P2P Cryptocurrency Network

Aneesh Garg, Raaghav Raaj, Shivam Goel

Task 2:

What are the theoretical reasons for choosing the exponential distribution for the interarrival between transactions generated by any peer?

Answer - We achieve the memoryless property by choosing an exponential distribution, which means that at any point in time, the expected time after which a node generates the next transaction does not change with time.

Task 4:

Reference: https://www.ndsu.edu/pubweb/~novozhil/Teaching/767%20Data/chapter_3.pdf

We have used theorem 3.11 (with $c = 3$). We basically choose whether an edge in the graph is present or not with probability $p = 3\frac{\log n}{n}$, and using the theorem the graph is expected to be connected with very high probability.

In case the resulting graph is not connected then we try again to get a connected graph.

So, the probability that our final graph is given connected graph G is equal to -

$$Np^{|E|}(1 - p)^{T-|E|},$$

where $T$is the total number of edges possible in the graph G i.e. $T = {}^{n}C_{2}$,

$|E|$ is the number of edges in the graph G,

N is the normalisation factor.

Task 5:

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

Answer -

c_{ij} - link speed between i and j

d_{ij} - queuing delay at node i to forward the message to node j

Now, if the link speed between i and j increases then there would be a lesser delay at node i to forward messages to node j, as the time taken for a particular message would be lesser.

Similarly, if the link speed between i and j decreases, then the delay would increase.

Hence, mean of d_{ij} is inversely related to that of c_{ij}.

Task7:

We choose a mean time taken to create a block(mean Tk) based on how much hashing (or CPU) power we want to assign to a particular peer. As

$$mean\ Tk \propto 1\ /\ hashing\ power$$

So, if we want to give a peer very high hashing power we assign mean Tk for that peer as low as possible and vice-versa.

Variation of Different Parameters -

We can see that as branching in a tree increases, the ratio of the number of blocks in the tree to the number of blocks in the longest chain of the tree increases, i.e.

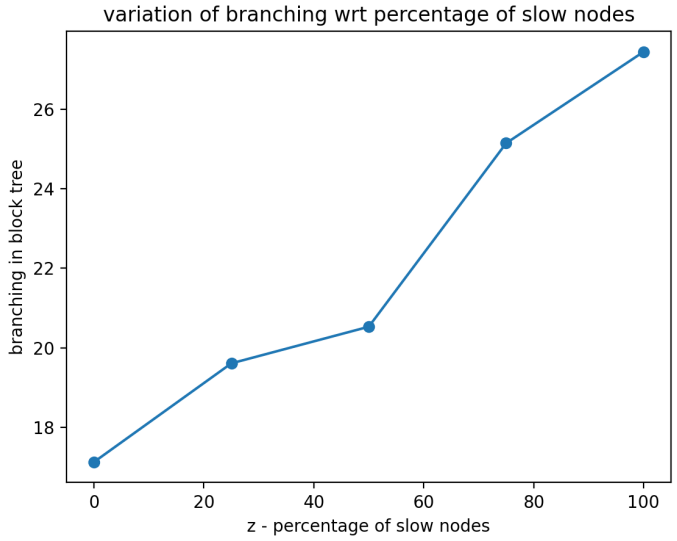
$$\text{Branching in the tree} \propto \frac{B_{Tree}}{B_{Longest\ chain}}$$

where

- B_{Tree} is the number of blocks in the block tree and
- $B_{Longest\ chain}$ is the number of blocks in the longest chain of the tree

Hence, in order to visualise and quantify the branching of the tree, we plot the ratio $\frac{B_{Tree}}{B_{Longest\ chain}}$ with respect to variation in different parameters.

