
Introduction to Blockchains

Course Code - CS765

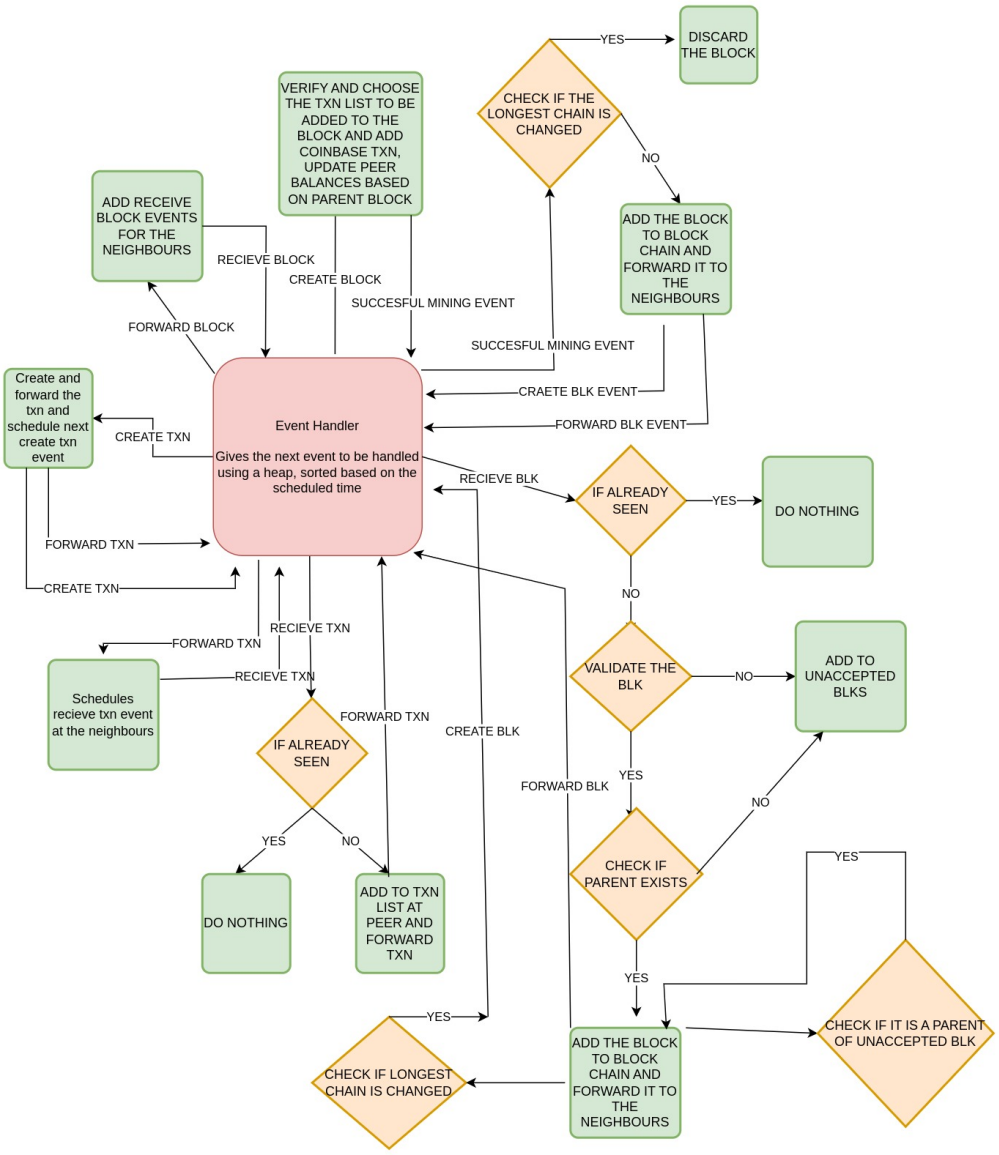
# Assignment 1

Report

**210050105,210050098,210050034**

February 2024

# 1 Code Flow




# 2 Questions

## 2.1 Q1. What are the theoretical reasons for choosing the exponential distribution?

1. The process of generation of transactions by peers is independent of each other
2. So the transactions arrive in the network randomly and independently at a constant average rate

3. This can be seen as a poisson process, the time between successive transactions, therefore follows an exponential distribution consistent with the properties of the poisson process

