a “double” selfish-mining attack on top the *discrete-event* simulator for a P2P cryptocurrency network, you built in the last assignment. This assignment can be done in groups consisting of **at most 3** persons.

You can use any programming language of your choice. If you include code from a publicly available source or generative AI (such as ChatGPT), then state the source in the comments of the code. **Not more than 20% of the code should be taken from such sources.**

Double Selfish Mining Attack This is the same attack which was proposed by Eyal and Sirer in the paper ”Majority is not Enough” (4th link mentioned in ”Useful links”), **with one main difference**. In this attack there are two such selfish miners who do not collaborate. Each behaves as if he is the only selfish miner in the system, unaware about the presence of the other attacker.

Honest miner: Each honest node mines on the longest chain visible to it. Tie breaks are resolved according to bitcoin rules, that is in case it sees multiple such longest chains, it mines on the first of these which it sees.

Selfish miner: Each selfish miner considers the longest visible chain (LVC) to him (excluding his current private blocks) as if it were the ”honest chain”, and tries to selfishly mine as in Eyal and Sirer. Note that the LVC may contain blocks released by the other selfish miner, but this selfish miner thinks that all other miners are honest.

Steps that any particular selfish miner follows are given below. The attack begins with the selfish miner trying to generate a secret chain on the genesis block. The rules for releasing secret blocks or starting a new attack on another block (that is, creating a new secret chain starting from another publicly visible block) are similar to the paper and described below.

1. If the lead of the selfish miner is 1 block over the LVC, and the lead now becomes zero, then the selfish miner immediately broadcasts the block he has mined secretly. The selfish miner continues to mine on top of his own block. If the LVC increases further in length, the selfish miner starts a new attack, starting from the last block of the LVC. However, if instead the selfish miner mines a block then he releases it immediately and starts a new attack starting from the block that he just released. This is similar to jumping from state 0’ to state 0 as discussed in class.

2. If the lead of the selfish miner over the LVC is 2 blocks and then one block gets added to the LVC, then the selfish miner broadcasts immediately all the blocks he has mined secretly. He starts a new attack on the last block of the longest chain visible to him.

3. If the lead of the selfish miner over the LVC is greater than 2, as soon as the LVC increases in length by 1 block, then the selfish miner makes public one more block, releasing a subchain that ends with that block which enters into competition with the new block at the end of the LVC. The selfish miner keeps mining on top of his secret chain.

4. At any point in time, if the LVC exceeds the length of the selfish miner’s private chain, then he starts a new attack on the last block of the longest chain visible to him.

5. The selfish miner does not forward blocks generated by other nodes. This is so that his own blocks propagate faster than blocks generated by others.

Marks:
Correct implementation of each step listed above. [15, 7, 8,3,2]

Simulate latencies L_{ij} between pairs of peers i and j connected by a link. Latency is the time between which a message m was transmitted from sender i and received by another node j. Choose the latency to be of the form $\rho_{ij} + |m|/c_{ij} + d_{ij}$, where ρ_{ij} is a positive minimum value corresponding to speed of light propagation delay, $|m|$ denotes the length of the message in bits, c_{ij} is the link speed between i and j in bits per second, and d_{ij} is the queuing delay at node i to forward the message to node j. d_{ij} is randomly chosen from an exponential distribution with some mean $96kbits/c_{i,j}$. Note that $d_{i,j}$ must be randomly chosen for *each message* transmitted from i to j. ρ_{ij} can be chosen from a uniform distribution between 10ms and 500ms at the start of the simulation. c_{ij} is set to 100 Mbps if both i and j are fast, and 5 Mbps if either of the nodes is slow. **Note that 50% of the honest nodes in the network are slow while an adversaries will always be fast.**

Experiment with choosing different values for different parameters (n, T_{tx}, mean of T_k, mining power of the two adversaries ζ_1 and ζ_2). **For simplicity you can consider $n = 100$ and $T_{tx} = 10$ ms, $\zeta_1 = 30\%, 40\%, 50\%$ and $\zeta_2 = 0\%, 30\%$.** You can try other combinations of ζ_1 and ζ_2 as well. At the demo time, you are expected to run the experiment for a specific parameter configuration. By running the experiment, find the following ratios (1). Furthermore, find the effect of the different adversary mining powers on these ratios.

Calculate these for each peer at each run

$$MPU_{node_{adv}} = \frac{\text{Number of block mined by an adversary in final public main chain}}{\text{Total number of blocks mined by this adversary overall}} \tag{1}$$

$$MPU_{node_{overall}} = \frac{\text{Number of block in the final public main chain}}{\text{Total number of blocks generated across all the nodes}} \tag{2}$$

Also, study how the fraction of the first attacker's blocks vary in the main chain with different parameter values. Draw a graph of this quantity with ζ_1 on the x-axis. Draw one such graph for each value of ζ_2.

Use an appropriate visualization tool to study the blockchain tree (suitable choices can be gnuplot, matlab, or other visualization tools). We expect you to show the blockchain tree at an adversary and one honest node using a visualization tool. You do not need to include the blockchain tree in the report, but a TA will ask you to show the tree at the demo time while running the experiment.

Marks:

Proper study using different parameters and use of a particular visualization tool to show the blockchain tree. [12]

Insight and critique of the observed values. [8]

In your submission on Moodle, submit a single zip file (filename format: RollNo1_RollNo2_RollNo3.zip) containing:

1. Source code for simulator. You need not submit any code for the visualization tool.
2. README file with instructions for compiling and running.
4. A report detailing your findings along with pictures of typical blockchain trees and appropriate insight.

Marks: Proper commenting of code [6], Design document [8], README file [6]

Useful links:

`https://people.orie.cornell.edu/mru8/orie3120/lec/lec10.pdf`
`http://cs.baylor.edu/~maurer/aida/desauto/chapter3.pdf`
`https://www.cs.cmu.edu/~music/cmsip/readings/intro-discrete-event-sim.html`
Majority is not enough - `https://arxiv.org/abs/1311.0243`