Simulation of a P2P Cryptocurrency Network
CS 765 Assignment 1

Mitesh Khemani	Het Patel	Bharat Patidar
23M0743	23M0751	23M0761

Q1) What are the theoretical reasons for choosing the exponential distribution?

Ans: For modeling the interarrival times between transactions in a peer-to-peer network simulation, an exponential distribution was chosen for a number of theoretical reasons:

Memorylessness: The memoryless property of the exponential distribution indicates that the likelihood of waiting a specific length of time for an event to occur remains constant regardless of the amount of time that has elapsed.Here's an illustration: Suppose you are standing in line for a bus. Whether you have been waiting for 10 minutes or have just arrived at the bus stop, there is an equal chance of waiting exactly 15 minutes for the bus if it arrives on average every 15 minutes.This property indicates that the time it takes for the next transaction to happen is independent of the time elapsed since the previous transaction in the context of transaction generation. The probability of the next transaction happening in the next minute doesn't change, regardless of how long ago a transaction occurred. It could have been one minute or an hour. When it comes to waiting for something to happen, time seems to be same regardless of the past.

Flexibility: By modifying the mean value (Ttx), the exponential distribution is readily adjustable. By simply changing this parameter, you can now model various scenarios with different transaction arrival rates. Because of its adaptability, it can be used in a variety of simulation scenarios.

Mathematical Proof:

Y is a random variable following exponential distribution with parameter λ.
a is the time by which the event has not occurred t
b is the time by which the event has occurred s

$P(Y>a+b \mid Y>a)=P(Y>b)$

Let's now demonstrate this property:

The probability density function (pdf) of Y can be found as follows:

$f_y(y)=\lambda e^{-\lambda y}$, $y \geq 0$

The cumulative distribution function (CDF) of $F_y(y)= 1-e^{-\lambda y}$, $y \geq 0$

Let's calculate the conditional probability $P(Y>a+b \mid Y>a)$:

$=P(Y>a+b \mid Y>a)= P(Y>a) / P(Y>a+b \text{ and } Y>a)$

$= P(Y>a) / P(Y>a+b)$

Using the CDF of Y:

$= 1-F_y(a+b) /1-F_y(a)$

$=1-(1-e^{-\lambda(a+b)}) / 1-(1-e^{-\lambda a})$

$=e^{-\lambda(a+b)} / e^{-\lambda a}$

$=e^{-\lambda a}$

Q2) Why is the mean of Dij inversely related to Cij ? Give justification for this choice.

Ans:

Since the mean of **D**ij represents the queuing delay at node i for the message to be forwarded to node j, it has an inverse relationship with **C**ij. When there is network congestion, queuing delay happens, and it is affected by the link speed (**C**ij) between nodes i and j.A high link speed (**C**ij) indicates faster data transmission between nodes, which reduces congestion and, consequently, delays in the queue. On the other hand, slower data transmission due to a low link speed raises the possibility of congestion and lengthens wait times.In terms of math, the queuing delay (**D**ij) is inversely proportional to the link speed (**C**ij), since shorter queuing delays result from a decrease in transmission time as **C**ij rises.

Q3) Explanation of choosing the mean equal to I/h_k

Ans:

Whole network's new block arrival time is I. That means on an average after I time we can expect new block to be generated by whole network. Miners have fraction of hashing power the whole network possesses. So for a miner to produce one block that will depend on it's hashing power. Total network has 100% of hashing power then it's able to create one block on an average. A single miner only has a fraction of that hashing power so miner should create one block on an average in I/hk time.

Analysis of Blockchain Dynamics:

Insights and Observations under Normal Parameters

At normal parameters (close to what actually happens), i.e.

- Avg. Block Interarrival Time = 600 sec (10 minutes)
- Avg. Txn Interarrival Time = 5 sec
- Avg. Latency of the network = ~0.5 sec
- Observation Period = 100 x Avg. Block Interarrival Time = 60000 sec (1000 minutes)

Slow Nodes %	Low CPU nodes %	Blocks in the Longest Chain	Total Blocks generated
10	10	107	107
10	90	98	98
90	10	104	104
90	90	85	85
50	50	102	102

Insights from the presented table reveal several key observations:

- **Absence of Forks:** The absence of forks across all scenarios can be attributed to the network propagation delay, which is notably shorter (~0.5 seconds) compared to the average block interarrival or generation time. Consequently, any block generated experiences swift propagation throughout the network. Thus, all participating nodes promptly recognize and begin mining on the latest block, provided it extends the length of the longest chain.

- **Consistency with Theoretical Expectations:** The outcomes obtained align consistently with established theoretical principles. Specifically, given the constant hashing power of the overall system, the number of generated blocks tends to remain relatively stable across the evaluated scenarios.

Simulation with different Parameters

These are the different parameters:

- total_peers: Represents total peers in the network
- Z0_percent: Represents percentage of slow peers
- Z1_percent: Represents percentage of low cpu peers
- Tb: Represents average block generation time (in milliseconds)
- Tx: Represents average transaction generation time (in milliseconds)
- To: Represents total observation time (in milliseconds)

To aid in the comprehension of our results, accompanying each simulation is an image depicting the blockchain structure (tree) of a randomly selected node.

