University of Pisa
Department of Informatics

Assignment
Peer to Peer System Course

TyniCoin Simulation

Tri Hong Nguyen - 553719

Teacher
Prof. Laura Ricci.

November 30, 2017

Table of Contents

1 Introduction . . . 2
2 Selfish Miner . . . 2
3 Tinycoin Simulation . . . 4
3.1 The Interaction of PeerSim in Tinycoin Simulation . . . 4
Configuration File . . . 4
Tinycoin Initialization . . . 5
Oracle . . . 5
Tinycoin Protocol . . . 5
Tinycoin Observer . . . 6
Weighted Transport in Network . . . 6
3.2 Block Chain Structure in Tinycoin . . . 6
Transaction . . . 6
Block . . . 6
Block Chain Structure . . . 6
Adding a New Block . . . 8
4 Experiments . . . 8
5 Conclusion . . . 9
6 Ackowledgements . . . 9

Abstract. This assignment is about "TinyCoin: Simulating fraudolent mining strategies in a simplified Bitcoin Network" (the final project in Peer to peer systems course at University of Pisa). In this project, the simulation of TinyCoin is implemented with two different strategies (normal and selfish [1] ways). The Aim of the project is to find the number of fork and the total time need to solve it. Because of the variety of the information, the project is experimented with different parameters such as: the percentage of selfish miners in total of honest miner created randomly, the power of selfish miners, the latency between two nodes and the latency of the network. However, other parameters also can need to consider: the number of nodes in the network, the degree or the connections between nodes.

1 Introduction

With the development of technology, nowadays there are a lot of cryptocurrency and one of the most popular cryptocurrency is Bitcoin. In the Bitcoin network, each of user has a public log called block chain. This block chain is a list of blocks in each of which another list of transactions is stored. To make the security of the block chain, a chain of cryptographic puzzles which is solved by some users called miners is utilized. When a miner solves a cryptopuzzle, it can create a new block containing a set of transaction and then take the rewards. Therefore, the goal of miners is about trying to solve cryptopuzzles, but to solve it the miners need to have a lot of resources or a powerful computation, which supports miners in increasing the probability resolving a cryptopuzzle. In addition, the Bitcoin needs a large number of honest miners who correctly follow the rules of Bitcoin protocol [1].
In the assignment, another cryptocurrency is used to make a simulation called Tinycoin whose idea is also similar to the simple Bitcoin's idea, but it eliminates some parts. Furthermore, another strategy is applied to consider the different from the honest strategy made by honest miners. The new strategy is called selfish strategy and based on the paper [1]. The main goal of this strategy is about supporting selfish miners who can get more rewards than honest miners, with less computation than honest's computation.
With the diversity of parameters not only from the network but also from Tinycoin protocol, the simulation utilizes a list of parameters such as: the number of nodes in the network, the number of degrees of each node, the percentage of selfish miner in the total miners, the power of selfish miners, the time of generation of two blocks and the latency of two nodes propagation. Based on these parameters, the purpose of the simulation is to consider about the number of forks generated and the total time consuming to solve these forks.
The remaining of the assignment is followed: Section 2 describes about selfish pool and its algorithm, the simulation is showed in Section 3 and two last Sections 4 and 5 represent about the experiments and conclusion, respectively.

2 Selfish Miner

This part is an overview of selfish strategy from paper [1]. The Bitcoin is considered a technique which is equitable to its participants. Therefore, the reward is believed that it is shared based on the resource of members in the pool. Nevertheless, in the paper [1], this wrong conventional wisdom is explained and then a strategy to exploit it called selfish strategy is described. Miners who follow the selfish strategy is called selfish miners.
The basic idea of this strategy is about keeping new blocks in a private and forking the public block chain. Whereas the honest miners try to mine a new block on the public block chain, the selfish nodes mine on its private chain. It hence exists two situations which can be:

- One of them is the case in which the private pool creates a lot of new blocks and continues keeping them in the private.
- Another case is that when the public reaches the length which is equal to the length of the private block chain, the private publishes blocks to public.

The analysis from the paper [1] shows that selfish and honest miners spend an amount of resources on mining cryptopuzzles, but the selfish miners will earn more rewards than honest miners while the honest miners waste more resources than other selfish miners. After that, the selfish pool can be appealing honest miners who will change the behavior and participate the selfish pool. As the result, the selfish pool will dominate and stop the

decentralization of the currency. To reach this goal, the paper [1] mentions about a threshold size of selfish mining pool. If the selfish pool exceeds this threshold, the Bitcoin system will be in a threat. After experiments, the paper concludes that the Bitcoin will be not safe if existing a selfish mining pool contains more than one third of the total power of the network, which means the selfish pool can be capable to get rewards more than the resources which is spent by this pool.

