CS 765 Assignment 1
Simulation of a P2P Cryptocurrency Network

Guramrit Singh 210050061 **Isha Arora** 210050070 **Karan Godara** 210050082

Indian Institute of Technology, Bombay

February 10, 2024

Problem 1

The inter-arrival between transactions generated by any peer is chosen from an exponential distribution. Give theoretical reasons for choosing the exponential distribution?

Solution 1

Memoryless Property
The exponential distribution has the memoryless property, which means that the probability of waiting a certain amount of time for an event to occur does not depend on how much time has already passed. Blockchain transactions follow this property i.e. a transaction being generated in the next time interval remains constant regardless of how long it has been since the last transaction. Due to this exponential distribution is a good candidate for modelling inter-arrival between transactions.
Mathematical Justification
Let δ be a very small time interval and t_0 be any time.
Then, Pr[Transaction is generated between $T \in [t_0, t_0 + \delta]$] $\approx T_{tx}\delta$.
W.L.O.G let $t_0 = 0$
Thus Pr[Transaction is not generated between $T \in [0, \delta]$] $= 1 - T_{tx}\delta$.
Pr[Transaction is not generated between $T \in [0, n\delta]$] $= (1 - T_{tx}\delta)^n$.
Substituting $n\delta$ by t, we get
Pr[Transaction is not generated between $T \in [0, t]$] $= (1 - T_{tx}\delta)^{t/\delta}$.
Now $\delta \to 0$,

$$\lim_{\delta \to 0} (1 - T_{tx}\delta)^{t/\delta} = e^{-T_{tx}\delta * t/\delta} = e^{-T_{tx} * t}$$

Therefore, Pr[Transaction is not generated between $T \in [0, t]$] $= e^{-T_{tx} * t}$.
So, the CDF is given by $Pr[T < t] = 1 - e^{-T_{tx} * t}$
This leads us to have PDF as $Pr[T = t] = T_{tx}e^{-T_{tx} * t}$ which is nothing but PDF of an exponential random variable having $\lambda = T_{tx}$.

Problem 2

Why is the mean of d_{ij} inversely related to c_{ij}? Give justification for this choice.

Solution 2

Queuing delay is a fundamental concept in networking that refers to the time a packet spends waiting in a queue at a network node (e.g., router, switch, etc.) before it can be transmitted. High link speed helps us to transmit blocks at a faster rate. Therefore, if we transmit blocks faster, we would build up a smaller queue and vice versa. Therefore, the queuing delay is inversely proportional to link speed.

Problem 3

Give an explanation for the choice of a particular mean of block inter-arrival time in PoW simulation.

Solution 3

We have chosen the inter-arrival time between blocks on average to be 50 seconds for a P2P network of 100 peers. This number was chosen because for a block which is about 1 MB in size, ρ_{ij} is 250 ms on average, d_{ij} is 96 ms on average and transmission delay is 200 ms in the worst case. Hence, to transfer a block between 2 peers it takes around 0.5s. So, in the worst kind of network topology i.e., linear topology a block would take around 50s to go from one extremity to the other. Hence, this choice of mean minimizes the chance of creating any forks in blockchain due to network delay for the block transmission.

We know the property of exponential distribution, that given exponential random variables $x_1, x_2, \cdots x_n$ with their means as $\alpha_1, \alpha_2 \cdots \alpha_n$, the probability that we would see the occurrence of say x_j before any other random variable is $\frac{\frac{1}{\alpha_i}}{\Sigma_{j=1}^{n} \frac{1}{\alpha_j}}$. So arguing for the fact that mean of T_k is set as $\frac{I}{h_k}$, we can see that since each T_k is an exponential random variable and thus the probability that agent i generates block earlier than anyone else is $\frac{\frac{h_i}{I}}{\Sigma_{j=1}^{n} \frac{h_j}{I}} = h_i$. Hence we see that how choosing such a mean for T_k, we make sure that probabilty of any peer generating the next block is proportional to their hashing power.

Visual Analysis

Note: Symbols Used in Figures and Tables

Symbol Key:

- z_0: Percentage of slow peers
- z_1: Percentage of peers with low CPU
- F: $\frac{\text{Mean interarrival time between blocks}}{\text{Mean transmission delay of a block between two peers}}$
- ND: Not defined $(\frac{0}{0})$