In the simulations presented below, the observation time or simulation time is set to 40 blocks generation time to ensure clear visualization of the resulting trees.

Note that in certain diagrams, nodes representing two distinct blocks may overlap due to geometric and spatial constraints imposed by the visualization software. Therefore, configurations resembling a diamond shape do not imply the existence of two separate preceding/parent blocks.

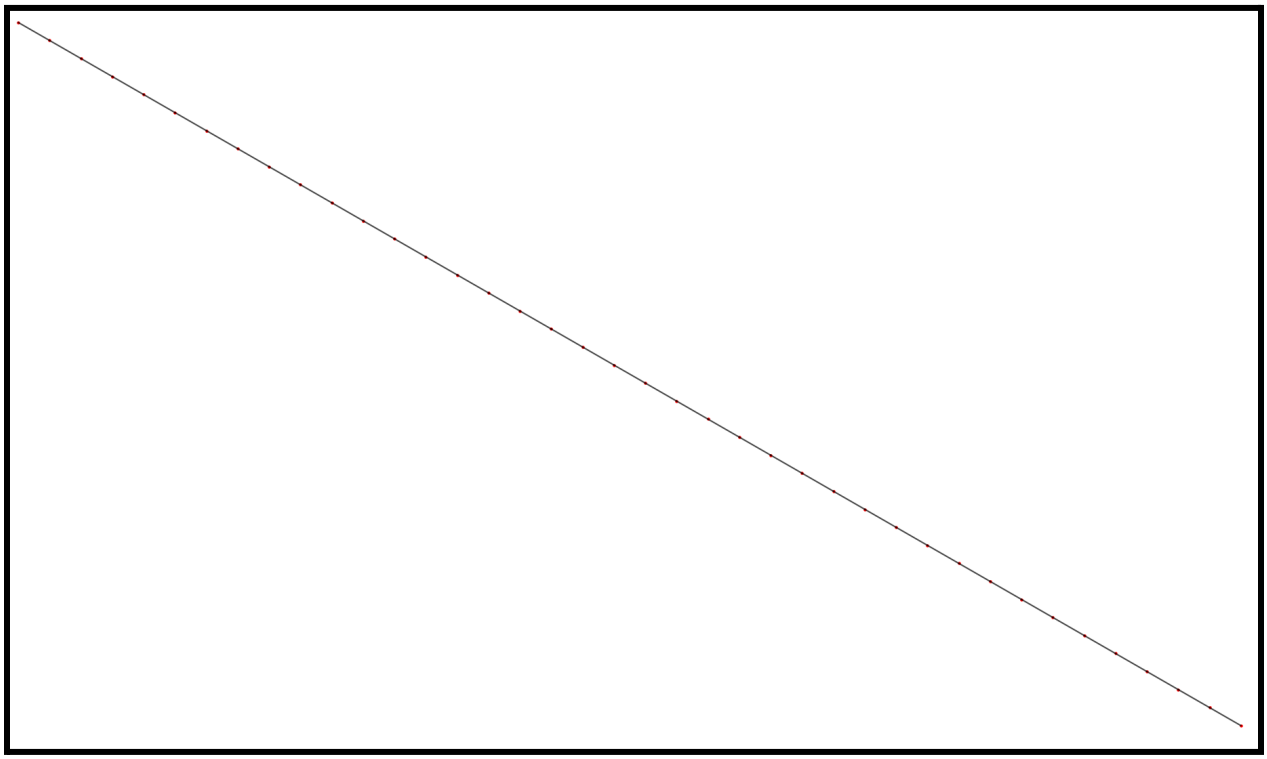
Ratio in the simulation results represent the the number of blocks generated by particular nodes that got into the longest chain to the total number of blocks generated by those nodes.

During our experiments, we observed that the transaction generation time doesn't have a significant impact on the final blockchain, as expected. Reducing it only increases the time required for completing the simulation. Therefore, we focus solely on factors that lead to significant changes in the final blockchain, which are presented below.

1) **Simulation 1: [Normal Parameters]**

python3 simulator.py --total_peers 100 --z0_percent 50 --z1_percent 50 --Tx 5000 --Tb 600000 --To 24000000

```
Total blocks in blockchain:  40
Longest chain size:  40
Low Speed, Low CPU - Longest chain: 3, Total: 3, Ratio: 1.0
Low Speed, High CPU - Longest chain: 18, Total: 18, Ratio: 1.0
High Speed, Low CPU - Longest chain: 2, Total: 2, Ratio: 1.0
High Speed, High CPU - Longest chain: 16, Total: 16, Ratio: 1.0
Branch: Total branches: 0, Max Branch length: 0, Min Branch length: 0, Average branch length: 0
```



