CS765 Project Part 1 Report

Indian Institute of Technology, Bombay

Ananya Kulashreshtha : 22B0906
Nitin Singh Patel : 22B0974
Tanish Agarwal : 22B0978

August 23, 2024

Q: Why is the exponential distribution used for interarrival time sampling in transaction generation?

A: The exponential distribution is chosen for the following reasons:

1. **Memorylessness Property:** The exponential distribution is the only continuous probability distribution that exhibits the memoryless property. This means that the probability of a new transaction occurring is independent of how much time has already passed since the last transaction. This is useful in modeling decentralized peer-to-peer systems, where transactions arrive randomly and do not depend on prior transactions.

2. **Poisson Process Modeling:** If transactions are generated independently and at a constant average rate, the number of transactions occurring in a fixed time follows a Poisson distribution. The time between consecutive transactions in a Poisson process follows an exponential distribution. Since peer transactions can be modeled as independent arrivals, using an exponential interarrival time is appropriate.

3. **Simple Parameterization:** The exponential distribution is defined by a single parameter, the mean transaction interarrival time T_{tx}. This makes it easy to control the rate of transaction generation in simulations.

4. **Observed in Real-World Systems:** Empirical studies in blockchain networks, distributed systems, and network traffic analysis show that interarrival times of transactions often follow an exponential distribution. This is due to the random nature of user activities and external influences.

5. **Mathematical Tractability:** The exponential distribution is analytically convenient for performance modeling. It enables closed-form solutions for various performance metrics, such as average transaction latency and system load.

Q: Why is the mean of d_{ij} (queuing delay) inversely related to c_{ij} (link speed)?

A: The mean of the queuing delay d_{ij} is chosen to be inversely proportional to the link speed c_{ij} for the following reasons:

1. **Queueing Theory Justification:** In a network system, the queuing delay at a node depends on the rate at which messages arrive and the rate at which they can be processed (transmitted). A higher link speed allows messages to be transmitted faster, reducing queuing delays.

2. **Realistic Network Behavior:** In real-world networks, congestion and queuing delays tend to be higher on slower links because they cannot process messages as quickly as fast links. By setting $\mathbb{E}[d_{ij}] = \frac{96\text{kbits}}{c_{ij}}$, we ensure that slower links experience higher delays, modeling congestion effects more realistically.

3. **Traffic Handling Efficiency:** Faster links can accommodate higher volumes of traffic, reducing the probability of messages waiting in queues. On the other hand, if a link has a lower bandwidth, packets are transmitted at a slower rate, increasing the likelihood of queuing and waiting times.

4. **Network Performance Consistency:** This choice ensures that as link capacities scale up, delays automatically adjust, maintaining a consistent network performance model where faster nodes benefit from lower latency and congestion effects.

****We used different mean block inter-arrival times for different experiments. The reasons for each choice are explained below.****

Insights

All the parameters involving time are in microseconds. We have derived our conclusions from the peer_stats and the block arrival times which we have printed in various files. This can be seen by running our code.

First Experiment

```
--peers 50
--slow_percent 80
--low_cpu_percent 20
--txn_interarrival 5000000
--block_interarrival_time 28000000
--end_time 1200000000
```

- Block interarrival time is 28 seconds and Network propagation time is < 1 second. So, here since these are comparable, we get forks.
- We observe that only high CPU nodes are able to mine blocks.
- The ratio of the number of blocks generated by them in the longest chain of the tree to the total number of blocks they generate at the end of the simulation ranges from 0 to 1. For most of them it is either 0/0 (the 0 in denominator signifies that it did not generate any block in the simulation time) or 1/1 and for some it's between 0 and 1.

