

Simulation of a P2P Cryptocurrency Network

CS765: Introduction to Blockchains,
Cryptocurrencies, and Smart Contracts

Sourab Jha
19305R003

September 7, 2021

The task in this assignment is to prepare a discrete event simulator for a P2P cryptocurrency network. I have used Python for this simulation. The implementation details follow.

1 Initialization

At the very start, we calculate the fixed delays corresponding to each pair of nodes, which will help us in calculating latencies later.

We also create a network graph that resembles a real-world cryptocurrency network. Existing research ([1] [2] [3]) suggests that cryptocurrency networks (and even the internet, in general) can be best represented by a small-world graph, where the degree distribution of nodes follows the Power Law. In accordance with this, I have used the Albert-Barabasi algorithm [4] to generate a random connected graph containing n nodes. Each of these nodes is a peer of the cryptocurrency network, represented by integers 0 to $n - 1$. Each of these nodes is marked slow with a probability z , or fast with a probability $1 - z$.

Finally, we also pre-populate the event queue with a set of m transactions from each of the n nodes. These transactions are of the form "*TrxID*: A paid B an amount of K coins." For each A , m B s are randomly chosen out of the remaining nodes, with replacement. The amount K is sampled from a uniform distribution with mean 10. The generation of the transactions can be thought of as a Poisson process. Hence, the inter-arrival times of the transactions are sampled from an exponential distribution with mean i .

These transaction events are added to a queue (implemented as a heap), and subsequently created events are added to the same queue.

At the start of time, we also schedule mining events at all nodes with a mean completion time m , and add these events to the queue.

2 Transaction Broadcast

Execution of each transaction happens as follows: for ease of processing, the node generating the transaction sends the transaction to itself at the same timestamp it is generated. Upon receiving any transaction, the node checks if it has seen the transaction before. If not, it sends it to all of its neighbour (except the one it received the transaction from). The reception events are queued for each nodes after a time delay corresponding to the network latency between the two nodes, which is given by $\rho_{ij} + \frac{|m|}{c_{ij}} + d_{ij}$, where i and j correspond to the source and destination nodes, ρ_{ij} to the speed of light propagation delay, c_{ij} to the link speed between i and j , and d_{ij} to the queuing delay at node i . d_{ij} is inversely proportional to c_{ij} . This is because the faster the link speed is between the two nodes, the lesser would be the probability of the message getting stuck for longer periods in the queue at the source, since the source would not be busy propagating the earlier queued messages.

3 Block Creation and Broadcast

Each block contains information about a set of transactions, about the creator of the block, and a pointer to its parent block. At the start, each node has the genesis block (which, in the likeness of bitcoin, has a non-existent peer as its creator, which is denoted by -1).

The passing of created blocks is exactly the same as the passing of transaction messages, except the actions taken by the receiving node after receiving the block.

Each node keeps track of the blocks received by it. If the received block forms a new longest than previously seen, the node starts a new mining event over this new block. While executing the mining event, if the block's parent is no longer the longest chain seen by the node, the new block scheduled for mining is discarded. While this discarding should be done whenever a new block is received, in this case it is being discarded at the end of the mine time so that the event queue (which is a heap in this case) does not need to be recreated.

4 Termination

The simulation runs until the time t_t , given by $t_{max} + 2m$, where t_{max} is the time of the last transaction and m is the average mining time. This gives the network enough time to create a block which contains the last transaction, but terminates it not very long after. It has been empirically verified that only mining events are scheduled after this in the queue.

At the end of the simulation, the information regarding which nodes have which blocks is written to a single file, which is named as per the parameters of the simulation.

5 Conclusions

For the purpose of the following results, all combinations of the following parameters have been used:

Number of peers, n : 10, 30 Number of transactions, t : 50, 500 Interarrival mean, i : 100, 500 Average Mining time, m : 200, 500, 1000 Fraction of slow nodes, z : 0.20, 0.30, 0.40

Thus, we have a total of 72 simulations.

