CS 765: Project Part 2

Team: Ansh Khurana (170050035), Kushagra Juneja(170050041), Onkar Deshpande(170050009)

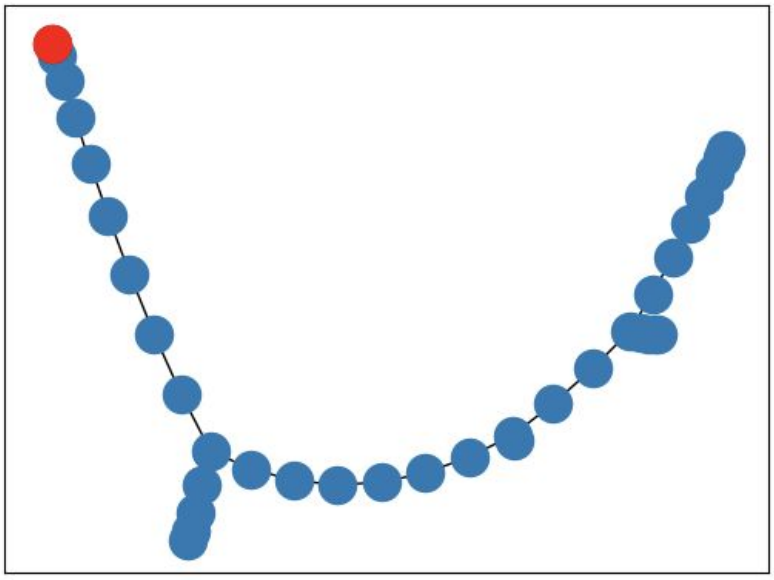
Parameters used for plotting:
Network Delay = 0.5 s
Inter arrival time = [1,2,4,6,8,10]

Algorithm for determining the longest chain to mine on:
When each node receives a new block, after it passes the verification checks, we store the level of this block in the tree as well.
The level of a received block = level of the block that matches with the previous hash (must be in the tree) + 1.
We choose the block with the highest level (which represents the longest chain) and mine on that. Ties (forks) are broken by considering the block which was received by this peer earlier similar to the assumption of Satoshi.

Other implementation details:

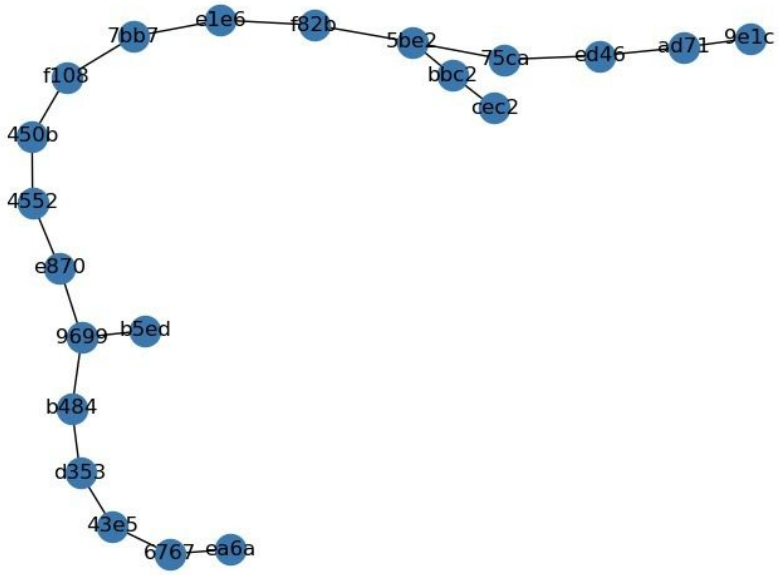
1. We iterate through the longest chain by using prev_hash as back pointers and terminate at the root node with prev_hash=genesis hash. This gives us the length of the longest chain.
2. We also mark blocks which are generated by the peer on its own or received from other peers in the network. This label helps us calculate the fraction of blocks by the adversary in the longest chain.

