CS765 Spring 2025 Semester
Project Part-1: Simulation of a P2P Cryptocurrency Network

Roll No: 24M0804
Roll No: 24M0755
Roll No: 24M0743

February 9, 2025

Contents

1 Theoretical Explanations

1.1 Theoretical Reasons for Choosing the Exponential Distribution for Interarrival Time Sampling

The exponential distribution is often used to model interarrival times because it has the following key properties:

1.1.1 Memoryless Property

The exponential distribution has the memoryless property, which states that:

$$P(X > s + t \mid X > t) = P(X > s)$$

for all $s, t \geq 0$. This means that the probability that an event occurs after $s + t$ units of time, given that it has already survived for t units, is the same as the probability that it survives for s units of time from the start.

Proof: We begin with the definition of conditional probability:

$$P(X > s + t \mid X > t) = \frac{P(X > s + t \cap X > t)}{P(X > t)}.$$

Since $X > s + t$ implies $X > t$, we can simplify this as:

$$P(X > s + t \mid X > t) = \frac{P(X > s + t)}{P(X > t)}.$$

For an exponential random variable X, the probability that X exceeds a certain value x is given by:

$$P(X > x) = e^{-\lambda x},$$

where λ is the rate parameter. Substituting this into the equation:

$$P(X > s + t \mid X > t) = \frac{e^{-\lambda(s+t)}}{e^{-\lambda t}}.$$

Simplifying the expression:

$$P(X > s + t \mid X > t) = e^{-\lambda s}.$$

Finally, since $P(X > s) = e^{-\lambda s}$, we conclude that:

$$P(X > s + t \mid X > t) = P(X > s).$$

This completes the proof of the memoryless property.

1.1.2 Exponential Distribution in Blockchain Simulations

The exponential distribution is ideal for modeling random event occurrences, such as transaction submissions in a blockchain network. It captures the randomness and independence of events, where transactions are initiated at unpredictable intervals. The distribution assumes a constant average rate of events (e.g., a node generating a transaction every 10 seconds), and its simple mathematical formula makes it efficient for simulating transaction arrivals and block creation times. This property allows for effective event scheduling in blockchain simulations, accurately representing how transactions and blocks propagate across the network.

1.2 Mean of d_{ij} (Queuing Delay) Inversely Related to c_{ij} (Link Speed)

The mean queuing delay d_{ij} is inversely related to the link speed c_{ij} due to the following reasons:

- **Queuing Delay**: d_{ij} is the time a packet waits in the queue before being transmitted over the network. It occurs when the network node is busy processing other packets.
- **Link Speed**: c_{ij} represents the rate at which data is transmitted over the link between nodes i and j. It is measured in bits per second (bps).
- **Higher Link Speed Decreases Queuing Delay**: When c_{ij} is higher (faster link), packets are transmitted more quickly, which reduces the time they spend waiting in the queue. Thus, the queuing delay d_{ij} decreases as c_{ij} increases.
- **Lower Link Speed Increases Queuing Delay**: When c_{ij} is lower (slower link), packets are transmitted more slowly, increasing the queuing time. Thus, the queuing delay d_{ij} increases as c_{ij} decreases.

This relationship can be expressed as:

$$\mathbb{E}[d_{ij}] \propto \frac{1}{c_{ij}}$$

where $\mathbb{E}[d_{ij}]$ is the expected queuing delay.

In summary, **higher link speeds result in lower queuing delays, while lower link speeds lead to higher queuing delays**.

1.3 Explanation for the Choice of T_k

In our simulation, the block mining time T_k is sampled from an exponential distribution with a mean of $\frac{I}{h_k}$, where h_k is the fraction of the total network hashing power possessed by node k. The parameter I controls the average block mining time across the network.

Reason for Choosing I:

To minimize forking in the blockchain, we need to balance the block mining time with network delays. In Satoshi's Bitcoin system, the block mining time is set to approximately 10 minutes, with network delays of the order of 10 seconds. This is because when the block mining time is much larger than the network latency, a successfully mined block can propagate to all nodes before multiple miners mine a block, reducing the probability of forks.

In our simulation, the network latency is modeled to be between 500-1000 milliseconds, which is significantly lower than Bitcoin's network latency assumptions. To ensure smooth block propagation and minimize forking, we set the ideal block mining time to I = 50 seconds. This allows for a suitable gap between block mining events, ensuring that a node's mined block is likely to be propagated throughout the network before another node mines its own block.