1. z (% age of slow nodes) **VS** Branching in Block Tree

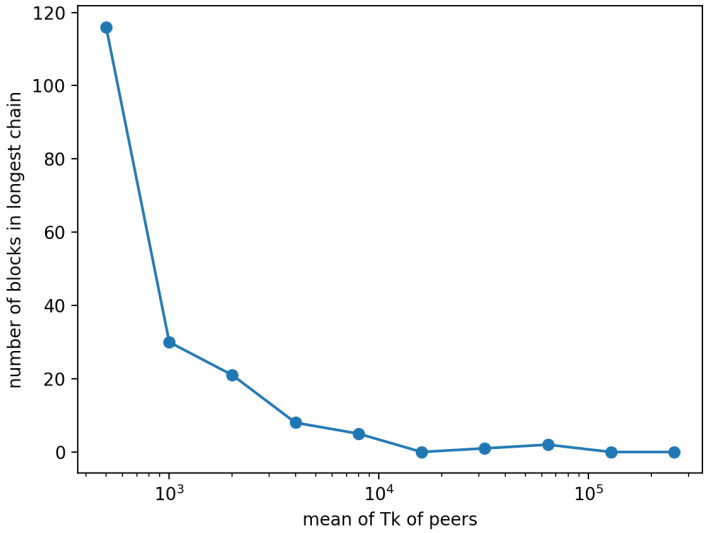


Fixed Parameters

Number of peers, *num*	200
T_x	200 milliseconds
Mean T_k	1000 milliseconds for all peers
Seed	25
Total Run Time	30 seconds
Invalid Status	False

As z increases, the percentage of slow nodes in the graph increases which leads to a high transmission time of blocks between peers. As a result, there is more chance of forking which leads to more branching in the block tree.

2. Mean of T_k of peers (in ms) **VS** Number of Blocks in Longest Chain



Fixed Parameters

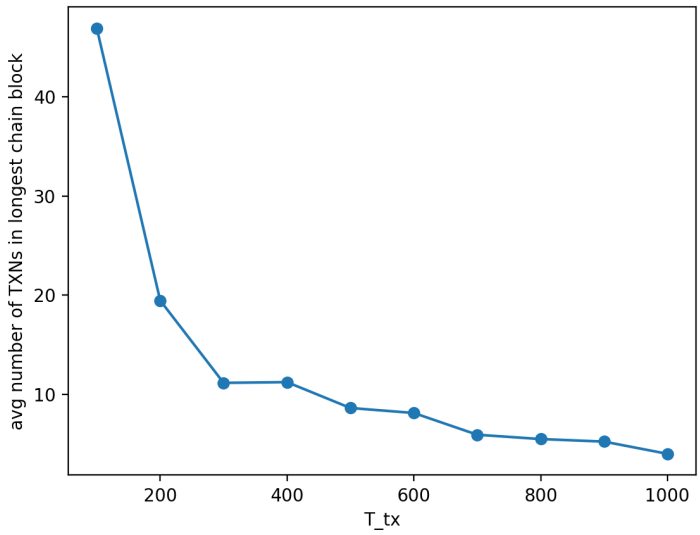
Number of peers, *num*	10
T_x	200 milliseconds
z	0
Seed	25
Total Run Time	60 seconds
Invalid Status	False

We know that:

$$mean\ Tk\ \propto 1\ /\ hashing\ power$$

So, as we can see in the graph as mean Tk increases, the number of blocks in the longest chain of the tree owned by the peer with that mean Tk decreases due to decrease in the hashing power.

3. T_{tx}- Mean Interarrival Time between Txns in ms **VS** Avg Number of Txns in Longest Chain Block

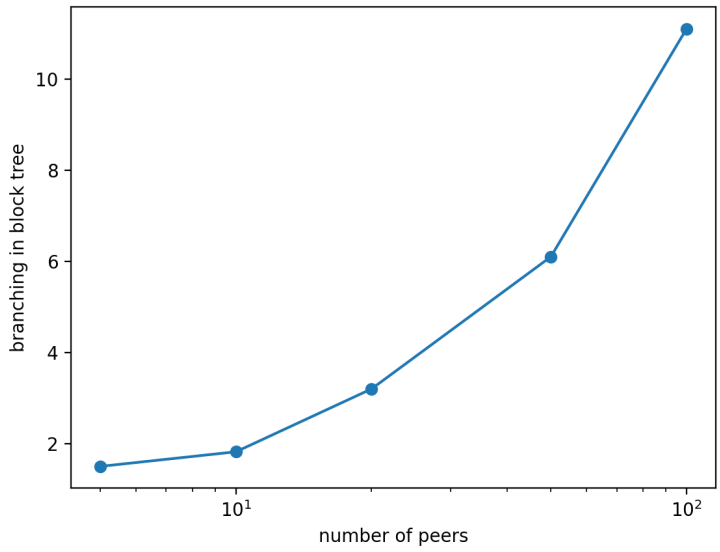


Fixed Parameters

Number of peers, *num*	10
Mean T_k	2000 milliseconds for all peers
z	50
Seed	25
Total Run Time	20 seconds
Invalid Status	False

