CS 765: Assignment 1
Simulation of a P2P Cryptocurrency Network

Parismita Das	Ankit Kumar Misra	Richeek Das
Roll No: 22M0815	Roll No: 190050020	Roll No: 190260036

15 February 2023

Contents

1 Theoretical Explanations

1.1 Why does transaction interarrival time follow the exponential distribution?

It can be proved mathematically that the transaction interarrival time follows an exponential distribution.

Consider a time interval of size Δ. Since transactions are uniformly likely to occur at any point in time, the probability that a transaction occurs within this interval is proportional to Δ. Let this probability be $\beta\Delta$. Then, assuming $t = 0$ marks a transaction, the probability that the next transaction occurs after n such time intervals is given by

$$\Pr[t > n\Delta] = (1 - \beta\Delta)^n,$$

since it is equal to the probability that there is no transaction in each of n intervals of size Δ. Rewriting this with $T = n\Delta$, we have

$$\Pr[t > T] = \left(1 - \frac{\beta T}{n}\right)^n.$$

To generalize the above probability to arbitrary and continuous values of T, suppose the intervals of size Δ are infinitesimally small, i.e., $\Delta \to 0$, and consequently $n = \frac{T}{\Delta} \to \infty$. We get

$$\lim_{n\to\infty} \left(1 - \frac{\beta T}{n}\right)^n = e^{-\beta T},$$

and thus $\Pr[t > T] = e^{-\beta T}$, which implies $\Pr[t \leq T] = 1 - e^{-\beta T}$.

The latter is exactly the cumulative distribution function (CDF) of an exponential distribution having mean $1/\beta$. Thus, we sample transaction interarrival time from an exponential distribution having mean $T_{tx} = 1/\beta$, for which the probability distribution function (PDF) is

$$p(t = T) = \frac{1}{T_{tx}} e^{-T/T_{tx}}.$$

1.2 Why is the mean of d_{ij} inversely related to c_{ij}?

Note that d_{ij} denotes the queueing delay at node i to forward a message to node j, whereas c_{ij} is the speed of the link between nodes i and j in bits per second. The queueing delay would be high if the link speed is low, and vice versa. To justify this, consider a packet P in a queue having n packets queued before it. The higher the link speed towards the destination, the more packets will be removed from the queue per unit time, and the faster packet P itself will get dequeued. In particular, if each packet has size k bits, and the link speed is c bits per second, the queueing delay for P would be

$$d = \frac{\text{number of bits queued before } P}{\text{link speed}} = \frac{nk}{c}.$$

Thus, we have an inverse proportionality between the mean of queueing delay d_{ij} and link speed c_{ij}.

1.3 Explanation for the choice of T_k

The block mining time T_k is sampled from an exponential distribution having mean I/h_k, where h_k is the fraction of hashing power possessed by node k. The hyperparameter that we can control here is I.

Ideally, we want to minimize forking along the blockchain. We know that Satoshi's Bitcoin assumes network delays of the order ~ 10 s, and tries to maintain an average block mining time ~ 10 minutes. This is because, if block mining time has order sufficiently larger than the network latencies, a single successfully mined block is likely be propagated to all the nodes before multiple nodes succeed in mining. This reduces the likelihood of forks in the blockchain.

The network latencies used in our simulation are of the order $\sim 500 - 1000$ ms. Thus, for an ideal block mining time, we assume $I = 100,000$ ms.

2 Experiments and Insights

2.1 Experiment 1

For this experiment, we use the following parameters:

1. Number of nodes, $n = 10$
2. Percentage of slow nodes: $z_0 = 0.5$
3. Percentage of nodes with low hashing power: $z_1 = 0.5$
4. Mean transaction inter-arrival time: $T_{tx} = 1{,}000$ ms
5. Average block mining time: $I = 100{,}000$ ms
6. Total simulation time: $T = 1{,}000{,}000$ ms

Command: `python main.py -I 100000 -T 1000000 -ttx 1000 -s`

The following blockchain is generated.

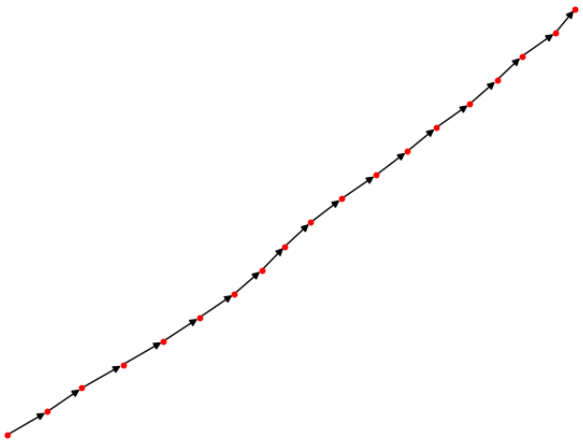

Figure 1: Blockchain generated by Experiment 1

The experiment yields the following statistics:

```
Length of longest chain (including genesis block): 19
Total number of blocks mined: 18
Fraction of mined blocks present in longest chain: 1.0

% blocks in longest chain mined by slow_low node: 0.0
% blocks in longest chain mined by slow_high node: 0.61
% blocks in longest chain mined by fast_low node: 0.0
% blocks in longest chain mined by fast_high node: 0.39

% blocks mined by slow_low node that made it to longest chain: 0.0
% blocks mined by slow_high node that made it to longest chain: 1.0
% blocks mined by fast_low node that made it to longest chain: 0.0
% blocks mined by fast_high node that made it to longest chain: 1.0

No branches were formed!
```

