# CS765 Assignment 1

Ameya Deshmukh, Mridul Agarwal & Shantanu Welling
210050011, 210050100 & 210010076

Feb 2025

## Part 2

The exponential distribution is commonly used for modeling inter-arrival times in systems like blockchain transactions for the following reasons:

1. **Memoryless Property**: The exponential distribution is the only continuous probability distribution with the memoryless property, i.e. the probability of a transaction arriving in the next time interval is independent of how much time has already passed. This aligns well with the assumption that each peer generates transactions randomly over time.
2. **Poisson Process Modeling**: When interarrival times follow an exponential distribution, the number of transactions generated in a given time period follows a Poisson distribution. This is a standard assumption in queuing and networks where a packet follows the Poisson model for simulating decentralized transaction arrivals.

## Part 5

- Queuing delay depends on transmission capacity
  - Queuing delay represents the time a packet spends waiting in a buffer before transmission.
  - When link speed $c_{ij}$ is **higher**, the transmission rate is faster, so messages spend **less time waiting** in the queue.
  - Conversely, when $c_{ij}$ is **lower**, the transmission rate is slower, leading to **longer queuing delays** as packets accumulate in the buffer.
  - In blockchain peer-to-peer communication, nodes with slow connections experience **higher delays**, affecting block propagation times.
  - This queuing delay formulation ensures that fast nodes propagate messages quickly, while slow nodes introduce **realistic network bottlenecks** where the packet waits in their queue for longer times until its turn arrives and the link is clear for transmitting it.

## Part 7

### Choice of block interarrival time $I$

For a stable blockchain a desirable property is to ensure that the miners are putting transactions in the longest blockchain at a similar rate as of their generation by the peers. A rough heuristic for this could be:

$$\text{Consumption rate} = \sum_{i \in \text{Peers}} \lambda_i f_i \overline{N}_i = \sum_{i \in \text{Peers}} \frac{h_i}{I} f_i \overline{N}_i$$

where for the $i$th peer, $\overline{N}_i$ is the average number of transactions per block generated, $\lambda_i$ is the Poisson parameter for the block hashing process, $f_i$ is the fraction of blocks that get inserted into the longest chain.

For simplifying purposes we assume that $\overline{N}_i = 500$, and $f_i = 0.25$ based on values from the experiments, which gives

$$\text{Consumption rate} \approx 500 \cdot 0.25 \cdot \sum_{i \in \text{Peers}} \frac{h_i}{I} = \frac{125}{I}$$

$$\text{Generation rate} = n \cdot \frac{1}{T_{\text{tx}}}$$

where $n$ is the number of peers

Hence, we aim for values that ensure that

$$\frac{\text{Generation rate}}{\text{Consumption rate}} = \frac{I \cdot n}{T_{\text{tx}} \cdot 125}$$

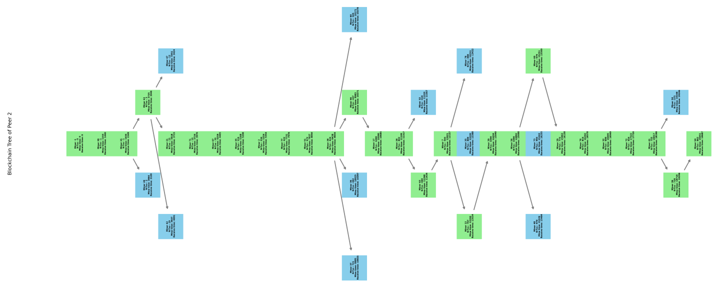
is not far from 1. For eg. for $n = 50, T_{\text{tx}} = 200, I = 600$ the ratio is 1.2.

## Observations from the experiments

We run the simulation for the following parameters

$$(n, z_0, z_1, T_{\text{tx}}, I) \in \{50, 80\} \times \{0.2, 0.5, 0.8\} \times \{0.2, 0.5, 0.8\} \times \{50, 100, 200\} \times \{400, 600\}$$

The trees for all the peers agree to most extent, and the **branches off the longest chain are 1 block long**, going up to 2 blocks rarely.



A tree for parameters $(n = 80, z_0 = 0.2, z_1 = 0.2, T_{\text{tx}} = 50, I = 600)$

## Empirical insights

Defining

$$f_i = \frac{\text{no. of blocks in the longest chain mined by Peer } i}{\text{no. of blocks mined by Peer } i}$$

we consider $\overline{f}$, the mean value of $f_i$, as a measure of the stability of the network, since lower values of $f_i$'s imply more competing forks, more stale branches.

