CS 765 Assignment 1: Simulation of a P2P Cryptocurrency Network

Hruday Nandan Tudu - 210050067
Shikhar Parmar - 210050145
Pratiksha Deka - 210050122

February 2024

1 Introduction

We have built a discrete-event simulator for a P2P cryptocurrency network. The simulator enables analysis of parameters offering valuable insights into network performance and scalability.

Events, representing actions like transaction generation, block generate, transaction receive and block receive, drive the simulation. It executes the earliest events from the queue and pushes new events as a result of the execution of events. Nodes, blocks, and transactions are key components, mimicking real-world network dynamics. Transactions originate from payers and are transmitted to random payees, while blocks are mined based on the longest chain rule, with miners selecting valid transactions for inclusion. Upon receiving a new block, nodes validate and propagate it, ensuring network synchronization.

2 Network of peers

This graph generation process ensures that each node in the network has between 3 to 6 peers while maintaining connectivity. The network topology is established by iteratively connecting nodes in an initially empty graph. Nodes are added one by one, ensuring they have a minimum number of connections while adhering to a maximum limit. Connections are chosen randomly from eligible peers, with excess connections managed through a queue. This process continues until all nodes meet the connectivity requirements. The resulting graph is then verified for connectivity, and the process is repeated if necessary until a fully connected network with the desired topology is achieved.

3 Transaction

In the transaction section, transactions are generated and processed through events representing transaction generation ("TransactionGen") and receipt ("Trans-

actionRec"). When a "TransactionGen" event occurs, a payer node randomly selects a payee and transfers a random amount of coins choosen with some constraints. This event triggers the printing of transaction details, "CurSimTime TxnID: IDx pays IDy C coins". Upon receipt of a transaction, a "TransactionRec" event appends the transaction to the receiver's list and schedules further propagation to other nodes.

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

The exponential distribution's probability density function is represented as:

$$f(x) = \beta e^{-\beta x}$$

Transactions in blockchain networks follow a Poisson process, characterized by transactions occurring independently at a constant average rate over time, akin to events in a Poisson distribution. This process exhibits memorylessness, meaning the time until the next transaction is independent of past occurrences.

Let β is a parameter proportional to the likelihood of a transaction being generated at any instant. Let Δ be a very small interval. The probability of an event occurring can be taken as $\beta\Delta$ Probability of no block for a time of $n\Delta$ is given by $(1 - \beta\Delta)^n$

Let $n\Delta$ be denoted by x. As Δ approaches zero and n tends to infinity to ensure a nonzero time interval, the probability of no transaction occurring for a time $n\Delta$ is given by:

$$P(I > x) = \lim_{n\to\infty}(1 - \frac{\beta x}{n})^n = e^{-\beta x}$$

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

The queuing delay, represented by d_{ij}, exhibits an inverse relationship with the link speed c_{ij}. This relationship is grounded in networking principles, where higher link speeds facilitate faster data transmission, resulting in shorter queuing delays. Conversely, lower link speeds constrain capacity, leading to increased congestion risks and longer queuing delays. By modeling d_{ij} inversely to c_{ij}, the simulation accurately captures this relationship, ensuring that queuing delays decrease as link speed increases, and vice versa.

4 Block generation

The simulation incorporates two key events crucial to the operation of a blockchain network: block generation and block receiving. These events are simulated through the BlockGen and BlockRec classes respectively. In the BlockGen

Event, nodes generate new blocks based on their longest chain. Valid transactions are included in the newly generated block, and the node earns a mining reward before propagating the block to its peers. On the other hand, the Block-Rec Event simulates the receipt of blocks by nodes. Upon receiving a block, a node validates its contents and adds it to its blockchain if deemed valid. The node then updates its longest chain and shares the received block with its neighbors

4.1 Choice of $\frac{I}{h_k}$ as mean

T_k (the time until the next block is mined), is chosen as $\frac{I}{h_k}$, where I represents the average interarrival time between blocks and h_k is the fraction of hashing power held by a node. This choice ensures that nodes with higher hashing power have shorter mean inter arrival time.

Using the independence of the random variables, the probability that the interarrival times of multiple peers, denoted as $I_1, I_2, \ldots, I_n$, are all greater than x can be expressed as the product of the individual probabilities:

$$P(I_1, I_2, \ldots, I_n > x) = P(I_1 > x) \cdot P(I_2 > x) \cdot \ldots \cdot P(I_n > x)$$

. When each peer's interarrival time has a mean of $\frac{I}{h_i}$, where h_i represents the hashing power fraction of the i-th peer, these terms simplify to

$$P(I_1, I_2, \ldots, I_n > x) = e^{-x/I}$$

. This demonstrates that the mean of this distribution is I, confirming the choice of $\frac{I}{h_i}$ for the mean time.