Balancing Forks and Mining Time:

By setting I = 50 seconds, we strike a balance between the desired block mining time and network latency. This value ensures that block generation does not occur too quickly (which could increase the likelihood of forks) and that blocks can propagate efficiently across the network before multiple miners mine blocks simultaneously.

2 Experiments

2.1 Experiment1

For this experiment, we use the following parameters:

- **Number of nodes:** $n = 10$
- **Percentage of slow nodes:** $z_0 = 50$
- **Percentage of nodes with low CPU:** $z_1 = 50$
- **Mean transaction interval time(in seconds):** $T_{tx} = 1$
- **Mean block interval(in seconds):** $I = 100$
- **Maximum simulation time(in seconds):** $T = 1000$

To run the experiment, we use the following command:

```
python3 main.py --z0 50 --z1 50 --n 10 --T 1000 --ttx 1 --tblk 100
```


Figure 1: Blockchain tree visualization

The experiment shows this result:

```
Length of the longest chain in blocktree (excluding genesis):  18
Number of mined blocks which are added in blocktree (exluding genesis and rejected blocks):  18
Fraction of mined blocks present in the longest chain to the total number of blocks in blocktree):  1.0


Fraction of blocks mined by low CPU and slow nodes to the total number of blocks present in the longest chain:  0.0
Fraction of blocks mined by low CPU and fast nodes to the total number of blocks present in the longest chain:  0.0
Fraction of blocks mined by high CPU and slow nodes to the total number of blocks present in the longest chain:  0.44
Fraction of blocks mined by high CPU and fast nodes to the total number of blocks present in the longest chain:  0.56

Fraction of blocks mined by node 0 which is fast and low cpu to the total number of blocks present in the longest chain: 0.0
Fraction of blocks mined by node 1 which is slow and high cpu to the total number of blocks present in the longest chain: 0.11
Fraction of blocks mined by node 2 which is fast and high cpu to the total number of blocks present in the longest chain: 0.28
Fraction of blocks mined by node 3 which is fast and high cpu to the total number of blocks present in the longest chain: 0.28
Fraction of blocks mined by node 4 which is fast and low cpu to the total number of blocks present in the longest chain: 0.0
Fraction of blocks mined by node 5 which is fast and low cpu to the total number of blocks present in the longest chain: 0.0
Fraction of blocks mined by node 6 which is slow and low cpu to the total number of blocks present in the longest chain: 0.0
Fraction of blocks mined by node 7 which is slow and low cpu to the total number of blocks present in the longest chain: 0.0
Fraction of blocks mined by node 8 which is slow and high cpu to the total number of blocks present in the longest chain: 0.11
Fraction of blocks mined by node 9 which is slow and high cpu to the total number of blocks present in the longest chain: 0.22

Length of each branch (branches are tree of blocks which is not part of the longest chain):  []
```

Figure 2: Results from Experiment 1

These parameters create a linear sequence without any branching. Since the block mining time significantly exceeds network latencies, forks do not occur in the blockchain. Nodes receive updates about a miner's success before completing their own mining attempts, making it highly likely that all nodes continue building on the same successfully mined block.

2.2 Experiment2

For this experiment, we use the following parameters:

- **Number of nodes:** $n = 10$
- **Percentage of slow nodes:** $z_0 = 50$
- **Percentage of nodes with low CPU:** $z_1 = 50$
- **Mean transaction interval time(in seconds):** $T_{tx} = 1$
- **Mean block interval(in seconds):** $I = 1$
- **Maximum simulation time(in seconds):** $T = 10$

To run the experiment, we use the following command:

```
python3 main.py --z0 50 --z1 50 --n 10 --T 10 --ttx 1 --tblk 1
```