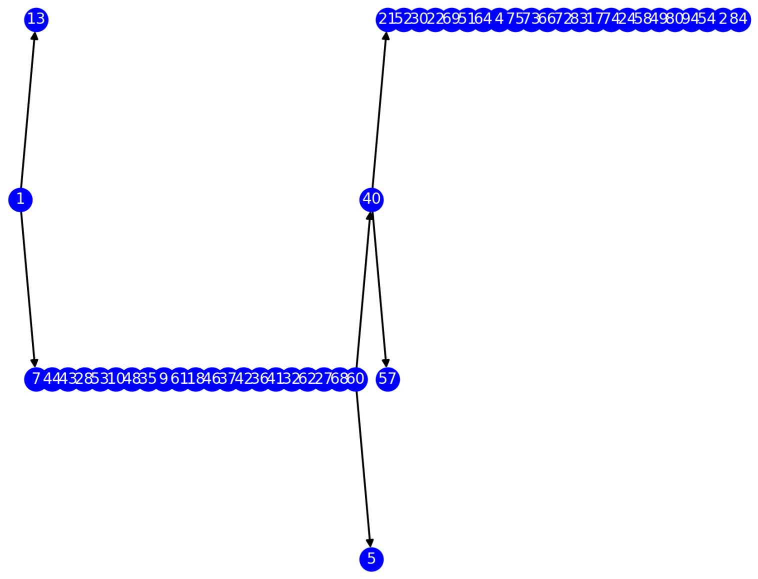

Figure 1: Blockchain tree

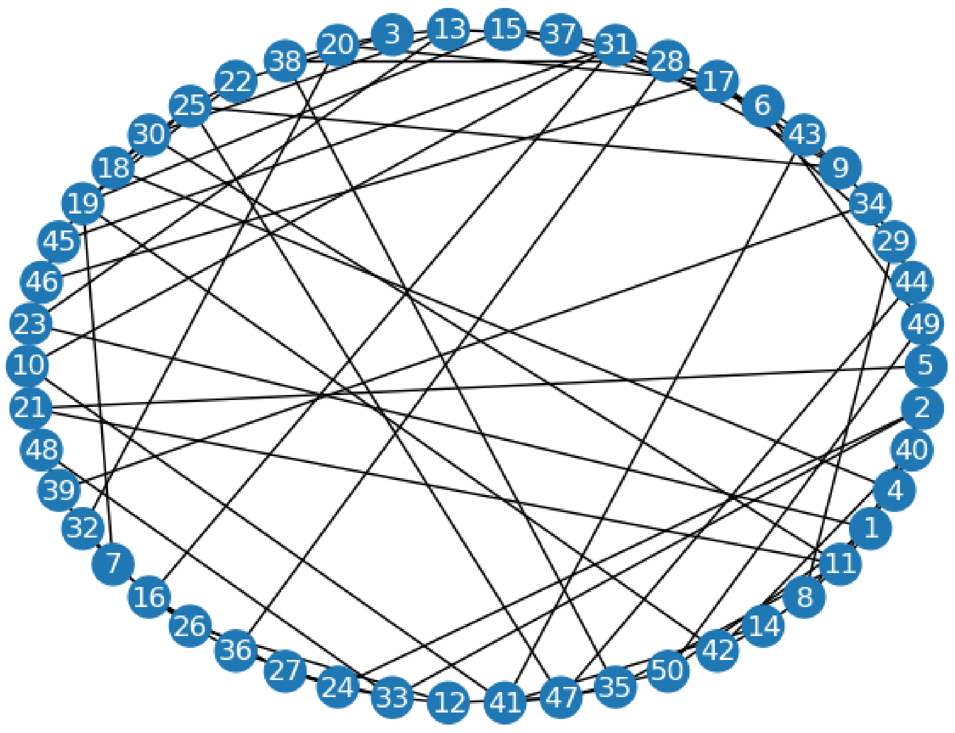

Figure 2: P2P network

Second Experiment

Here, we have further decreased the block interarrival time, so we expect the forks to increase.

```
--peers 50
--slow_percent 80
--low_cpu_percent 20
--txn_interarrival 5000000
--block_interarrival_time 10000000
--end_time 1200000000
```

- Block interarrival time is 10 seconds and Network propagation time is $<$ 1 second. So, here since these are much more comparable, we get more forks.
- We observe that some of the low CPU nodes are able to mine blocks.
- The ratio of the number of blocks generated by them in the longest chain of the tree to the total number of blocks they generate at the end of the simulation ranges from 0 to 1. For most of them it is either 0/0 (the 0 in denominator signifies that it did not generate any block in the simulation time) or 1/1 and for some it's between 0 and 1.