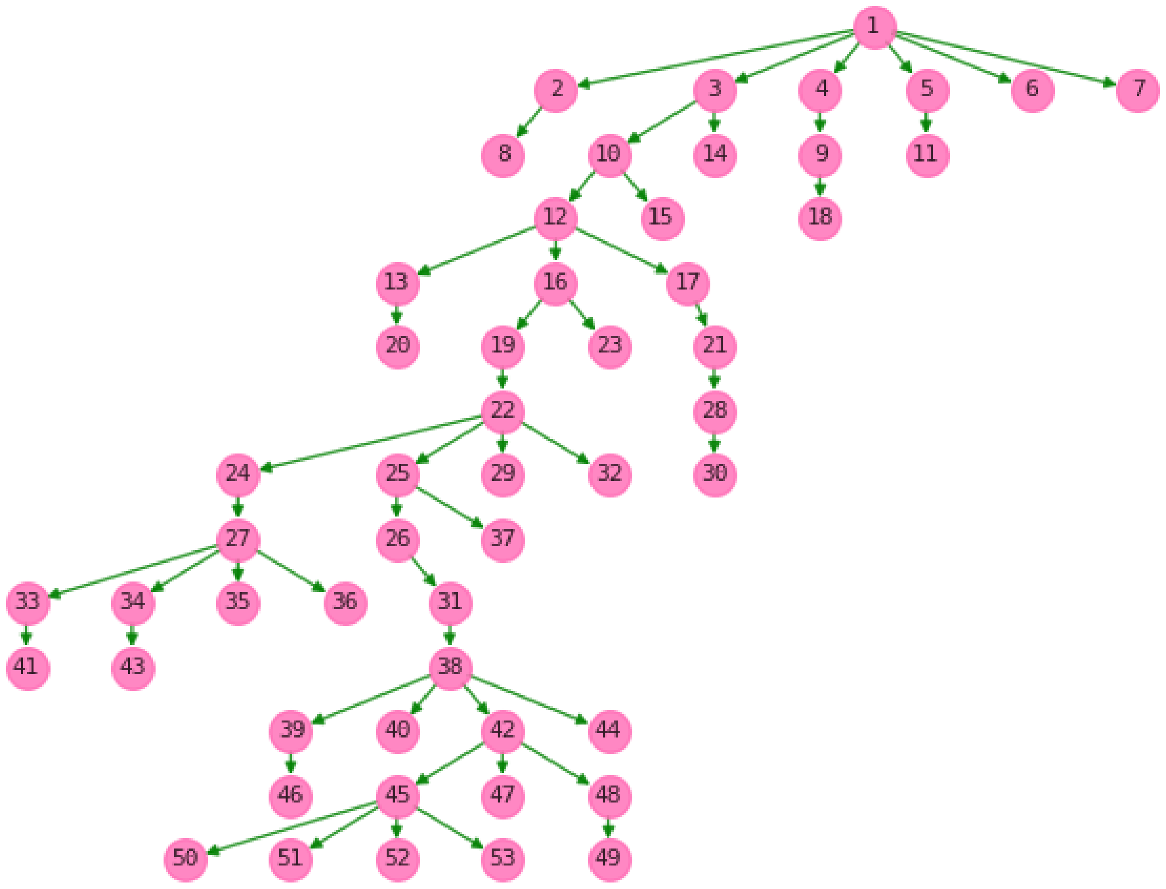

Figure 1: Parameters: $z_0 = 50$, $z_1 = 50$, $F = 1$

Statistic	**Value**
Minimum number of blocks in a branch	1
Maximum number of blocks in a branch	8
Average number of blocks in a branch	1.7727272727272727
Total number of branches	22
Length of longest chain	14
Number of orphaned blocks	39