Figure 2: Statistics from Experiment 1

As expected, these parameters form a single chain without branches. The block mining time is clearly large enough to exceed the network latencies, which does not allow forking in the blockchain. Nodes are informed about a miner's success before they are done mining, so it is highly probable that a single successfully mined block is further mined upon by all nodes in the network.

2.2 Experiment 2

We keep the same parameters as Experiment 1, except we reduce I to 1,000 ms and T to 10,000 ms. The block mining time is now closer to the network latency values.

Command: `python main.py -I 1000 -T 10000 -ttx 1000 -s`

The following blockchain is generated.

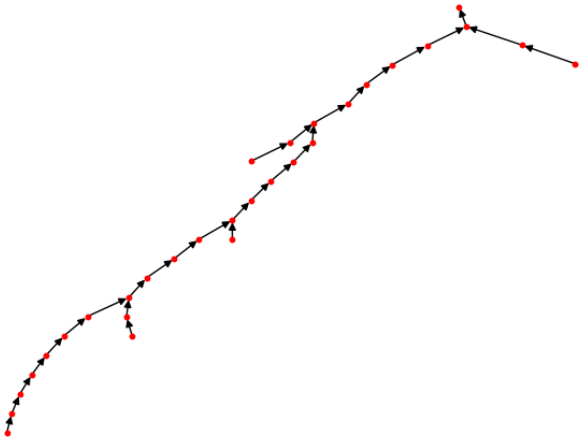

Figure 3: Blockchain generated by Experiment 2

The experiment yields the following statistics:

```
Length of longest chain (including genesis block): 23
Total number of blocks mined: 29
Fraction of mined blocks present in longest chain: 0.759

% blocks in longest chain mined by slow_low node: 0.05
% blocks in longest chain mined by slow_high node: 0.45
% blocks in longest chain mined by fast_low node: 0.09
% blocks in longest chain mined by fast_high node: 0.41

% blocks mined by slow_low node that made it to longest chain: 0.33
% blocks mined by slow_high node that made it to longest chain: 0.83
% blocks mined by fast_low node that made it to longest chain: 1.0
% blocks mined by fast_high node that made it to longest chain: 0.75

Lengths of branches: [2, 1, 2, 2]
Average length of branch: 1.75
```

Figure 4: Statistics from Experiment 2

We now observe that the blockchain has short branches at several places along its length. The block mining time is no longer much larger than the latencies in the network. Thus, some nodes are not informed about successfully mined blocks in time, and they start mining on other blocks, resulting in forks in the network.

2.3 Experiment 3

We keep the same parameters as Experiment 1, except we reduce now I to 100 ms and T to 1000 ms, along with T_{tx} to 10 ms. The block mining time is now likely to be even smaller than the network latencies, which are of the order ~ 500 ms.

Command: `python main.py -I 100 -T 1000 -ttx 10 -s`

The following blockchain is generated.

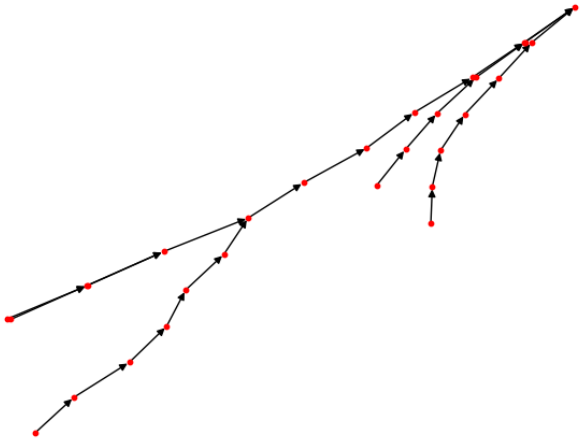

Figure 5: Blockchain generated by Experiment 3

The experiment yields the following statistics:

```
Length of longest chain (including genesis block): 13
Total number of blocks mined: 28
Fraction of mined blocks present in longest chain: 0.429

% blocks in longest chain mined by slow_low node: 0.0
% blocks in longest chain mined by slow_high node: 1.0
% blocks in longest chain mined by fast_low node: 0.0
% blocks in longest chain mined by fast_high node: 0.0

% blocks mined by slow_low node that made it to longest chain: 0.0
% blocks mined by slow_high node that made it to longest chain: 0.63
% blocks mined by fast_low node that made it to longest chain: 0.0
% blocks mined by fast_high node that made it to longest chain: 0.0

Lengths of branches: [3, 6, 5]
Average length of branch: 4.667
```

Figure 6: Statistics from Experiment 3

We now observe that the blockchain still has several forks along its length. However, the important thing to note is that these branches are now much longer than they were in Experiment 2. This shows that network latencies are now considerably large enough to result in different sets of miners working on different branches for a large amount of time before becoming aware of the existence of a longer branch.

