# CS765-Assignment 2

Yash Khemchandani - 170050025, Devansh Garg - 170050029, Ajay Sheoran - 170050042

December 8, 2020

## 1 Implementation Details

- The blockchain P2P network is built on top of the P2P network created as a part of Assignment 1. This can be seen in the code as **BlockchainPeer** class is inherited from the **Peer** class implemented in Assignment 1.
- The block header is a binary string of **64 bits**, the components of which are as follows:
  - The first 16 bits are the last 16 bits of the hash of the previous block.
  - Next 16 bits represent the Merkel root which, for this assignment, is random.
  - The final 32 bits encode the timestamp, which is the the number of seconds that have elapsed since January 1, 1970 at 00:00:00 GMT.
- For the ease of implementation, the hashing power fraction of a particular node is **static** (i.e. it doesn't change dynamically when new peers connect to the network) and has to be passed as an argument when running the node.
- Since this assignment was tested on a single machine, with different nodes having different port numbers, an **artificial network delay** was introduced manually. Precisely, whenever a node receives a block from a peer socket, it pushes the block into the network queue. After the network delay (specified when running the node), the block is transferred to the validation queue for further processing.
- If the incoming block passes the validation tests (timestamp is within 1 hour of the current time and the hash in the block is the hash of some other existing block), it is added into the blockchain and the mining is restarted with a new mining time (exponential random variable). However, if the validation tests fail, then the block is discarded and the mining is restarted with the current mining time
- Every peer stores the blockchain locally in a **list of lists** structure, where each index in the outer list represents the level of the blockchain tree and the inner list contains the blocks in that level. Thus, the last level block(s) correspond to the longest chain. If the addition of a new block increments the longest chain, a new level is introduced in the list, otherwise the block is added to the desired level.
- In the experiments that we have performed to generate the blocks, the adversary sends the invalid blocks to the target nodes every **0.5 seconds**. This time interval was taken so that the output files are readable and neat. Moreover, we have tested that having the invalid block generation time to be very low (0.001 seconds) doesn't cause much difference to the plots.
- For our experiments, the P2P network had **10 peers**, out of which one was adversary. This network was facilitated by a single seed