5 Observations

5.0.1 Block generation

1. Number of nodes, $n = 10$, fraction of slow nodes, $z_0 = 0.3$, fraction of Low CPU, $z_1 = 0.3$, transaxtion delay mean time $T_{tx} = 0.3s$, Average block inter arrival time, $I = 3s$.

2. Number of nodes, $n = 10$, fraction of slow nodes, $z_0 = 0.5$, fraction of Low CPU, $z_1 = 0.5$, transaxtion delay mean time $T_{tx} = 0.3s$, Average block inter arrival time, $I = 3s$.

In Table 1 and 2, it's evident that the Ratio, which represents the proportion of blocks generated by each node in the Longest Chain relative to its total block generation, is influenced by both the hash power and the speed of the node. Notably, nodes with higher CPU power exhibit a significant advantage over those with lower CPU power. This advantage arises because higher CPU power contributes to an increased denominator in the ratio, reflecting the total number of blocks generated in the overall blockchain. Moreover, faster nodes

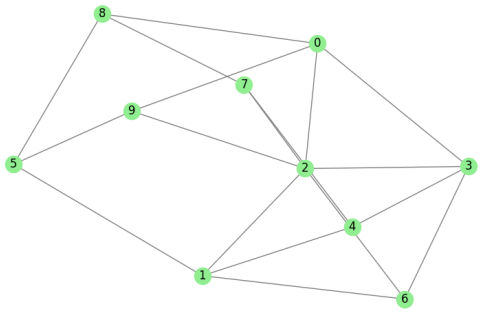

Figure 1: Network-1

contribute to the numerator by enhancing the number of blocks added to the Longest Chain network. High CPU Power with Fast Network holds the highest precedence, followed by High CPU Power with Slow Network, Low CPU Power with Fast Network, and Low CPU Power with Slow Network.

Node	**CPU Power**	**Slow Node**	**Ratio**
n_0	Low	Slow	1.0
n_1	Low	Fast	0.00
n_2	High	Fast	0.8
n_3	High	Slow	0.43
n_4	High	Slow	0.16
n_5	High	Fast	0.6
n_6	High	Fast	0.57
n_7	High	Fast	0.6
n_8	High	Fast0	0.66
n_9	Low	Fast	0.00

Table 1: Network(n=10, $z_0 = 0.3$, $z_1 = 0.3$, $T_{tx} = 0.3s$, I(block mean interarrival time)$= 1s$

5.1 Tree analysis

1. Number of nodes, $n = 10$, fraction of slow nodes, $z_0 = 0.1$, fraction of Low CPU, $z_1 = 0.9$, transaxtion delay mean time $T_{tx} = 0.3s$, Average block inter arrival time, $I = 1s$.

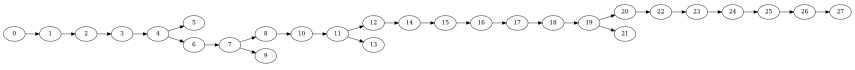

Figure 2: Network-2-tree

In the first observation, the number of slow nodes is reduced while the number of low CPU nodes is increased. With a higher number of low CPU

Node	CPU Power	Slow Node	Ratio
n_0	High	Fast	0.75
n_1	Low	Fast	0.00
n_2	Low	Fast	0.0
n_3	Low	Fast	0.00
n_4	High	Slow	0.83
n_5	High	Slow	0.375
n_6	High	Slow	0.2
n_7	High	Slow	0.6
n_8	Low	Fast	0.00
n_9	Low	Slow	1.0

Table 2: Network(n=10, $z_0 = 0.5$, $z_1 = 0.5$, $T_{tx} = 0.3s$, I(block mean interarrival time)$= 1s$

nodes, the network takes comparatively more time to generate blocks. Simultaneously, due to the lower number of slow nodes, the network propagates blocks at a faster rate, resulting in less branching.

2. Number of nodes, $n = 10$, fraction of slow nodes, $z_0 = 0.9$, fraction of Low CPU, $z_1 = 0.1$, transaxtion delay mean time $T_{tx} = 0.3s$, Average block inter arrival time, $I = 1s$.

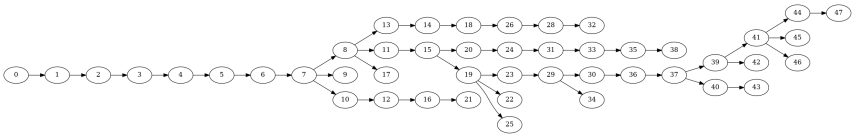

Figure 3: Network-3-tree

In contrast, in the second observation, the number of slow nodes is increased while the number of low CPU nodes is reduced. With a higher number of slow nodes, the network propagates blocks at a slower rate. Additionally, due to the lower number of low CPU nodes, the network generates blocks more quickly, resulting in increased branching.