2.4 Experiment 4

We keep the same parameters as Experiment 3, except we now increase n to 20 nodes instead of 10.

Command: `python main.py -n 20 -I 100 -T 1000 -ttx 10 -s`

The following blockchain is generated.

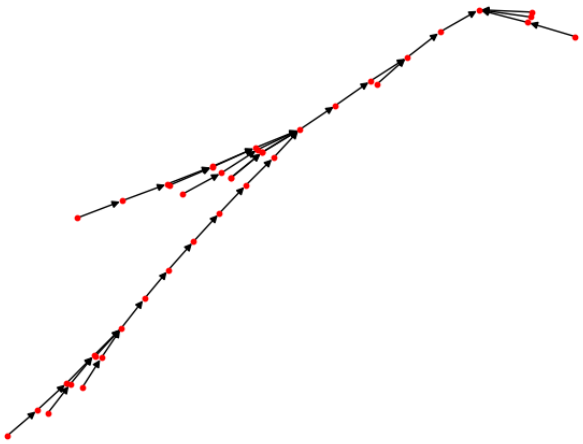

Figure 7: Blockchain generated by Experiment 4

The experiment yields the following statistics:

```
Length of longest chain (including genesis block): 17
Total number of blocks mined: 40
Fraction of mined blocks present in longest chain: 0.4

% blocks in longest chain mined by slow_low node: 0.0
% blocks in longest chain mined by slow_high node: 0.25
% blocks in longest chain mined by fast_low node: 0.0
% blocks in longest chain mined by fast_high node: 0.75

% blocks mined by slow_low node that made it to longest chain: 0.0
% blocks mined by slow_high node that made it to longest chain: 0.27
% blocks mined by fast_low node that made it to longest chain: 0.0
% blocks mined by fast_high node that made it to longest chain: 0.48

Lengths of branches: [2, 1, 2, 3, 2, 5, 2, 1, 2, 1, 1]
Average length of branch: 2.0
```

Figure 8: Statistics from Experiment 4

On doubling the number of nodes in our network, we observe that there is now a lot more forking in the blockchain than there was before. The number of branches, as can be seen in the output screen, is much larger now. This proves that, with more number of nodes competing as miners in the blockchain network, there is more competition and thus a greater likelihood of forks being produced in the chain.

2.5 Experiment 5

We keep the same parameters as Experiment 3, except we now set $z_1 = 0$, i.e., all nodes are high CPU nodes. This corresponds to an equitable distribution of hashing power among the nodes in the network.

Command: `python main.py -z1 0 -I 100 -T 1000 -ttx 10 -s`

The following blockchain is generated.

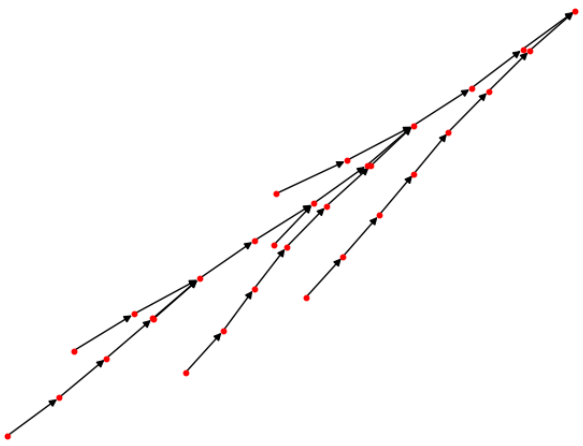

Figure 9: Blockchain generated by Experiment 4

The experiment yields the following statistics:

```
Length of longest chain (including genesis block): 12
Total number of blocks mined: 30
Fraction of mined blocks present in longest chain: 0.367

% blocks in longest chain mined by slow_low node: 0.0
% blocks in longest chain mined by slow_high node: 1.0
% blocks in longest chain mined by fast_low node: 0.0
% blocks in longest chain mined by fast_high node: 0.0

% blocks mined by slow_low node that made it to longest chain: 0.0
% blocks mined by slow_high node that made it to longest chain: 0.52
% blocks mined by fast_low node that made it to longest chain: 0.0
% blocks mined by fast_high node that made it to longest chain: 0.0

Lengths of branches: [1, 2, 1, 2, 6, 7]
Average length of branch: 3.167
```

Figure 10: Statistics from Experiment 4

We now notice that there are more forks (and longer branches) in the blockchain than there were in the original Experiment 3. This is because the nodes now have equal hashing powers, which levels the playing field and gives them all equal chances of mining the next block. It is likely that two or more of them successfully mine their own blocks before learning about each other's blocks.