Code 1: Selfish-Mine algorithm [1].

```
1  on Init
2   publicChain <- publicly known blocks
3   privateChain <- publicly known blocks
4   privateBranchLen <- 0
5   Mine at the head of the privateChain.
6
7  on My pool found a block
8   DeltaPrev <- length(privateChain) - length(publicChain)
9   append new block to private chain
10  privateBranchLen <- privateBranchLen + 1
11  if DeltaPrev = 0 and privateBranchLen = 2
12   then
13    (Was tie with branch of 1)
14    publish all of the private chain
15    (Pool wins due to the lead of 1)
16    privateBranchLen <- 0
17  Mine at the new head of the private chain.
18
19 on Others found a block
20  DeltaPrev <- length(privateChain) - length(publicChain)
21  append new block to publicChain
22  if DeltaPrev = 0
23   then
24    private chain<-public chain (they win)
25    privateBranchLen <- 0
26  else if DeltaPrev = 1
27   then
28    publish last block of the privateChain
29    (Now same length. Try our luck)
30  else if DeltaPrev = 2
31   then
32    publish all of the privateChain
33    (Pool wins due to the lead of 1)
34    privateBranchLen <- 0
35  else DeltaPrev > 2
36    publish first unpublished block in private block.
37  Mine at the head of the private chain.
```

Particularly, let assume that existing public and private chains and the length of private chain is greater than the length of public chain. Hence, the honest miners mine the the shorter chain (public chain), while the selfish miners mine the longer chain (private chain). Because of the less computation of the total mining power in selfish miners, the public chain will reach the length close to the private chain, which means the private chain cannot be always ahead of the chain. In this situation, the private chain publishes blocks in its chain to the public. At this case, the honest miners waste their computation and move to a new received block from the private chain. The behavior of the selfish miners is presented by the Algorithm 1. The strategy is depended on the length of the private chain versus the public chain. Hence, there are some situations which can be considered.

When the public chain is longer than the private chain because the powers are different between them, the probability of selfish miners overtake the public chain is quite small. As a result, the selfish miners have to get the public chain and mine from this chain. If the honest miners find a new block and publish it, the selfish miners update and continue mining at the public chain.
When the selfish miners find a new block, instead of publishing this block to the public and spreading this block to other miners, the selfish miner will keep it in its private block chain. In the next step, there are two possible cases which can be considered: the honest miners find a new block on the public block chain which leads to the equal to the length of the private chain or the selfish miners continue finding another new block and extend its private chain.
In the first case, when a honest miner finds a new block and makes the length of public chain being equal to the length of private chain. At this situation, the private chain immediately publishes it private block. Thus, the selfish miners add and extend the received block from the honest miners, while the honest miners will choose a block to continue mining, which depends on the propagation. In case of selfish miners who want to mine from the received block from honest miners, the private chain will publish another block to enjoy the rewards from both of them. After a private block published by private chain, if the honest miners find a new block, the private pool will get the revenue of the block whereas the honest get the revenue from their block. Another case is that if the honest miners continue mining a block from their block, they will totally get the revenue of their blocks and the private pool cannot get anything.
Another case in which a selfish miner finds a second private block. Then, the private chain will continue leading the chain. As a result, the selfish miners go on mining from this private chain. The private chain will publish its first private block in case of receiving another block from honest miners. Because of the less power of computation from selfish miners, the private chain cannot always lead the chain. With this reason, when the public chain needs only one block to reach the private chain, the private chain publishes all of its blocks.

3 Tinycoin Simulation

In this part, the simulation is described based on two other small pieces. First subsection is a short explanation of PeerSim [2] frame work and the general idea in the simulation. Another one is about the specific Tinycoin structure and the way it work.

