Simulation of a P2P Network

B.S.S.R. Saran K. Sree Nikhil V. Mahanth Naidu

CS 765 Assignment - 1

1 Introduction

In this report, we present a simulation of a P2P cryptocurrency network that was implemented in Python. In our implementation, we used a discrete event simulator that maintains a timed priority queue. A timed priority queue is a data structure that manages a collection of events with their associated timestamps, where events are executed according to their timestamps.

2 Code Explanation

We defined several classes and functions that collectively simulate the behavior of a P2P cryptocurrency network:

- `Block` and `Blockchain` : These two classes together represent the basic building blocks of our blockchain. The `Block` class encapsulates information such as the block ID, timestamp, list of transactions, miner name, and parent link hash value. The `Blockchain` class manages the chain of blocks and provides functionality for adding new blocks, validating the chain, resolving conflicts, and visualizing the blockchain.
- `Transaction` : The `Transaction` class defines the structure of transactions within the network, including attributes such as transaction ID, sender, recipient, amount, and timestamp.
- `Peer` : The `Peer` class represents individual nodes in the P2P network. Each peer maintains its copy of the blockchain and participates in transaction validation, block mining, and network communication. This peer class contains most of the important functions used for generating and sending transactions, generating and sending blocks, checking the validation of transactions and calculating the propagation delay, etc.
- `TimedPriorityQueue` : This class is for discrete event simulation and it contains a priority queue based on timestamps for managing events within the simulation. It allows for scheduling and executing events at their specified times.

- `Network` : The `Network` class manages the simulation by creating and linking peers, initially generating transactions randomly, propagating them across the network, and tracking information such as block arrival times and transaction propagation delays.

3 Implementation Details

The simulation begins by initializing a network of peers with input parameters such as total number of events`(N)` in the simulation, number of nodes in the network`(n)`, percentage of slow nodes`(z0)`, percentage of low cpu nodes`(z1)`, and mean times for the exponential distributions`(Ttx and Tk)`. Peers are connected in a peer-to-peer fashion, creating a `decentralized network` topology. Each peer is randomly connected to between `3` and `6` other peers. Initially, the transactions are generated by all the peers to randomly selected peers and then broadcasted to their neighboring peers. It's implemented such that all the peers start mining from the starting time. The simulation then proceeds in discrete time steps, during which the peer generates transactions randomly in time with the interarrival between transactions generated by any peer chosen from an exponential distribution whose meantime is `(Ttx)`.

Upon receiving a transaction, each peer validates it against its local copy of the blockchain. Valid transactions are added to the transaction pool, from which miners select transactions for inclusion in the next block. After validating a new block before adding it into the tree we always find the longest chain present in the tree and then we add it to the longest chain. If there are more than one longest chains in the tree, then we select the longest chain based on the time of arrival of the last blocks present in these longest chains i.e., we add the new block into the longest chain which has the shortest time of arrival for the last block. In this way the `Forks` were resolved. Then we broadcast the newly added block and this process goes on repeating until the total number of events matches input paramter`(N)`.

4 Justifications

Q1. What are the theoretical reasons behind choosing exponential distribution for the interarrival time between the transactions generated by any peer?

- The exponential distribution is a "memoryless" distribution. Memoryless indicates that the probability of an event occurring in the next time interval is independent on how much time has already elapsed. In our context, if we use exponential distribution for the interarrival time between the transactions then this property implies that the probability of a transaction generated by any peer occurring in the next time interval is not influenced by how long it has been since the last transaction of that peer.

- The exponential distribution is indeed commonly used to model phenomena where small values occur more frequently than large values. This property makes it suitable for modeling various types of random variables in different contexts, including transaction times and other scenarios where the occurrence of events is probabilistic and the likelihood of longer durations decreases exponentially.

Q2. Why is the mean of d_{ij} inversely related to c_{ij} while calculating propagation delay? Give justification for this choice.

- If the c_{ij} is lower (i.e., the link speed between nodes i and j is smaller), implies that the link connecting node i and node j has a lower capacity to transmit data. Then the messages have to wait for longer times in the queue before they can be forwarded. Thus, in networks with slower link speed(`$c_{ij}$`), queuing delays(`$d_{ij}$`) tend to be longer.

- Conversely when the c_{ij} is higher (i.e., the link speed between nodes i and j is greater), implies that the link connecting node i and node j has a greater capacity to transmit data. As a result, messages don't need to wait for much time in queues and can be forwarded quickly. Thus, in networks with higher link speed(`$c_{ij}$`), queuing delays(`$d_{ij}$`) tend to be smaller. Hence, the mean of d_{ij} is inversely related to c_{ij} for any i, j.

Q3. Give justification for chosen mean value `Tk` for generation of new blocks.