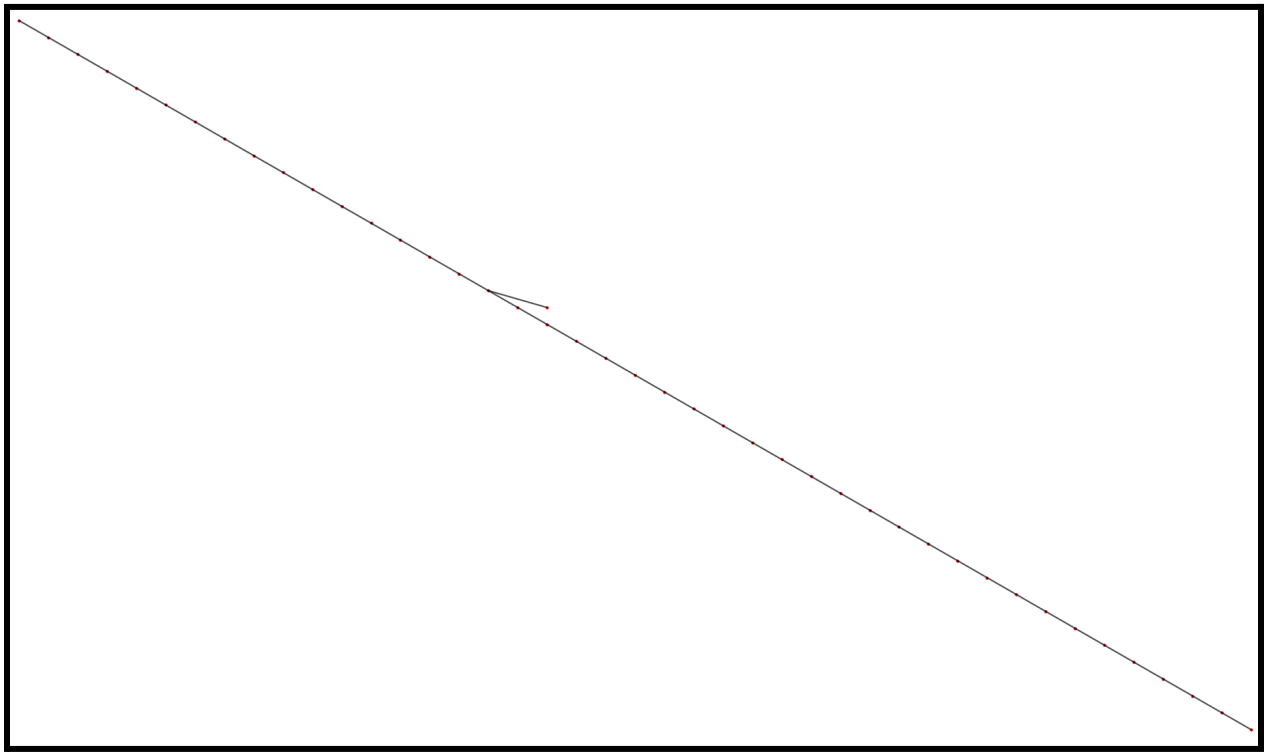
As expected, no forks when the parameters are normal, i.e. Block generation time = 10 min (600000 ms) and Transaction generation time = 5 sec (5000 ms). The reason for absence of forks is stated in above section as well. We can also see that for all types of nodes, the ratio is 1, since all the blocks that were generated got into the longest chain. High CPU nodes generate ~7x more blocks than Low CPU nodes in the longest chain.

2) **<u>Simulation 2:</u> [Reducing Block Generation Time]**

python3 simulator.py --total_peers 100 --z0_percent 50 --z1_percent 50 --Tx 5000 --Tb 25000 --To 1000000

Results:

```
Total blocks in blockchain:  44
Longest chain size:  43
Low Speed, Low CPU - Longest chain: 2, Total: 2, Ratio: 1.0
Low Speed, High CPU - Longest chain: 16, Total: 16, Ratio: 1.0
High Speed, Low CPU - Longest chain: 2, Total: 2, Ratio: 1.0
High Speed, High CPU - Longest chain: 22, Total: 23, Ratio: 0.96
Branch: Total branches: 1, Max Branch length: 1, Min Branch length: 1, Average branch length: 1.0
```

