Simulation of a P2P Cryptocurrency Network

Venkatesh K S, Veeresh B, Vinu Rakav S

February 2024

1 Introduction

We have implemented a discrete event simulator for a peer to peer cryptocurrency network. It maintains an event queue and global clock. We execute earliest event from the event queue which generates new events and push into event queue waiting for other events to run.

2 What are the theoretical reasons of choosing the exponential distribution?

In this type of Network, based on observations of experiments as well as queuing model analysis, the process of generating Transactions follows Poisson distribution. Hence, we need exponential distribution to generate inter-arrival time of two events.

3 Why is the mean of d_{ij} inversely related to c_{ij}? Give justification for this choice

d_{ij} is Queuing delay and c_{ij} is Link speed. We know that queuing delay is dependant on Rate of departure and Rate of arrival. Rate of departure depends on transmission time. Transmission time is inversely proportional to link speed and hence, queuing delay is inversely related to link speed.

For example, we have 5 packets in a Queue waiting to be sent. Fifth packet is queued and sent once when forth packet is sent. If we increase Link speed, first four packets are sent faster than before and hence, queuing delay of fifth packet will be reduced.

4 Mean = I/h_k

Keeping mean of T_k, we are satisfying following conditions:

- average interarrival time between any two blocks is I.

- Nodes with higher hashing power should be able to generate block in less time

5 Fork Resolution

Fork is resolved by focusing on mining the longest blockchain. I change focus from mining only when I receive a block that changes the longest blockchain. For example, if a new block arrives on the parent of the last block of the longest chain, the longest chain is unchanged, and hence, I do not stop mining. If a new block is added to the longest chain, I stop mining and focus on the new block now.

6 Loopless Transaction and block forwarding

We maintain lists of transactions at each peer. If we receive a transaction, we check if the transaction does not exist in the list before forwarding it. If the transaction exists in the list, we do not forward it to prevent the looping problem.

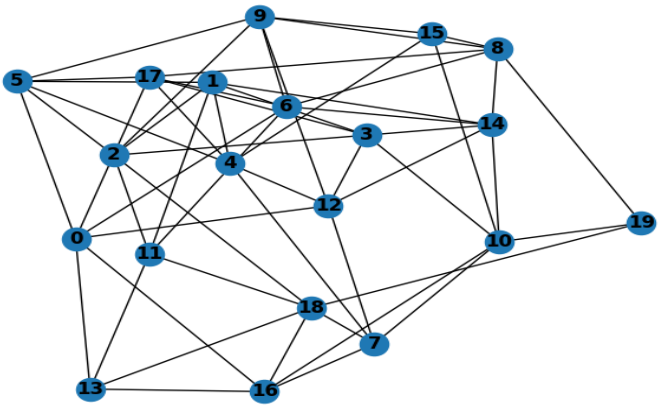
7 Validation of Blocks and Transactions

We have access to every peer's balances beforehand. When we receive a new block, we use the new block and every peer's old balances to calculate new balances. This will generate new balances, and we will invalidate the block if any balance goes negative.

8 Longest Chain Identification

Every block has a depth variable attached to it. We use this variable to find out the maximum depth. We maintain a global variable to track the block with the longest depth.

9 Network



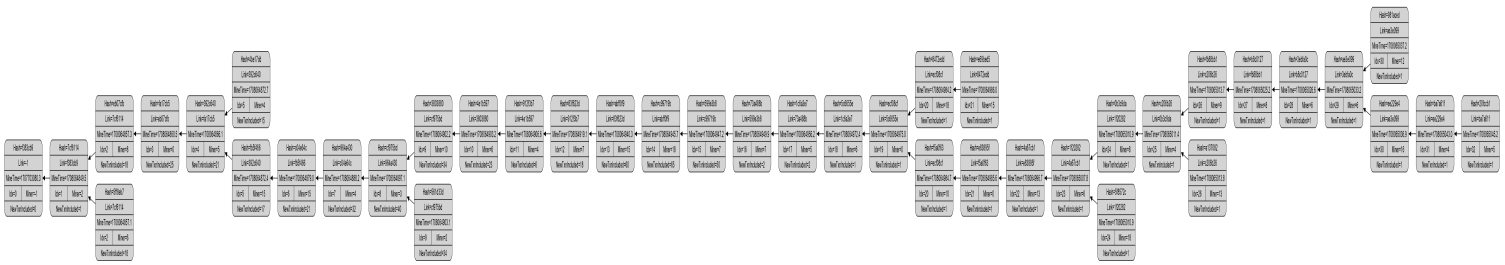
This is image of connected graph network created using 20 nodes. We used **Networkx** module to create such a network.

