CS765 Project Part-1 : Report

Atharva Bendale
22B0901
atharvaab@cse.iitb.ac.in

Vardan Verma
22B0902
22B0902@iitb.ac.in

Vishal Bysani
22B1061
vishalbysani@cse.iitb.ac.in

August 9, 2025

Justification for Using the Exponential Distribution for Transaction Inter-Arrival Time Sampling

Transaction generation is memoryless because the probability of a new transaction occurring in the next instant is independent of when the last transaction occurred. Let T denote the random variable representing the inter-arrival time of transactions. Consider a small time interval Δ.
The probability of a transaction occurring within any Δ is given by:

$p = k\Delta$ [Due to memorylessness, this probability remains same for any Δ time interval]

The probability that no transaction occurs within n such intervals (i.e., the inter-arrival time exceeds $n\Delta$) is:

$$\mathbb{P}(T > n\Delta) = (1-p)^n = (1-k\Delta)^n.$$

Setting $t = n\Delta$, we rewrite the expression as:

$$\mathbb{P}(T > t) = \left(1 - \frac{kt}{n}\right)^n.$$

Taking the limit as $n \to \infty$ (i.e., for an infinitesimally small Δ), we obtain:

$$\mathbb{P}(T > t) = \lim_{n\to\infty} \left(1 - \frac{kt}{n}\right)^n = \exp(-kt)$$

$$\mathbb{P}(T \leq t) = 1 - \exp(-kt)$$

This is the cumulative distribution function (CDF) of an exponential distribution with mean $1/k$. Therefore, the exponential distribution is a natural choice for modeling the interarrival times of transactions.

Justification for the Inverse Relationship Between Queuing Delay and Link Speed

The queuing delay d_{ij} represents the time a message spends waiting to be forwarded over the link from node i to node j. In our simulation, the mean of d_{ij} is defined to be

$$\text{Mean}(d_{ij}) = \frac{96\,\text{kbits}}{c_{ij}},$$

where c_{ij} is the link speed in bits per second.

Reasoning

1. **Service Rate Analogy:** In queuing theory, the service rate (the rate at which packets are transmitted) is analogous to the link speed c_{ij}. A higher link speed means that the link can process (i.e., transmit) data more quickly. Thus, when c_{ij} increases, packets spend less time in the queue waiting for transmission.

2. **Proportional Relationship:** By defining the mean queuing delay as inversely proportional to c_{ij}, the simulation realistically mimics the behavior of networks: faster links incur lower delays, while slower links suffer higher delays. This ensures that the simulated latency, given by

   $$\text{Latency} = \rho_{ij} + \frac{|m|}{c_{ij}} + d_{ij},$$

   accurately reflects the interplay between propagation delay, transmission delay, and queuing delay.

In summary, the mean of the queuing delay d_{ij} is inversely related to the link speed c_{ij} because a higher link speed implies a higher service rate. This leads to quicker transmission of packets and consequently shorter queuing times. This design choice in the simulation reflects realistic network behavior where a faster link leads to a more efficient handling of traffic and a reduction in waiting times.

Explanation for the choice of mean for block inter-arrival time in PoW simulation

Let I be our parameter h_k be the hashing power of k^{th} peer, the block inter-arrival for each such peer is sampled from an exponential distribution, which was derived due to the process being memoryless with probability of success $\beta_k = \frac{h_k}{I}$. Thus:

$$Pr(T_k < \delta) = \beta_k \cdot \delta$$

Here δ is an infinitesimally small period of time.
Let n be the number of miners, then the probability that atleast one block is generated by any of the peers is:

$$Pr\left(T_1 < \delta \ or \ T_2 < \delta \ or \ \dots\right) = 1 - \prod_{1<k<n}\left(1 - Pr\left(T_k < \delta\right)\right)$$

$$Pr\left(T_1 < \delta \ or \ T_2 < \delta \ or \ \dots\right) = 1 - \prod_{1<k<n}\left(1 - \beta_k \cdot \delta\right)$$

As higher powers of δ are negligible

$$Pr\left(T_1 < \delta \ or \ T_2 < \delta \ or \ \dots\right) \approx \sum_{1<k<n}\left(\beta_k \cdot \delta\right)$$

As $\Sigma\left(h_k\right) = 1$

$$Pr\left(T_1 < \delta \ or \ T_2 < \delta \ or \ \dots\right) \approx \frac{\delta}{I}$$

Then we can derive an exponential distribution for this:

$$Pr\left(T_1 < x \ or \ T_2 < x \ or \ \dots\right) \approx e^{-\frac{x}{I}}$$

This means that I is actually the mean block generation time for the blockchain as a whole. In real bitcoin blockchain, the value of this is maintained to be **10 min**.