Drawing the blockchain tree:



The drawing has been made by exporting the tree as a PNG file using networkx library. The red node denotes the first node in the blockchain which has prev_hash=genesis hash.

The forks can be seen as small branches coming out of the longest chain.

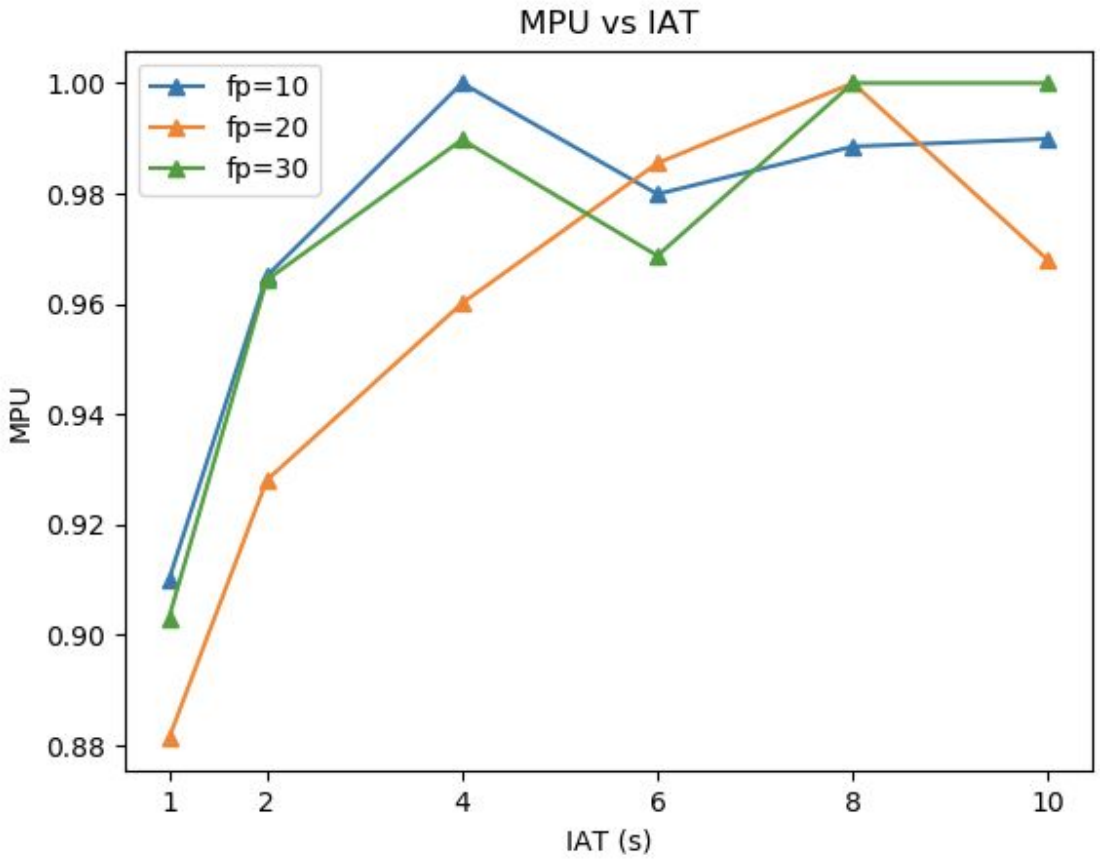


Earlier we had hash of the blocks also printed on the chain. However, this made the plots very cluttered and thus we removed the hashes from the final submission.