Table 1: Branch statistics for parameters: $z_0 = 50$, $z_1 = 50$, $F = 1$

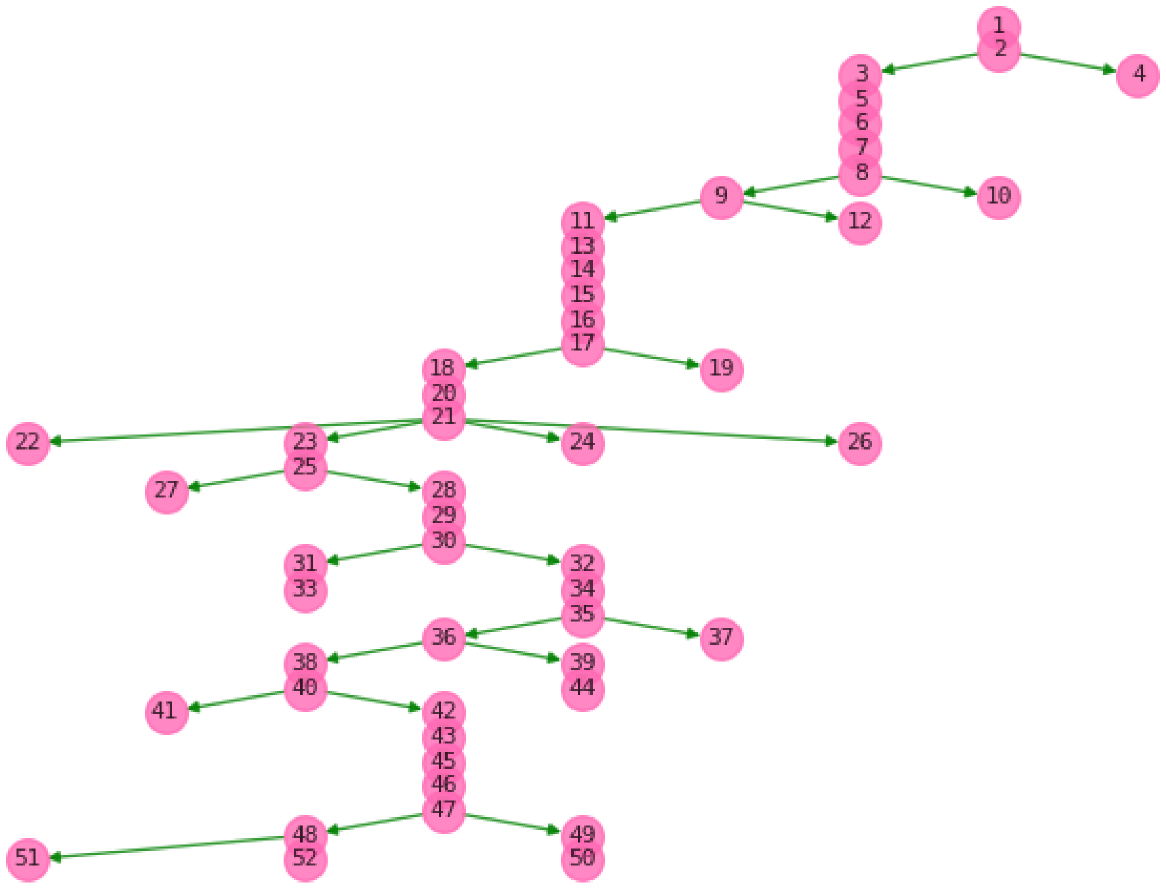

Figure 2: Parameters: $z_0 = 50$, $z_1 = 50$, $F = 10$

Statistic	**Value**
Minimum number of blocks in a branch	1
Maximum number of blocks in a branch	2
Average number of blocks in a branch	1.2142857142857142
Total number of branches	14
Length of longest chain	35
Number of orphaned blocks	17

Table 2: Branch statistics for parameters: $z_0 = 50$, $z_1 = 50$, $F = 10$

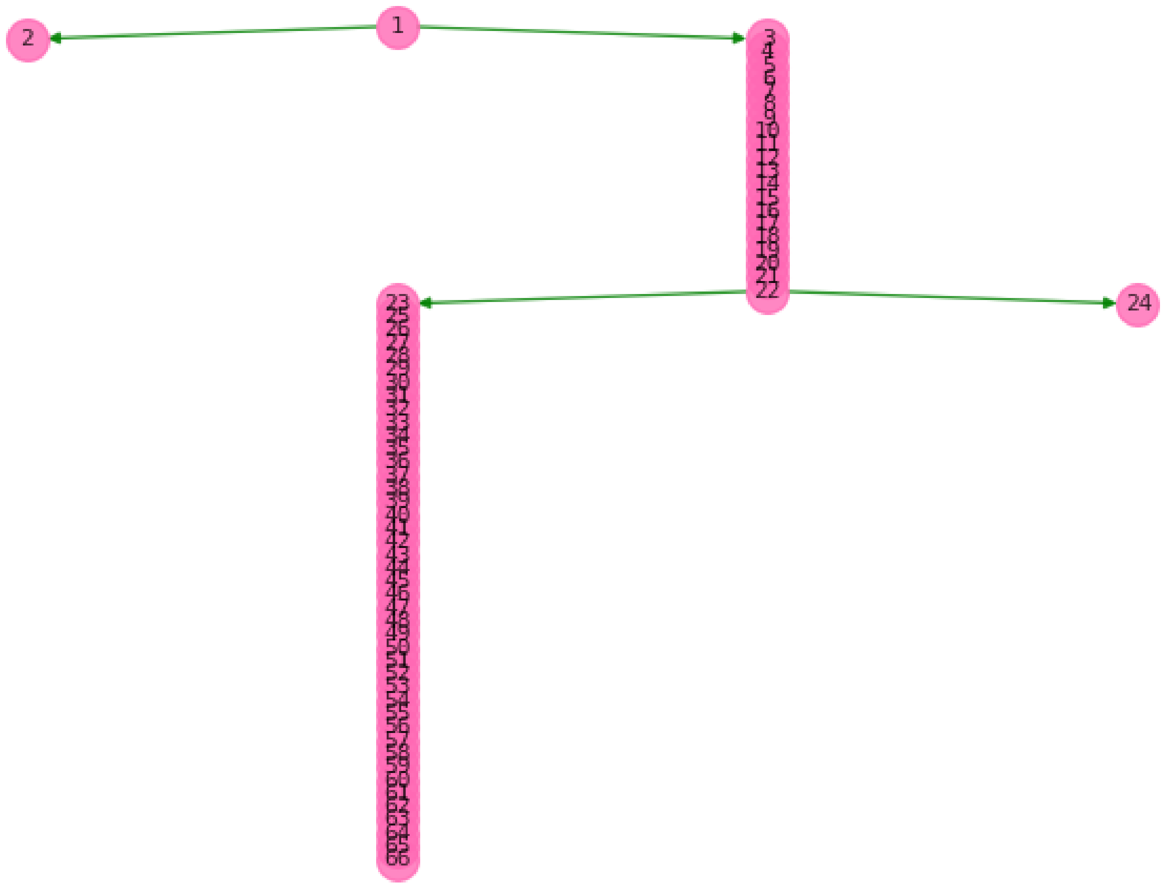

Figure 3: Parameters: $z_0 = 50$, $z_1 = 50$, $F = 100$

Statistic	**Value**
Minimum number of blocks in a branch	1
Maximum number of blocks in a branch	1
Average number of blocks in a branch	1.0
Total number of branches	2
Length of longest chain	64
Number of orphaned blocks	2