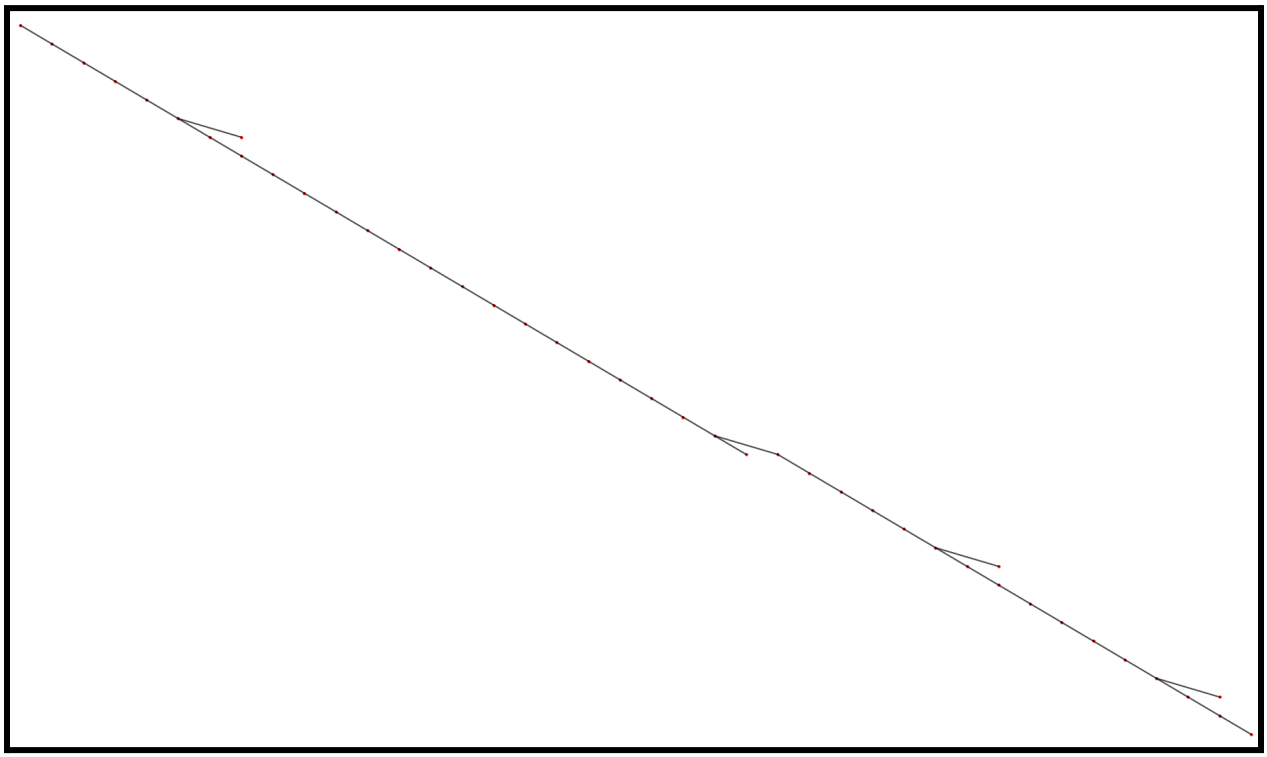


When the block generation time is reduced to 25 seconds, minimal forks are observed in the blockchain. This phenomenon arises due to a reduced block generation time, which is no longer significantly higher than the network delay as in the previous scenario. Consequently, occasional forks may occur in the chain, particularly when a node successfully mines a block and completes its proof of work before receiving information about a newly generated block with the same preceding block ID. The ratio of all node types remains nearly equal to 1. High CPU nodes generate ~10x more blocks than Low CPU nodes in the longest chain.

3) **Simulation 3:** **[Further Reducing Block Generation Time]**
python3 simulator.py --total_peers 100 --z0_percent 50 --z1_percent 50 --Tx 5000 --Tb 10000 --To 400000

Results:

```
Total blocks in blockchain:  43
Longest chain size:  39
Low Speed, Low CPU - Longest chain: 2, Total: 2, Ratio: 1.0
Low Speed, High CPU - Longest chain: 19, Total: 20, Ratio: 0.95
High Speed, Low CPU - Longest chain: 2, Total: 2, Ratio: 1.0
High Speed, High CPU - Longest chain: 15, Total: 18, Ratio: 0.83
Branch: Total branches: 4, Max Branch length: 1, Min Branch length: 1, Average branch length: 1.0
```