10 minutes of runtime with 3 peer nodes :-

Attacker Hashing power = 33%
Victim Hashing power = fp (flood percentage)
Honest miner hashing power = 100-33-fp

MPU vs IAT:

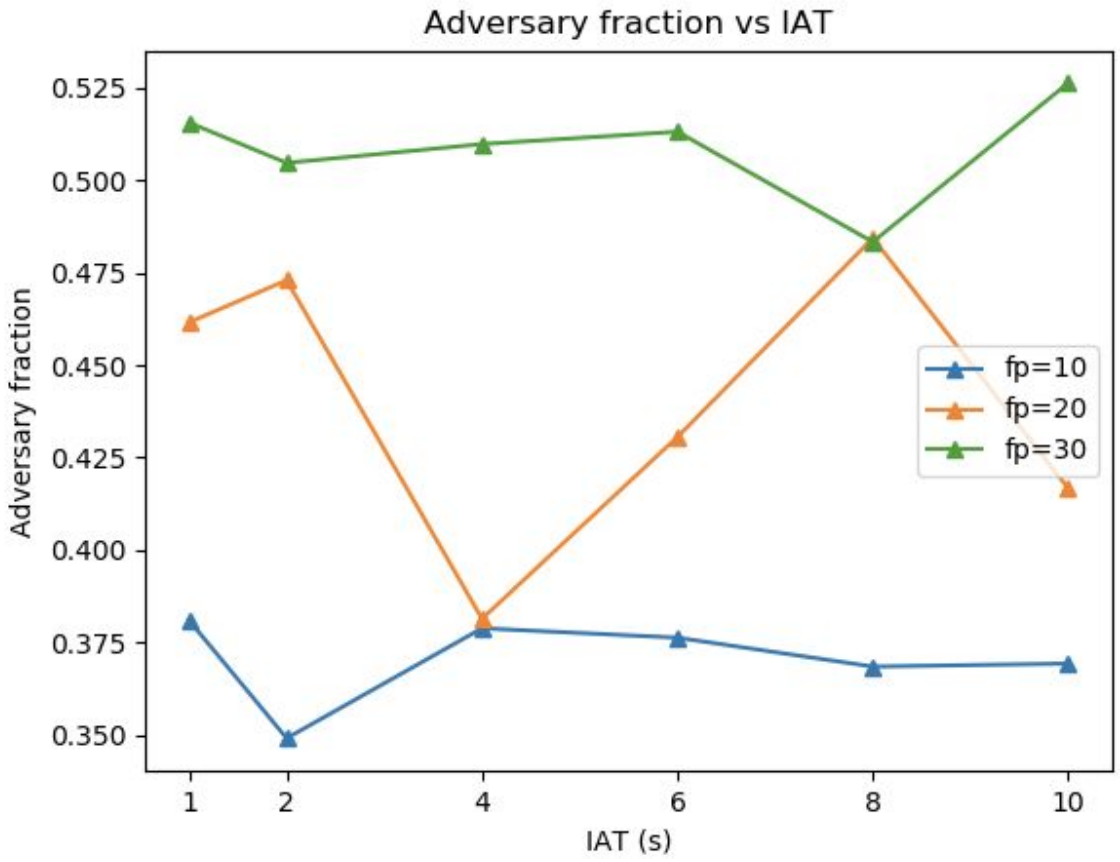


Observations :-

Network delay (ND) is fixed at 0.5 seconds throughout the experiments. As we can see, as the interarrival time increases, the Mining Power Utilization increases. This is because as IAT increases, ratio of ND/IAT decreases. This reduces the probability of fork because fork is

caused only when two nodes generate a block within a ND period of time which is less likely if ND is small in comparison to IAT. As the probability of fork decreases, the MPU increases. Also, there is no visible variation of the MPU with changing flooding percentage at the same IAT value.