- Nodes with higher CPU power have higher $f_i$

| $(n, z_0, z_1, T_{\text{tx}}, I)$ | **$\overline{f}$ for low, slow** | **$\overline{f}$ for high, slow** | **$\overline{f}$ for low, fast** | **$\overline{f}$ for high, fast** |
|---|---|---|---|---|
| (50, 0.2, 0.5, 100, 400) | 0.14 | **0.50** | 0.10 | **0.58** |
| (50, 0.5, 0.5, 100, 400) | 0.12 | **0.86** | 0.00 | **0.79** |
| (50, 0.8, 0.5, 100, 400) | 0.14 | **0.82** | 0.20 | **0.78** |

  This is a result of these nodes mining much more quickly that the low-CPU nodes.

- Increasing the fraction of slow nodes leads to more stability indicated by higher $\overline{f}$

| $(n, z_0, z_1, T_{\text{tx}}, I)$ | **$\overline{f}$ over all peers** |
|---|---|
| (50, 0.2, 0.5, 100, 400) | 0.21 |
| (50, 0.5, 0.5, 100, 400) | 0.25 |
| (50, 0.8, 0.5, 100, 400) | 0.28 |

  This presumably happens since slow links leads to slower propagation of the blocks in the network which may reduce the number of competing forks.

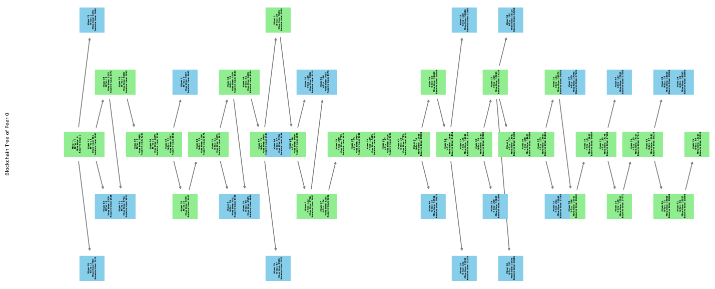
- Increasing the fraction of low-CPU nodes leads to more stability indicated by higher $\overline{f}$

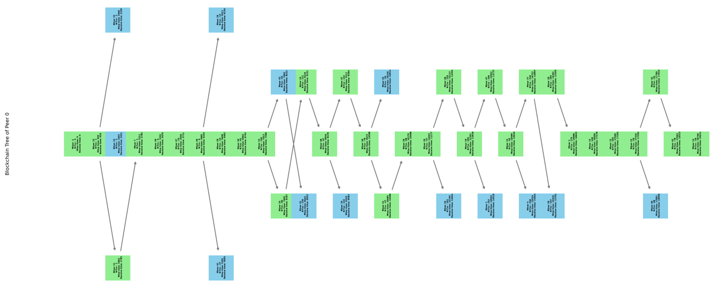
| $(n, z_0, z_1, T_{\text{tx}}, I)$ | **$\overline{f}$ over all peers** |
|---|---|
| (50, 0.5, 0.2, 100, 400) | 0.14 |
| (50, 0.5, 0.5, 100, 400) | 0.25 |
| (50, 0.5, 0.8, 100, 400) | 0.30 |

  Low-CPU blocks become aware of the longest chain faster than they can mine, which leads to termination of mining before a stale branch can be formed.

- Increasing $n$ leads to a wider tree, lower $\overline{f}$

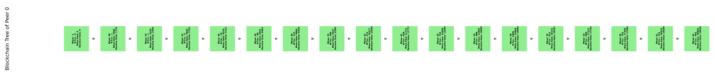
| $n$ | **Average of $\overline{f}$ over all experiments** |
|---|---|
| 50 | 0.21 |
| 80 | 0.18 |

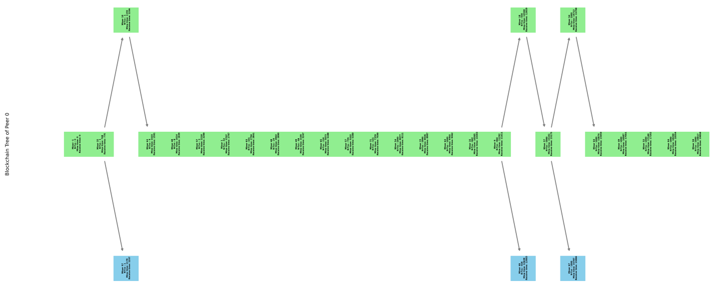

A tree for parameters $(n = 80, z_0 = 0.2, z_1 = 0.2, T_{\text{tx}} = 200, I = 400)$


A tree for parameters $(n = 50, z_0 = 0.2, z_1 = 0.2, T_{\text{tx}} = 200, I = 400)$

This can be explained by more competition in adding to the chain as the number of peers increases.

- High values of both $z_0, z_1$ such as $(0.8, 0.8)$ lead to stable chains with very few forks


A tree for parameters $(n = 50, z_0 = 0.8, z_1 = 0.8, T_{\text{tx}} = 50, I = 600)$


A tree for parameters $(n = 80, z_0 = 0.2, z_1 = 0.2, T_{\text{tx}} = 200, I = 400)$