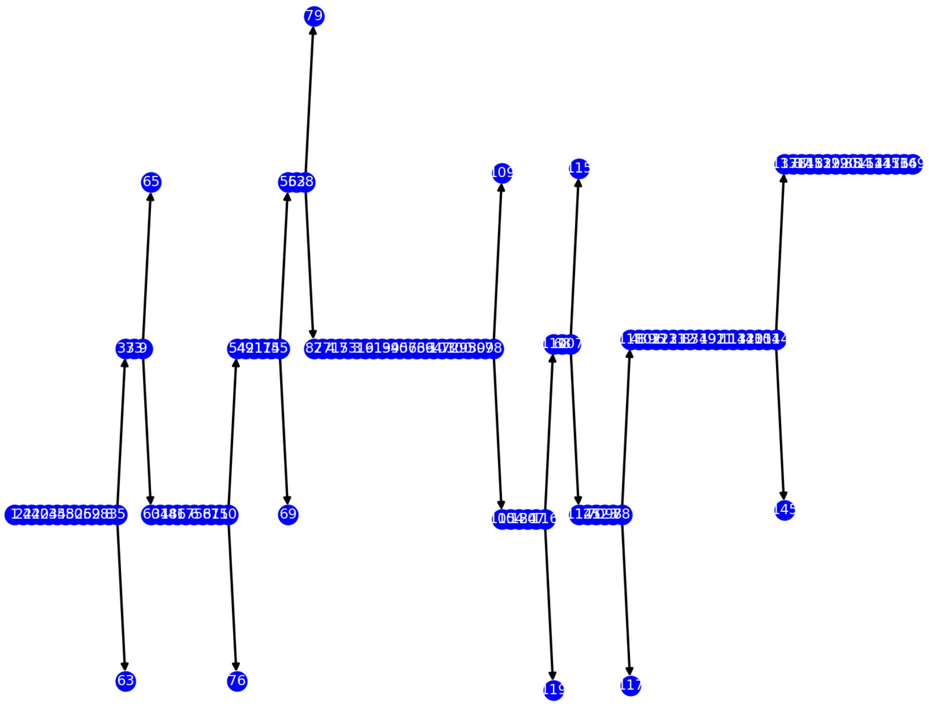

Figure 3: Blockchain tree

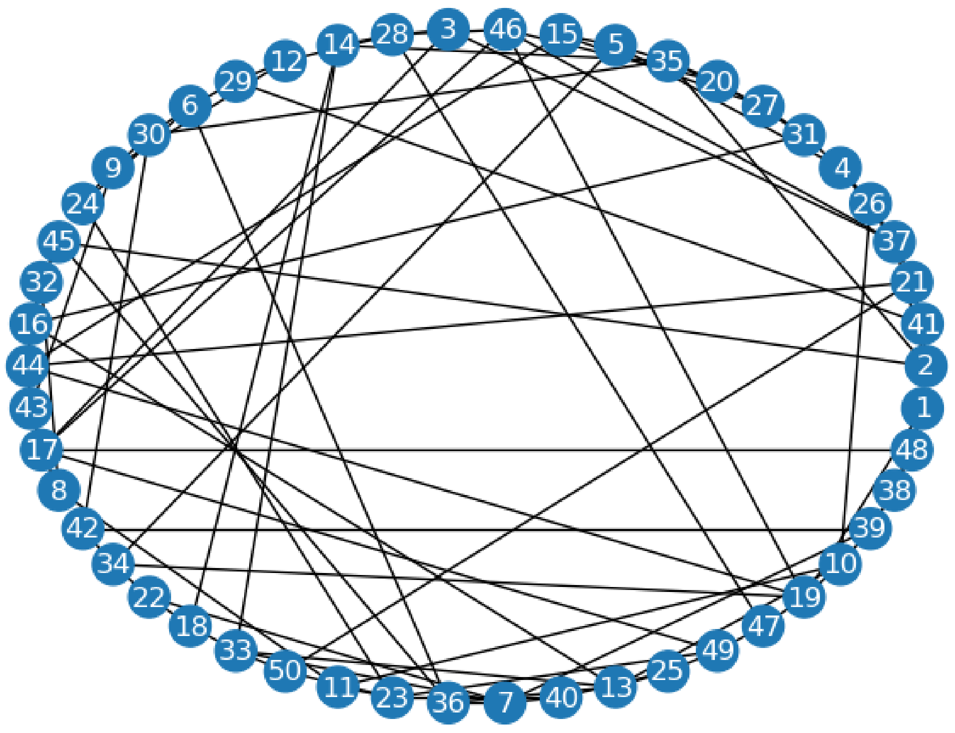

Figure 4: P2P network

Third Experiment

```
--peers 50
--slow_percent 80
--low_cpu_percent 20
--txn_interarrival 5000000
--block_interarrival_time 600000000
--end_time 12000000000
```