## 2.2 Q2. Why is the mean of $d_{ij}$ inversely related to $c_{ij}$? Give justification for this choice

1. $d_{ij}$ represents the queuing delay at the node, this queuing delay arises due to congestion of the packets at the node.
2. So a higher link speed means that more number of packets can be transmitted in a unit time.
3. Consequently, packets spend less time waiting in queues because they can be transmitted more quickly

## 2.3 Q3.Why mean = I/$h_k$?

1. We aim to determine the value of $T_k$ such that the average inter-arrival time between any two blocks from any two nodes is I
2. We need to ensure that peers with more hashing power can generate a block in less time, So the mean of $T_k$ is inversely proportional to $h_k$
3. If we keep mean of $T_k$ as I/$h_k$, then it satisfies both the conditions one and two

## 2.4 Fork resolution (Q7)

1. When does this situation occur? For instance, if I've added a block to my chain but haven't received another miner's block on the same parent, resulting in a fork.
2. How is this issue resolved? By focusing on mining the longest blockchain.
3. How do I address this? By creating a new block, such as block A, only when there's a change in the longest chain.
4. What actions do I take based on chain updates? If I'm mining on the longest chain and receive a new block like B, I stop creating blocks and switch to B. Conversely, if block B arrives on the parent of the last block of the longest chain, like C, I continue mining because my longest chain is still unchanged. If another block, D, arrives on B, indicating a change in the longest chain, I stop mining and transition to mining on block D.

## 2.5 Loop-less transaction and block forwarding

1. Solution is simple, We don't forward to the block from which we received it.
2. We maintained the list of all transactions and blocks the peer has received at that peer, if the received item exists in these lists we don't forward it

## 2.6 Validation of Blocks and Transactions

1. We validated the transactions of the received blocks before accepting them into the chain by calculating the peer balances using the transactions in the chain to which the block is getting into, if any peer balance becomes less than zero, we add the block into the cache blocks.
2. Because they might be valid because of the transactions received in the future blocks, then we validate again and add the block to the chain
3. To reduce program runtime for calculating the peer balances on the whole chain, we maintained a peer balances list in the blocks, whose balance is calculated based on it's ancestors

### 2.7 Longest chain Identification

1. We stored all the blockchains as a tree, and the longest chain is identified by the depth parameter in the block, defined by depth of child = 1 + depth of parent.
2. If the depth is maximum for a block, then it is the last block of the longest chain, and all its ancestors are part of the blockchain

## 3 Assumptions

- We assumed the number of peers in the network is more than 6, because we were asked to give 3 to 6 neighbors for each node.
- We assumed every peer has a balance of 100 BTC initially.
- To reduce the number of invalid blocks, we assumed each transaction involves only 1 to 3 BTC.

## 4 Analysis and Observations

### 4.1 Block-interarrival Time

- As shown from the below figures 1 and 2, it is evident that the branching increases as the block inter-arrival time decreases
- Because more blocks can be created even before a block propagates to all the nodes

