Simulation of a P2P Cryptocurrency Network
CS 765 Assignment 1

Manish Kumavat - 200050071
Nagendra Singh - 200050081

1 Reason for choosing Exponential Distribution

Let δ be a small time interval, then the probability of generating a transaction in this small time, say p_δ is $\beta\delta$ where β is some constant. Let I denote the interarrival time between transactions, then

$$P(I = n\delta) = (1 - \beta\delta)^{n-1}\beta\delta$$

or,

$$P(I > n\delta) = (1 - \beta\delta)^n$$

Put $x = n\delta$ and let $n \to \infty$, $\delta \to 0$ then

$$P(I > x) = (1 - \frac{\beta x}{n})^n$$

which goes to $e^{-\beta x}$. The PDF of this distribution is the exponential distribution and thus we choose it.

2 Reason why the mean of d_{ij} is inversely related to c_{ij}

The queuing delay depends on the rate at which packets are pushed out of the queue. Packets which are deeper in the queue will have less queuing delay if the speed of the link is high and it processes the previous packets quickly. This is why average queuing delay is inversely proportional to c_{ij}.

2.1 Reason for choice of I

We chose a default of 20s, the reason is that the time taken to transmit a block between two peers is less than 10th of a sec and so it'll probably take less 1 sec for the block to propagate to all the peers. Hence, forks would be likely if we were to pick a time very close to 1 sec as that would mean more blocks will be generated over a time interval and since our goal is to keep forks at a minimum we chose the time of 20 secs which makes the mean of Tk to be around in the range of a few thousands (because of hashing power distribution).

3 Visualization of Multiple Simulations

We ran a total of 4 simulations and report the statistics observed in each of them. Please see the **visualization** folder for the graphs of blockchain of each simulation.

3.1 Simulation 1

```
./simulator -Ttx 20 -I 20 -blks 100 -n 40 -z1 0.4
Total blocks : 100
Longest chain length : 100
Longest chain length / Total blocks : 1

Total Branches : 1          // no branches
Branch lengths : Longest : 100 Smallest : 100 Mean : 100

SLOW And LOW CPU, Mean of fraction of blocks: 0.4
SLOW And HIGH CPU, Mean of fraction of blocks: 1
FAST And LOW CPU, Mean of fraction of blocks: 0.181818
FAST And HIGH CPU, Mean of fraction of blocks: 1
```

With the hash powers that we assign to peers where the maximum is of the order of a few hundredths, the mean of interarrival between blocks with I set to 20 comes out to be a few thousands which is way higher than the delay due to block transmission and hence the possibility of a fork is negligible and we get a straight blockchain.

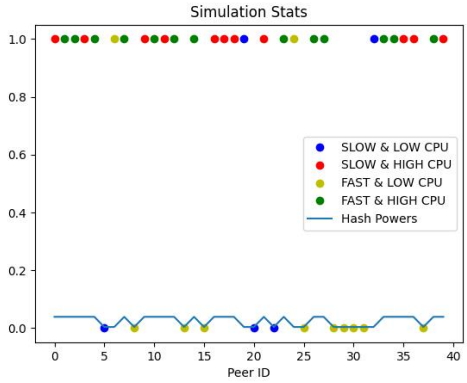

Figure 1: Simulation 1

As we expected, the fraction of blocks in the longest chain to the number of blocks generated to be a function of the hashing power, it is low for "low CPU" peers and high for "high CPU" peers.

3.2 Simulation 2

This time we try to bring interarrival time of blocks to close to Ttx. Now since the mining time has been reduced significantly we expect an increase in the number of forks in the blockchain. The results we got are as follows,

```
./simulator -Ttx 5 -I 0.4 -blks 100 -n 40 -z1 0.4
Total blocks : 100
Longest chain length : 87
Longest chain length / Total blocks : 0.87

Total Branches : 14
Branch lengths : Longest : 87 Smallest : 7 Mean : 42.0714

SLOW And LOW CPU, Mean of fraction of blocks: 0.6
SLOW And HIGH CPU, Mean of fraction of blocks: 0.884091
FAST And LOW CPU, Mean of fraction of blocks: 0.363636
FAST And HIGH CPU, Mean of fraction of blocks: 0.81337
```

We can see that the number of forks has indeed increased from the previous simulation due to reduced mining time which also caused some blocks to be rejected from the Longest Chain which is proved the new ratio of 0.87 as compared to 1.0 of last simulation. The fractions for all types of blocks are close dwindling because of the same reason of reduced mining time and forks.

