CS 765: Assignment 1

Aditya Agarwal (180050004)
Bhaskar Gharu (1800500xx)
Pratik Mistry (1800500xx)

August 30, 2021

1 What are the theoretical reasons of choosing the exponential distribution?

Since, if we divide time into small blocks of size Δt, and model the arrival of a transaction in each block of time as a Bernoulli trial with probability $\lambda\Delta t$, the probability P that the blocks arrive after nth block is

$$P = \Big(1 - \lambda\Delta t\Big)^n$$

Let $t = n\Delta t$. Now, as the blocks get smaller and smaller, ie $\Delta t \to 0$, $n \to \infty$. So, we get

$$P = \lim_{n\to\infty} \Big(1 - \lambda\frac{t}{n}\Big)^n$$

$$\ln P = \lim_{n\to\infty} n \ln (1 - \lambda\frac{t}{n})$$

$$\ln P = \lim_{n\to\infty} \frac{\ln (1 - \lambda\frac{t}{n})}{\frac{1}{n}}$$

By L'Hopital's Rule

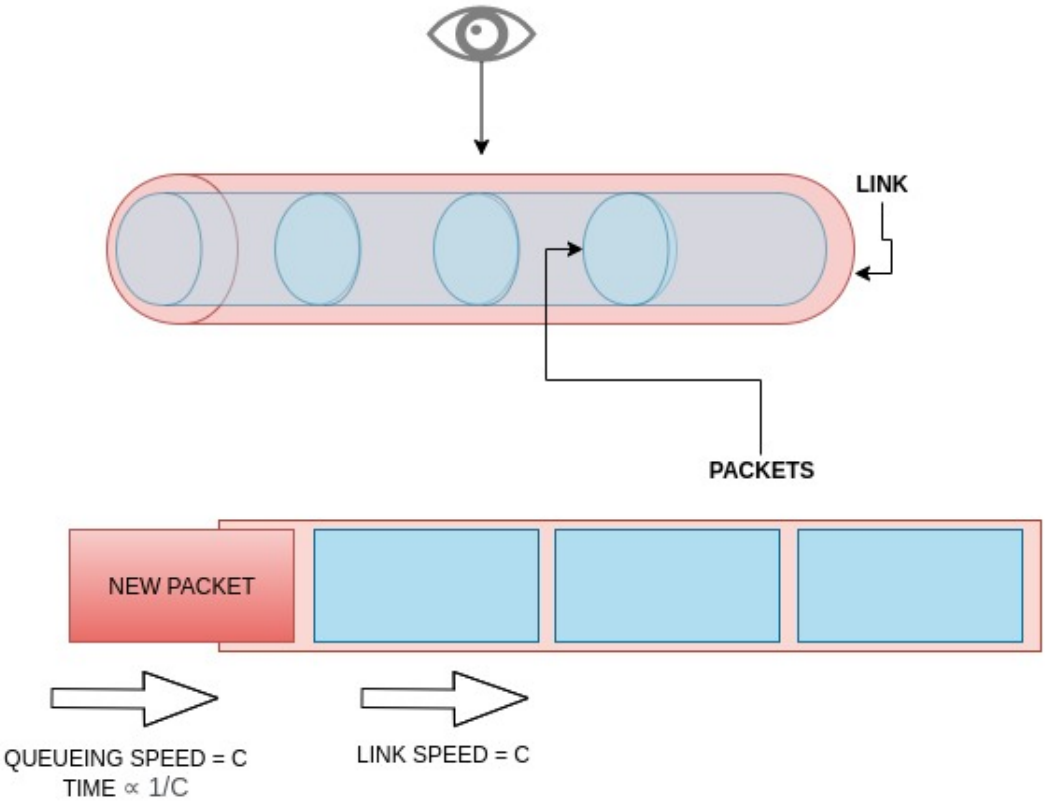
$$P = P(t > t_0) = e^{-\lambda t_0}$$

Which is the exponential distribution!