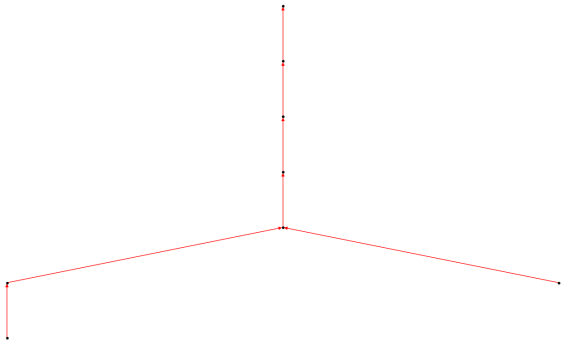

Figure 3: Blockchain tree visualization

The experiment shows this result:

```
Length of the longest chain in blocktree (excluding genesis):  6
Number of mined blocks which are added in blocktree (exluding genesis and rejected blocks):  7
Fraction of mined blocks present in the longest chain to the total number of blocks in blocktree):  0.86


Fraction of blocks mined by low CPU and slow nodes to the total number of blocks present in the longest chain:  0.0
Fraction of blocks mined by low CPU and fast nodes to the total number of blocks present in the longest chain:  0.0
Fraction of blocks mined by high CPU and slow nodes to the total number of blocks present in the longest chain:  0.83
Fraction of blocks mined by high CPU and fast nodes to the total number of blocks present in the longest chain:  0.17

Fraction of blocks mined by node 0 which is fast and high cpu to the total number of blocks present in the longest chain: 0.0
Fraction of blocks mined by node 1 which is fast and low cpu to the total number of blocks present in the longest chain: 0.0
Fraction of blocks mined by node 2 which is fast and low cpu to the total number of blocks present in the longest chain: 0.0
Fraction of blocks mined by node 3 which is fast and low cpu to the total number of blocks present in the longest chain: 0.0
Fraction of blocks mined by node 4 which is slow and high cpu to the total number of blocks present in the longest chain: 0.5
Fraction of blocks mined by node 5 which is slow and high cpu to the total number of blocks present in the longest chain: 0.0
Fraction of blocks mined by node 6 which is slow and high cpu to the total number of blocks present in the longest chain: 0.33
Fraction of blocks mined by node 7 which is slow and low cpu to the total number of blocks present in the longest chain: 0.0
Fraction of blocks mined by node 8 which is slow and low cpu to the total number of blocks present in the longest chain: 0.0
Fraction of blocks mined by node 9 which is fast and high cpu to the total number of blocks present in the longest chain: 0.17

Length of each branch (branches are tree of blocks which is not part of the longest chain):  [1]
Average length of branches:  1.0
```

Figure 4: Results from Experiment 2

We can now see that the blockchain exhibits short branches at multiple points. The block mining time is no longer significantly greater than network latencies, causing delays in block propagation. As a result, some nodes remain unaware of newly mined blocks and begin mining on different ones, leading to forks in the network.

2.3 Experiment3

For this experiment, we use the following parameters:

- **Number of nodes:** $n = 10$
- **Percentage of slow nodes:** $z_0 = 50$
- **Percentage of nodes with low CPU:** $z_1 = 50$
- **Mean transaction interval time(in seconds):** $T_{tx} = 0.1$
- **Mean block interval(in seconds):** $I = 1$
- **Maximum simulation time(in seconds):** $T = 20$

To run the experiment, we use the following command:

```
python3 main.py --z0 50 --z1 50 --n 10 --T 20 --ttx 0.1 --tblk 1
```

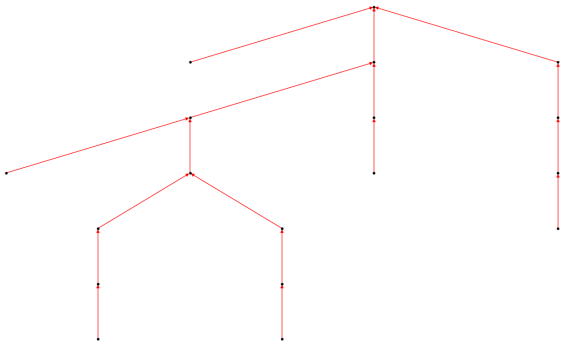

Figure 5: Blockchain tree visualization

The experiment shows this result:

```
Length of the longest chain in blocktree (excluding genesis):  6
Number of mined blocks which are added in blocktree (exluding genesis and rejected blocks):  17
Fraction of mined blocks present in the longest chain to the total number of blocks in blocktree):  0.35


Fraction of blocks mined by low CPU and slow nodes to the total number of blocks present in the longest chain:  0.0
Fraction of blocks mined by low CPU and fast nodes to the total number of blocks present in the longest chain:  0.0
Fraction of blocks mined by high CPU and slow nodes to the total number of blocks present in the longest chain:  0.5
Fraction of blocks mined by high CPU and fast nodes to the total number of blocks present in the longest chain:  0.5

Fraction of blocks mined by node 0 which is fast and high cpu to the total number of blocks present in the longest chain: 0.33
Fraction of blocks mined by node 1 which is fast and low cpu to the total number of blocks present in the longest chain: 0.0
Fraction of blocks mined by node 2 which is fast and low cpu to the total number of blocks present in the longest chain: 0.0
Fraction of blocks mined by node 3 which is slow and low cpu to the total number of blocks present in the longest chain: 0.0
Fraction of blocks mined by node 4 which is slow and high cpu to the total number of blocks present in the longest chain: 0.5
Fraction of blocks mined by node 5 which is fast and high cpu to the total number of blocks present in the longest chain: 0.17
Fraction of blocks mined by node 6 which is slow and low cpu to the total number of blocks present in the longest chain: 0.0
Fraction of blocks mined by node 7 which is slow and low cpu to the total number of blocks present in the longest chain: 0.0
Fraction of blocks mined by node 8 which is fast and high cpu to the total number of blocks present in the longest chain: 0.0
Fraction of blocks mined by node 9 which is slow and high cpu to the total number of blocks present in the longest chain: 0.0

Length of each branch (branches are tree of blocks which is not part of the longest chain):  [3, 4, 2, 1, 1]
Average length of branches:  2.2
```

