CS765 : PROJECT PART 2

Simulation of Double Selfish Mining Using P2P Cryptocurrency Network

Submitted On :

Tuesday, March 26, 2024

Submitted By :

Swetha M (23M0756)

Chaitra Gurjar (23M0831)

Index

1 Properties of the P2P Network

1) There are two independent selfish miners in the network, unaware of each other's intentions. Remaining miners are completely honest.

2) Honest miners follow the bitcoin rules and mine of the longest chain visible with the usual fork resolutions.

3) Latency between two peers depends on three parameters as : $\varrho_{ij} + |m|/c_{ij} + d_{ij}$ where

ϱ_{ij}	: speed of light propagation delay
$\|m\|/c_{ij}$	: length of message / link speed between the peers
d_{ij}	: random value from exponential distribution with mean 96 kbits/c_{ij}

4) Half of the honest miners have slow link speed and the rest have fast link speed, while the selfish miners are always fast.

5) We fix the fraction of hashing power of the selfish miners and assume that the remaining honest miners have equal fraction of hashing power, such that the sum of all is one.

2 Simulation of Selfish Mining

1) The fraction of hashing power of the two selfish miners is chosen from $\zeta_1 = \{0.3, 0.4, 0.5\}$ and $\zeta_2 = \{0.0, 0.3\}$.

2) The selfish miner starts attacking from the genesis block by forming a secret chain. The honest miners also start mining on the genesis block.

3) We will define lead of the selfish miner (L) = lead of the miner's secret chain over the longest visible chain.

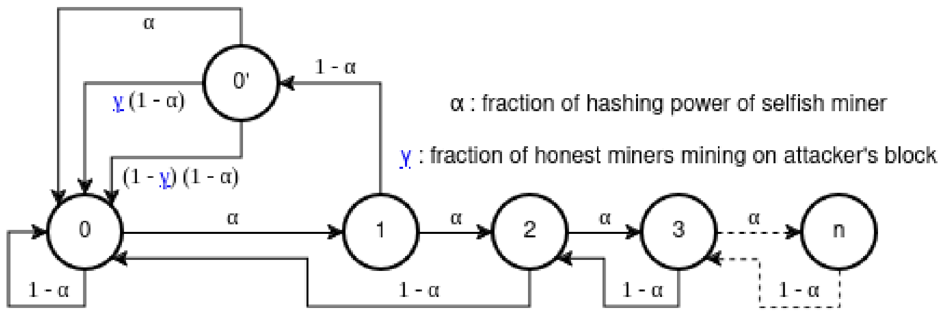
4) If $L > 2$ the selfish miner is in State > 2 in the given figure. (S)He releases only one block to the longest visible chain, when the chain increases by one block. (S)He keeps mining on the secret chain and goes to State = State − 1.

5) If $L = 2$ the selfish miner is in State = 2 in the given figure. (S)He releases both the blocks to the longest visible chain, when the chain increases by one block. (S)He continues the attack on the last block of the longest visible chain and goes to State = 0.

6) If $L = 1$ the selfish miner is in State = 1 in the given figure. (S)He releases the block to the longest visible chain, when the chain increases by one block. It is now important that the selfish miner keeps mining on its own block and then continues the attack on the new longest visible chain. The miner goes to State = 0'.

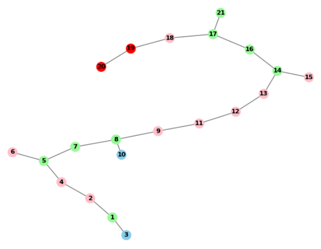
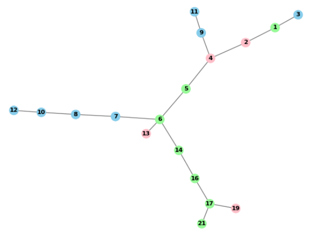
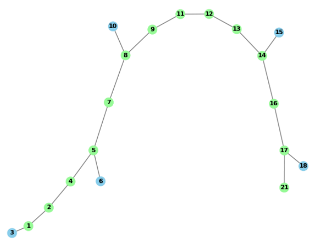
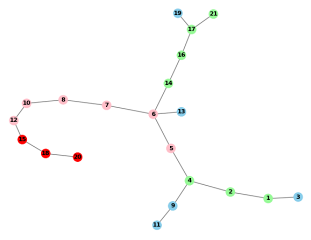
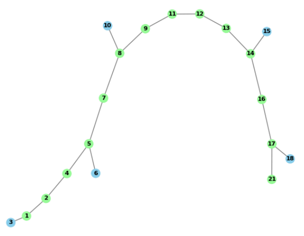
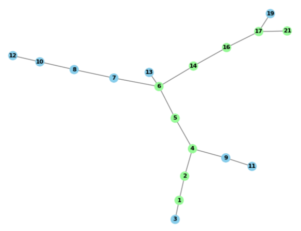
7) At any point, if the length of the longest visible chain becomes more than the secret chain, the selfish miner switches the attack on the latest block in the chain.