2 Random Graph Algorithm

- First generated a random sequence of numbers with min = $\lfloor\frac{n}{6}\rfloor$ and max = $\lfloor\frac{n}{2.5}\rfloor$.
- Checked if its a graphical sequence (degree sequence) using `networkx.is_graphical` function. If not, repeated steps 1 & 2.
- Used `networkx.random_degree_sequence_graph` to generate a connected undirected graph with the above degree sequence.
  Note: Internally it uses the research paper referenced at `https://networkx.org/documentation/stable/reference/generated/networkx.generators.degree_seq.random_degree_sequence_graph.html`

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



Note that d_{ij} is the queueing delay for the message and c_{ij} is the link speed. Also we assume that the network is packet switched (which is a fair assumption for computer networks). Now for a packet to be completely sent from the node to the link, there needs to be some space for the packet in the link. And the time taken for that space to be created (and hence the new packet to be inserted) is $\frac{\textbf{packet size}}{\textbf{link speed}}$, which is inversely related to the link speed (c_{ij})

4 Justify the chosen mean value for T_k.

(Assume latency to be negligible) Lets say that peer i's block generation time is given by the random variable T_i. So the overall block generation time is given by

$$Y = \min\{T_1, T_2, ..., T_n\} = \min_i(T_i)$$

Now, each of these (independent) random variables is distributed exponentially, and as explained here: **Link**, Y is also exponentially distributed with parameter

$$\lambda = \sum_i \lambda_i$$

where λ_i is the parameter for peer i's exponential random variable. So the corresponding Poisson process (for overall block generation) is $Poisson(\lambda)$. So the average rate is given by λ blocks/sec. So if we need 1 block in t seconds, then rate is $1/t$ blocks/sec. And hence, we

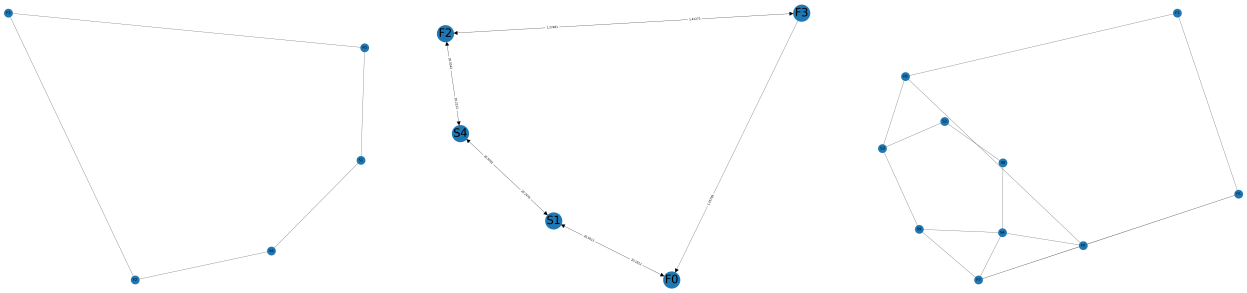
need that

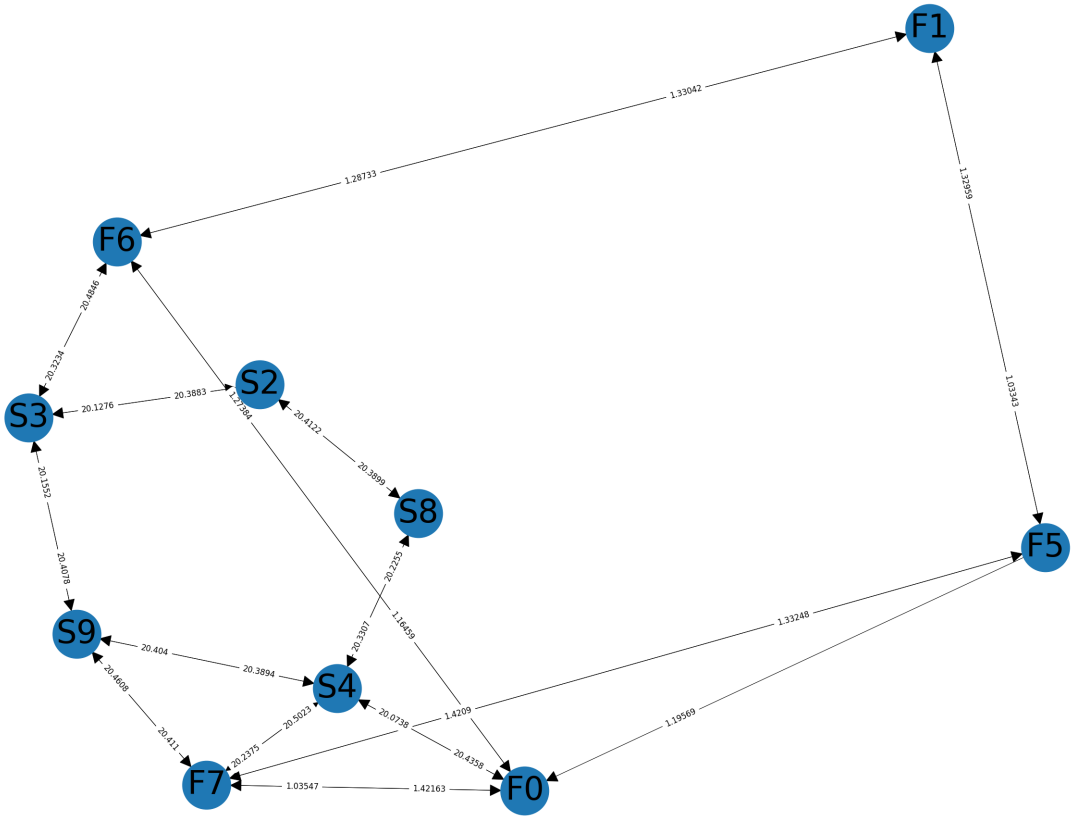
$$\frac{1}{t} = \sum_i \lambda_i = \sum_i \frac{1}{t_i}$$