Table 3: Branch statistics for parameters: $z_0 = 50$, $z_1 = 50$, $F = 100$

Conclusion 1

We first analyzed the effect of changing the value of the ratio between mean inter-arrival time between blocks and the mean transmission delay of a block between two peers. For this set of experiments, we had 40 peers, $z_0 = 50$ and $z_1 = 50$. We can see in Figure 3, how if F is sufficiently large (100 in this example), we have very little to no forking cause whenever a block is produced, it reaches every other peer earlier than any other block generated after it. Whereas, when we decrease F such that it is 10, we see how the number of forks and thus the blocks orphaned due to not being part of the largest chain increases considerably as can be seen in Figure 2. This is due to the fact before a generated block can reach all the nodes in the network, some other block gets created and is observed by nodes in the network hence creating forks. This effect is amplified even further when $F = 1$, because now this phenomenon occurs way too frequently, hence we see how the longest chain has very few blocks in it and the majority of the blocks get orphaned as observed in Figure 1. All of these observations are backed quantitatively and can be verified from the tables 3, 2 and 1.

Hence, we can conclude that to prevent extensive forking and therefore to curb the wastage of resources, our inter-arrival time between blocks should be sufficiently larger than the time it takes for the block to get propagated in the entire network.

Conclusion 2

Now, let's observe the contribution of each kind of miner in the final blockchain. Following the data observed during experiments, we can see that whatever the setting(value of z_0 and z_1), all of the peers of type Fast link speed and High CPU power have ratios close to 1. They are able to generate blocks at a fast rate and also able to transmit them at a high speed to other peers. Whereas peers of type Slow link speed and High CPU have high ratios, they are in general less than those Fast/High peers because by the time they are able to transmit a block, the longest chain changes and thus their block gets orphaned. Moving on to peers of type Fast link speed, Low CPU and Slow link speed, Low CPU, their contribution is very less in the final blockchain as their blocks gets dropped frequently. Note, the ratio of many of such peers is Not Defined (ND) because they are unable to transmit any block as by the time they generate a block, current longest chain in the blockchain changes due to blocks being made at faster pace by high CPU peers. All of these observations are backed quantitatively and can be verified from tables 4, 5, 6 and 7 and figures 4, 5, 6 and 7.

Conclusion 3

Now doing an in-depth analysis of the effects of link speed and CPU speed on the number of blocks produced and also on the number of blocks actually becoming part of the longest chain. We can see how it's the CPU speed that majorly determines whether the contribution in the blockchain would be high or not, and not the link speed. If a node has a high CPU speed, it is able to generate a block at a faster rate. If additionally, it has a high link speed, it will be able to transmit the block to its immediate neighbors at a faster rate but for a large network, the overall delay that would take place to transmit the block to all peers would be approximately same as the delay for a block generated by a high CPU node with a low link speed. This is because the time taken for a block to propagate through the entire network depends not only on the link speed of the generator of the block but mainly on the overall topology and link speeds of the network. Whereas, if a node has a Low CPU speed, they are unable to generate blocks that still form the longest chain post generation because by the time it creates a block a high CPU speed node has already created a new block and therefore has changed the longest chain. Thus, there are much fewer blocks generated by low CPU nodes propagating in the network and thus consequently fewer blocks in the blockchain. Hence, all in all, it's CPU speed that plays a major role in determining that how many blocks of the peer would be in the blockchain and the longest chain. Again, all of these observations are backed quantitatively and can be verified from tables 4, 5, 6 and 7 and figures 4, 5, 6 and 7.