3.1 The Interaction of PeerSim in Tinycoin Simulation

The simulation of the assignment is based on the event-based engine PeerSim frame work [2] from the University of Bologna. With this frame work, the properties of Peer-to-peer system (the nodes be added and removed continuously) can be experimented and become an easy task [2].
Because of this advantages of PeerSim, the project is built based on the frame work. Therefore, the simulation has one config "txt" file and other java files to construct the protocol and to control the flow of the simulation as below:

- "configFile.txt" is to set the parameters and declare which protocols, initial points and the flow of execution.
- "protocolTinyCoin.java" is the definition of the protocol.
- "InitTinyCoin.java" is the initialization of the protocol.
- "observerTinyCoin.java" is the observer which can take node the result of simulation.
- "Oracle.java" is the thing which chooses a next node creating a new block, instead of solving a cryptopuzzles.
- "weightedTransportTinyCoin.java" is the definition of the weight of the network of nodes in PeerSim.

Configuration File To start the configuration "txt" file, in this file the general information of the network is defined such as: the number of nodes in the network, the degree of each node, the protocol is chosen, the initialization of protocol is chosen, the weight or the latency of the network, and the behavior of observer to take values depend on the requirements. Additionally, the information of the parameters is mentioned by this file such as: the percentage of selfish miners, the power of selfish miners, the latency between two nodes and the time between.

Tinycoin Initialization At the starting point of Tinycoin protocol, each node in the network randomly has an amount of money (it is used to create transaction). The public block chain and temporary buffer are also generated the same for every node at this state. The temporary buffer is used to store some received blocks which cannot find the previous block in the public block chain.
Additionally, the number and the power of miners are randomly generated, then the number of selfish miners is also created based on the parameters the percentage and the power of these selfish miners. In more detail, after taking the number of nodes in network, this state randomly chooses some of them becoming miner nodes and also randomly set the kind of miner for these nodes. In this simulation, four kinds of miners including: CPU, GPU, FPGA and ASIC are considered. Thus, when a node becomes a miner, this node randomly chooses a kind of miner. Because each kind of miner will have a different computation, the probability of different kind of miners is chosen by oracle to generate a new block in the simulation. The simulation considers the CPU will have one computation, GPU is two, while FPGA and ASIC being equal to each other are three. After that from two parameters the percentage and the power of selfish miners the, number of selfish miners is generated.

Oracle The aim of oracle is used to randomly choose a next miner who generates a new block. This object takes a parameter which is the time creating a new block. Then, after an amount of this time, the oracle randomly chooses a next miner. However, the choice of oracle is based on the probability of kinds of miners, which means FPGA and ASIC have higher probability of choosing than GPU and CPU. The reason corresponds to the probability to become a next miner corresponds to the power or computation.

Tinycoin Protocol The Tinycoin protocol is built based on the basic idea of the Bitcoin protocol without some parts. In this protocol, each node can create transactions (based on the amount of money it has) and spread it to its neighbors. It also can create a new block in case of the miner node, otherwise it only creates transactions. However, the protocol also considers about another strategy which is the selfish strategy [1].
The processes of Tinycoin protocol for miner node is divided into two strategies that are honest and selfish strategies. Therefore, each node follows the honest strategy or selfish strategy which depends on the definition of initial state.
In the honest strategy, each node in every cycle considers to randomly create a transaction including random two parameters a target node and an amount of money sent to this target. However, the generation of the transaction has to be satisfied a condition which is that the amount of money sent to target is less than the total amount of money owned by this node. Moreover, in the same cycle if this node is a miner, it will check to generate a new block or not. A node miner creates a new block in case of choosing by an Oracle. The Oracle is a object which will randomly choose a miner node to generate a new block in next cycle (The random of choosing miner is based on the power of node or the computation of node based on the kind of miners which is set from the initial step). In case of a transaction or a block is created, this node will propagate this message to its neighbors.
When a honest miner node or a normal node is received a message (it can be a transaction or a block) from one of its neighbors, this node will check the existing of this message in the transaction pool in case of transaction message or the temporary buffer and public block chain in case of block message. If this message exists before, this node will stop propagating this message to its neighbor, otherwise it will add this message to the transaction pool or public block chain or temporary buffer (it depends on the algorithm) and then propagate the message to its neighbors. Particularly, if the message is the transaction, it will be checked in transaction pool and in all blocks known by this node. In case of existing the transaction from the transaction pool or one of blocks known by this node, this node stops propagating the message, otherwise this transaction is added into the transaction pool. If the received message is a block, this node will check the existing of the block in its public block chain and temporary buffer. If this block has already existed, the node will stop propagating to its neighbors, otherwise this block is added to public block chain or temporary buffer (If the previous block of the message exists on the public block chain, this message will be added into public block chain, otherwise it is added into the temporary buffer). When a new block is added into the public block chain, each block of temporary buffer checks the new block being its previous block or not. If the new block is not its previous block, it will stay at the temporary buffer, otherwise it will be added into the public block chain and other blocks continue checking these new added blocks.