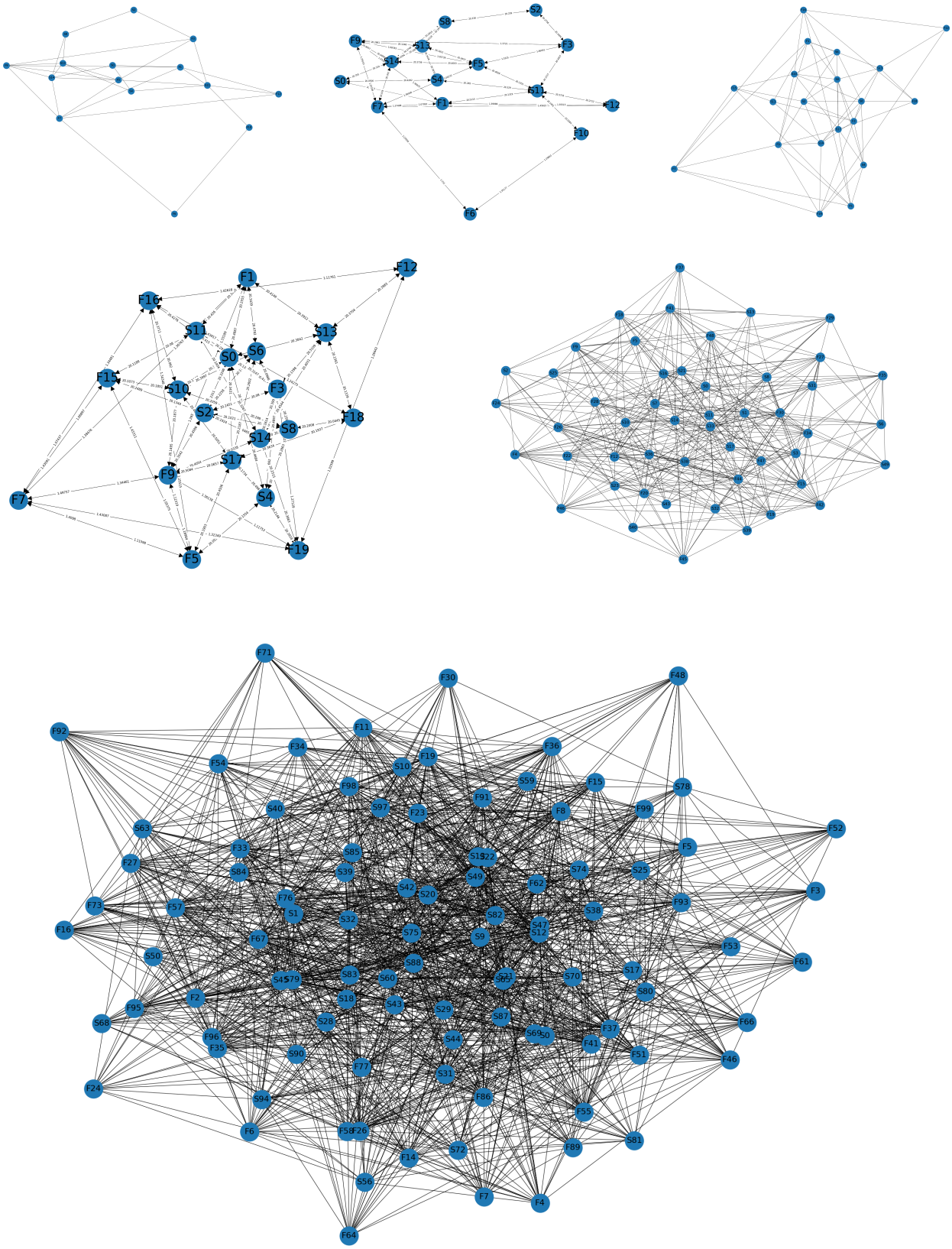
where t_i is peer i's interarrival time for blocks.
So we choose t_i(s) such that the above equation holds. Note that $1/t_i$ is proportional to the hashing power, so we generate a random array of hashing powers and choose appropriate t_i(s).