Analysis

All of the experiments in the following are performed with parameters $\mathbf{z_0 = 0.6}$, $\mathbf{z_1 = 0.4}$, **Ttx = 5 sec** and **50 peers** unless specified otherwise.

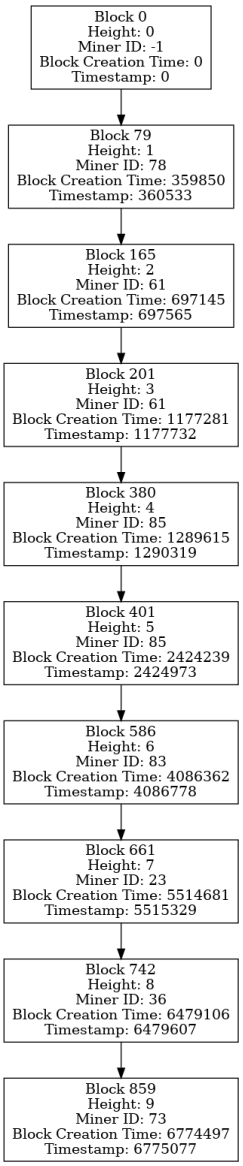
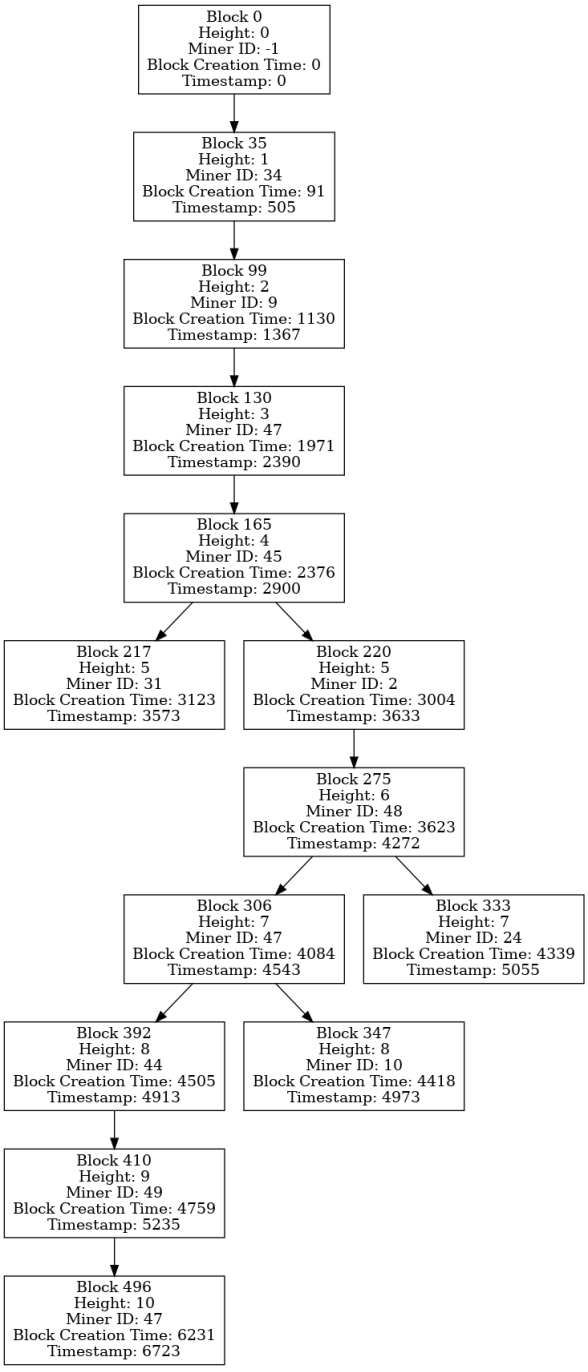