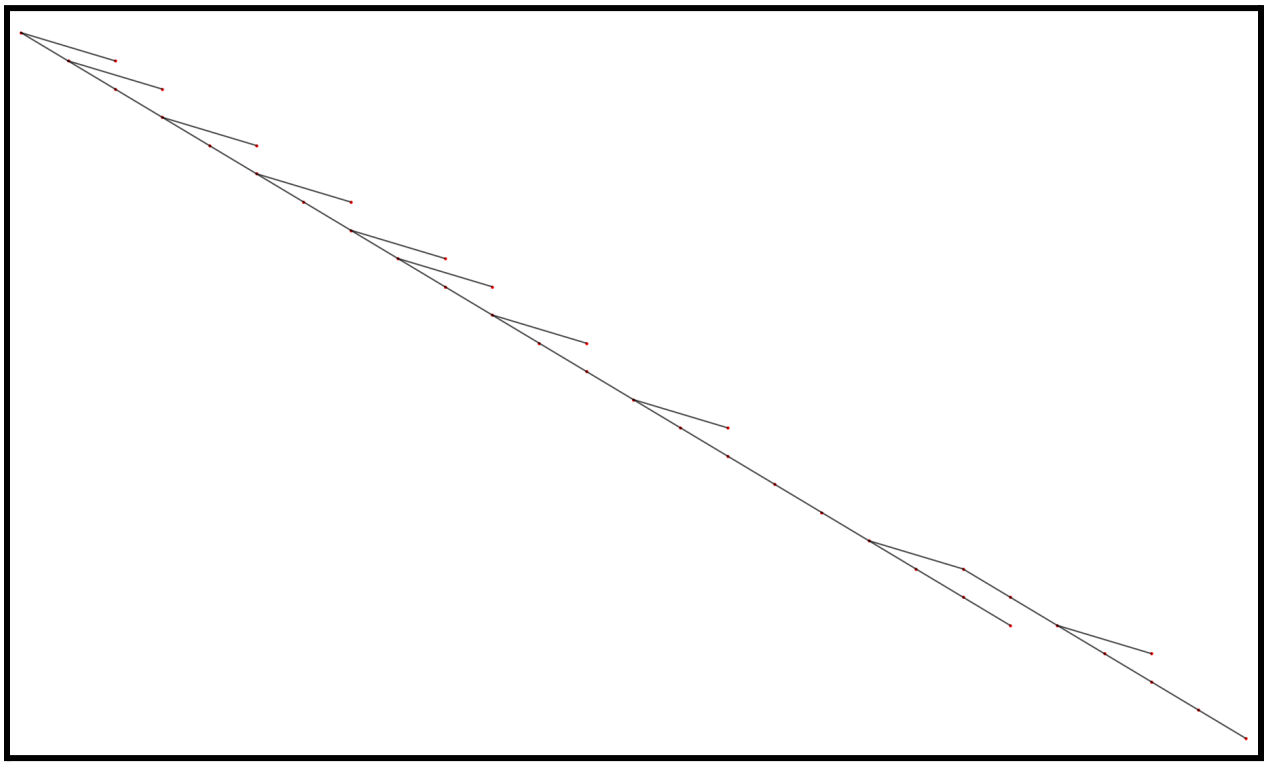


As we shorten the time it takes to generate blocks, we see more instances of forking in the chain. Additionally, nodes with high CPU power produce about ~9x more blocks than nodes with low CPU power in the longest chain. The ratios for nodes with high CPU power and high speed, as well as those with high CPU power and low speed, decrease slightly as all node types compete to generate the next block quickly. If we run the simulation for a longer time, say, until 500 block generations, these ratios will likely decrease even more, possibly bringing the ratios for nodes with low CPU power and high speed, and those with low CPU power and low speed, close to zero.

4) **<u>Simulation 4:</u> [Further Reducing Block Generation Time]**
python3 simulator.py --total_peers 100 --z0_percent 50 --z1_percent 50 --Tx 5000 --Tb 1000 --To 40000

Results:

```
Total blocks in blockchain:  38
Longest chain size:  26
Low Speed, Low CPU - Longest chain: 2, Total: 3, Ratio: 0.67
Low Speed, High CPU - Longest chain: 12, Total: 18, Ratio: 0.67
High Speed, Low CPU - Longest chain: 0, Total: 0, Ratio: No Block
High Speed, High CPU - Longest chain: 11, Total: 16, Ratio: 0.69
Branch: Total branches: 10, Max Branch length: 3, Min Branch length: 1, Average branch length: 1.2
```