In case of selfish strategy from selfish miners, each selfish node follows the same behavior of honest, but it has another block chain called private block chain, which means these selfish nodes can generate transactions and blocks based on some conditions which is mentioned in Section **??**.
However, in stead of adding and spreading immediately a new block created by this node into public block chain, the new block will be kept on the private block chain. In the simulation of the selfish miners, in stead of creating one or some private pools for some selfish miners, each selfish miner has a different private pool, which means that these selfish miners cannot share their private block chain to each other. Therefore, each selfish miner will publish its private block based on the algorithm 1.
When a new block (honest block) is sent to the selfish miner, it will add the new block into the public block chain and check the difference between the length of public block chain and private block chain. If this difference is equal to zero, the private chain or the selfish miner loses and the private chain is assigned by the new public chain. If this difference is equal to one, which means the public and private chain are equal to each other. The private block chain will publish its last block. If this difference is equal to two, the selfish miner will publish all blocks in the private chain. If this difference is greater than two, the selfish will publish the first unpublished block in private chain.

Tinycoin Observer The observer of Tinycoin is used to take the results (this result can be taken based on the time of execution) after executing the flow or the simulation. In more detail, this part takes the number of forks (the number of branches are different from the main or the longest chain) from the public Block chain of each node in the network and the time consuming to solve these forks (the amount of time is calculated based on the number of blocks which are not belong to the longest chain). Specifically, the observer writes to a file the last result of nodes in the network, which means it counts the number of forks and the total time of solving these forks in every public block chain of each node and then takes the averages.

Weighted Transport in Network The weighted transport of the network is the latency between two nodes. Furthermore, this part describes about the latency of the propagation of messages (block or transaction) from nodes. The latency of a block is also depended on the number of transactions in each block. In addition, the simulation also has a range of the latency (the min and max of the delay) to control the delay.

3.2 Block Chain Structure in Tinycoin

The structure block chain contains some components including transactions and blocks. To start describing about the structure of block chain, it should be clear about the structure of its components.

Transaction The structure of transactions has the source, destination nodes and the amount of money sent by this transaction. Additionally, it should have a identification to recognize the transaction.

Block The structure of block includes the identified miner, identification of the block (the identification of each block is unique) and the identification of the previous block. In addition, the latency of the block in transportation is different from other blocks, which means the latency of the block depends on the number of transactions in each block. The rewards to the miner node of the block is also based on the number of transactions in the block. Therefore, when a miner creates a new block, it receives a reward of money which is the sum of reward of creating a new block and rewards from the number of transactions in the block.

Block Chain Structure The block chain structure is built based on the idea of tree structure because blocks can generate forks when more than two block having the same previous block. Therefore, each node of the tree public block chain represents a list of blocks while each of its children represents lists of blocks whose first blocks have the previous blocks being the last block of its parent list of block. Clearly, the Fig. 1 illustrates the block chain structure.

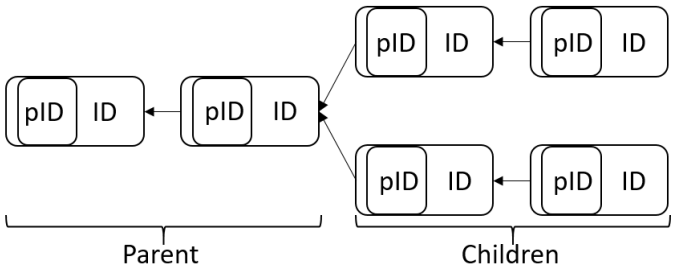
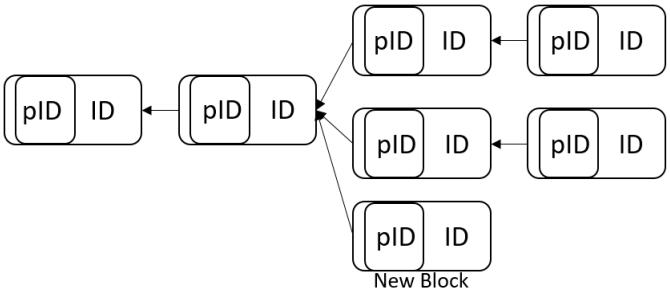
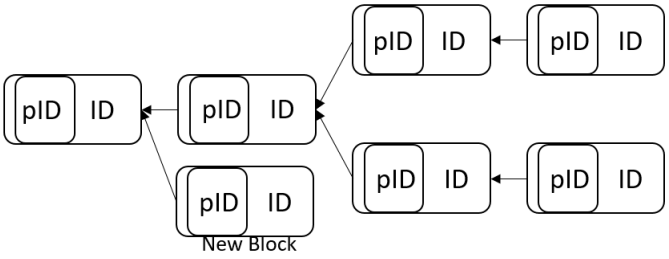