- Block interarrival time is 600 seconds (the usual value in bitcoin) and Network propagation time is < 1 second. So, here since these are not comparable, we don't get forks.
- We observe that only high CPU nodes are able to mine blocks. In fact only some of them are able to mine blocks.
- The ratio of the number of blocks generated by them in the longest chain of the tree to the total number of blocks they generate at the end of the simulation ranges from 0 to 1. For all of them it is either 0/0 (the 0 in denominator signifies that it did not generate any block in the simulation time) or 1/1.

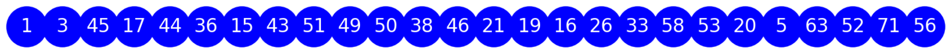

Figure 5: Blockchain tree

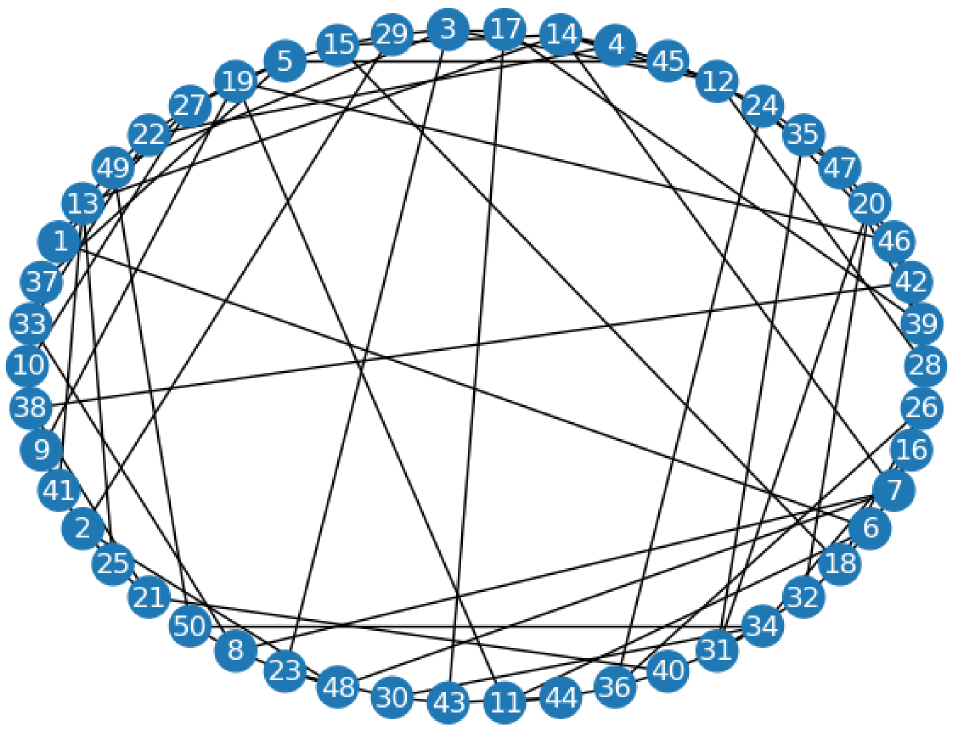

Figure 6: P2P network

Fourth Experiment

```
--peers 50
--slow_percent 50
--low_cpu_percent 50
--txn_interarrival 5000000
--block_interarrival_time 6000000000
--end_time 1200000000
```

- Block interarrival time is 6000 seconds and Network propagation time is < 1 second. So, here since there is a huge disparity, we get no forks.
- We observe that only high CPU nodes are able to generate blocks. In fact only a few of them(2 or 3) are able to generate blocks. We get a very sparse blockchain.
- The ratio of the number of blocks generated by them in the longest chain of the tree to the total number of blocks they generate at the end of the simulation ranges from 0 to 1. For most of them it is 0/0 (the 0 in denominator signifies that it did not generate any block in the simulation time). It is 1/1 for 2 or 3 nodes as described above.

