P2P Cryptocurreny-Network

Arijit Saha 210050017
Aryan Mathe 210050021
Khushang Singla 210050085

February 2024

1 Introduction

Blockchains has been of great interest now-a-days because of its various advantages such as decentralization, building trust, security etc. For this assignment we created an event based **P2P Cryptocurrency Network** in order to analyze and understand blockchains via simulating various events in a blockchains which are **Block Genaration, Block Recieve, Transaction Generation and Transaction Recieve**. This simulator has been built using **C++** language along with some **python** for experimenting and visualizing results.

2 Questions on Simulator Construction

2.1 Theoritical reasons for choosing transaction inter-arrival time as exponential distribution ?

Transactions in a blockchain should be independently generated with respect to generation time of other available transactions. Also, the transactions in a real blockchain network are a part of a distributed system which leads to a complete independence among them.

Since we know that the exponential distribution has Memorylesness property which implies that the time of generation of every transaction will be independent of the time of generation of other transactions. This can be mathematically represented as follows

$$Pr(T > s + t \mid T > s) = Pr(T > t)$$

where T is the random variable from exponential distribution which represents the interarrival time between transaction genaration.

2.2 Dependence between link speed (c_{ij}) and queuing delay (d_{ij})

When link speed between nodes i and j less then it will have larger queuing delays, on the other hand if link speed is high then this will result in less delay.
This relation is evident from the fact that when link speed gets higher the packets in the network encounters less delay which results in reduced congestion and less queuing at nodes in the network. One the other hand if link speed decreases this will lead to delays in packet transfer because of queuing at nodes in the network.

3 Simulator Running Instructions

```
$ ./blockSimWrapper.py
usage: blockSimWrapper.py [-h] [-t INTERARRIVAL_TRANSACTION_TIME]
                          [-k INTERARRIVAL_BLOCK_TIME] [-b MAX_BLOCKS] [-a INITIAL_AMT]
                          [-z FRAC_SLOW] [-o FRAC_LOW_CPU] [-s SEED] [-x MAX_TRANSACTIONS]
                          [-n NUM_PEERS]

options:
  -h, --help   show this help message and exit
  -t INTERARRIVAL_TRANSACTION_TIME, --interarrival_transaction_time
                        mean interarrival transaction time
  -k INTERARRIVAL_BLOCK_TIME, --interarrival_block_time INTERARRIVAL_BLOCK_TIME
                        mean interarrival block time
  -b MAX_BLOCKS, --max_blocks MAX_BLOCKS
                        number of blocks to terminate on
  -a INITIAL_AMT, --initial_amt INITIAL_AMT
                        initial coins for each node
  -z FRAC_SLOW, --frac_slow FRAC_SLOW
                        fraction of high latency nodes
  -o FRAC_LOW_CPU, --frac_low_cpu FRAC_LOW_CPU
                        fraction of low cpu(low hashing power) nodes
  -s SEED, --seed SEED  seed for randomness in simultaion
  -x MAX_TRANSACTIONS, --max_transactions MAX_TRANSACTIONS
                        Number of Transactions to terminate on
  -n NUM_PEERS, --num_peers NUM_PEERS
                        Number of peers in blockhchain network(default 100)
```

4 Experiments

The blockchain shown in these figure shows only the top 200 blocks mined in the blokchain network.

4.1 Insights and Critique on forking for variation in Slow (*Networking delay*) Nodes

```
./blockSimWrapper.py −−interarrival_block_time 10 −−frac_low_cpu 0.2 −frac_slow 0.2
./blockSimWrapper.py −−interarrival_block_time 10 −−frac_low_cpu 0.2 −frac_slow 0.4
```

On changing number of slow nodes in the network keeping other parameters same, we expect that there should be not much differences in the network because of the value of inter-arrival block time which is **10 units** and is much higher as compared to worst case network latency for a block transmission which is around **1-2** units. Hence we expect to see almost the same amount of forking in the blockchain tree.