Figure 6: Results from Experiment 3

The blockchain continues to exhibit multiple forks, but a key observation is that these branches are now significantly longer compared to Experiment 2. This indicates that network latencies have increased substantially, causing different groups of miners to work on separate branches for an extended period before eventually discovering a longer chain.

2.4 Experiment4

For this experiment, we use the following parameters:

- **Number of nodes:** $n = 20$
- **Percentage of slow nodes:** $z_0 = 50$
- **Percentage of nodes with low CPU:** $z_1 = 50$
- **Mean transaction interval time(in seconds):** $T_{tx} = 0.1$
- **Mean block interval(in seconds):** $I = 1$
- **Maximum simulation time(in seconds):** $T = 20$

To run the experiment, we use the following command:

```
python3 main.py --z0 50 --z1 50 --n 20 --T 20 --ttx 0.1 --tblk 1
```

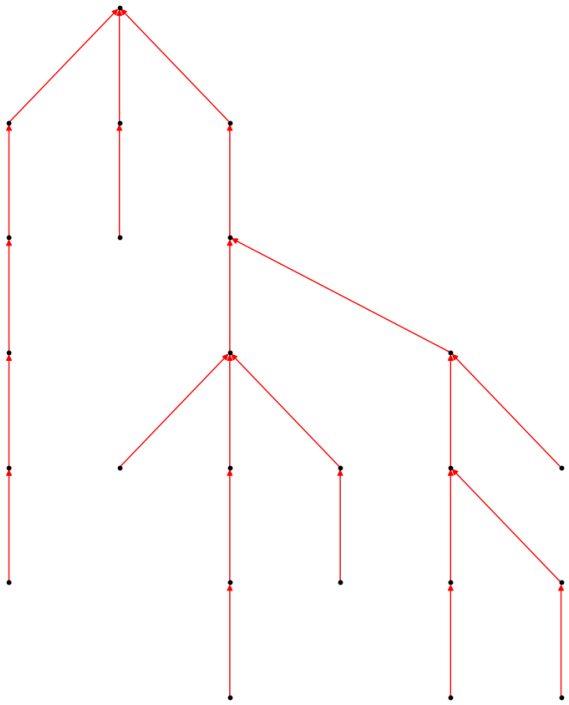

Figure 7: Blockchain tree visualization

The experiment shows this result:

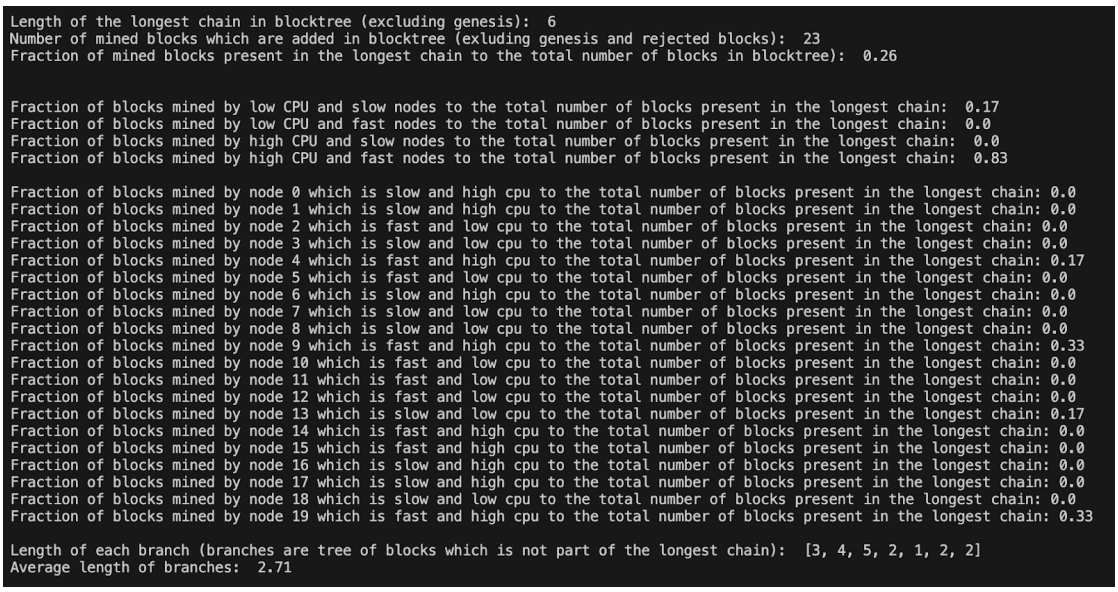

Figure 8: Results from Experiment 4

After doubling the number of nodes in the network, we notice a significant increase in forking compared to before. The output screen clearly shows a larger number of branches. This demonstrates that as more nodes participate as miners, competition intensifies, leading to a higher probability of forks occurring in the blockchain.

2.5 Experiment5

For this experiment, we use the following parameters:

- **Number of nodes:** $n = 10$
- **Percentage of slow nodes:** $z_0 = 50$
- **Percentage of nodes with low CPU:** $z_1 = 0$
- **Mean transaction interval time(in seconds):** $T_{tx} = 0.1$
- **Mean block interval(in seconds):** $I = 1$
- **Maximum simulation time(in seconds):** $T = 20$

To run the experiment, we use the following command:

```
python3 main.py --z0 50 --z1 0 --n 10 --T 20 --ttx 0.1 --tblk 1
```

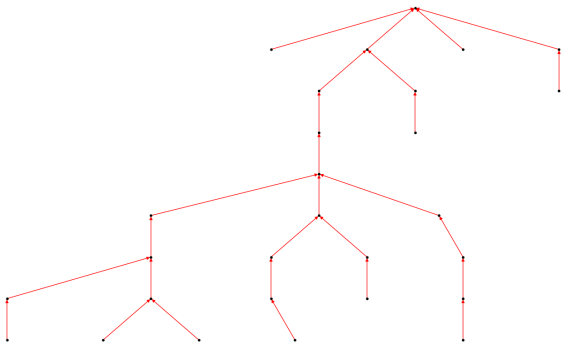

Figure 9: Blockchain tree visualization

The experiment shows this result:

```
Length of the longest chain in blocktree (excluding genesis):  8
Number of mined blocks which are added in blocktree (exluding genesis and rejected blocks):  27
Fraction of mined blocks present in the longest chain to the total number of blocks in blocktree):  0.3


Fraction of blocks mined by low CPU and slow nodes to the total number of blocks present in the longest chain:  0.0
Fraction of blocks mined by low CPU and fast nodes to the total number of blocks present in the longest chain:  0.0
Fraction of blocks mined by high CPU and slow nodes to the total number of blocks present in the longest chain:  0.5
Fraction of blocks mined by high CPU and fast nodes to the total number of blocks present in the longest chain:  0.5

Fraction of blocks mined by node 0 which is slow and high cpu to the total number of blocks present in the longest chain: 0.5
Fraction of blocks mined by node 1 which is slow and high cpu to the total number of blocks present in the longest chain: 0.0
Fraction of blocks mined by node 2 which is slow and high cpu to the total number of blocks present in the longest chain: 0.0
Fraction of blocks mined by node 3 which is slow and high cpu to the total number of blocks present in the longest chain: 0.0
Fraction of blocks mined by node 4 which is fast and high cpu to the total number of blocks present in the longest chain: 0.12
Fraction of blocks mined by node 5 which is fast and high cpu to the total number of blocks present in the longest chain: 0.0
Fraction of blocks mined by node 6 which is fast and high cpu to the total number of blocks present in the longest chain: 0.38
Fraction of blocks mined by node 7 which is slow and high cpu to the total number of blocks present in the longest chain: 0.0
Fraction of blocks mined by node 8 which is fast and high cpu to the total number of blocks present in the longest chain: 0.0
Fraction of blocks mined by node 9 which is fast and high cpu to the total number of blocks present in the longest chain: 0.0

Length of each branch (branches are tree of blocks which is not part of the longest chain):  [4, 4, 2, 2, 3, 2, 2, 1, 1]
Average length of branches:  2.33
```

Figure 10: Results from Experiment 5

We observe an increase in both the number of forks and the length of branches compared to the original Experiment 3. This occurs because all nodes now have equal hashing power, creating a fair competition where each has an equal chance of mining the next block. As a result, multiple nodes may successfully mine their own blocks before becoming aware of each other's blocks.