Fig. 1: Block chain structure is based on the tree structure: parent part is the list of blocks and children are other lists whose first blocks have the same previous ID of the last block of its parent.

(a) Adding a new block into the parent which does not have children.



(b) Adding a new block into the end of parent which has children.



(c) Adding a new bock into the parent which has children.

Fig. 2: These are three cases of adding a new block in the block chain structure.

Adding a New Block Because the block chain structure is based on the tree structure, to add a new block to the block chain there are two different cases: adding a new block created by this node to the public chain of the same node and adding a new block received from another miner to the public chain.
When a node want to create a new block, it needs to find the longest path of its block chain and then add the new block in this path. Thus, in this case of adding a new block created and added to the block chain by the same node, the block chain first finds the longest path to get the last block and then assigns it to the previous block of the new block. Therefore, the new block can be created with enough information and added to the block chain.
On another hand, when a node want to add a new block received from another node, the block chain of the node first find the list of block which contains the previous block of the new block or the tree who contains the previous block of the new block. After that the new block can be added to the list of block from the public chain. The way to add a new block to a list of blocks is described by the Fig. 2

4 Experiments

In this part, the simulation is experimented with various parameters to understand the changes of number of forks and the time of consuming to solve them.

Table 1: Experiments with the variety of parameters.

Network Size	No Degree	Selfish Percent	Selfish Power	Time of 2 blocks	Latency of 2 nodes	No Forks-Avg	Time Solving Forks-Avg
100	2	10	15	20	10	673 - 6.73	939 - 9.39
100	20	10	15	20	10	1221 - 12.21	2062 - 20.62
100	50	10	15	20	10	1262 - 12.62	2017 - 20.17
100	2	50	70	20	10	433 - 4.33	487 - 4.87
100	20	50	70	20	10	1214 - 12.14	2001 - 20.01
100	2	1	1	20	10	695 - 6.95	1009 - 10.09
100	20	1	1	20	10	1243 - 12.43	2148 - 21.48
100	2	100	50	20	10	790 - 7.9	1051 - 10.51
100	20	100	50	20	10	1232 - 12.32	2034 - 20.34
100	2	10	15	50	10	419 - 4.19	427 - 4.27
100	20	10	15	50	10	701 - 7.01	901 - 9.01

From the Table 1, the first row represents about parameters while other rows illustrate different experiments from parameters. The two last columns showing the results of the experiments are the sum of the number of forks and the amount of time to solve these forks.
With values of the table, it can be clear that with the low number of degree of each node, the number of forks and the time to consume are also low. However, in the first three experiments, the number of degree is increasingly changed 2, 20, 50 which leads to the rise of the number of forks and the time of consuming it, but the time of consuming forks decreases in case of the number of degree equal to 50.

Another observation is when the selfish miner is 100 percent, the number of forks also increase in comparison with other case (when the percentage of selfish miners is 50 or 10 or 1). The last experiment can get the low number of forks because of the low number of created blocks. Need to note that because of these experiments based on the random functions, the results is not a perfect one.

5 Conclusion

This assignment implements and simulates the Tinycoin, one of cryptocurrency, with two honest and selfish [1] strategies. The experiments of the simulation is implemented with PeerSim [2] frame work to create a network of nodes. After experiments, it can be clear that the number of selfish miners effects to generating forks, but the connections is the main of problem to create forks.

6 Ackowledgements

First and foremost, I would like to show my gratitude to Prof. Laura Ricci and her assistant PhD. Emanuele Carlini about lessons in Peer to Peer systems course. I also want to say thank my classmates who supported and explained my problems through discussions and conversations.

References

1. Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *International conference on financial cryptography and data security*, pages 436–454. Springer, 2014.
2. Gian Paolo Jesi. Peersim howto: Build a new protocol for the peersim 1.0 simulator. *Peersim. surcefge. net*, 2005.