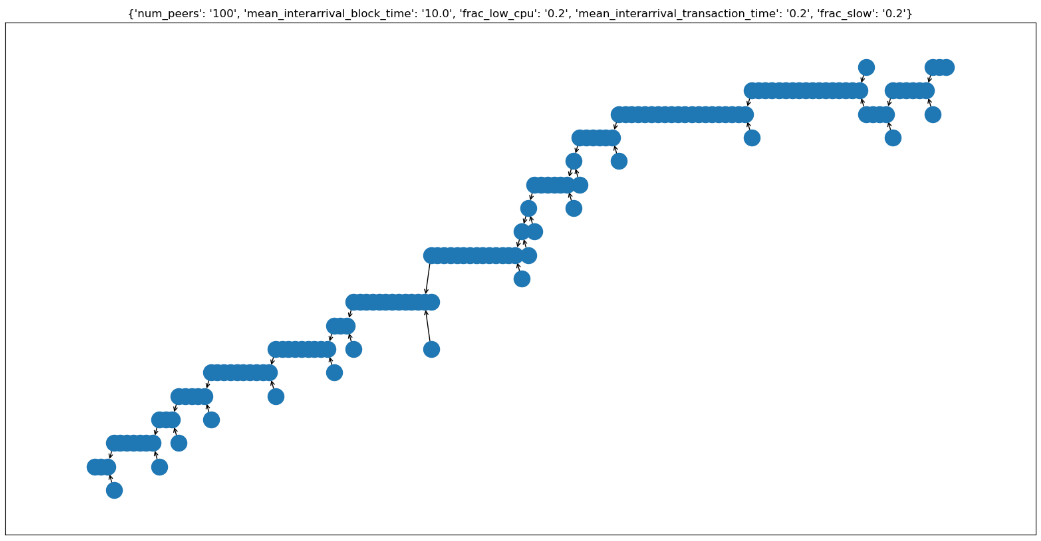
{'num_peers': '100', 'mean_interarrival_block_time': '10.0', 'frac_low_cpu': '0.2', 'mean_interarrival_transaction_time': '0.2', 'frac_slow': '0.2'}

Figure 1: Frac Low CPU: 0.2, Frac Slow: 0.2, Mean Inter-arrival Block Time : 10

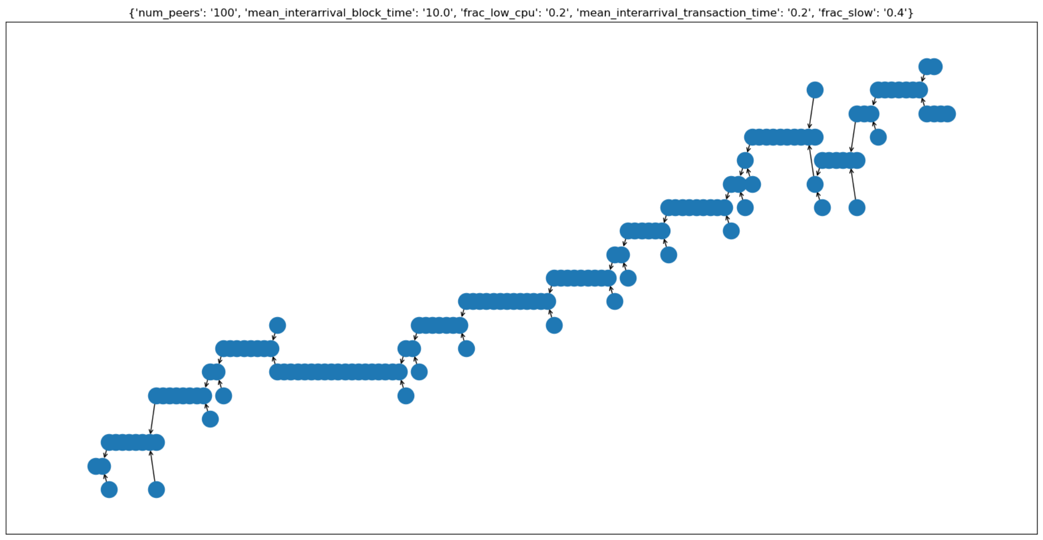
{'num_peers': '100', 'mean_interarrival_block_time': '10.0', 'frac_low_cpu': '0.2', 'mean_interarrival_transaction_time': '0.2', 'frac_slow': '0.4'}

Figure 2: Frac Low CPU: 0.2, Frac Slow: 0.4, Mean Inter-arrival Block Time : 10

Here the results meet our expectations and we don't see much differences in the pattern of block tree. Both the trees have almost same amount of forking as can be seen from the figures shown above.

4.2 Insights and Critique on forking for variation in Low CPU Nodes Fraction

```
./blockSimWrapper.py ——interarrival_block_time 10 ——frac_low_cpu 0.2 —frac_slow 0.2
./blockSimWrapper.py ——interarrival_block_time 10 ——frac_low_cpu 0.4 —frac_slow 0.2
```

When the fraction of nodes having low cpu is increases, we expect that the amount of branching in the blockchain decreases by some amount. Suppose a **low CPU** node tries to generate a block on the blockchain which will lead to forking, but since this node has low cpu power the longest chain might change during the creation of this block with more probability as compared to high cpu node and hence the forking might decrease by some amount.