Adversary Fraction vs IAT:



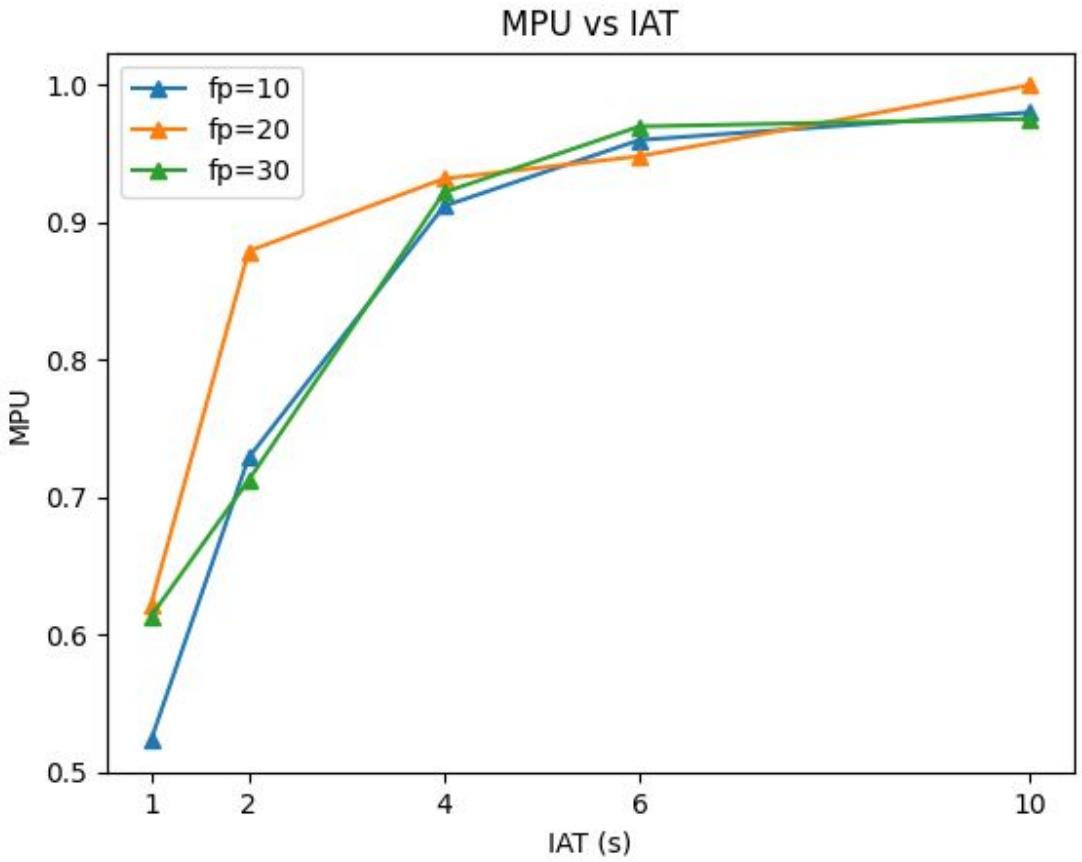
Observations :-

Here, as expected, given a fixed value of IAT, the fraction of block mined by the adversary increases with increasing flooding percentage. This is because as the flooding percentage increases, more percentage of hashing power is being overwhelmed by the adversary and is rendered useless. Hence, the relative percentage of adversary hash power to the active network's total hash power increases hence the adversary fraction increases. At a fixed flooding percentage, there is no visible trend observed in adversary fraction on varying IAT.