As T_{tx} increases less transactions are generated in the total run time of the code, which results in less number of average transactions in a block in the longest chain.

4. *num*-Number of Peers **VS** Branching in Block Tree



Fixed Parameters

T_x	500 milliseconds
Mean T_k	2000 milliseconds for all peers
z	100
Seed	50
Total Run Time	15 seconds
Invalid Status	False

As the number of peers increases, the chance of forking increases which results in more branching in the block tree.

5. Variation of r for Fast and Slow nodes

   r - Ratio of Number of Blocks generated by each node in the Longest Chain of the tree **to** the Total Number of Blocks it generates at the end of the simulation

Fixed Parameters

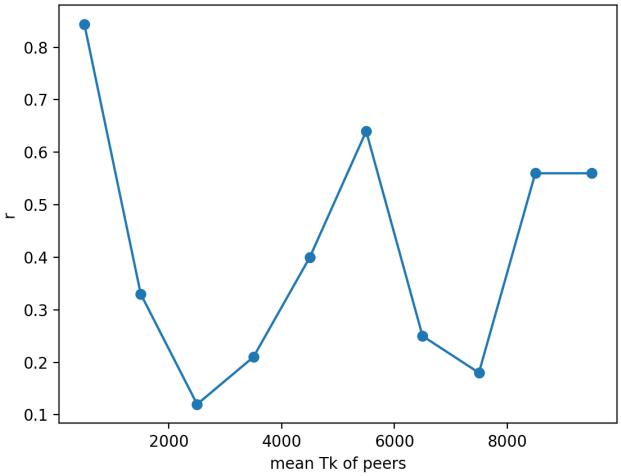
Number of peers, *num*	100
T_x	500 milliseconds
Mean T_k	2000 milliseconds for all peers
z	50
Seed	1
Total Run Time	60 seconds
Invalid Status	False

- Avg Value of r for **Slow** Nodes = 0.0598036
- Avg Value of r for **Fast** Nodes = 0.156012

We can say that for a slow node, the time it will take to receive a block will be greater than that for the fast node. As a result, there will be greater probability for a slow node to generate a block that will not be a part of the longest chain of the block tree, due to which it has a smaller avg value of r.

6. Variation of *r* with CPU power

   r - Ratio of Number of Blocks generated by each node in the Longest Chain of the tree **to** the total number of blocks it generates at the end of the simulation