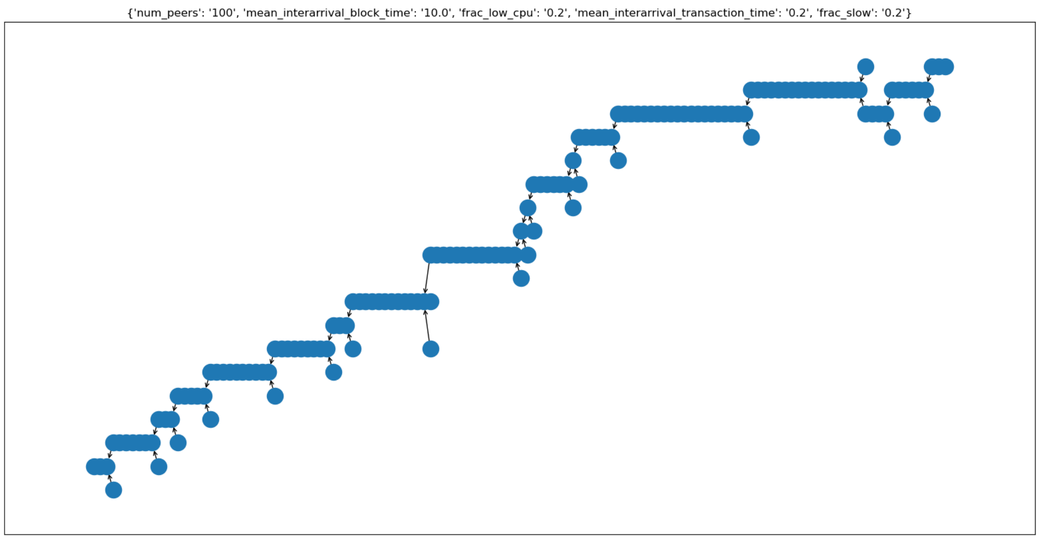
{'num_peers': '100', 'mean_interarrival_block_time': '10.0', 'frac_low_cpu': '0.2', 'mean_interarrival_transaction_time': '0.2', 'frac_slow': '0.2'}

Figure 3: Frac Low CPU: 0.2, Frac Slow: 0.2, Mean Inter-arrival Block Time : 10

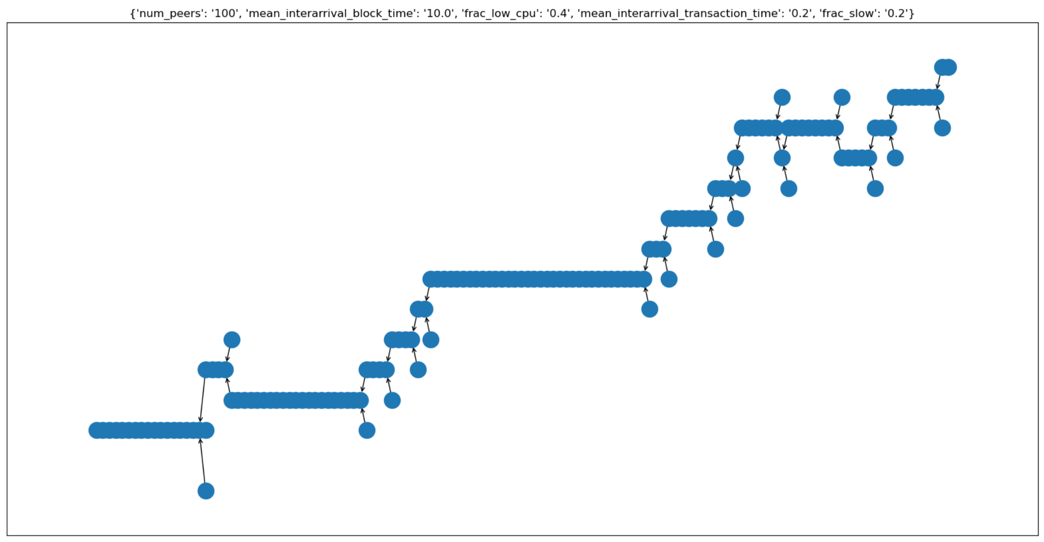
{'num_peers': '100', 'mean_interarrival_block_time': '10.0', 'frac_low_cpu': '0.4', 'mean_interarrival_transaction_time': '0.2', 'frac_slow': '0.2'}

Figure 4: Frac Low CPU: 0.4, Frac Slow: 0.2, Mean Inter-arrival Block Time : 10

In this case we don't observe a lot of decrease in the forking on increasing the amount of **low cpu nodes** and even the forking is almost the same which could be because of the fact that the total hashing power of all the nodes remains same so the total amount of forking in the overall blockchain balances out.