- We have to choose `Tk` such that the peers with high hashing power(`h`) should generate new blocks quickly and conversely the peers with low hashing power need to generate new blocks slowly. If the mean interarrival time for a peer is chosen as `I/h` (where h is the hashing power of that peer) then it's clear that for the above mean value, the peers with high hashing power generate new blocks quickly when compared to peers with lower hashing values.

- We have to choose `Tk` such that the mean of interarrival time between any two blocks from any two peers is `I`. Now the average inter-arrival time between blocks from any two peers is given by $P(I_1, I_2, ..I_n > x)$, which is the product of $P(I_1 > x)$,$P(I_2 > x)$,....$P(I_n > x)$. If we take the meantime as I/h_i for a peer `i`, then $\prod_{i=1}^{n} e^{-h_i x/I} = e^{-\sum h_i x/I} = e^{-x/I}$ which implies mean of this distribution is `I`. Hence the choice of the mean value `Tk` is I/h for any peer.

5 Diagrams

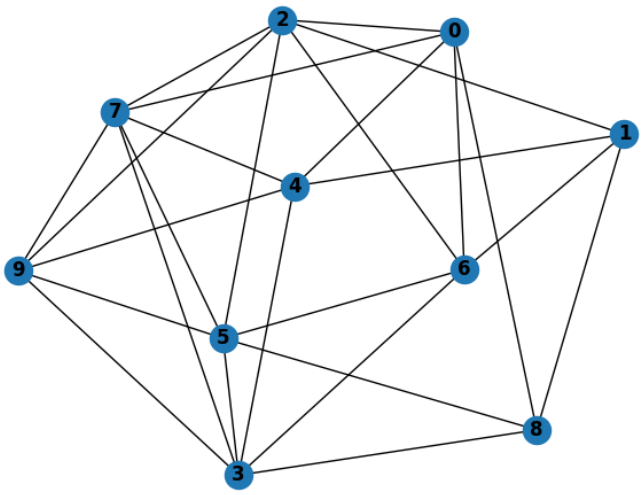


Figure 1: Network with 10 peers

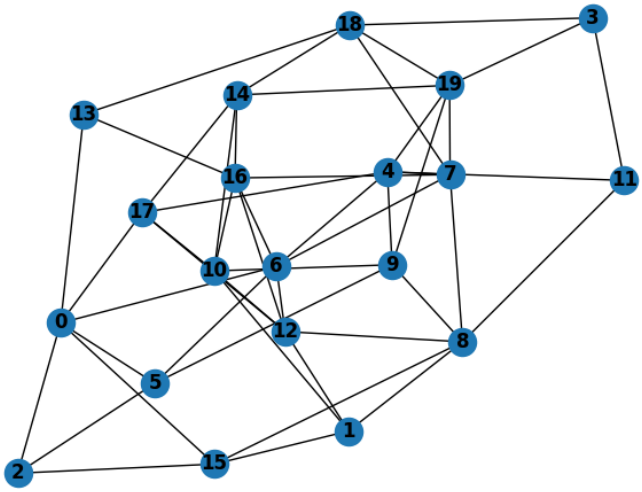
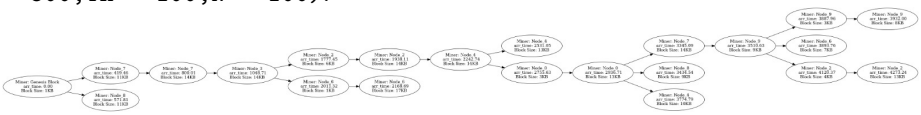


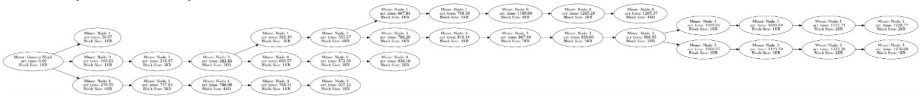
Figure 2: Network with 20 peers

- Blockchain Tree for the input paramaters (`n = 10,z0 = 10,z1 = 10,Ttx = 500,Tk = 100,N = 100`).



- Blockchain Tree for the input paramaters (`n = 10,z0 = 10,z1 = 90,Ttx = 500,Tk = 100,N = 100`).

- Blockchain Tree for the input paramaters (`n = 5,z0 = 10,z1 = 90,Ttx = 500,Tk = 10,N = 100`).



- Blockchain Tree for the input paramaters (`n = 5,z0 = 10,z1 = 90,Ttx = 500,Tk = 300,N = 100`).

- Blockchain Tree for the input paramaters (`n = 5,z0 = 90,z1 = 10,Ttx = 500,Tk = 500,N = 100`).

- Blockchain Tree for the input paramaters (`n = 5,z0 = 90,z1 = 90,Ttx = 500,Tk = 300,N = 100`).