5 Proper study with a particular visualization tool using different parameters. Insight and critique of the observed values.

5.1 Examples of some randomly generated network topologies







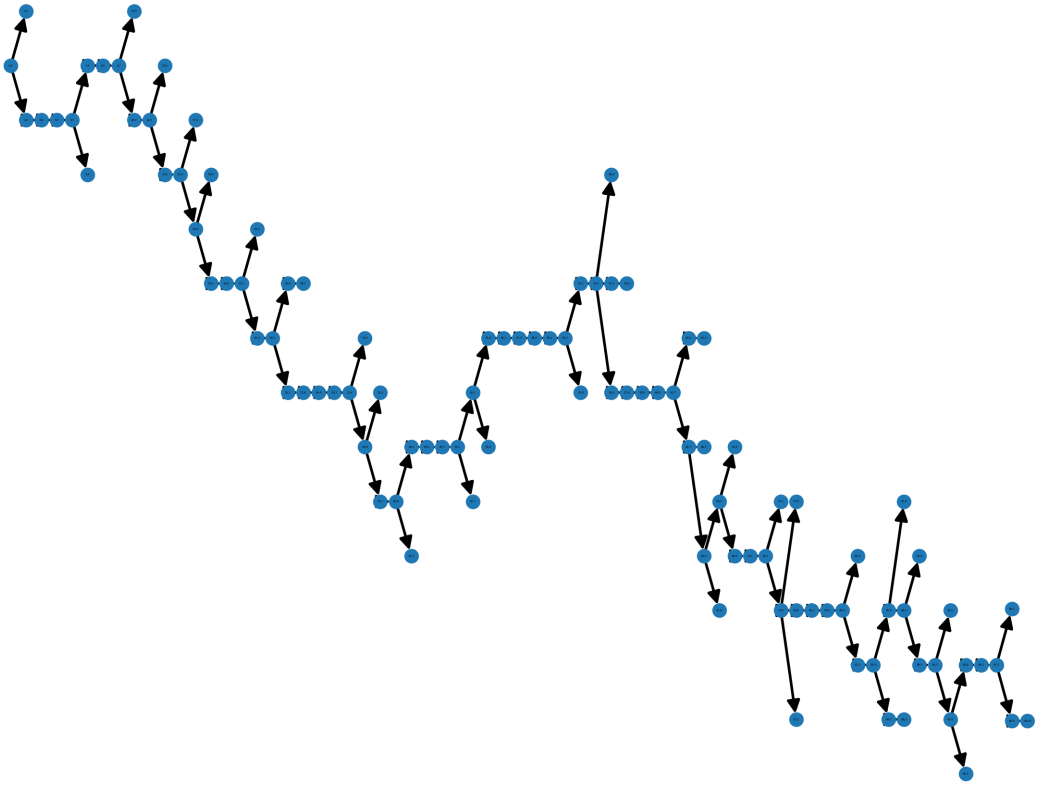
As you can see, the latencies are typically $\tilde{1}$ second (for Fast-Fast) or $\tilde{2}0$ seconds (for others). (Its more or less the same regardless of the parameters)

5.2 Different blockchain trees with the parameters

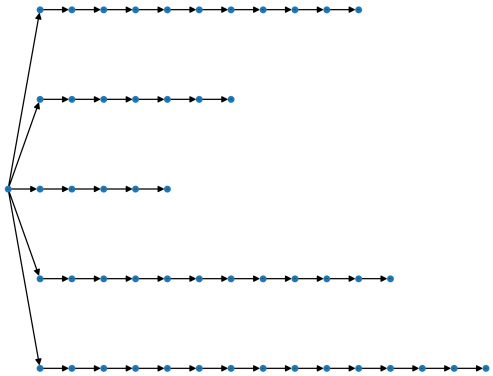
The parameters we are using are:

- n = num_peers
- z = slow_node_fraction
- t = average interarrival time for transactions (overall)
- T = average interarrival time for blocks (overall)
- M = Maximum blocks mined

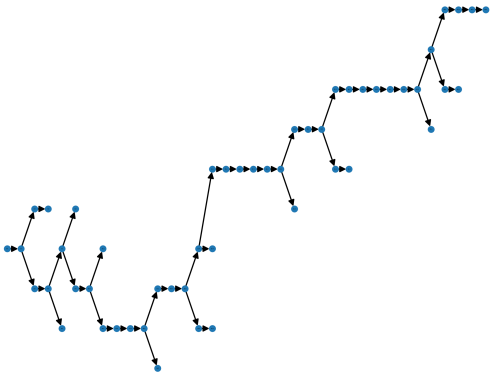
The following are the blockchains we got (Seed used: 570436):



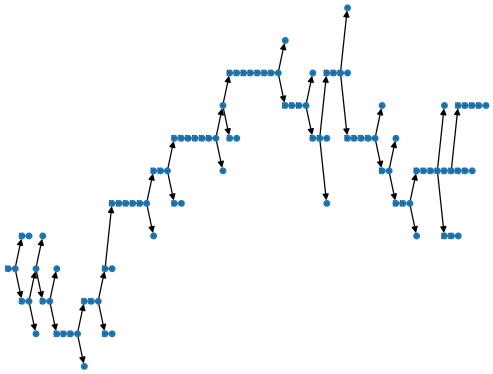
This is for 10 peers, z = 0.5, t = 0.1, T = 1.0, M = 100.
Turn page for rest of the figures



1. **(n = 5, z = 0.0, t = 1e-4, T = 1e-3, 50)**

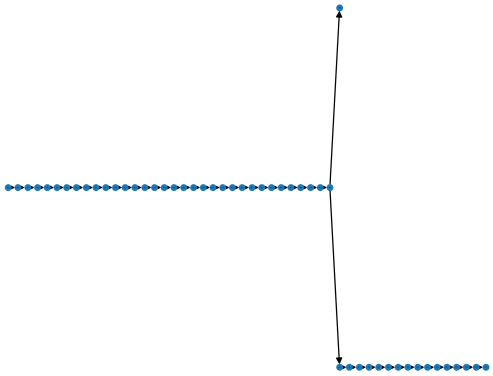


2. **(n = 5, z = 0.0, t = 0.1, T = 1, 50)**

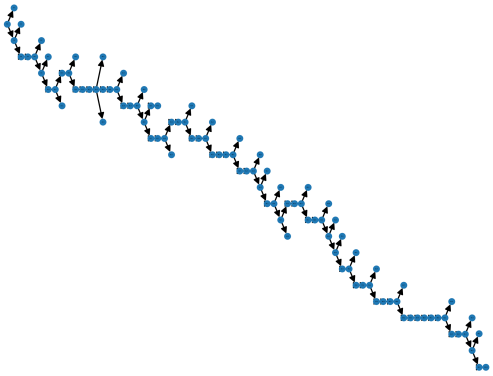


3. **(n = 5, z = 0.0, t = 0.1, T = 1, 100)**

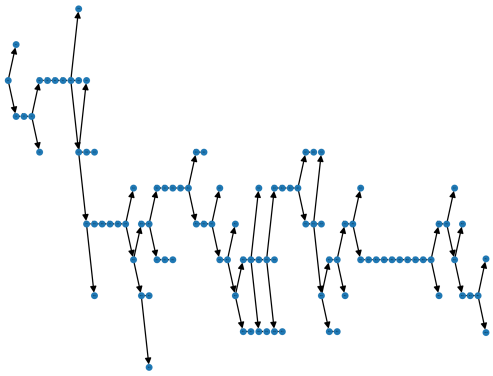
4. **(n = 5, z = 0.0, t = 20, T = 600, 100)**



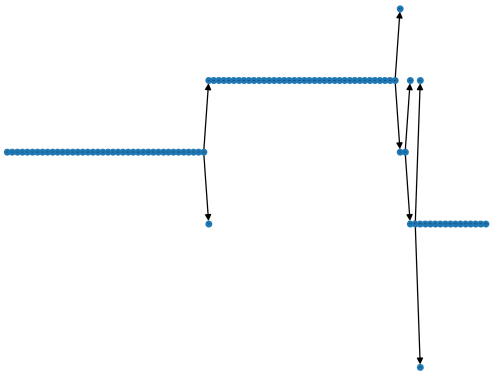
5. **(n = 5, z = 0.25, t = 0.5, T = 10, 50)**