4.3 Insights and Critique on forking for variation in Mean Inter-arrival Block Time

We expect that smaller inter-arrival block time would result in higher number of forks because the network delays in case of smaller inter-arrival time might have a larger effect on generation of forks. Delays caused during transmission will result in the mining of a new block on wrong chain (because the block forming the current longest chain might not have been received by this node) and hence resulting in more frequent forks.

$$Mean\ Interarrival\ Block\ Time \uparrow \implies Forking \downarrow$$

```
./blockSimWrapper.py ——interarrival_block_time 1 ——frac_low_cpu 0.2 —frac_slow 0.2
./blockSimWrapper.py ——interarrival_block_time 20 ——frac_low_cpu 0.2 —frac_slow 0.2
```

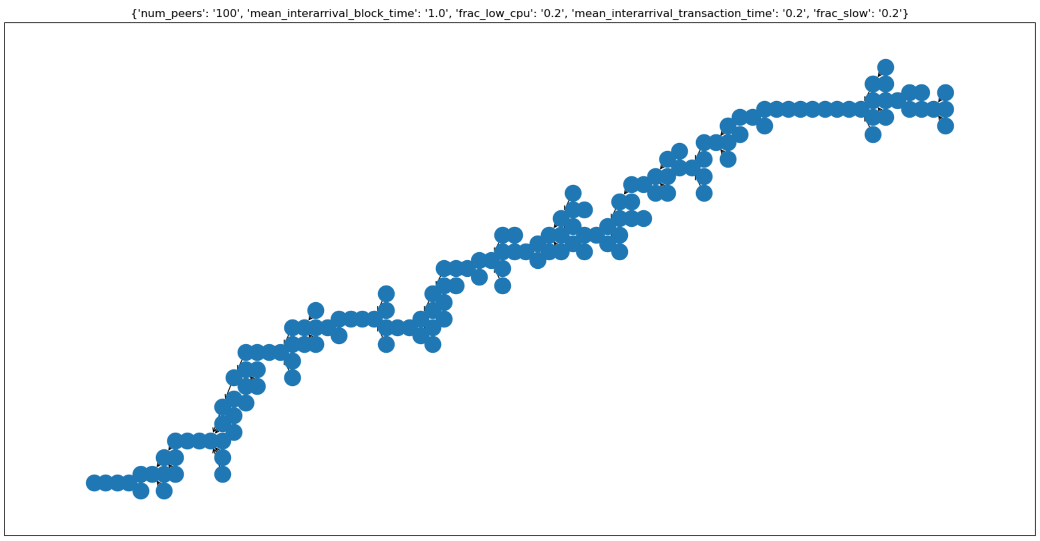
{'num_peers': '100', 'mean_interarrival_block_time': '1.0', 'frac_low_cpu': '0.2', 'mean_interarrival_transaction_time': '0.2', 'frac_slow': '0.2'}