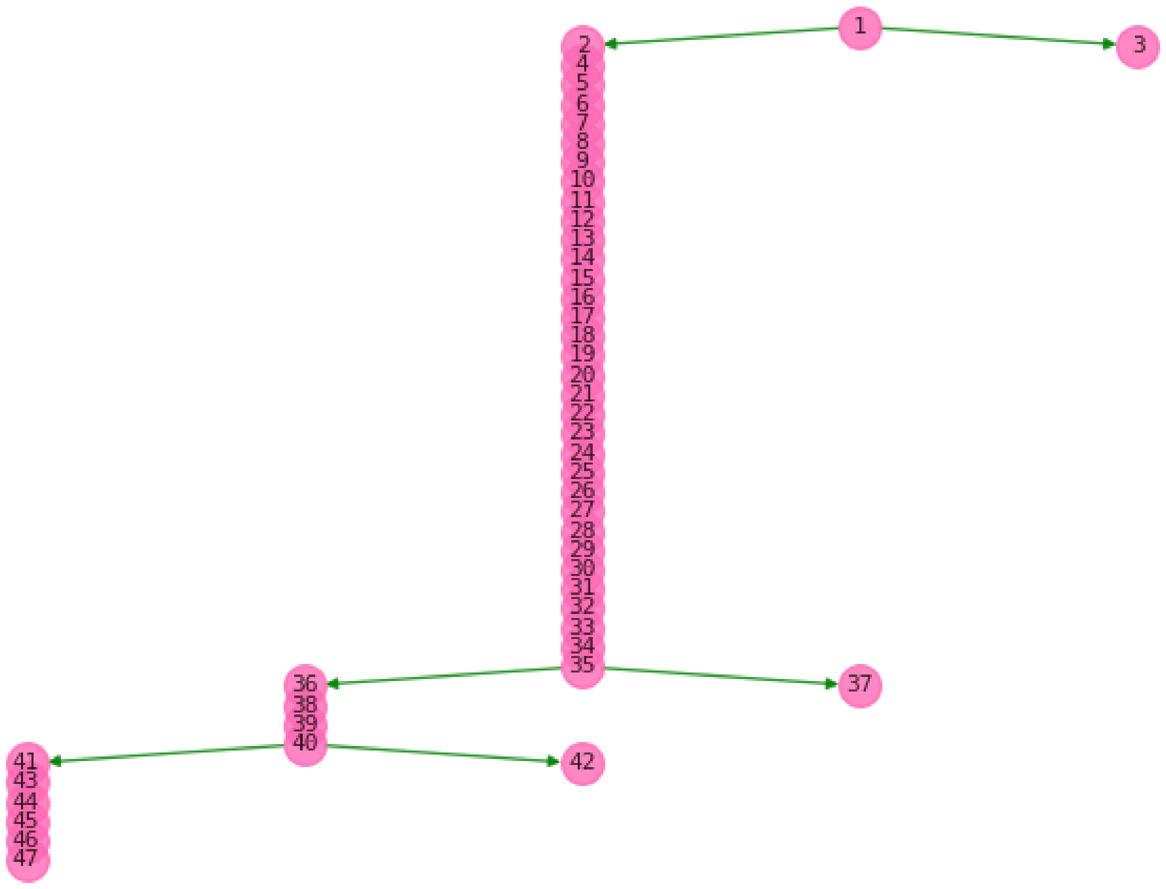

Figure 4: Parameters: $z_0 = 20$, $z_1 = 20$, $F = 100$

(a) Category of miners

Category	Number
Slow, Low CPU miners	1
Fast, Low CPU miners	5
Slow, High CPU miners	9
Fast, High CPU miners	25
Low CPU miners	6
High CPU miners	34

(b) Overall miner statistics I

Category	Ratio of blocks in longest chain
Slow, Low CPU miners	1.0
Fast, Low CPU miners	ND
Slow, High CPU miners	1.0
Fast, High CPU miners	0.9210526315789473

(c) Overall miner statistics II

Category	Number of miners who didn't mine a single block
Low CPU miners who did not contribute	5
High CPU miners who did not contribute	5

(d) Per miner statistics

ID	Slow/Fast	CPU	#Blocks in Longest Chain	#Blocks Generated	$\frac{\text{\#Blocks in Longest Chain}}{\text{\#Blocks Generated}}$
1	Slow	High CPU	1	1	1.0
2	Fast	High CPU	2	2	1.0
3	Fast	High CPU	2	2	1.0
4	Slow	High CPU	2	2	1.0
5	Fast	High CPU	1	1	1.0
6	Fast	High CPU	1	1	1.0
7	Fast	High CPU	0	0	ND
8	Slow	High CPU	0	0	ND
9	Fast	High CPU	1	1	1.0
10	Slow	High CPU	0	0	ND
11	Fast	High CPU	1	1	1.0
12	Fast	High CPU	1	1	1.0
13	Fast	Low CPU	0	0	ND
14	Slow	High CPU	1	1	1.0
15	Fast	High CPU	1	1	1.0
16	Fast	High CPU	1	1	1.0
17	Fast	High CPU	1	1	1.0
18	Fast	High CPU	1	1	1.0
19	Fast	High CPU	2	2	1.0
20	Fast	High CPU	1	1	1.0
21	Fast	High CPU	3	3	1.0
22	Fast	High CPU	2	2	1.0
23	Fast	High CPU	1	1	1.0
24	Fast	High CPU	2	3	0.6666666666666666
25	Slow	High CPU	1	1	1.0
26	Slow	High CPU	1	1	1.0
27	Fast	High CPU	3	3	1.0
28	Slow	High CPU	0	0	ND
29	Fast	Low CPU	0	0	ND
30	Fast	Low CPU	0	0	ND
31	Fast	High CPU	0	0	ND
32	Fast	High CPU	1	1	1.0
33	Fast	Low CPU	0	0	ND
34	Fast	High CPU	2	2	1.0
35	Fast	High CPU	2	2	1.0
36	Fast	Low CPU	0	0	ND
37	Slow	Low CPU	1	1	1.0
38	Fast	High CPU	2	4	0.5
39	Slow	High CPU	1	1	1.0
40	Fast	High CPU	1	1	1.0