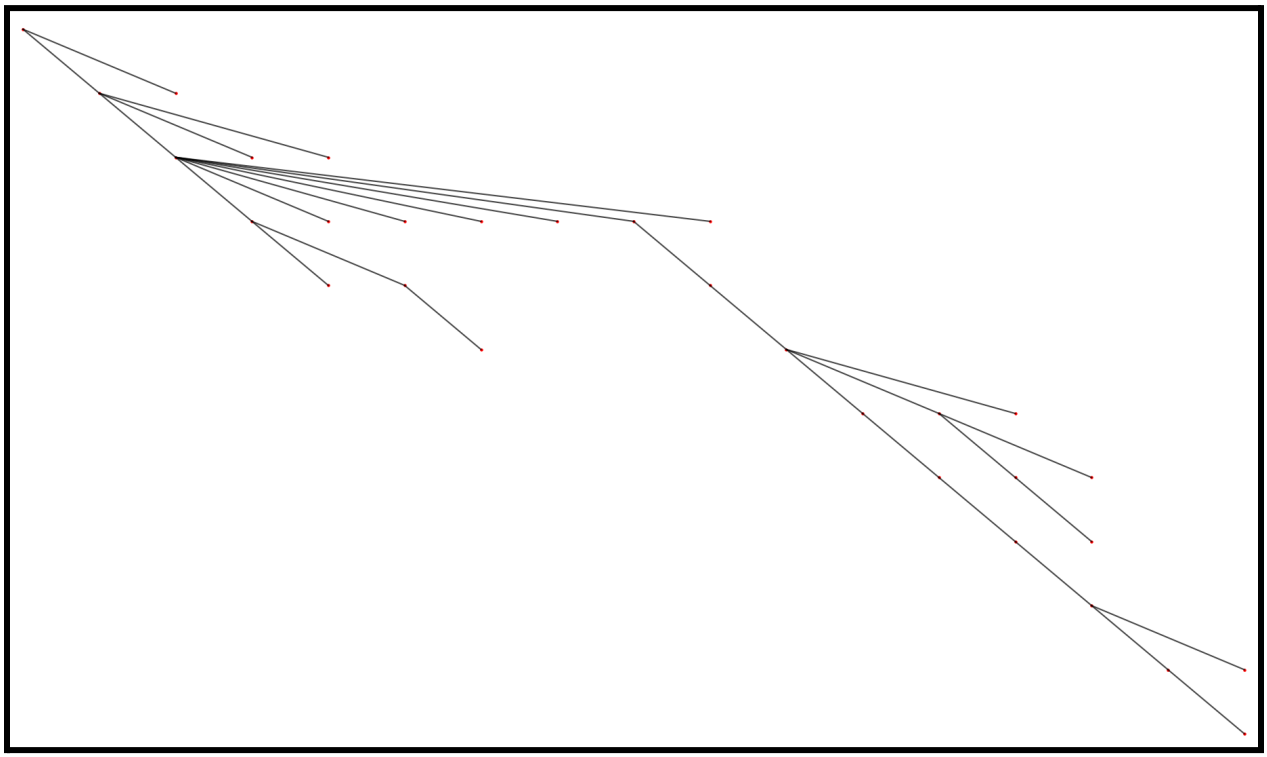


When the block generation time is reduced to 1 second, nearly matching the network delay, numerous forks occur in the chain, resulting in a significantly reduced size of the longest chain. In this scenario, nodes with high CPU capacity have generated approximately ~11x more blocks than nodes with low CPU capacity. Interestingly, the average branch length exceeds 1, indicating instances where a branch with more than one block becomes orphaned. This occurs when multiple nodes in close proximity continue to mine on their local chains, receiving a longer chain from the network at a later time.

5) **Simulation 5: [Minimum Block Generation Time]**
python3 simulator.py --total_peers 100 --z0_percent 50 --z1_percent 50 --Tx 50 --Tb 200 --To 8000

Results:

```
Total blocks in blockchain:  30
Longest chain size:  12
Low Speed, Low CPU - Longest chain: 0, Total: 0, Ratio: No Blocks
Low Speed, High CPU - Longest chain: 6, Total: 14, Ratio: 0.43
High Speed, Low CPU - Longest chain: 0, Total: 3, Ratio: 0.0
High Speed, High CPU - Longest chain: 5, Total: 12, Ratio: 0.42
Branch: Total branches: 14, Max Branch length: 3, Min Branch length: 1, Average branch length: 1.43
```



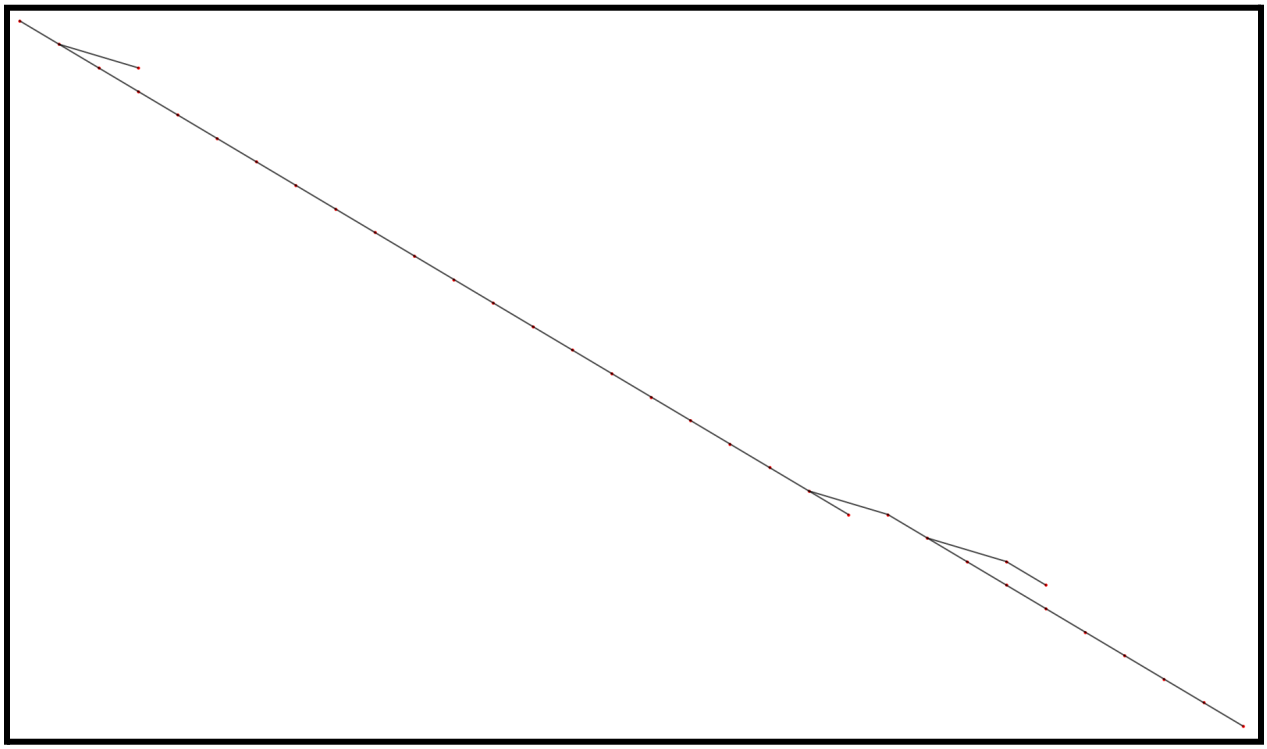
Reducing the block generation time to 0.2 seconds introduces a significant amount of forking in the blockchain. The ratios for low CPU nodes decrease to 0, and the ratios for high CPU nodes also decrease significantly. All blocks in the network are generated by high CPU nodes. Additionally, the average branch length increases as expected. In the longest chain, all blocks are of igh CPU nodes.