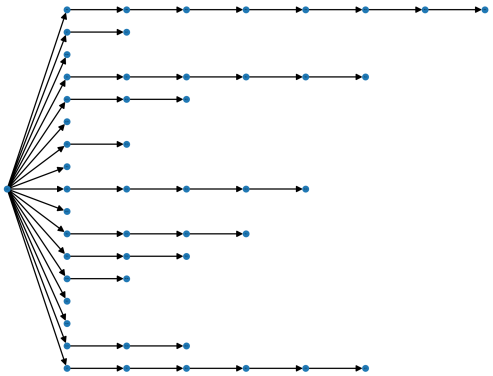
6. **(n = 10, z = 0.0, t = 0.1, T = 1, 100)**



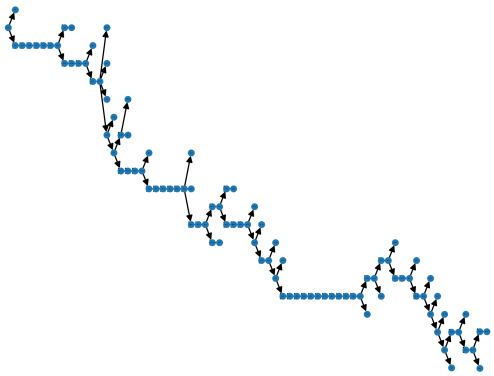
7. **(n = 10, z = 1.0, t = 0.1, T = 1, 100)**



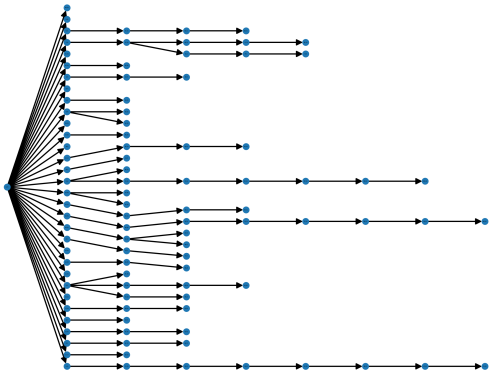
8. **(n = 10, z = 1.0, t = 2, T = 10, 100)**



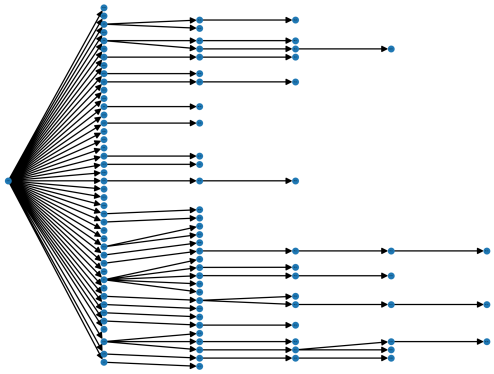
9. **(n = 20, z = 0.5, t = 1e-4, T = 1e-3, 50)**



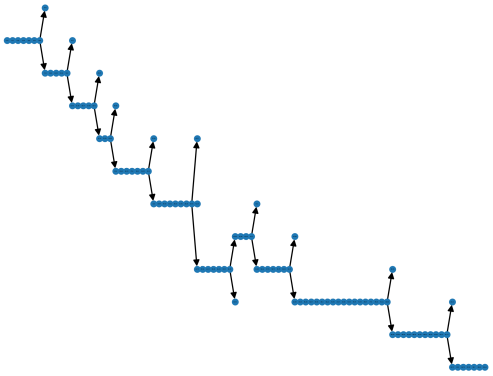
10. **(n = 20, z = 0.5, t = 0.1, T = 1, 100)**



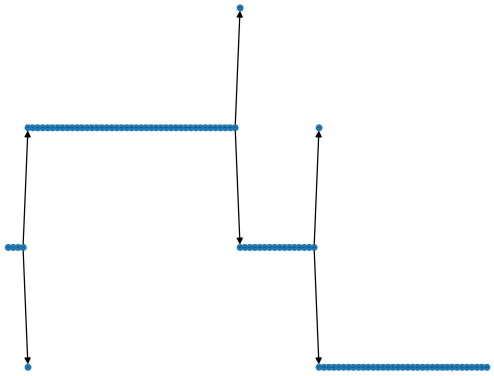
11. **(n = 50, z = 0.5, t = 1e-4, T = 1e-3, 100)**