## 2 Plots

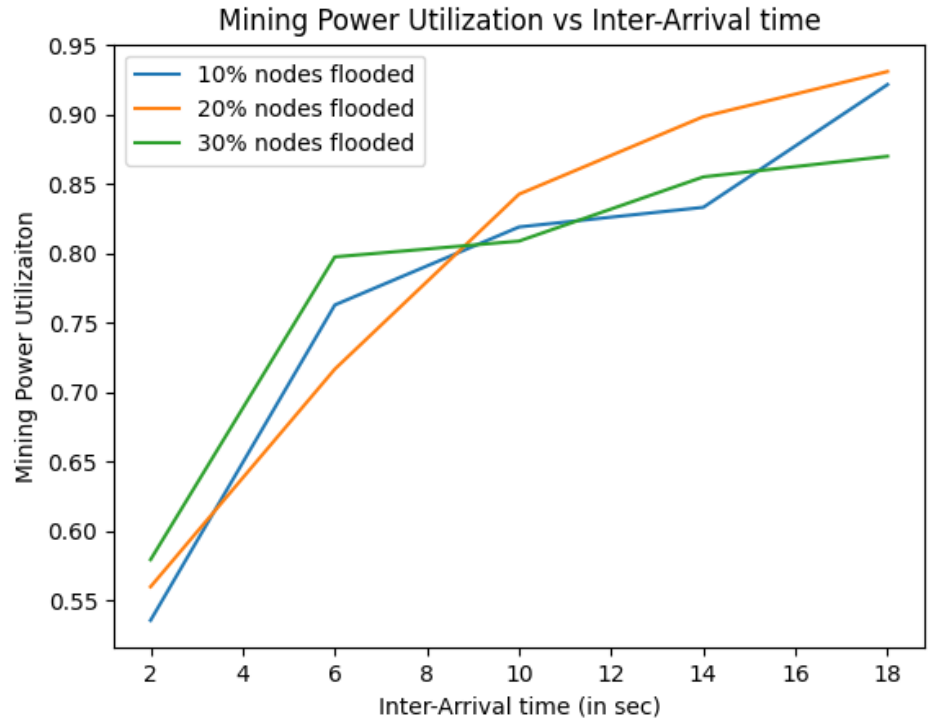


Figure 1: Mining Power Utilization

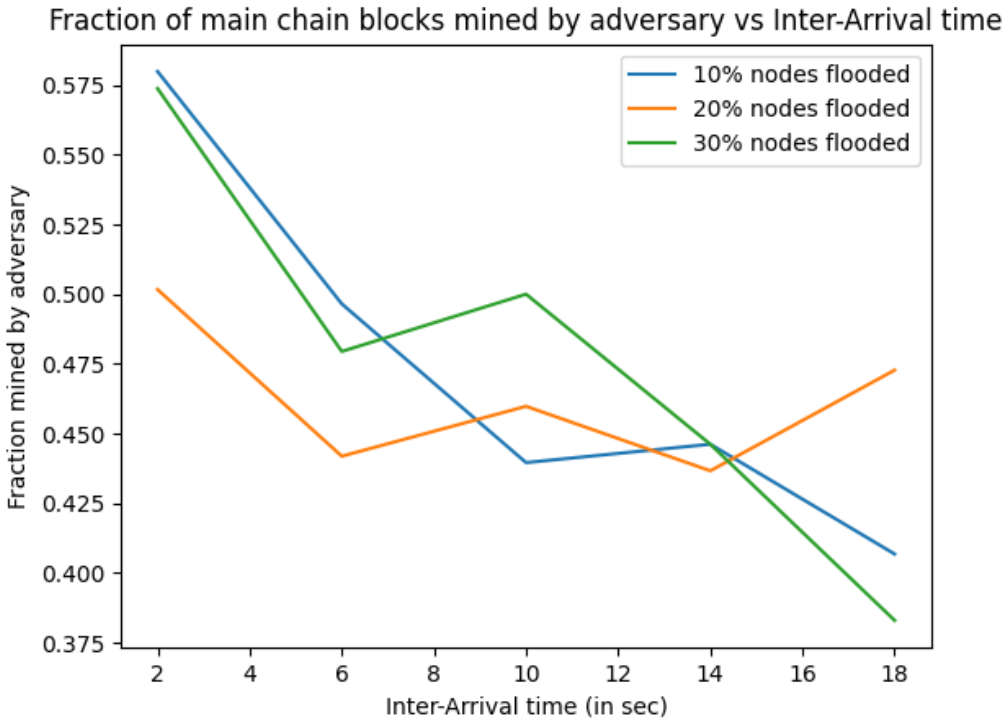


Figure 2: Fraction of main chain blocks mined by adversary

We can observe that the mining power utilization increases with increasing inter-arrival time, which is expected because larger the inter-arrival time, less will be the number of forks in the chain and thus mining power utilization will be larger. If the number of nodes flooded in the network will be more, the number of valid blocks generated will be less but so will the number of forks, thus there shouldn't exist a visible correlation between percentage of nodes flooded and mining power utilization, which is evident from the plot.

Some decreasing trend is observed in the plot of Fraction of main chain blocks mined by adversary vs Inter-Arrival times. This can possibly be explained by the fact that when inter-arrival time is less, the forks caused can cause less number of blocks by the honest nodes to go in the main chain, and since their hashing power is already low, it might be a double disadvantage for them. However, this is just a speculation and the shape of this plot might also be due to the randomness in the network.

## 3 Sample Blockchain Diagrams

We output the local blockchain diagram as 2 files, a spiral blockchain diagram in a `png` file, and a hierarchical blockchain diagram as a `dot` file which can be converted to a png as described in the README. Apart from that, we also write the blockchain to a `txt` file which is easily readable.

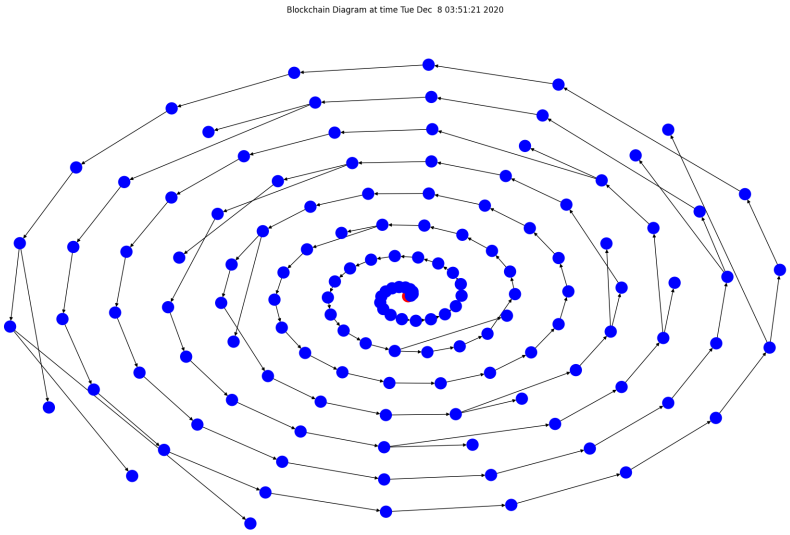


Figure 3: Spiral Blockchain Diagram

Figure 4: Hierarchical Blockchain Diagram