Fixed Parameters

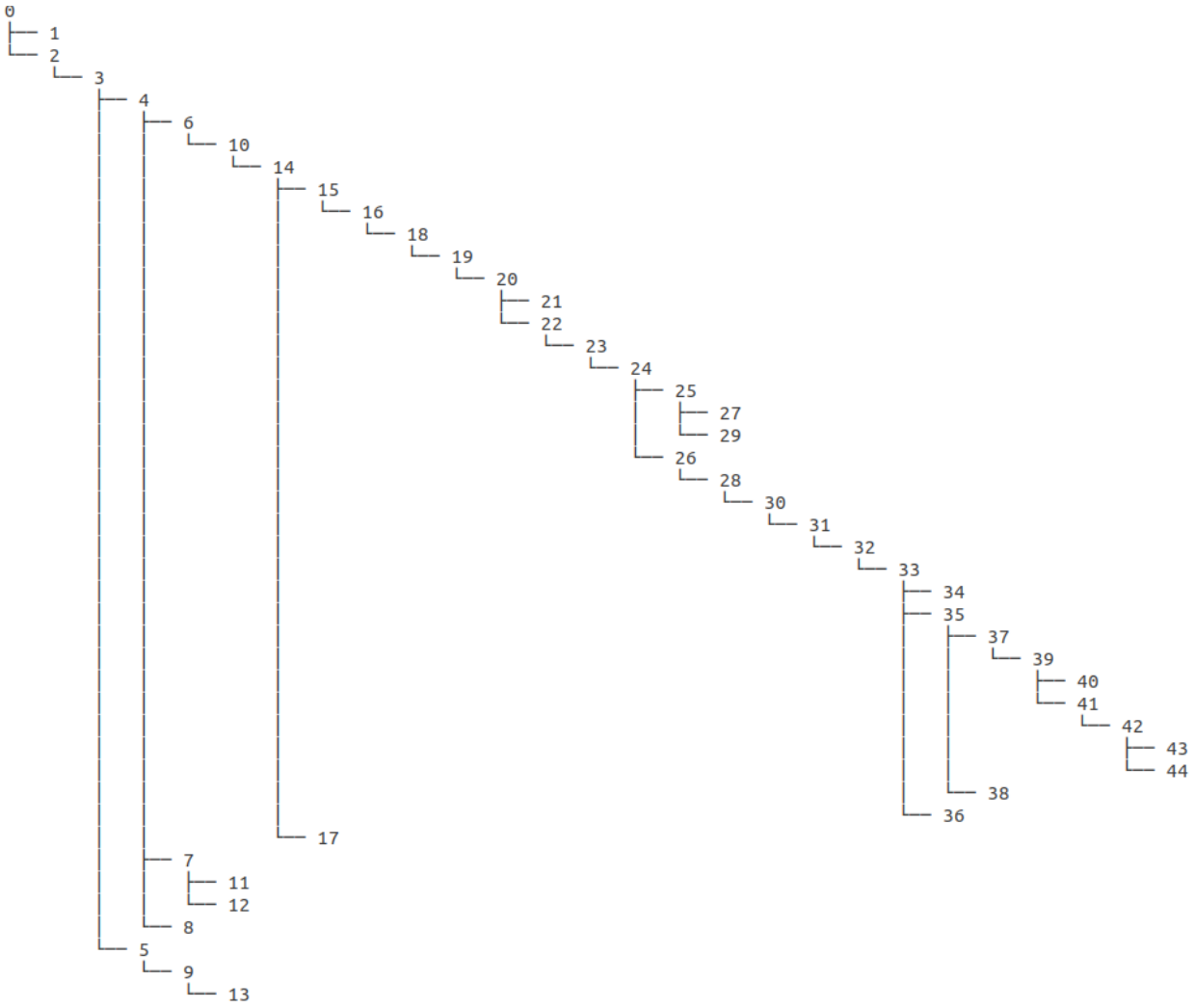
Number of peers, *num*	10
T_x	500 milliseconds
z	50
Seed	1
Total Run Time	60 seconds
Invalid Status	False

As we can see in the graph, mean tk of peers has no direct relation with r. Since, r depends on the probability that your block will belong to the longest chain and not on how fast you are creating the blocks.

7. Block Tree at the end of simulation

Fixed Parameters

Number of peers, *num*	10
Mean T_k	4000 milliseconds for all peers
T_x	250 milliseconds
z	100
Seed	0
Total Run Time	20 seconds
Invalid Status	False