Figure 5: Frac Low CPU: 0.2, Frac Slow: 0.2, Mean Inter-arrival Block Time : 1

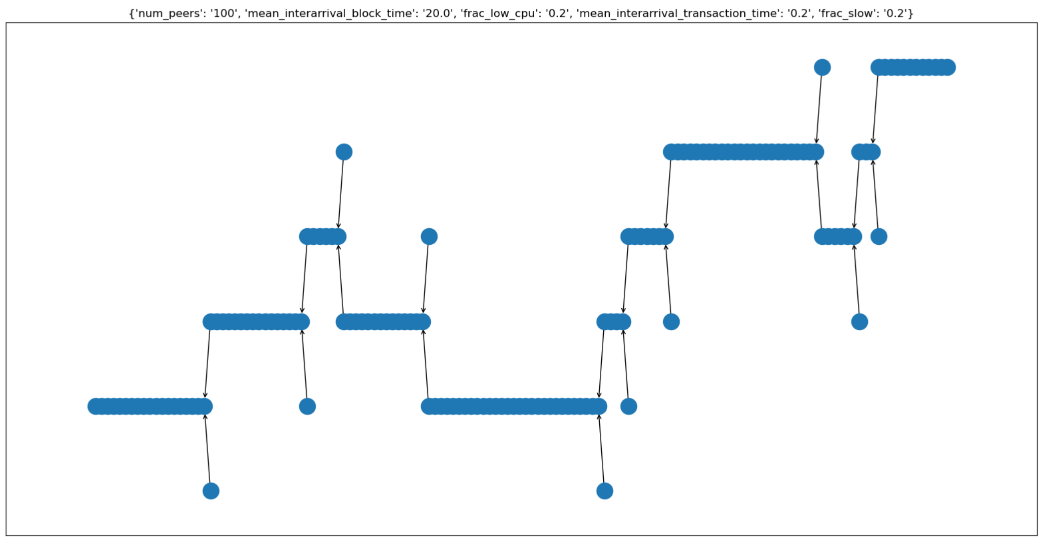
{'num_peers': '100', 'mean_interarrival_block_time': '20.0', 'frac_low_cpu': '0.2', 'mean_interarrival_transaction_time': '0.2', 'frac_slow': '0.2'}

Figure 6: Frac Low CPU: 0.2, Frac Slow: 0.2, Mean Inter-arrival Block Time : 20

It can be clearly observed from the figures, that the results match our expectations. Lesser mean inter-arrival block time lead to larger number of forks while larger inter-arrival time has resulted in less forking.

4.4 Insights and Critique on contribution of blocks by Nodes in blockchain

4.4.1 Case of Longest Blockchain

We had expected that the node which are **Fast** and have **High CPU** would have a larger contribution in the blocks of the longest blockchain which is clearly evident from the fact that such nodes can perform **PoW** more quickly as compared to low CPU nodes. Also, such nodes will mine on the true blockchain for most of the cases because of better network capabilities which keeps them up-to-date w.r.t the blockchain.

We expect **slow and high CPU** nodes to perform better than **fast and low CPU** nodes when average latency in network for receiving a block is very less as compared to mean inter-arrival block time. Also **slow but low CPU** nodes are expected to perform worst because with **PoW**, it's all about how much computing and network capabilities you have!