Table 4: Branch statistics for parameters: $z_0 = 20$, $z_1 = 20$, $F = 100$

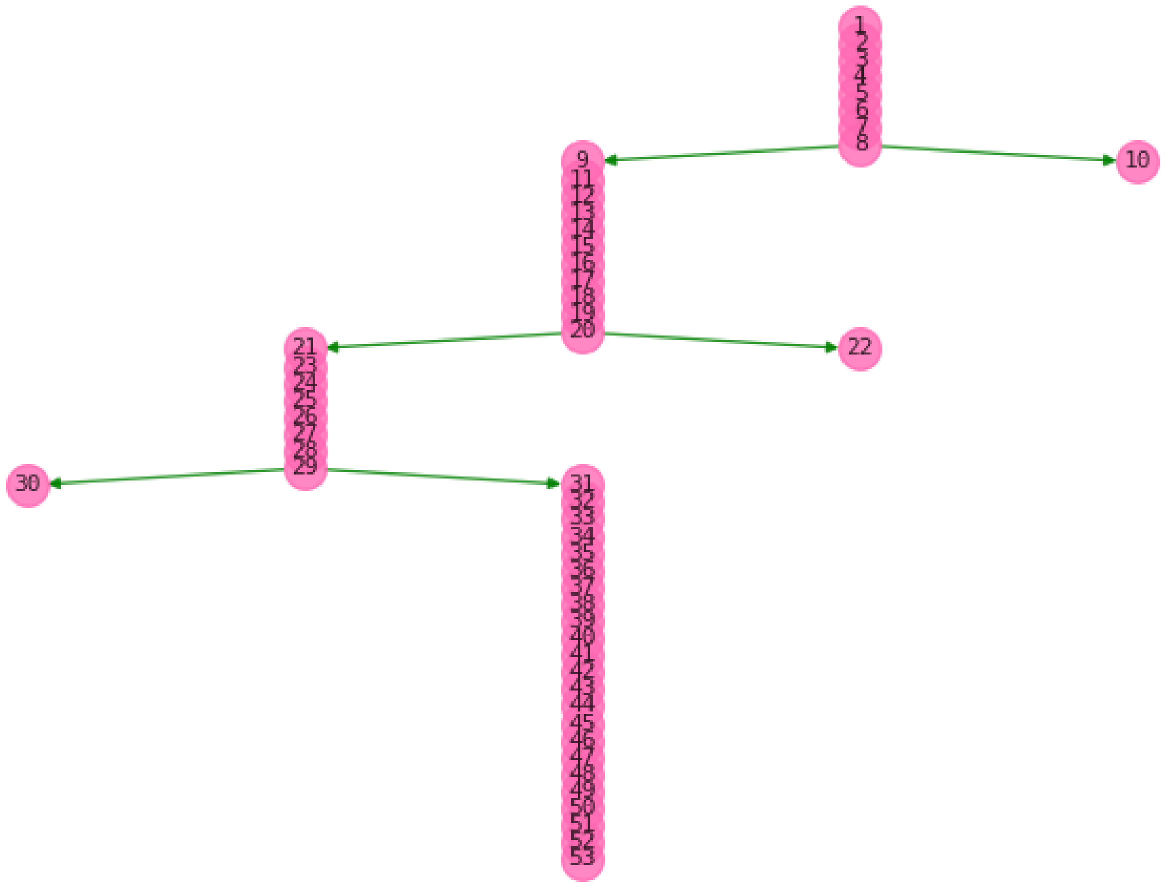

Figure 5: Parameters: $z_0 = 20$, $z_1 = 80$, $F = 100$

(a) Category of miners

Category	Number
Slow, Low CPU miners	9
Fast, Low CPU miners	25
Slow, High CPU miners	2
Fast, High CPU miners	4
Low CPU miners	34
High CPU miners	6

(b) Overall miner statistics I

Category	Ratio of blocks in longest chain
Slow, Low CPU miners	1.0
Fast, Low CPU miners	0.875
Slow, High CPU miners	0.9285714285714286
Fast, High CPU miners	0.9615384615384616

(c) Overall miner statistics II

Category	Number of miners who didn't mine a single block
Low CPU miners who did not contribute	23
High CPU miners who did not contribute	0

(d) Per miner statistics

ID	Slow/Fast	CPU	#Blocks in Longest Chain	#Blocks Generated	$\frac{\text{\#Blocks in Longest Chain}}{\text{\#Blocks Generated}}$
1	Fast	Low CPU	0	0	ND
2	Fast	Low CPU	0	0	ND
3	Fast	Low CPU	0	0	ND
4	Slow	Low CPU	0	0	ND
5	Fast	Low CPU	0	0	ND
6	Slow	High CPU	7	7	1.0
7	Fast	Low CPU	0	0	ND
8	Fast	High CPU	7	8	0.875
9	Fast	Low CPU	0	0	ND
10	Fast	Low CPU	0	0	ND
11	Fast	Low CPU	0	0	ND
12	Slow	Low CPU	1	1	1.0
13	Slow	Low CPU	1	1	1.0
14	Slow	Low CPU	0	0	ND
15	Slow	Low CPU	0	0	ND
16	Fast	Low CPU	1	1	1.0
17	Fast	Low CPU	0	0	ND
18	Slow	High CPU	6	7	0.8571428571428571
19	Fast	Low CPU	0	0	ND
20	Fast	High CPU	9	9	1.0
21	Slow	Low CPU	1	1	1.0
22	Fast	High CPU	3	3	1.0
23	Fast	Low CPU	0	0	ND
24	Fast	Low CPU	0	0	ND
25	Fast	Low CPU	1	1	1.0
26	Fast	Low CPU	1	1	1.0
27	Fast	Low CPU	1	1	1.0
28	Fast	Low CPU	0	0	ND
29	Fast	Low CPU	1	1	1.0
30	Slow	Low CPU	0	0	ND
31	Slow	Low CPU	1	1	1.0
32	Fast	Low CPU	0	0	ND
33	Fast	High CPU	6	6	1.0
34	Fast	Low CPU	2	2	1.0
35	Fast	Low CPU	0	0	ND
36	Fast	Low CPU	0	0	ND
37	Slow	Low CPU	0	0	ND
38	Fast	Low CPU	0	0	ND
39	Fast	Low CPU	0	0	ND
40	Fast	Low CPU	0	1	0.0