3. Number of nodes, $n = 5$, fraction of slow nodes, $z_0 = 0.5$, fraction of Low CPU, $z_1 = 0.5$, transaxtion delay mean time $T_{tx} = 0.3s$, Average block inter arrival time, $I = 1s$.

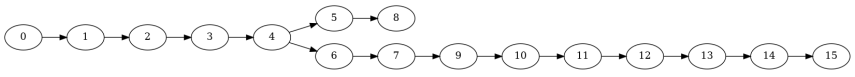

Figure 4: Network-4-tree

4. Number of nodes, $n = 15$, fraction of slow nodes, $z_0 = 0.5$, fraction of Low CPU, $z_1 = 0.5$, transaxtion delay mean time $T_{tx} = 0.3s$, Average block inter arrival time, $I = 1s$.

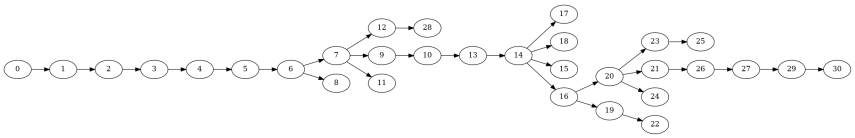

Figure 5: Network-5-tree

In observations 3 and 4, an evident trend emerges: with an increase in the number of nodes, branching within the network also rises. As the network's size expands, the time required for a block to propagate through the network also increases. Consequently, the likelihood of generating a block within this extended time interval also rises.

5. Number of nodes, $n = 5$, fraction of slow nodes, $z_0 = 0.1$, fraction of Low CPU, $z_1 = 0.5$, transaxtion delay mean time $T_{tx} = 0.3s$, Average block inter arrival time, $I = 0.6s$.

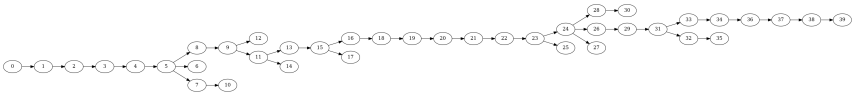

Figure 6: Network-6-tree

6. Number of nodes, $n = 5$, fraction of slow nodes, $z_0 = 0.1$, fraction of Low CPU, $z_1 = 0.5$, transaxtion delay mean time $T_{tx} = 0.3s$, Average block inter arrival time, $I = 3.5s$.

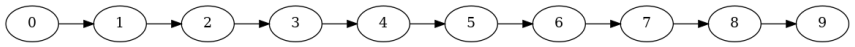

Figure 7: Network-7-tree

In observation 5 and 6 as the average block inter arrival time increases the branching decreases. When the inter-arrival time for blocks increases, it means that there is more time between the creation of successive blocks. With this increased time interval, blocks have more time to propagate through the network before the next block is created. As a result, nodes in the network have more opportunity to synchronize and reach a consensus on the accepted version of the blockchain. This increased synchronization reduces the likelihood of branching, as there is more time for all nodes to converge on a single version of the blockchain before the next block is generated. When the inter-arrival time for blocks increases, it means that blocks are generated less frequently. With fewer blocks being generated, the overall number of blocks in the blockchain decreases over time.

7. Number of nodes, $n = 10$, fraction of slow nodes, $z_0 = 0.3$, fraction of Low CPU, $z_1 = 0.3$, transaxtion delay mean time $T_{tx} = 0.3s$, Average block inter arrival time, $I = 1s$.

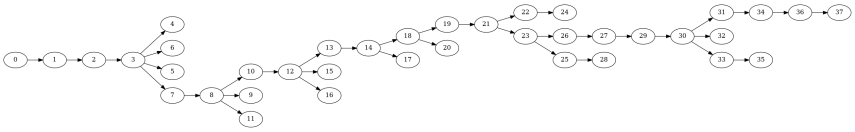

Figure 8: Network-8-tree

8. Number of nodes, $n = 10$, fraction of slow nodes, $z_0 = 0.3$, fraction of Low CPU, $z_1 = 0.3$, transaxtion delay mean time $T_{tx} = 0.03s$, Average block inter arrival time, $I = 1s$.

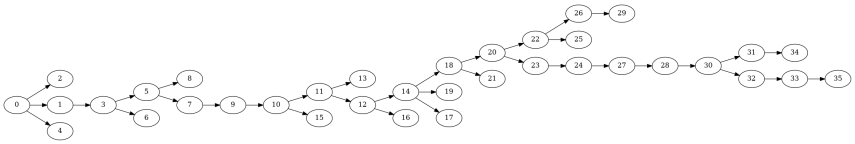

Figure 9: Network-9-tree

In observation 7 and 8 changing the transaction time doesn't significantly affect the blockchain because blocks and transactions operate independently.