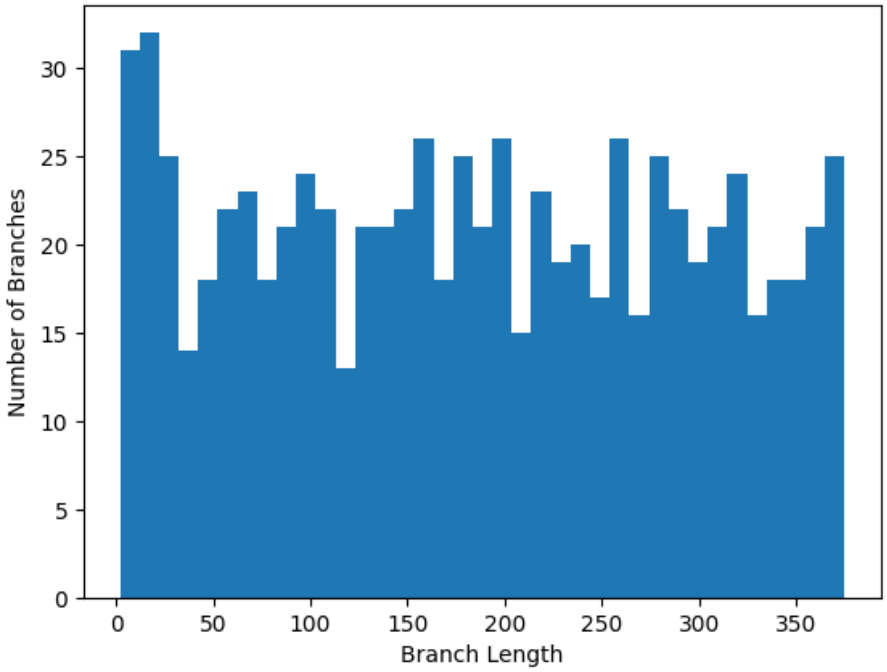


8. <u>Variation of Branch lengths seen for z = 0 and z = 100</u>

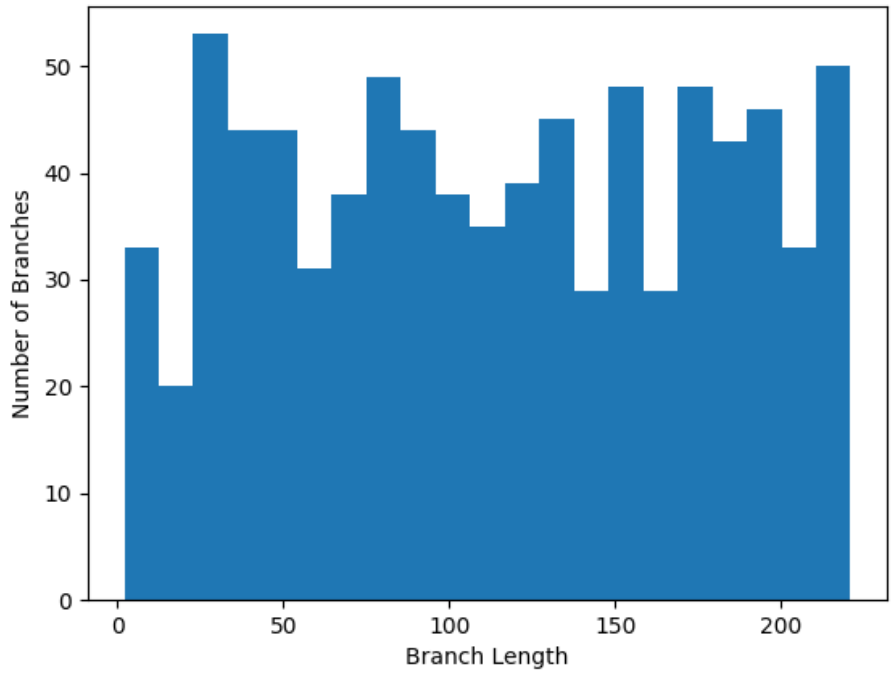
Number of peers, *num*	50

Mean T_k	2000 milliseconds for all peers
T_x	250 milliseconds
Seed	0
Total Run Time	60 seconds
Invalid Status	False

z = 0



z = 100



z = 0 means all nodes are fast, which results in less transmission time. As a result the total hashing power is mostly used on the updated longest chain. Since, this results in more efficient use of hashing power so it results in longer branch lengths as compared to the case of z = 100.