- Blockchain Tree for the input paramaters (`n = 30,z0 = 10,z1 = 90,Ttx = 500,Tk = 200,N = 100`).

6 Analysis

- If the number of nodes(`n`) increases, then there will be more blocks generated almost at the same time which leads to forks which further leads to high chances of branching in the blockchain tree.
- If the percentage of slow nodes(`z0`) is high means that most of the linkspeeds in the network will be low then the chances of branching also increase in the blockchain tree.

- If the percentage of low CPU nodes(`z1`) is low, it implies that new blocks get generated quickly by most nodes and less time to resolve forks, hence there will be high chances of branching in the blockchain tree.
- If the meantime for interarrival of blocks(`Tk`) is increased, then the blocks take more time to broadcast through the network which decreases the branching in the blockchain tree.

Note: For all the above experiments and analysis, we fixed the total number of events as N = 100.

7 Summary

Q. Consider the ratio of the number of blocks generated by each node in the Longest Chain of the tree to the total number of blocks it generates at the end of the simulation. How does this ratio vary depending on whether the node is fast, slow, low CPU, high CPU power, etc.? How long are branches of the tree measured in the order of number of blocks?

- **Number of Peers(n):** As the number of peers in the network decreases, in general, the ratio of the number of blocks in the longest chain to the total number of blocks increases. So, the ratio is inversely proportional to the number of peers.
  - For `n = 20, z0 = 20, z1 = 20, Ttx = 500, Tk = 200` the ratio of the number of blocks in the longest chain to the total number of blocks is `0.433`.
  - For `n = 10, z0 = 20, z1 = 20, Ttx = 500, Tk = 200` the ratio of the number of blocks in the longest chain to the total number of blocks is `0.477`.
  - For `n = 5, z0 = 20, z1 = 20, Ttx = 500, Tk = 200` the ratio of the number of blocks in the longest chain to the total number of blocks is `0.500`.
- **Percentage of slow nodes(z0):** As the percentage of slow nodes in the network increases, in general, the ratio of the number of blocks in the longest chain to the total number of blocks increases. So, the ratio is directly proportional to the percentage of slow nodes. A fast node typically has a lower latency and can propagate new blocks more quickly to other nodes in the network, so there will be more probability for the new blocks generated by a fast node to be a part of the longest chain in the blockchain tree. So the ratio of the number of blocks in the longest chain generated by a node in the longest chain is generally directly proportional to the speed of that node.

- For `n = 5, z0 = 20, z1 = 20, Ttx = 500, Tk = 500` the ratio of the number of blocks in the longest chain to the total number of blocks is `0.590`.
- For `n = 5, z0 = 80, z1 = 20, Ttx = 500, Tk = 500` the ratio of the number of blocks in the longest chain to the total number of blocks is `0.678`.

- **Percentage of Low CPU nodes(z1):** As the percentage of low CPU nodes in the network decreases, in general, the ratio of the number of blocks in the longest chain to the total number of blocks decreases. So, the ratio is directly proportional to the percentage of low CPU nodes. And if a node has high CPU power, then it can mine new blocks quickly so there will be more probability for the new blocks generated by a high CPU node to be a part of the longest chain in the blockchain tree. So the ratio of the number of blocks in the longest chain generated by a node in the longest chain is generally directly proportional to the CPU power of that node.
  - For `n = 5, z0 = 20, z1 = 80, Ttx = 500, Tk = 500` the ratio of the number of blocks in the longest chain to the total number of blocks is `0.733`.
  - For `n = 5, z0 = 20, z1 = 20, Ttx = 500, Tk = 500` the ratio of the number of blocks in the longest chain to the total number of blocks is `0.537`.

- **Mean of block interarrival time(Tk):** As the mean of block interarrival time increases, in general, the ratio of the number of blocks in the longest chain to the total number of blocks increases majorly. So, the ratio is directly proportional to the mean of block interarrival time.
  - For `n = 5, z0 = 20, z1 = 20, Ttx = 500, Tk = 100` the ratio of the number of blocks in the longest chain to the total number of blocks is `0.403`.
  - For `n = 5, z0 = 80, z1 = 20, Ttx = 500, Tk = 500` the ratio of the number of blocks in the longest chain to the total number of blocks is `0.690`.
  - For `n = 5, z0 = 80, z1 = 20, Ttx = 500, Tk = 10000` the ratio of the number of blocks in the longest chain to the total number of blocks is `0.956`.

- **Mean of Transaction Time(`Ttx`):** The change in mean transaction time, in general, does not show any impact on the blockchain tree. This is because of the independence of the generation of blocks and generation of transactions. So the mean transaction time does not show any major impact on the ratio of the number of blocks in the longest chain to the total number of blocks in the blockchain.