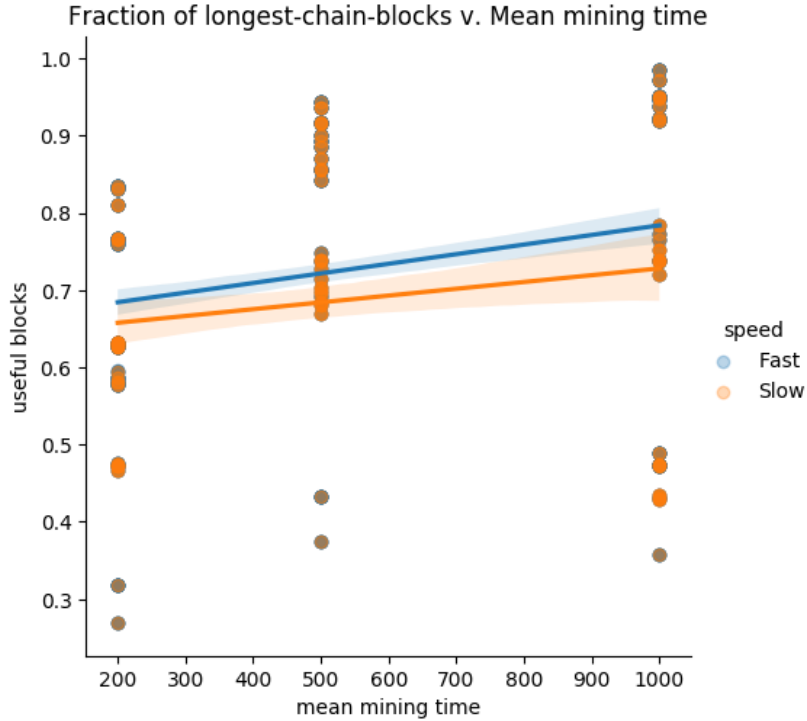


Figure 1: Comparison with mean mining time

As can be seen from the graphs 4, 3, 1, and 2, the average fraction of blocks at any node that make up the longest chain in the network increases as the average mining time increases and the interarrival between the transactions decreases. This can be explained as follows: the more blocks are created while the older blocks are still being circulated around the network, the higher is the number of wasted blocks (blocks not in the longest chain) we have.

In case of fast nodes, the fraction of wasted blocks increases as we increase the mean transaction interarrival time. This might be because as the speed of incoming transactions approaches the mining time, the faster nodes actually spend more time trying to create new blocks out of the new transactions.

Fig 5 shows a typical blockchain tree, wherein there are a few small chains

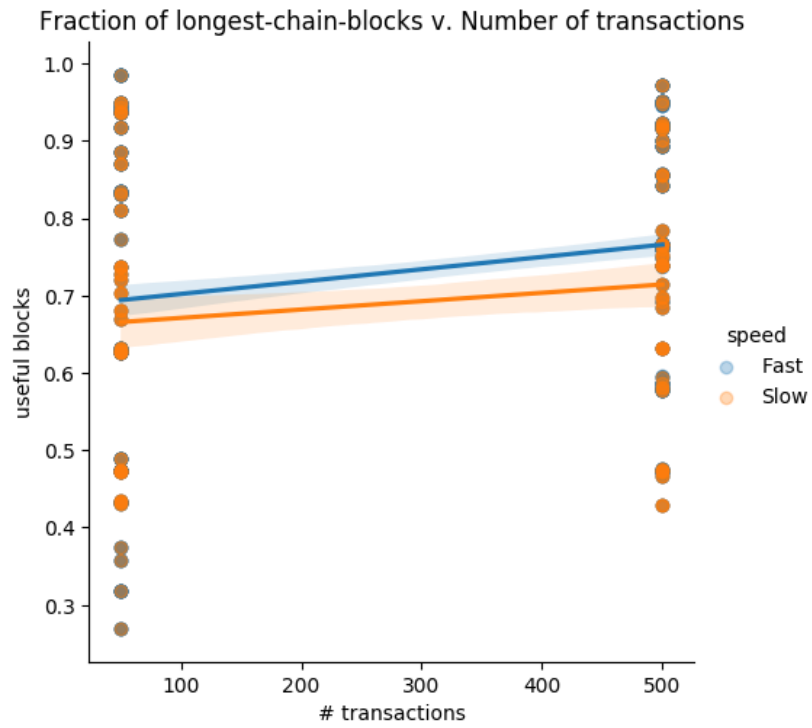


Figure 2: Comparison with number of transactions

that are abandoned soon after. Since the graph has over 133 blocks, many of the pointer arrows have been eliminated.