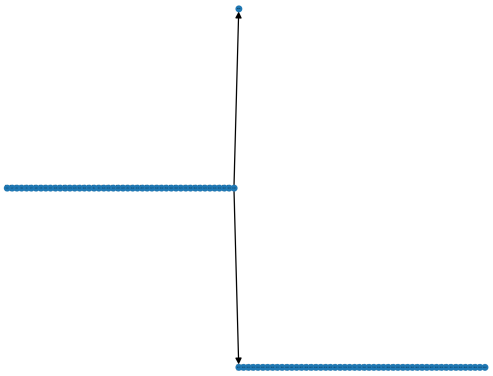
12. **(n = 100, z = 0.5, t = 1e-4, T = 1e-3, 100)**



13. **(n = 100, z = 1.0, t = 0.1, T = 1, 100)**



14. **(n = 100, z = 1.0, t = 0.5, T = 10, 100)**



15. **(n = 100, z = 1.0, t = 2, T = 10, 100)**

16. **(n = 100, z = 1.0, t = 10, T = 50, 100)**

Note that the transmission times are approximately 1 and 20 as discussed earlier.
Note the variation in blockchains. At the **left extreme** (figure 1,9), when the transmission times are way larger than the overall block interarrival time, every peer just keeps on mining on their own chains (since by the time they get 1 block of others, they have already generated too many of their own blocks), resulting in n chains each originating at the genesis block itself.
At the **right extreme**, which is the more realistic case, when overall block interarrival time is way larger than the transmission time, by the time next block is generated, every peer receives the complete update blockchain, so every peer more or less mines on the same longest chain! This results in a linear chain, with forks in rare cases. (Figure 4, 16).
Now the rest of the cases are **middle cases**, where we get longer or smaller side branches depending upon which extreme we are closer to. Most of these cases have heavy branching (forks). (Figure 2, 3, 6, 7, 10, 13)

5.3 Effects of CPU power and transmission time on Ratio (R) = $\frac{\textbf{No. of blocks in longest chain by the peer}}{\textbf{Total blocks by that peer}}$

Note, we have assigned uneven hashing powers for analysis purposes only:

```
Peer i (speed), hash power:  R
Peer 0 (slow), 36.32:  0.00297619
Peer 1 (slow), 42.8265:  1
Peer 2 (slow), 8.49378:  0
Peer 3 (slow), 12.3597:  0
```

For the left extreme case, the peer with highest hashing power mines the longest chain which becomes the longest chain in the end (in the left extreme). Similar observation with fast peers.

```
Peer i (speed), hash power:  R
Peer 0 (slow), 51.3382:  1
Peer 1 (slow), 3.65564:  1
Peer 2 (slow), 35.1145:  1
Peer 3 (slow), 9.89162:  1
```

In the **right extreme**, there is just one chain, so all have fraction 1.

```
Peer i (speed), hash power:  R
Peer 0 (slow), 51.3382:  0.828678
Peer 1 (slow), 3.65564:  0.366667
Peer 2 (slow), 35.1145:  0.522388
Peer 3 (slow), 9.89162:  0.353535
```

In the **middle case**, the one with higher hashing power still has higher fraction!

```
Peer i (speed), hash power:  R
Peer 0 (slow), 51.3382:  0.960317
Peer 1 (slow), 3.65564:  0.878049
Peer 2 (slow), 35.1145:  0.899713
Peer 3 (slow), 9.89162:  0.897196
```

As we **progress towards right extreme**, fractions start to level!

So we can see that the peer with higher hashing power has better R.

Now note the speeds:

```
Peer i (speed), hash power:  R
Peer 0 (fast), 13.1616:  0.126437
Peer 1 (fast), 15.5195:  0.322148
Peer 2 (fast), 3.07797:  0.0327869
Peer 3 (slow), 4.4789:  0
Peer 4 (fast), 0.897258:  0
Peer 5 (fast), 15.706:  0.544304
Peer 6 (fast), 14.0835:  0.242321
Peer 7 (fast), 11.6224:  0.0321285
Peer 8 (slow), 4.2073:  0
Peer 9 (fast), 17.2456:  0.626582
```