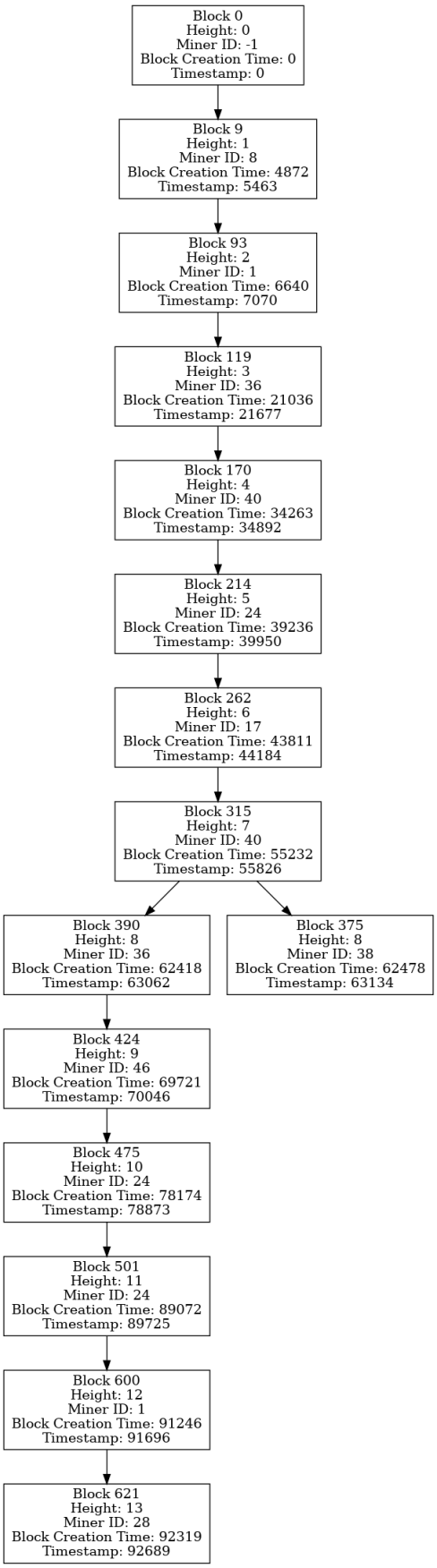
Figure 1: Blockchain Tree at $I = 600$ sec

At **I = 600 sec** and **TTx = 5 sec**, we observed a straight line blockchain for all miners with no forks. Every miner's generated block got into the longest chain. Hence we have conducted experiments on **I** values much lower than 600sec like 50 sec, 20 sec, 10 sec, etc to get more forks and have some orphaned blocks, because these times are almost comparable to the 0.75 sec upper limit of latency between two peers.

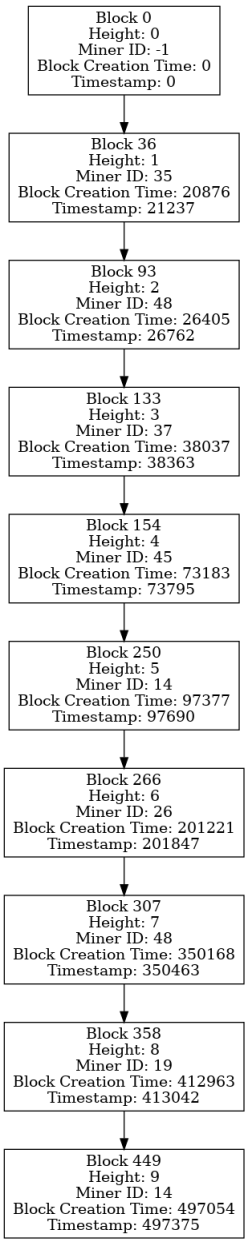
2.1 BlockChain Tree vs Average Inter-Arrival Time (I)



I = 0.6 sec



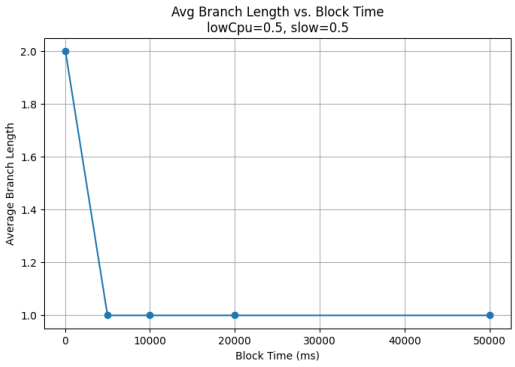
I = 6 sec

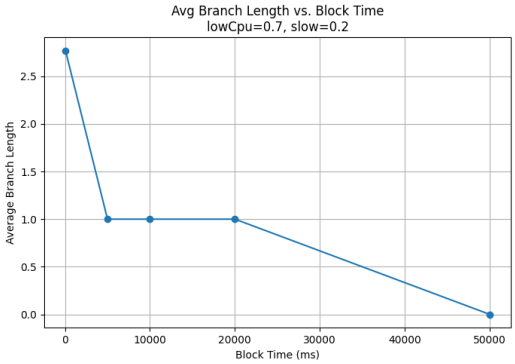


I = 60 sec

When blocks are generated too quickly, nodes in the network may not receive and verify a new block before another competing block is found. This leads to multiple miners solving blocks almost simultaneously, causing more frequent forks. A fast block generation rate may exceed the network's ability to propagate blocks quickly enough. Nodes might receive different versions of the blockchain, leading to temporary chain splits or forks until consensus is reestablished.