6) **Simulation 6: [Increasing Network Latency, Normal Block Generation time]**
python3 simulator.py --total_peers 100 --z0_percent 50 --z1_percent 50 --Tx 5000 --Tb 600000 --To 24000000

Latency delay added **50 seconds** (explicitly in the code and not through command line argument)

Results:

```
Total blocks in blockchain:  35
Longest chain size:  31
Low Speed, Low CPU - Longest chain: 0, Total: 0, Ratio: No Blocks
Low Speed, High CPU - Longest chain: 14, Total: 18, Ratio: 0.78
High Speed, Low CPU - Longest chain: 1, Total: 1, Ratio: 1.0
High Speed, High CPU - Longest chain: 15, Total: 15, Ratio: 1.0
Branch: Total branches: 3, Max Branch length: 2, Min Branch length: 1, Average branch length: 1.33
```



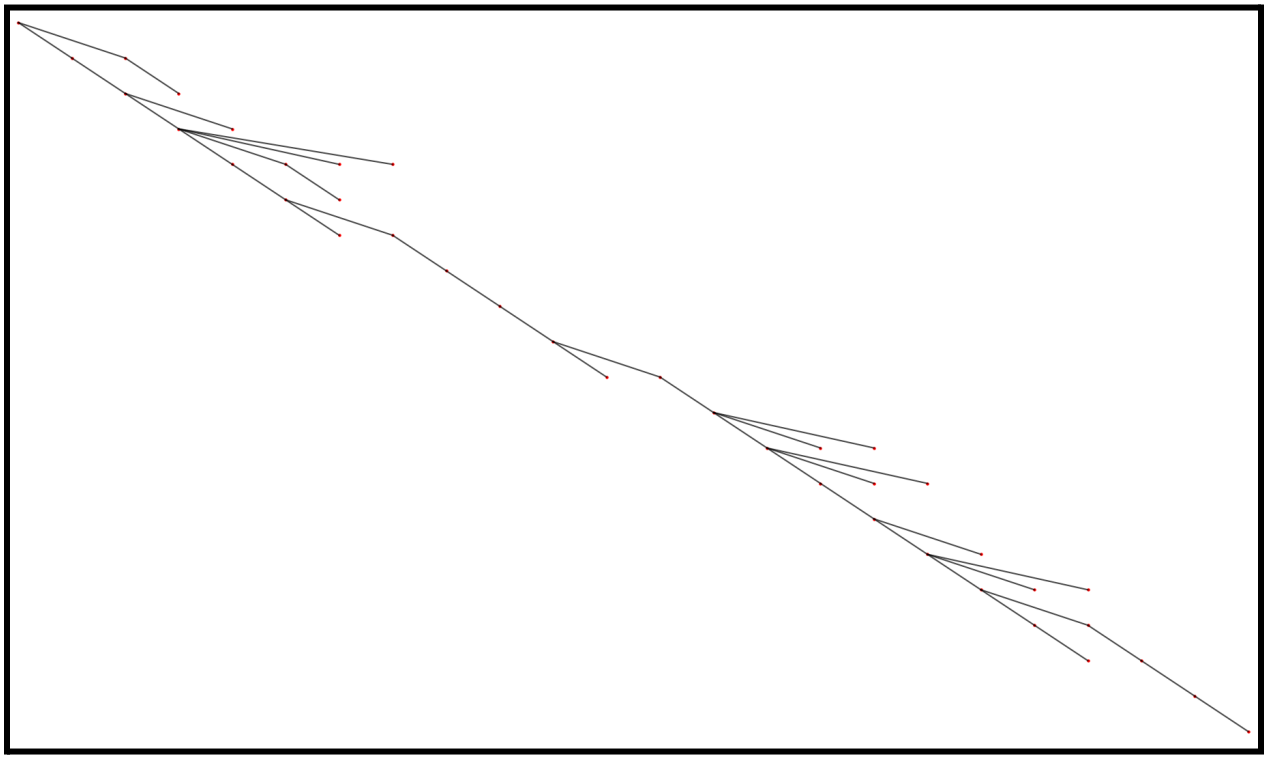
Under normal conditions, blockchains shouldn't show any forks due to the network delay being approximately 1 second, significantly shorter than the 600 seconds block mining time. However, with the added network delay of 50 seconds, minimal forks are now observed. When a peer transmits a block to another peer, it takes 50 seconds on just one link, increasing the likelihood of forks. The Low CPU nodes are able to add only 1 block in the longest chain and all other blocks are added by High CPU nodes.