Lazy Peer generates Invalid Blocks

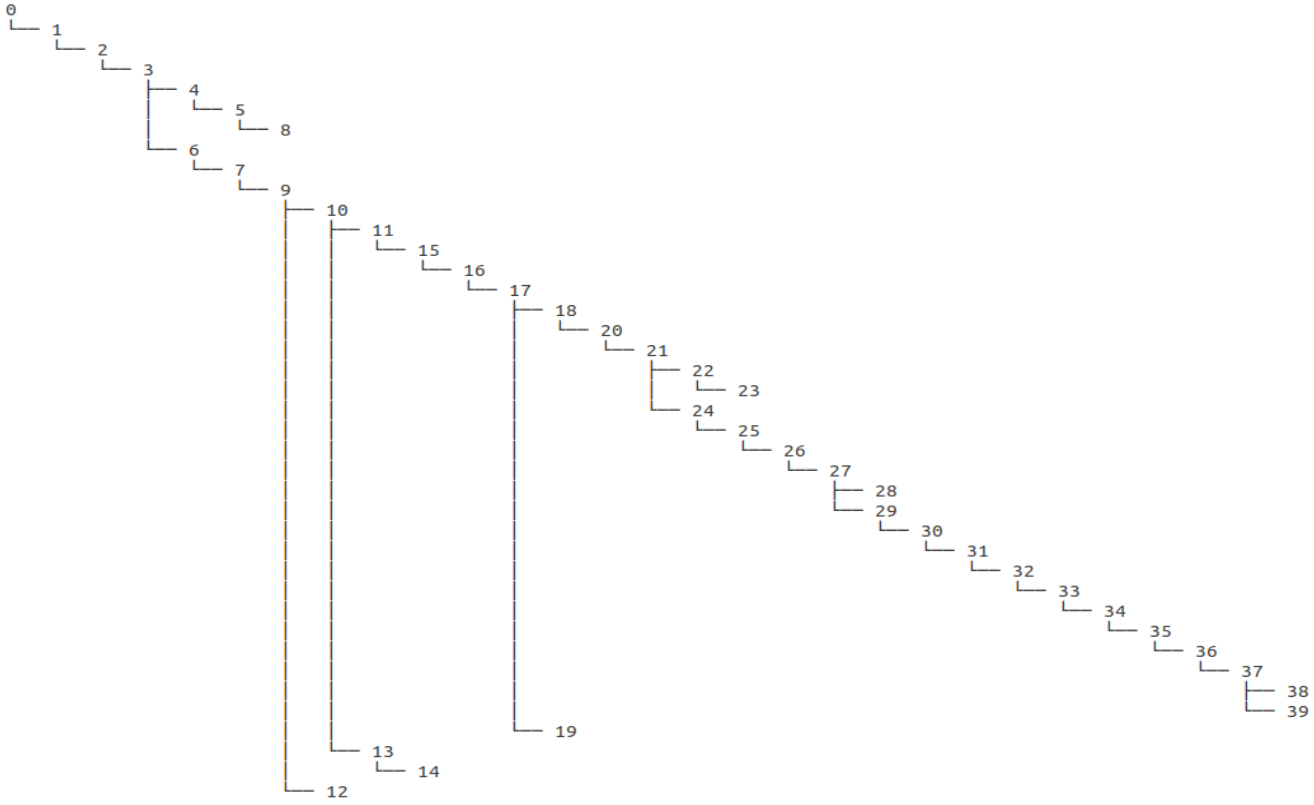
In the main.cpp file, if we set the **invalid** parameter to be **true,** then the 0th peer becomes lazy, i.e some of the blocks it generates may be invalid.

1. All peers have same hashing power

Number of peers, *num*	5
Mean T_k	4000 milliseconds for all peers
T_x	250 milliseconds
z	100
Seed	10
Total Run Time	30 seconds

Invalid Status	True

Invalid Block IDs: 4 5 8 14 22 23 28;