8) The selfish miner does not forward the blocks generated by the honest miners.

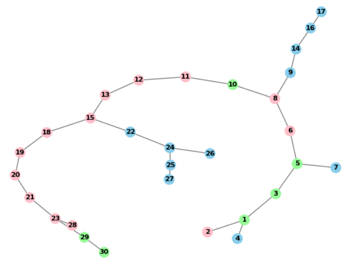
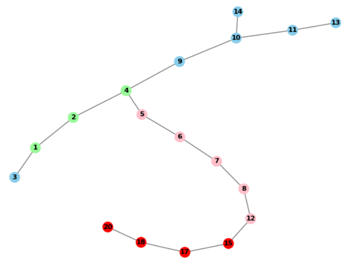
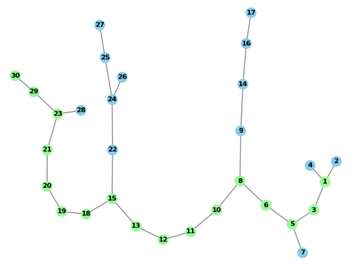
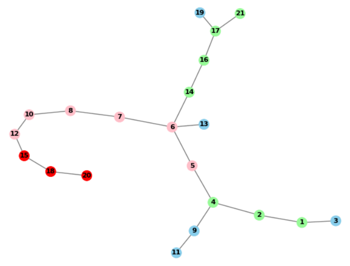
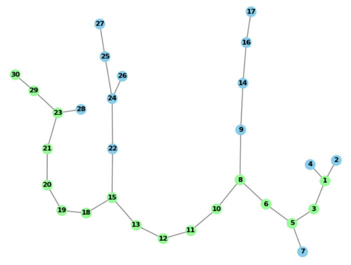
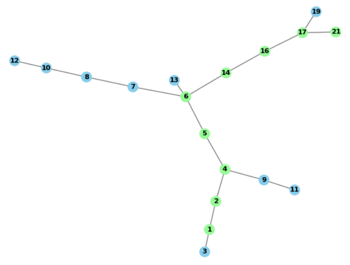

α
0'
1 - α
γ (1 - α)
α : fraction of hashing power of selfish miner
γ : fraction of honest miners mining on attacker's block
(1 - γ) (1 - α)
0
α
1
α
2
α
3
α
n
1 - α
1 - α
1 - α
1 - α

3 Changing Parameters

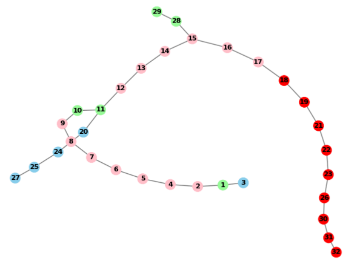
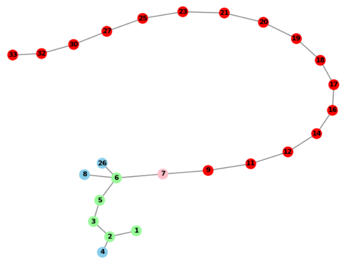
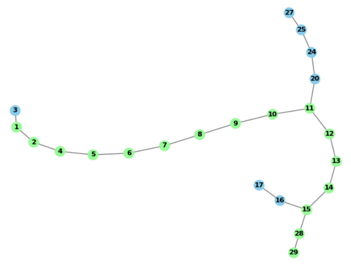
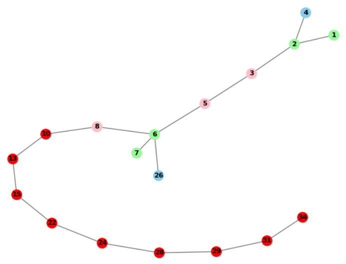
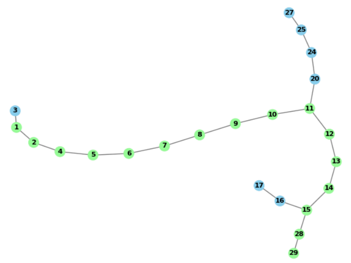
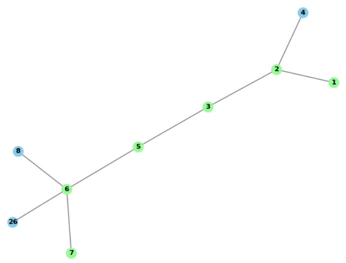
3.1 $\zeta_1 = 30\%$ for $\zeta_2 = 0\%, 30\%$ while n=100, Ttx =10s, I=30s, SimTime=600