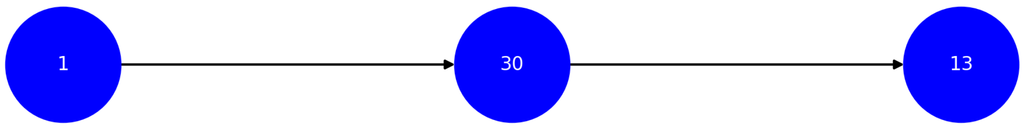

Figure 7: Blockchain tree

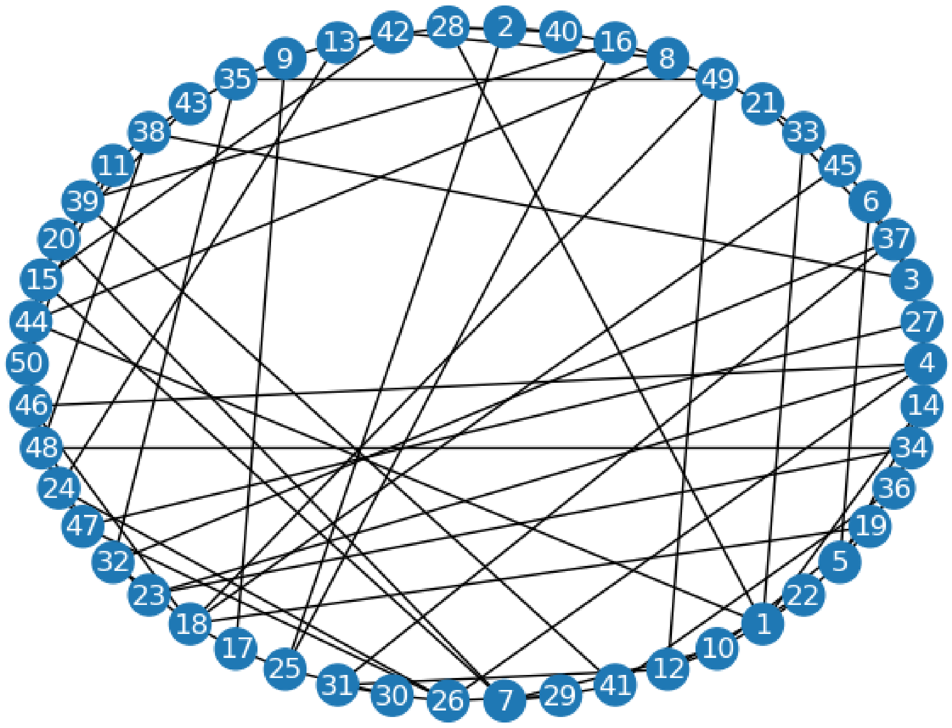

Figure 8: P2P network

Some Other Important Observations/Conclusions

- When the percentage of slow nodes is 100%, even then we do not observe a significant increase in travelling times (time taken to travel from node i to node j given a particular message size) than what we got in the usual setting.
- The probability of forks increases in general on :-
  1. Decreasing I (i.e. block-interarrival time) - More blocks leading to more forks.
  2. Increasing z0 (i.e. the number of slow peers) - More network propagation time, more the chances of some node mining a block amidst the propagation of another block.
  3. Increasing z1 slightly(i.e. the number of high-CPU peers) - Basically, when there will be more miners, then more blocks will be mined, leading to more forks. However, behaviour at z1=0 and z1=100 is equivalent since all nodes have same hashing power. So, increasing z1 first increases the hashing power of some nodes and then decreases the hashing power of those nodes.
- The probability of observing forks is independent of Ttx(transaction interarrival time). Even in the absence of transactions, the blocks containing only coinbase transactions can still be mined.

- Decreasing transaction interarrival time increases the average number of transactions in a block as expected.
- As the ratio of interarrival times to the virtual end time of simulation decreases, in general, more blocks are generated.
- Since the simulation is random, tuning the above mentioned factors above may not work sometimes, that's why we have written ”in general”.