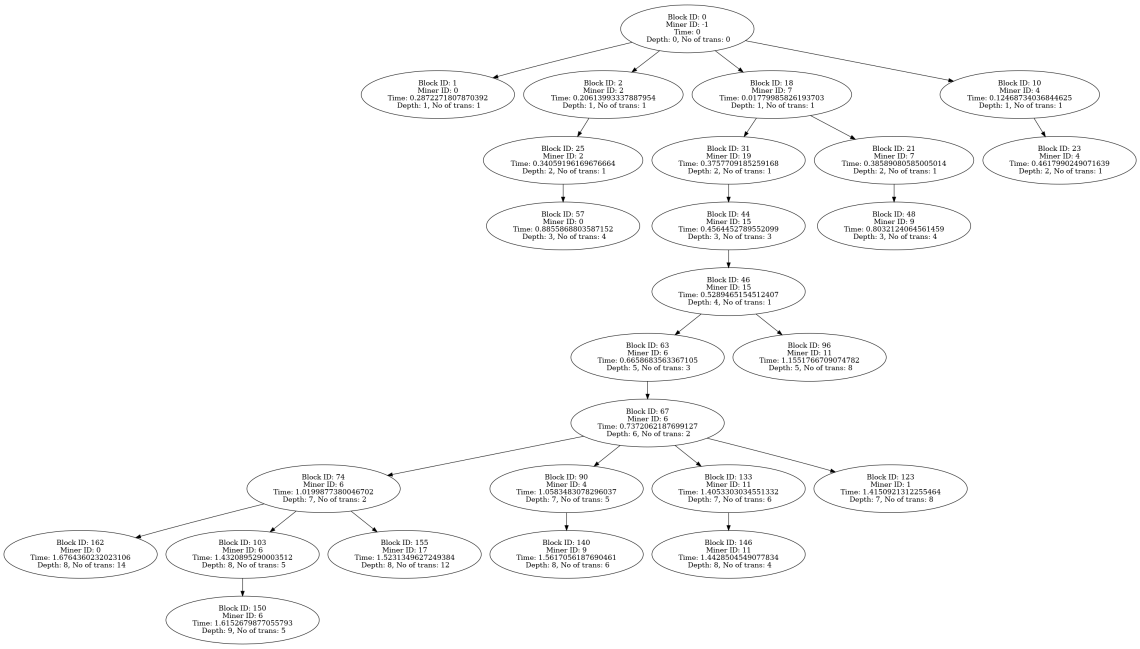



Figure 1: for 20 peers,50 slow,50 lowCPU,100(ms) block interarrival time,10000 iterations

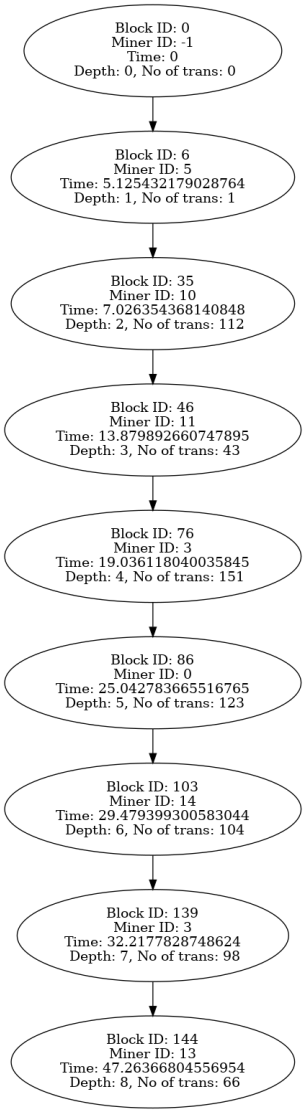



Figure 2: for 20 peers,50 slow,50 lowCPU,10000(ms) block interarrival time,200000 iterations

## 4.2 Ratio

```
no of peers: 20
 slow_percent: 50
 lowCPU_percent: 50
 tmean: 1.0
 max_iterations: 200000
 mining_time: 1.0
Blocks in the longest chain: 44
Total Blocks created by all peers: 59
Ratio of Blocks in the longest chain is 0.7457627118644068


Blocks in the longest chain created by slow and low CPU peers: 0
Total Blocks created by slow and low CPU peers: 1
Ratio of Blocks created by slow and low CPU peers in the longest chain: 0.0


Blocks in the longest chain created by slow and high CPU peers: 28
Total Blocks created by slow and high CPU peers: 37
Ratio of Blocks created by slow and high CPU peers in the longest chain: 0.7567567567567568


Blocks in the longest chain created by fast and lowCPU peers: 4
Total Blocks created by fast and lowCPU peers: 6
Ratio of Blocks created by fast and lowCPU peers in the longest chain: 0.6666666666666666


Blocks in the longest chain created by fast and high CPU peers: 12
Total Blocks created by fast and high CPU peers: 15
Ratio of Blocks created by fast and high CPU peers in the longest chain: 0.8
```

1. Consider the case where number of peers = 20, percentage of slow peers = 50 and percentage of low CPU = 50, I = 1000(ms) , transaction interarrival time = 1000(ms) and number of iterations = 2,00,000
2. Here it is evident from the above data that even though there are equal number of peers with low and high hashing power, the peers with high CPU created 53 blocks, whereas the peers with lowCPU created only 7 blocks
3. The ratio of blocks in the longest chain is higher for the peers with high hashing power
4. Increasing the number of low hashing power nodes results in longer block generation times, consequently, this extended time facilitates the resolution of forks, thereby reducing the occurrence of branching, keeping the other parameters constant increasing the low CPU percentage increases the ratio of blocks in the longest chain
5. The slow peers have less ratio than the fast peers because it takes more time for the blocks to arrive at the peer, so there is a higher chance that another block will be created.
6. Increasing the block inter-arrival time increases the ratio of blocks in the longest chain, because of the extra time for the fork resolution.