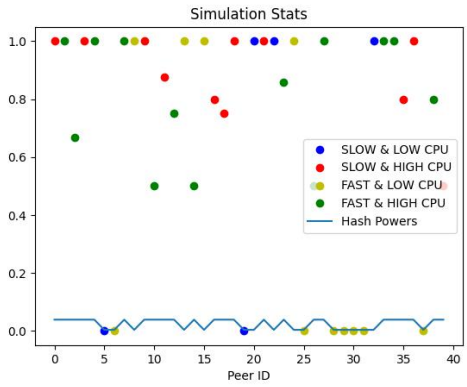

Figure 2: Simulation 2

We see that as compared to the last simulation there is larger variance of fraction, even FAST and HIGH CPU peers are getting a lesser fraction of blocks in their Longest Chain due to an increase in forks.

3.3 Simulation 3

In this simulation we tried to see what happens when we decrease the interarrival times of both blocks and txns even further and the results are as one might've expected,

```
./simulator -Ttx 0.05 -I 0.1 -blks 100 -n 40 -z1 0.4
Total blocks : 100
Longest chain length : 32
Longest chain length / Total blocks : 0.32

Total Branches : 47
```

```
Branch lengths : Longest : 32 Smallest : 3 Mean : 18.383

SLOW And LOW CPU, Mean of fraction of blocks: 0
SLOW And HIGH CPU, Mean of fraction of blocks: 0.0333333
FAST And LOW CPU, Mean of fraction of blocks: 0.0909091
FAST And HIGH CPU, Mean of fraction of blocks: 0.532051
```

We see a drastic decrease in the length of the Longest Chain and in the fraction of blocks that peers have in the Longest Chain. SLOW peers are struggling to get even a few blocks in the chain, most of the blocks are of the FAST and HIGH CPU peers as they generate blocks way faster due to reduced mining time and even transmit them quickly because of less latency.

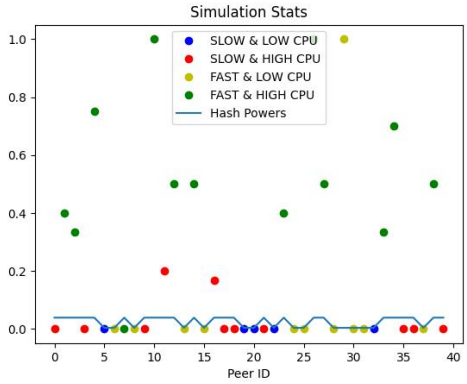

Figure 3: Simulation 3

As expected due to increase forking, all peers struggle to get their blocks in the Longest Chain, some can't even get 1 block in.

3.4 Simulation 4

Lastly we tried a simulation with reduced number of peers, the results were as follows

```
./simulator -Ttx 20 -I 20 -blks 100 -n 20 -z1 0.4

Total blocks : 100
Longest chain length : 100
Longest chain length / Total blocks : 1

Total Branches : 1
Branch lengths : Longest : 100 Smallest : 100 Mean : 100

SLOW And LOW CPU, Mean of fraction of blocks: 1
SLOW And HIGH CPU, Mean of fraction of blocks: 1
FAST And LOW CPU, Mean of fraction of blocks: 1
FAST And HIGH CPU, Mean of fraction of blocks: 1
```

This time since we've lesser number of peers with other parameters same, there is less competition between the peers and everyone's blocks get accepted in the Longest Chain.

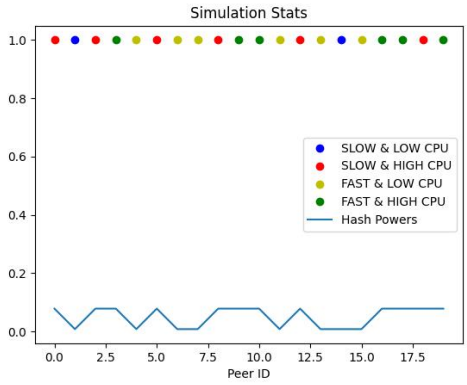

Figure 4: Simulation 4

4 References

- Discret Event Simulator
- Command Line Args Parsing