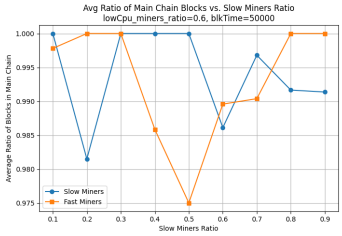
2.2 Average Branch Length vs Block Inter-Arrival Time



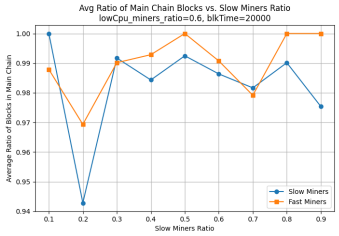


The average branch length decreases as block interarrival time increases. This inverse relationship occurs because blocks require more time to form at higher interarrival times, making it less likely for two peers to generate and release blocks with the same parent simultaneously. Consequently, this leads to fewer forks overall and shorter fork lengths when they do occur.

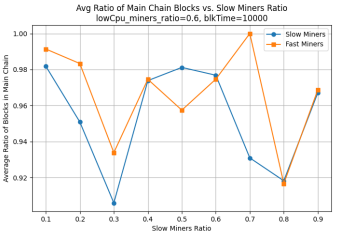
2.3 Average Ratio of Blocks in Main Chain & in the Tree vs fraction of Slow Miners (For Slow and Fast Miners)



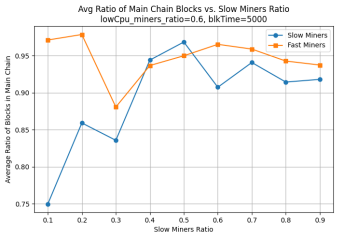
I = 50 sec



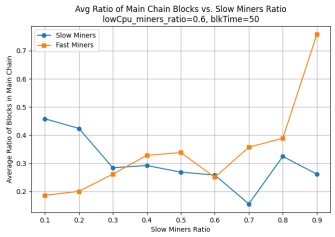
I = 20



I = 10 sec



I = 5 sec

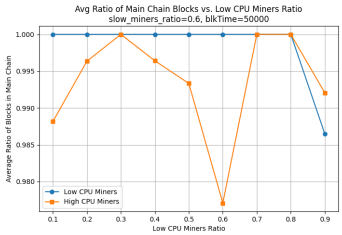


I = 0.05 sec

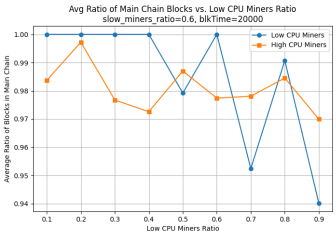
Here we observe the following:

- The ratio of blocks of miner in main chain to total blocks in the tree decreases as block inter arrival time decreases, this is because of increase of forks as discussed in previous part
- Fast miners propagate their blocks quicker, making it more likely that their blocks reach the majority first and remain in the main chain.
- Slow miners face a higher risk of their blocks being orphaned as competing blocks may reach the network earlier.
- Moreover,, fast miners tend to have a better success rate because if they have some fast neighbor, information is propagated quickly to it on average as compared to slow miners, and hence fast miners can switch to the longest chain (if any switch is there) with much less wastage of already mined blocks.

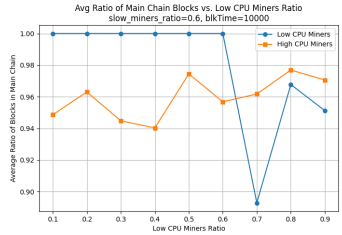
2.4 Average Ratio of Blocks in Main Chain & in the Tree vs fraction of Low-CPU Miners (For Low-CPU and High-CPU Miners)



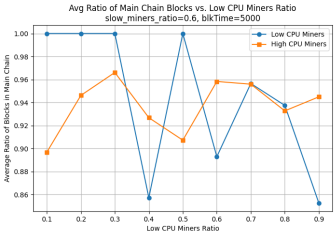
I = 50 sec



I = 20 sec



I = 10 sec



I = 5 sec

Here we observe that:

- Longer block inter-arrival times result in a higher main chain block ratio, minimizing forks and orphaned blocks. Shorter block times (10 sec) increase forking due to network delays and contention among miners.
- When slow miners control a significant portion of the network, the blockchain experiences more frequent forking due to overall increased latency; this causes the ratio to dip down for both types of miners.
- In general, high CPU miners produces more blocks than slow CPU miners, and in case of lower block inter-arrival times, higher ratio of high CPU miners' blocks gets rejected which reduced their ratio overall.