10 Experiment and Analysis

10.1 Influence of T_k value

We have observed that **very less T_k value** would cause blocks to be mined faster and increases probability of blocks being generated faster and creating forks and hence **wasting hashing power** of many peers. A **very high value of T_k also seems bad**. It will cause a lot of pending transactions and probability of attacks are high. Keeping T_k **slightly higher than interarrival time** seems like good idea because it reduces forks and reduce hashing power wastage of peers.

10.2 Equal Mining time and Interarrival time values



Values Used → n=20, $z_0 = 40\%$, $z_1 = 40\%$, interarrival_mean_time = 5, interarrival_mean_block_time = 5
High Hash Power - Fast Node:

- No of blocks created: 17
- No of blocks in longest chain: 15
- Ratio: 88%

High Hash Power - Slow Node:

- No of blocks created: 25
- No of blocks in longest chain: 24
- Ratio: 96%

Low Hash Power - Fast Node:

- No of blocks created: 2
- No of blocks in longest chain: 2
- Ratio: 100%

Low Hash Power - Slow Node:

- No of blocks created: 1
- No of blocks in longest chain: 1
- Ratio: 100%

Here, we see general case where it is neither perfectly linear nor forked too much when Interarrival time(T_{tk}) and Mining time(T_k) values are similar, but it is **almost linear leading to higher values of R**.

10.3 Very higher mining time

Values Used → n=20, z_0 = 40%, z_1 = 40%, interarrival_mean_time = 0.1, interarrival_mean_block_time = 2
High Hash Power - Fast Node:

- No of blocks created: 19
- No of blocks in longest chain: 8
- Ratio: 42%

High Hash Power - Slow Node:

- No of blocks created: 12
- No of blocks in longest chain: 26
- Ratio: 50%

Low Hash Power - Fast Node:

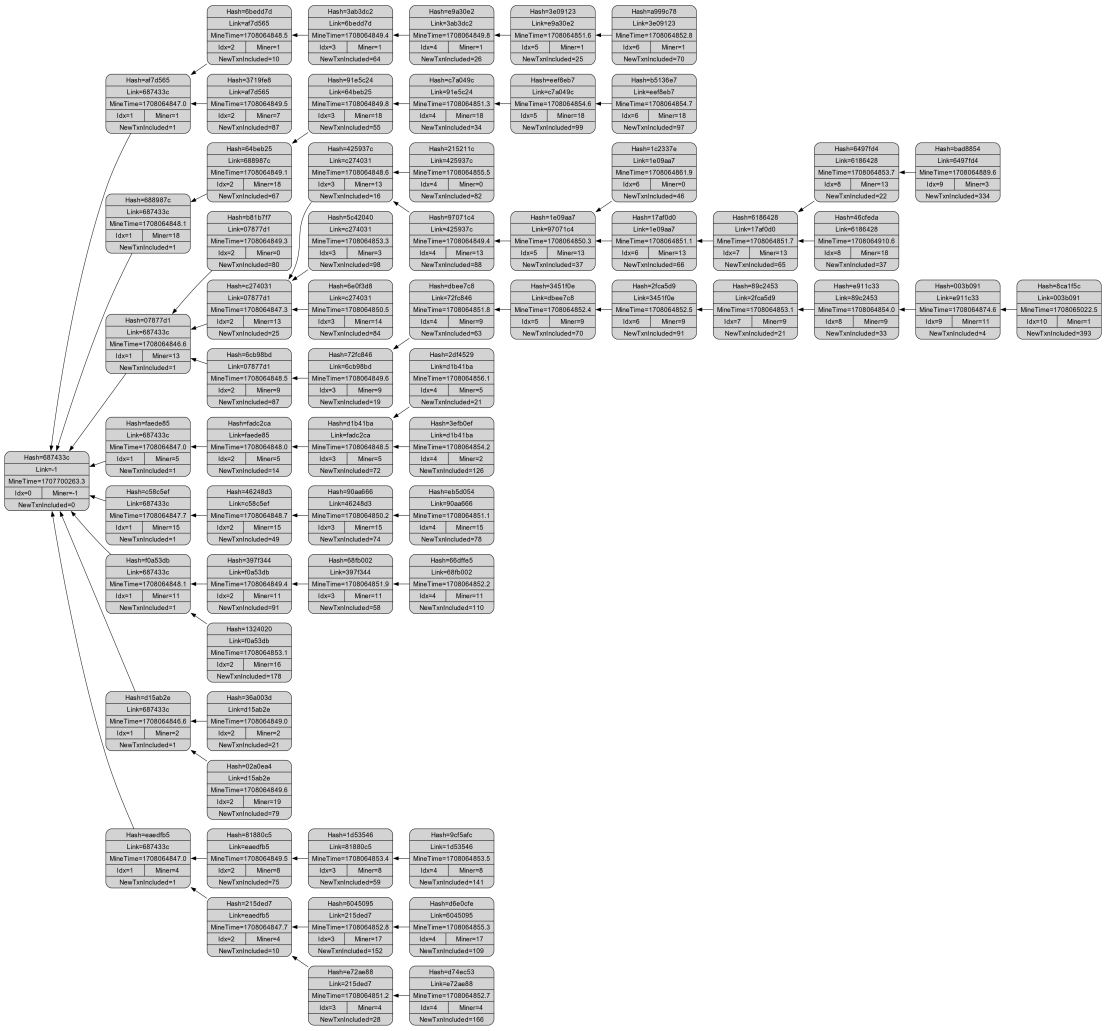
- No of blocks created: 0
- No of blocks in longest chain: 0
- Ratio: Undefined