	ζ_2 = 0%	ζ_2 = 30%
Blocks created by selfish adversary 1	11	4
Blocks created by selfish adversary 2	0	9
Length of the longest chain for honest miner	14	9
Total blocks created by all miners	20	20
$MPU_{adversary1}$	0.43	0.22
$MPU_{adversary2}$	0.00	0.22
$MPU_{overall}$	0.70	0.45
Selfish Adversary 1 **Legend**: *Green* - *Honest visible chain,* *Pink* - *Created by node and released* *Red* - *Created by node and not yet released* *Blue* - *Created by other nodes*		
Selfish Adversary 2		
Honest Miner		

3.2 $\zeta_1 = 40\%$ for $\zeta_2 = 0\%, 30\%$ while n=100, Ttx =10s, I=30s, SimTime=600

	ζ_2 = 0%	ζ_2 = 30%
Blocks created by selfish adversary 1	13	9
Blocks created by selfish adversary 2	0	6
Length of the longest chain for honest miner	17	8
Total blocks created by all miners	29	19
$MPU_{adversary1}$	0.65	0.63
$MPU_{adversary2}$	0.00	0.25
$MPU_{overall}$	0.59	0.42
Selfish Adversary 1 **Legend**: *Green* - *Honest visible chain,* *Pink* - *Created by node and released* *Red* - *Created by node and not yet released* *Blue* - *Created by other nodes*		
Selfish Adversary 2		
Honest Miner		

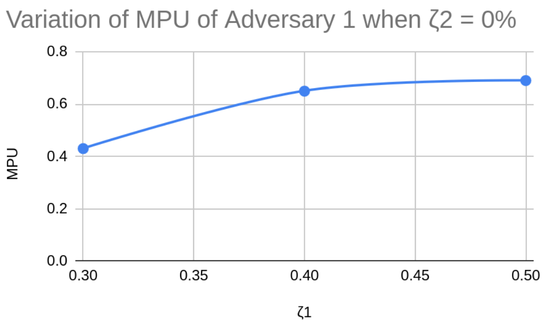
3.3 $\zeta_1 = 50\%$ for $\zeta_2 = 0\%, 30\%$ while n=100, Ttx =10s, I=30s, SimTime=600

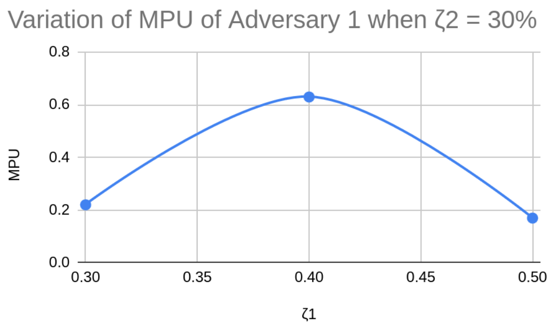
	ζ_2 = 0%	ζ_2 = 30%
Blocks created by selfish adversary 1	22	17
Blocks created by selfish adversary 2	0	12
Length of the longest chain for honest miner	16	6
Total blocks created by all miners	31	33
$MPU_{adversary1}$	0.69	0.17
$MPU_{adversary2}$	0.00	0.33
$MPU_{overall}$	0.52	0.18
Selfish Adversary 1 **Legend**: *Green* - *Honest visible chain,* *Pink* - *Created by node and released* *Red* - *Created by node and not yet released* *Blue* - *Created by other nodes*		
Selfish Adversary 2		
Honest Miner		