Table 5: Branch statistics for parameters: $z_0 = 20$, $z_1 = 80$, $F = 100$

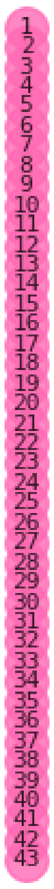

Figure 6: Parameters: $z_0 = 80$, $z_1 = 20$, $F = 100$

(a) Category of miners

Category	Number
Slow, Low CPU miners	4
Fast, Low CPU miners	2
Slow, High CPU miners	25
Fast, High CPU miners	9
Low CPU miners	6
High CPU miners	34

(b) Overall miner statistics I

Category	Ratio of blocks in longest chain
Slow, Low CPU miners	ND
Fast, Low CPU miners	ND
Slow, High CPU miners	1.0
Fast, High CPU miners	1.0

(c) Overall miner statistics II

Category	Number of miners who didn't mine a single block
Low CPU miners who did not contribute	6
High CPU miners who did not contribute	10

(d) Per miner statistics

ID	Slow/Fast	CPU	#Blocks in Longest Chain	#Blocks Generated	$\frac{\text{\#Blocks in Longest Chain}}{\text{\#Blocks Generated}}$
1	Slow	High CPU	0	0	ND
2	Slow	High CPU	3	3	1.0
3	Fast	High CPU	2	2	1.0
4	Slow	Low CPU	0	0	ND
5	Fast	High CPU	2	2	1.0
6	Slow	Low CPU	0	0	ND
7	Fast	High CPU	3	3	1.0
8	Slow	High CPU	3	3	1.0
9	Fast	High CPU	2	2	1.0
10	Slow	High CPU	1	1	1.0
11	Slow	Low CPU	0	0	ND
12	Slow	High CPU	3	3	1.0
13	Slow	High CPU	1	1	1.0
14	Slow	High CPU	1	1	1.0
15	Slow	High CPU	2	2	1.0
16	Slow	High CPU	1	1	1.0
17	Slow	High CPU	1	1	1.0
18	Fast	Low CPU	0	0	ND
19	Slow	High CPU	0	0	ND
20	Slow	High CPU	0	0	ND
21	Fast	High CPU	2	2	1.0
22	Slow	Low CPU	0	0	ND
23	Slow	High CPU	2	2	1.0
24	Fast	High CPU	1	1	1.0
25	Slow	High CPU	0	0	ND
26	Fast	High CPU	2	2	1.0
27	Fast	High CPU	3	3	1.0
28	Slow	High CPU	0	0	ND
29	Slow	High CPU	1	1	1.0
30	Slow	High CPU	1	1	1.0
31	Slow	High CPU	1	1	1.0
32	Fast	Low CPU	0	0	ND
33	Slow	High CPU	0	0	ND
34	Fast	High CPU	0	0	ND
35	Slow	High CPU	1	1	1.0
36	Slow	High CPU	1	1	1.0
37	Slow	High CPU	2	2	1.0
38	Slow	High CPU	0	0	ND
39	Slow	High CPU	0	0	ND
40	Slow	High CPU	0	0	ND