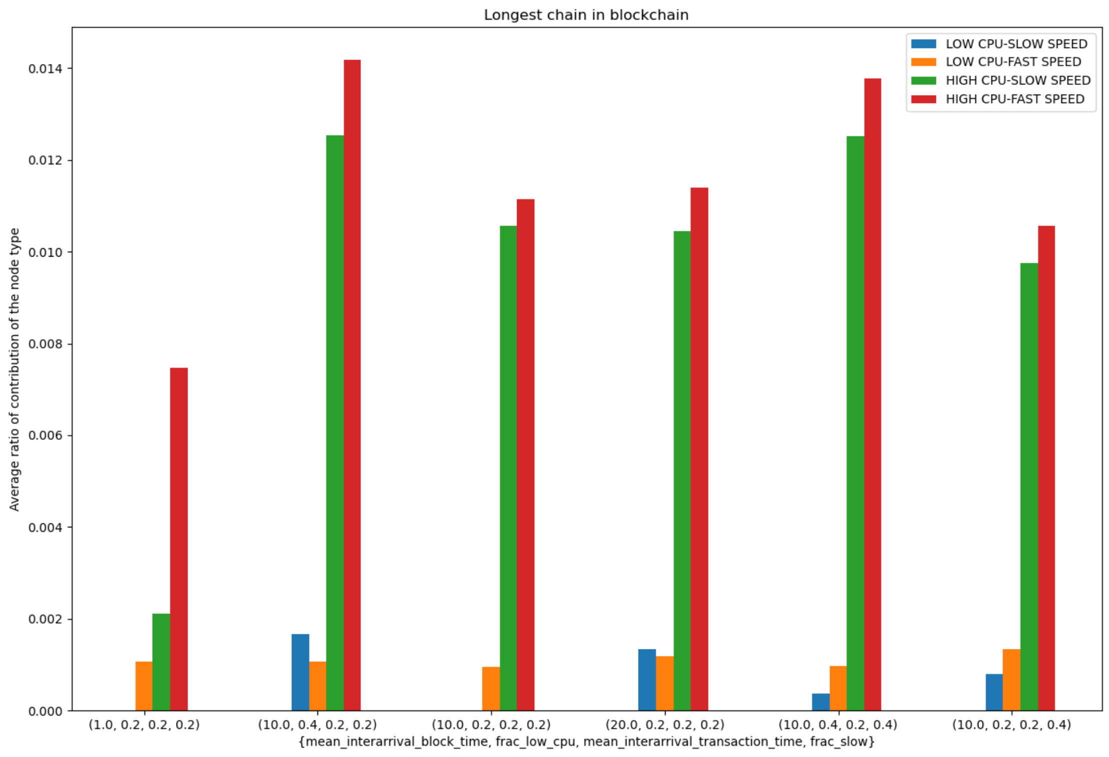

Figure 7: Block Ratio in Full Block Tree

It can be seen from the bar graph shown above that the results somewhat meets what we have expected. It is very interesting to note that, in the case when ($mean\ interarrival\ block\ time = 1.0$), the contribution of blocks by **fast and high CPU** in the blockchain are much higher as compared to **slow and high CPU** nodes which is because of comparable latencies and inter-arrival block time. This might have resulted in slow nodes mine in the wrong chain and hence low contribution in longest chain.

4.4.2 Case of Complete Blockchain Tree

In this case we were expecting that nodes which have high computing power will contribute more in overall block tree because of its high probability to keep itself up-to-date with the blockchain. While nodes which have low computing power will have very low contribution in the overall block tree.

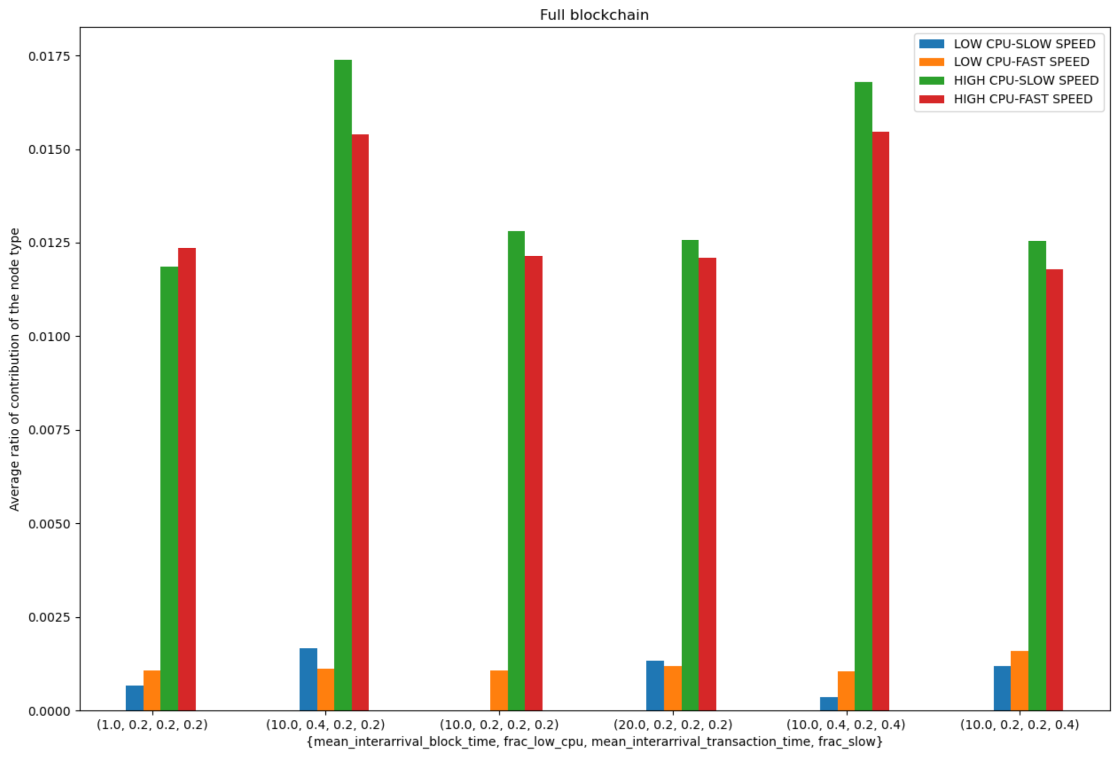

Figure 8: Block Ratio in Full Block Tree

As we can see in the figure shown above that high CPU nodes made a larger contribution overall. But an interesting insight is to note that nodes which have **high CPU but slow networking** mined more blocks than **nodes with high CPU and fast networking**. One of the reason could be that, slow nodes will face less interruption while performing PoW as compared to fast nodes (but many of their blocks could be on the wrong chain) and hence resulting in larger overall contribution. Also it's clear from the figure that low CPU nodes have very low contribution in block generation.

END OF REPORT