Low Hash Power - Slow Node:

- No of blocks created: 1
- No of blocks in longest chain: 0
- Ratio: 0%

We can see Blockchain is almost linear due to higher time for mining, which is higher than time taken to propagate the mined block, hence **very less forks** are created. We can see **very higher values of R due to linearity of blockchain**.

10.4 Mining time and Interarrival time less than link speed



Values Used → n=20, z_0 = 40%, z_1 = 40%, interarrival_mean_time = 0.05, interarrival_mean_block_time = 0.1
High Hash Power - Fast Node:

- No of blocks created: 79
- No of blocks in longest chain: 11
- Ratio: 13%

High Hash Power - Slow Node:

- No of blocks created: 38

- No of blocks in longest chain: 1
- Ratio: 3%

Low Hash Power - Fast Node:

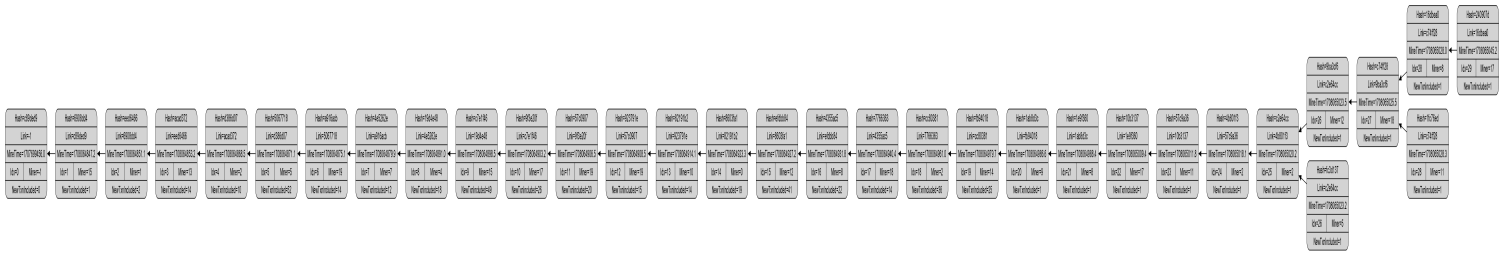
- No of blocks created: 24
- No of blocks in longest chain: 1
- Ratio: 4%

Low Hash Power - Slow Node:

- No of blocks created: 14
- No of blocks in longest chain: 0
- Ratio: 0%

Since we can now mine faster as well as create transactions faster, we can see **multiple forks created due to flooding of networks**. This will further reduce Ratio for almost all the peers. **R values are very much reduced** because of forks created in the network

10.5 Increasing amount of High hashing power machines



Values Used → n=20, $z_0 = 50\%$, $z_1 = 20\%$, interarrival_mean_time = 5, interarrival_mean_block_time = 5
High Hash Power - Fast Node:

- No of blocks created: 11
- No of blocks in longest chain: 10
- Ratio: 91%

High Hash Power - Slow Node:

- No of blocks created: 18
- No of blocks in longest chain: 17

- Ratio: 94%

Low Hash Power - Fast Node:

- No of blocks created: 1
- No of blocks in longest chain: 1
- Ratio: 100%

Low Hash Power - Slow Node:

- No of blocks created: 1
- No of blocks in longest chain: 1
- Ratio: 100%

We can see the **blockchain being almost linear with very high value of R**. We can also see that **high hashing power machines creates the most blocks** due to high hashing power.

10.6 Conclusion

- We can conclude that **higher mining time** causes **linearity** of blockchain and **reduces forks**.
- We can see that **high CPU machines** create many blocks than low CPU machines.
- Keeping **interarrival time less than link speed causes forking** as seen above.