Table 6: Branch statistics for parameters: $z_0 = 80$, $z_1 = 20$, $F = 100$

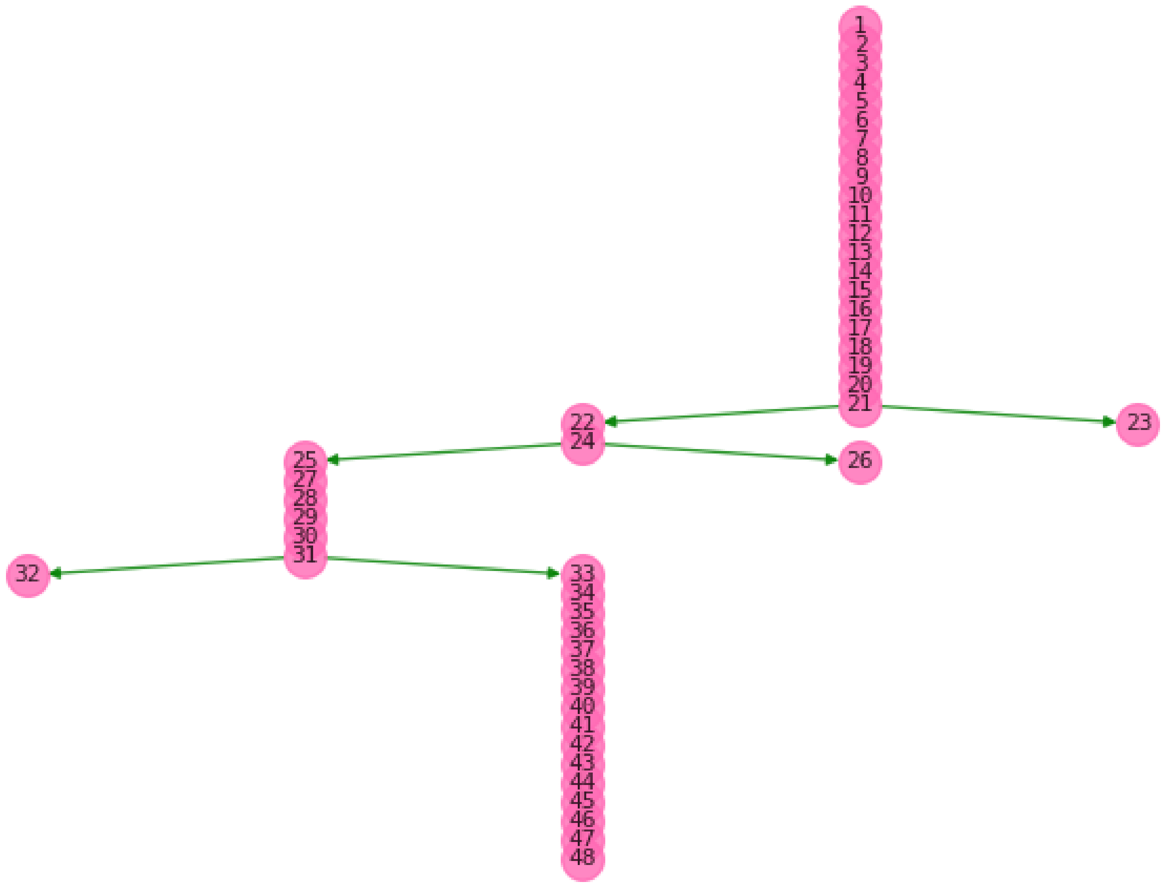

Figure 7: Parameters: $z_0 = 80$, $z_1 = 80$, $F = 100$

(a) Category of miners

Category	Number
Slow, Low CPU miners	26
Fast, Low CPU miners	5
Slow, High CPU miners	7
Fast, High CPU miners	2
Low CPU miners	31
High CPU miners	9

(b) Overall miner statistics I

Category	Ratio of blocks in longest chain
Slow, Low CPU miners	1.0
Fast, Low CPU miners	1.0
Slow, High CPU miners	0.9285714285714286
Fast, High CPU miners	0.8571428571428571

(c) Overall miner statistics II

Category	Number of miners who didn't mine a single block
Low CPU miners who did not contribute	22
High CPU miners who did not contribute	0

(d) Per miner statistics

ID	Slow/Fast	CPU	#Blocks in Longest Chain	#Blocks Generated	$\frac{\text{\#Blocks in Longest Chain}}{\text{\#Blocks Generated}}$
1	Slow	High CPU	5	5	1.0
2	Fast	High CPU	2	3	0.6666666666666666
3	Fast	Low CPU	0	0	ND
4	Slow	High CPU	6	6	1.0
5	Slow	Low CPU	1	1	1.0
6	Slow	Low CPU	0	0	ND
7	Slow	Low CPU	2	2	1.0
8	Slow	Low CPU	0	0	ND
9	Fast	High CPU	4	4	1.0
10	Slow	Low CPU	1	1	1.0
11	Slow	Low CPU	0	0	ND
12	Slow	Low CPU	0	0	ND
13	Slow	Low CPU	1	1	1.0
14	Slow	Low CPU	0	0	ND
15	Slow	Low CPU	0	0	ND
16	Slow	Low CPU	0	0	ND
17	Slow	Low CPU	0	0	ND
18	Slow	Low CPU	2	2	1.0
19	Fast	Low CPU	0	0	ND
20	Slow	Low CPU	0	0	ND
21	Slow	Low CPU	0	0	ND
22	Slow	Low CPU	2	2	1.0
23	Slow	Low CPU	0	0	ND
24	Fast	Low CPU	0	0	ND
25	Slow	High CPU	4	4	1.0
26	Slow	Low CPU	0	0	ND
27	Slow	Low CPU	0	0	ND
28	Slow	Low CPU	0	0	ND
29	Slow	High CPU	3	3	1.0
30	Fast	Low CPU	0	0	ND
31	Fast	Low CPU	1	1	1.0
32	Slow	Low CPU	0	0	ND
33	Slow	High CPU	1	2	0.5
34	Slow	Low CPU	0	0	ND
35	Slow	Low CPU	1	1	1.0
36	Slow	High CPU	3	3	1.0
37	Slow	Low CPU	0	0	ND
38	Slow	Low CPU	1	1	1.0
39	Slow	Low CPU	0	0	ND
40	Slow	High CPU	4	5	0.8

Table 7: Branch statistics for parameters: $z_0 = 80$, $z_1 = 80$, $F = 100$