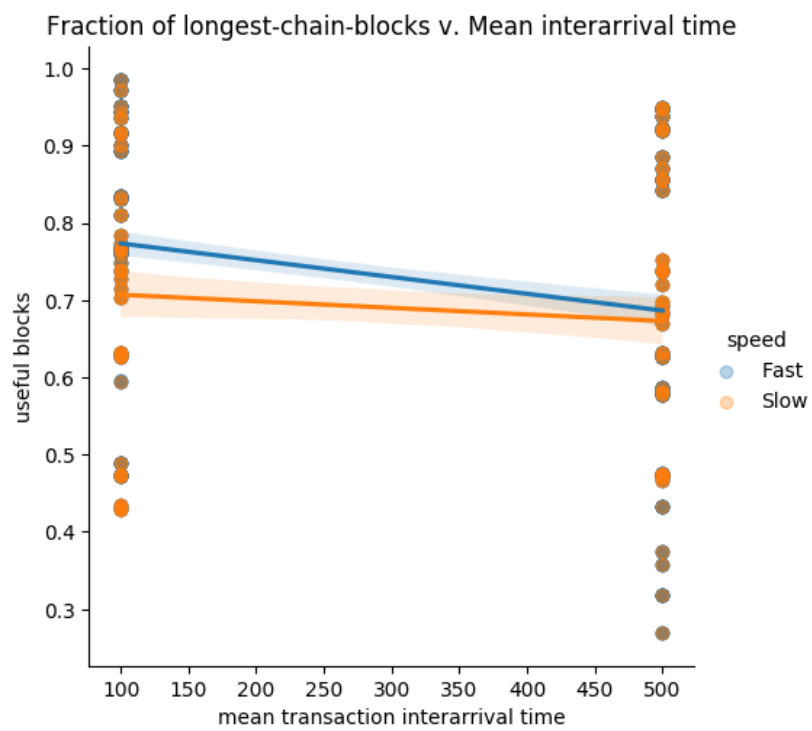


Figure 3: Comparison with mean interarrival time

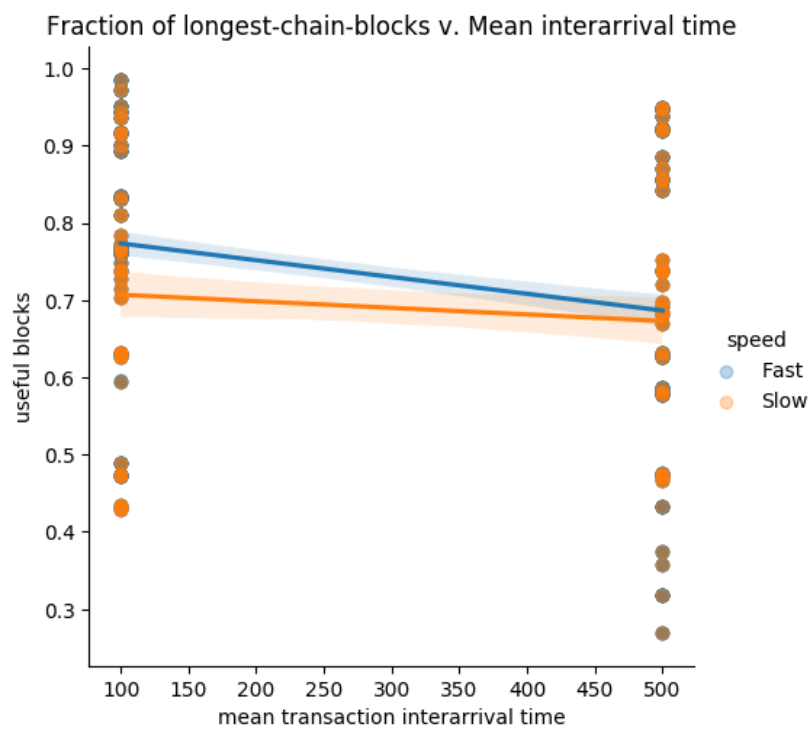


Figure 4: Comparison with speed of nodes

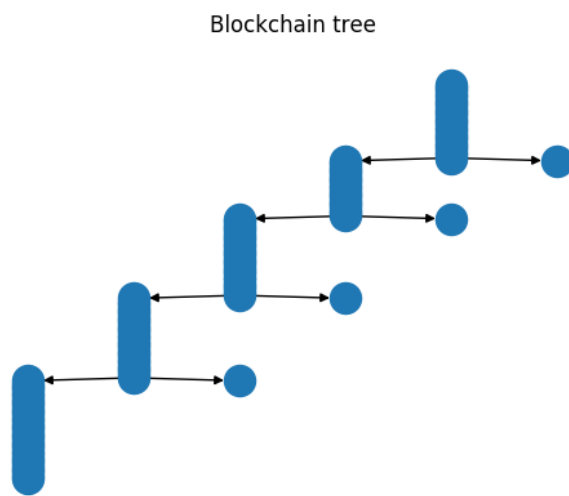


Figure 5: Typical Blockchain Tree

References

- [1] A. Baumann, B. Fabian, and M. Lischke, “Exploring the Bitcoin Network,” vol. 1, Apr. 2014.
- [2] L. Cocco and M. Marchesi, “Modeling and Simulation of the Economics of Mining in the Bitcoin Market,” *PLOS ONE*, vol. 11, p. e0164603, Oct. 2016. Publisher: Public Library of Science.
- [3] M. Javarone and C. Wright, “From Bitcoin to Bitcoin Cash: a network analysis,” pp. 77–81, June 2018.
- [4] A.-L. Barabási and R. Albert, “Emergence of scaling in random networks,” *science*, vol. 286, no. 5439, pp. 509–512, 1999. ISBN: 0036-8075 Publisher: American Association for the Advancement of Science.