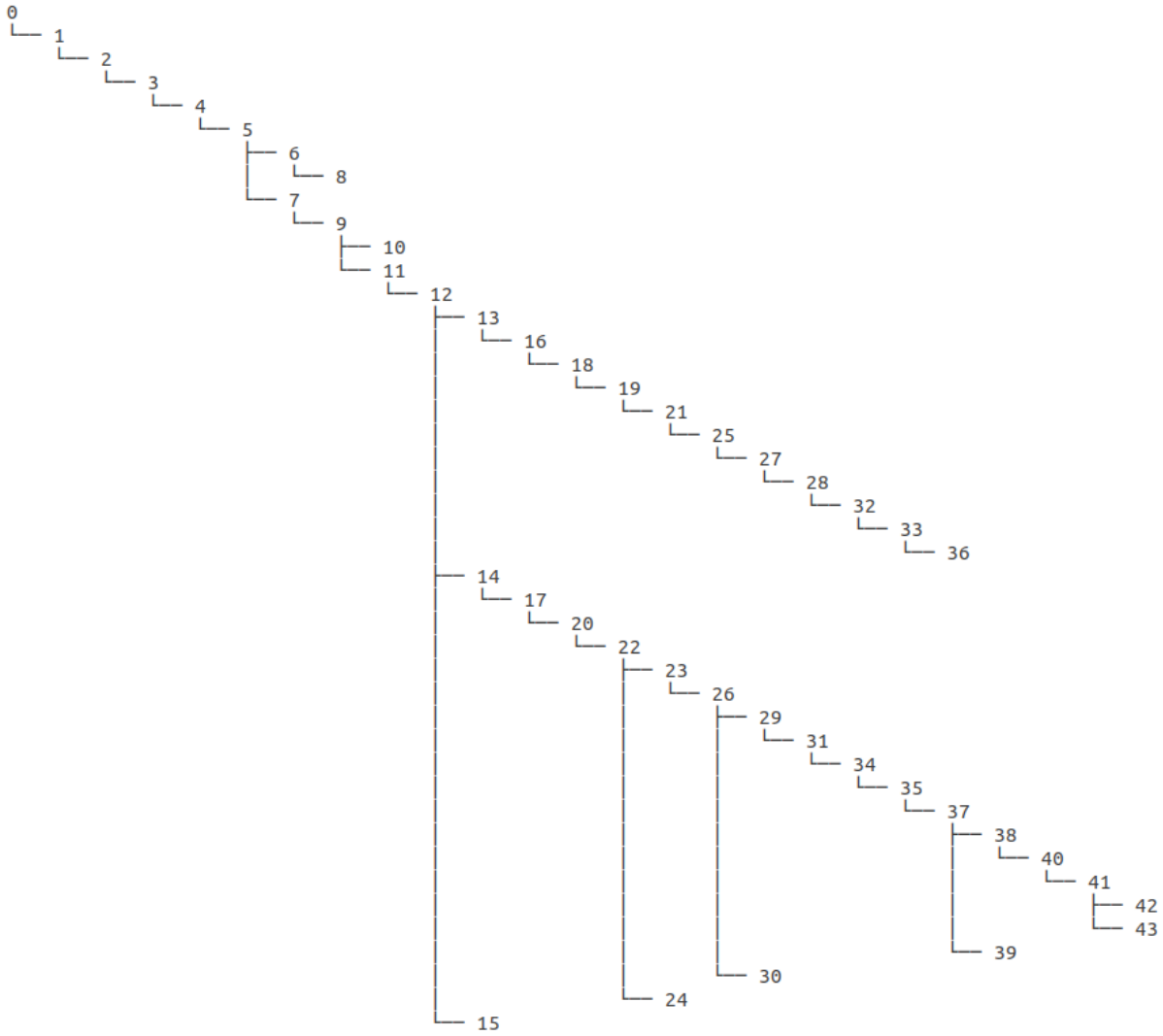
It is evident from the tree that all the chains containing invalid block IDs didn't progress much. This is because all other peers except the 0th peer rejected those blocks and formed a longer chain very soon.

2. Lazy peer has highest but less than 50 percent of the total hashing power

Number of peers, *num*	5
Mean T_k	{2000, 3000, 4000, 50000, 6000} ms for all peers
T_x	250 milliseconds
z	100
Seed	10
Total Run Time	30 seconds
Invalid Status	True

Invalid Block IDs: 6 8 13 16 18 19 21 25 27 28 32 33 36 42

Block tree looks like:



So, in this case the 0th peer can create much longer chains with invalid blocks but soon the remaining peers will form a longer chain since they have the majority of the hashing power. This will result in the 0th peer mining on their chain.