7) **<u>Simulation 7:</u> [Further increasing Network Latency, Normal Block Generation Time]**
python3 simulator.py --total_peers 100 --z0_percent 50 --z1_percent 50 --Tx 5000 --Tb 600000 --To 24000000

Latency delay added **200 seconds** (explicitly in the code and not through command line argument)

Results:

```
Total blocks in blockchain:  39
Longest chain size:  21
Low Speed, Low CPU - Longest chain: 0, Total: 1, Ratio: 0.0
Low Speed, High CPU - Longest chain: 11, Total: 17, Ratio: 0.65
High Speed, Low CPU - Longest chain: 0, Total: 1, Ratio: 0.0
High Speed, High CPU - Longest chain: 9, Total: 19, Ratio: 0.47
Branch: Total branches: 15, Max Branch length: 2, Min Branch length: 1, Average branch length: 1.2
```



On Further increasing the network delay to 200 seconds, the forks increased even more. It can be seen that all blocks in longest chain are from high cpu nodes. Total number of branches also got increased compared to last time due to the high propagation delay.

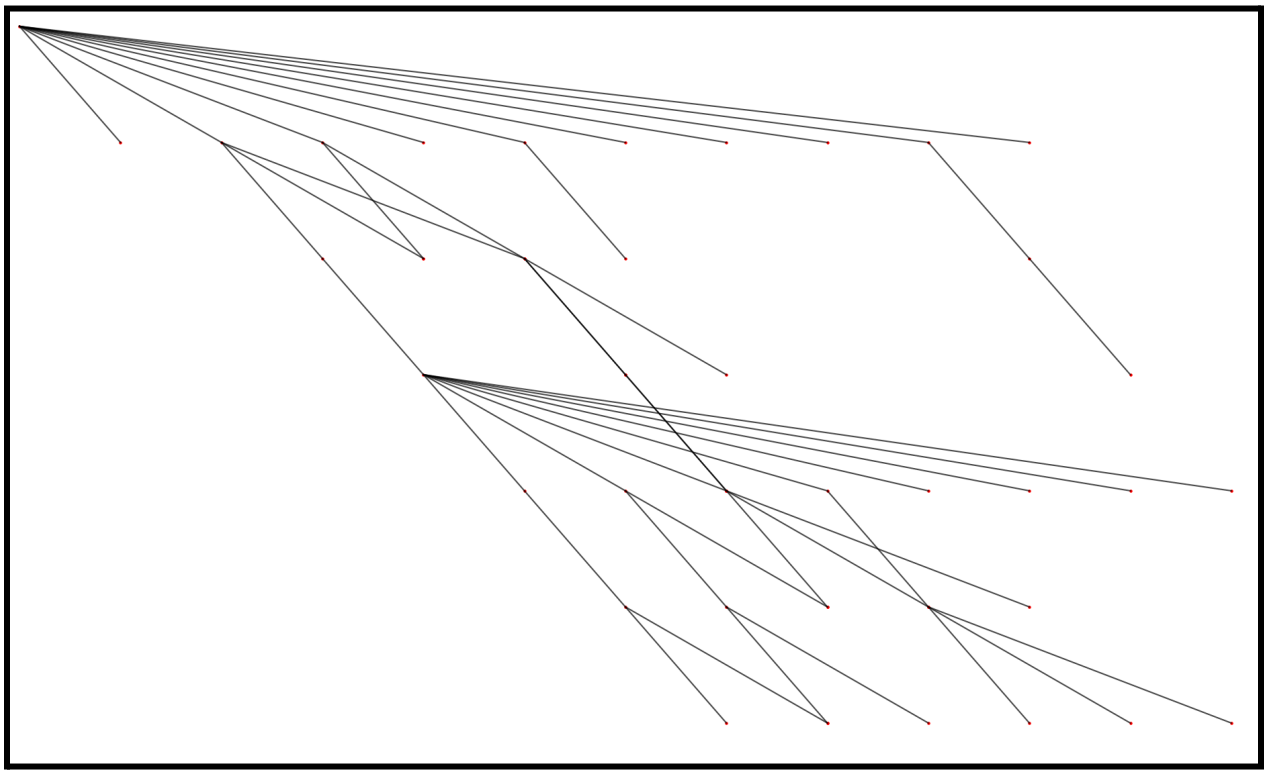
8) **Simulation 8:**

python3 simulator.py --total_peers 100 --z0_percent 50 --z1_percent 50 --Tx 5000 --Tb 600000 --To 24000000

Latency delay added **2000 seconds** (explicitly in the code and not through command line argument)

Results:

```
Total blocks in blockchain:  47
Longest chain size:  7
Low Speed, Low CPU - Longest chain: 0, Total: 3, Ratio: 0.0
Low Speed, High CPU - Longest chain: 4, Total: 25, Ratio: 0.16
High Speed, Low CPU - Longest chain: 0, Total: 1, Ratio: 0.0
High Speed, High CPU - Longest chain: 2, Total: 17, Ratio: 0.12
Branch: Total branches: 27, Max Branch length: 4, Min Branch length: 1, Average branch length: 1.89
```



On significantly increasing the propagation delay to 2000 seconds, we can observe that many blocks are attached to any given block. This is because the time to propagate a block to the network is now three times greater than the time it takes to create a block. Therefore, the observed significant increase in forks is justified. Also the average branch length is greater than any of the above cases.