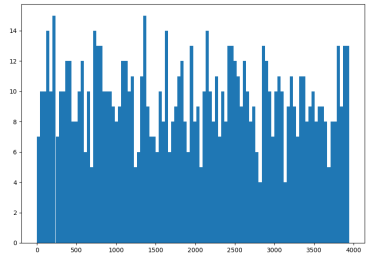
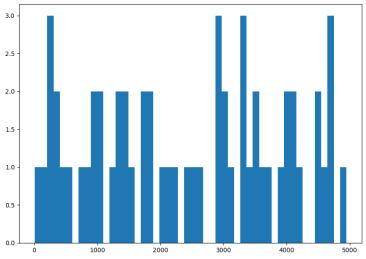
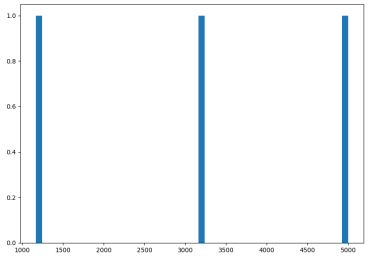
```
Peer i (speed), hash power:  R
Peer 0 (fast), 13.1616:  0.155797
Peer 1 (fast), 15.5195:  0.351852
Peer 2 (slow), 3.07797:  0.0454545
Peer 3 (slow), 4.4789:  0
Peer 4 (slow), 0.897258:  0
Peer 5 (slow), 15.706:  0.471154
Peer 6 (slow), 14.0835:  0.205036
Peer 7 (slow), 11.6224:  0.0577778
Peer 8 (slow), 4.2073:  0
Peer 9 (slow), 17.2456:  0.564565
```

For peer 2, fast $\rightarrow$ slow $\implies$ slight increase
For peer 5, fast $\rightarrow$ slow $\implies$ decrease
For peer 6, fast $\rightarrow$ slow $\implies$ decrease
For peer 7, fast $\rightarrow$ slow $\implies$ slight increase
For peer 9, fast $\rightarrow$ slow $\implies$ decrease

So overall trend seems to be that **when a peer gets slower, its R also decreases.** And its expected, since if it becomes slow, it spends more time in mining on the "wrong" (non-longest) chain.

5.4 How long are branches of the tree measured in number of blocks?

Below are the lists containing the aforementioned for each corresponding figure in section 5.2. The observations are consistent with the inferences drawn in the end of section 5.2. Below is the variation of frequency histograms as we go from the left extreme to the right extreme. (for 5000 blocks)

1. [5, 7, 12, 15, 11,]
2. [4, 31, 33, 35, 25, 21, 15, 15, 11, 7, 5, 3,]
3. [4, 31, 33, 44, 46, 46, 49, 54, 67, 69, 65, 63, 59, 56, 49, 40, 25, 21, 15, 15, 11, 7, 5, 3,]
4. [100,]
5. [34, 49,]
6. [6, 17, 24, 48, 70, 69, 68, 65, 58, 54, 51, 49, 47, 44, 41, 40, 38, 37, 34, 30, 27, 22, 20, 14, 14, 10, 8, 5, 2, 1,]
7. [4, 11, 18, 18, 21, 31, 33, 35, 34, 42, 43, 55, 61, 61, 58, 57, 45, 40, 40, 39, 32, 29, 27, 25, 16, 10, 11, 9, 9, 1,]
8. [40, 80, 82, 95, 82, 78,]
9. [1, 4, 5, 2, 8, 3, 1, 6, 1, 6, 3, 1, 3, 1, 1, 2, 2,]
10. [9, 15, 30, 32, 51, 53, 63, 68, 67, 65, 62, 61, 60, 58, 55, 39, 38, 36, 35, 26, 26, 20, 17, 17, 14, 14, 14, 12, 1,]
11. [8, 3, 4, 2, 1, 3, 2, 2, 1, 1, 5, 5, 2, 1, 3, 1, 1, 3, 3, 3, 3, 3, 4, 2, 2, 8, 4, 4, 2, 1, 2, 1, 3, 2, 2, 1, 2, 7, 2,]
12. [2, 4, 3, 1, 1, 3, 5, 3, 1, 2, 4, 3, 2, 2, 5, 4, 5, 3, 2, 1, 1, 1, 3, 2, 2, 1, 1, 1, 4, 1, 2, 2, 1, 2, 2, 1, 3, 1, 1, 1, 2, 1, 3, 2, 1, 3, 1, 2, 2, 1, 2, 2, 2, 1,]
13. [7, 42, 88, 82, 71, 53, 46, 35, 35, 27, 20, 17, 12,]
14. [4, 63, 97, 47,]
15. [99, 48,]
16. [99, 93,]