ANALYSIS

The above tables are summarized for the parameters n=100, Ttx =10s, I=30s, SimTime=600 when adversary2 has 0% and 30% hashing power. The observation is that

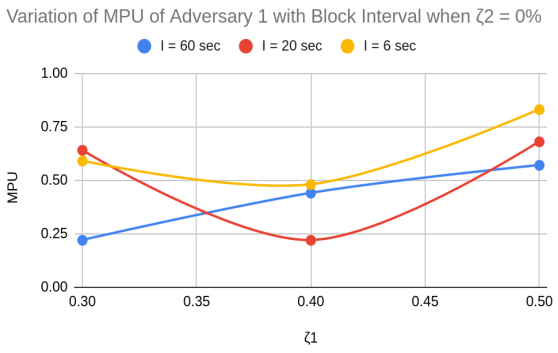
- The MPU of adversary 1 is ***higher when there is no other selfish miner*** with a considerable hashing power. This is because there are greater chances for the selfish miner's blocks to make it to the longest chain.
- Also observed that when the private chain lead is high (>2), the honest miners haven't yet seen the blocks of adversary 1. **MPU is less** because only a small fraction of the created blocks have been released. In spite of this, the adversary is at an advantage and will win when the lead eventually gets to 2. This leads to orphaning of a very high number of honest blocks in the future. ***Table 3.3 Case 2*** explains this scenario.

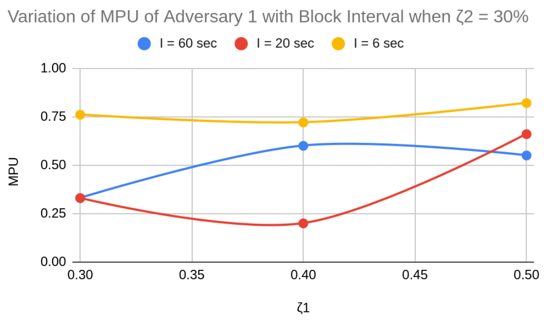




Changing the number of blocks created by tweaking Block Interval

- The following table shows the MPU variations when Block Interval is varied between 6s, 20s and 60s. The general trend is that **higher the number of blocks created (lower Block Interval) higher the MPU**. This gives enough time for selfish miners to orphan multiple other nodes. In a real blockchain network where the PoW mining target threshold is modified, this gives an even higher advantage to the selfish miner.
- When there are **less blocks created (higher Block Interval)**, *most selfish blocks are not part of the longest honest chain.* They are released later than an honest block of the same chain length.





CONCLUSION:

The influence of the selfish miners is studied in detail. The forks formed are really high and *a lot of PoW is wasted* because of orphaning of blocks. The second adversary sometimes *inversely affects* the first adversary. In most cases when the hashing power is 40% to 50%, the *private chain is much longer* than the visible chain and each time an honest block is released to blockchain, a selfish block is also released. The attack is sometimes slow and the selfish adversary wins only after a *considerable amount of blocks are created.*

Appendix

Some implementation details are as follows.

1. This part of the project is implemented in **Python**. It provides classes and objects to easily deal with entities such as transactions, peers, links and trees. It also provides some standard libraries like ‘random’ which is helpful to sample from various distributions.
2. **Inheritance of Selfish Miner class from Peer Class**: The functions of a selfish miner are almost same as that of a normal peer. It has extra attributes like a private chain, a flag to see if it is in the zero dash state of the “Majority is not enough” paper state diagram. Some functions which are enhanced from peer are adding, removing from private chain and checking length of the longest visible chain (excluding private chain).
3. **SimPy** : SimPy is a python library used to simulate events. The function call ‘env.process’ schedules events in the simulation's event queue, specifying when actions associated with the process should occur.
4. **NetworkX Graph** : In NetworkX, nodes can be any hashable object e.g. a text string, an image, an XML object, another Graph, a customized node object, etc. We noticed that sometimes, the creation of graphs using this library can lead to flooding of resources, which leads to a pause in the execution. In such cases, forceful termination and restart reloads the correct execution.
5. **matplotlib** : Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. We have used this to visualize the network and peer trees containing blocks.