3. Lazy peer has more than 50 percent of the total hashing power

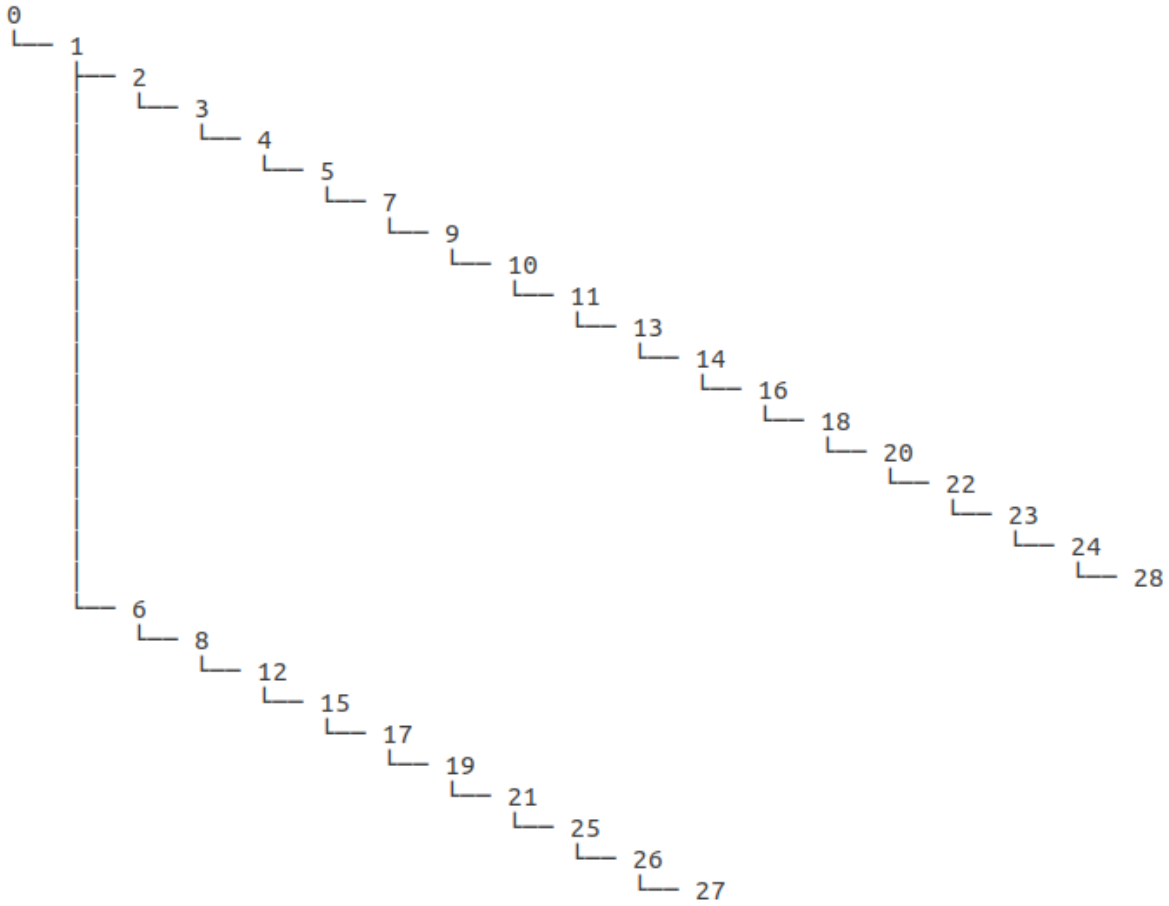
   Parameters:

Number of peers, *num*	5

Mean T_k	{2000, 4000, 8000, 160000, 32000} ms for all peers
T_x	250 milliseconds
z	100
Seed	10
Total Run Time	30 seconds
Invalid Status	True

Invalid Block IDs: 2 3 4 5 7 9 10 11 13 14 16 18 20 22 23 24 28

Block tree looks like:



In this case since the 0th peer has more than 50% of the total hashing power, it will continue to mine on the invalid blocks and form a longer chain, whereas others will mine on a different chain since they will reject those blocks.