20 minutes of runtime with 10 peer nodes :-

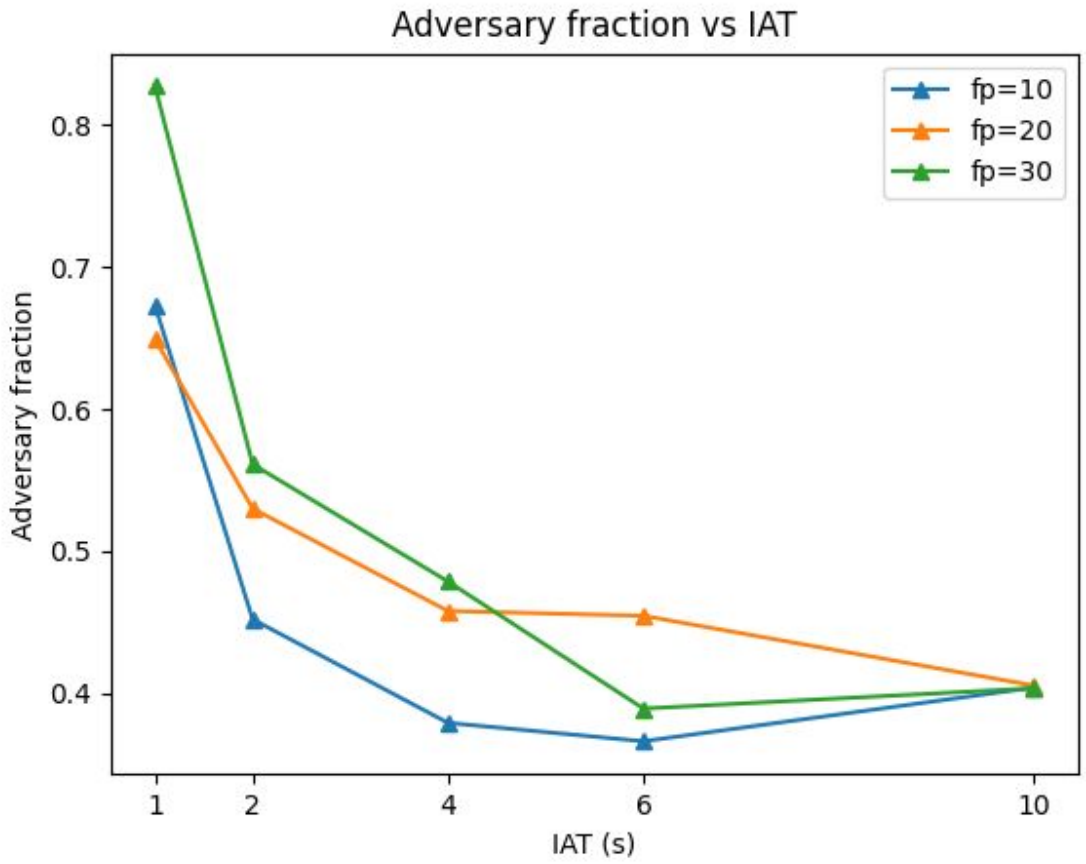
Attacker Hashing power = 33%
Victim total Hashing power is fp (flood percentage) distributed equally between fp/10 victim nodes.
Honest miner total hashing power = 100-33-fp distributed equally between 10-fp/10-1 honest miner nodes.

MPU vs IAT:



Observations :-
The previous justification of the trend of MPU holds here as well. In addition to that, we would like to comment that here MPU value at IAT=1 is small i.e. 0.5 as compared to 0.9 in the previous case (with 3 nodes). This is because now we have increased the number of nodes in the network from 3 to 10 which increases the probability of forks happening and hence decreases MPU considerably.

Adversary Fraction vs IAT:



Observations :-

Here, as expected, given a fixed value of IAT, the fraction of block mined by the adversary increases with increasing flooding percentage. This trend is not very clear in this graph because of randomness but can be seen in the initial values of IAT.

At a fixed flooding percentage, the adversary fraction decreases and then converges to some value on increasing IAT. This is because at small values of IAT the network delay is of the order of IAT (note here that since there are 10 nodes, the network delay between two peers can be more than 0.5 because the shortest path connecting the nodes can be in general longer). Now, even honest miners can only share their mined block after network delay time but they start mining on their generated block as soon as the block is generated. Now, since ND~IAT, in this case there is a significant time difference between when different miners receive a block and hence start mining on that block. The miner who generated the previous block hence has an advantage in generating the next block as well since it receives the previous block the earliest. Hence, this leads to an inherent "selfish mining" type effect at small IAT values in the blockchain where a miner with higher hashing power than its peers will generate a much higher number of blocks than its peers. As we can see here, the malicious node has 33% hashing power which is much higher than any other individual node which is at max 10% hashing power. This effect will exist even without

malicious flooding. This effect will not be present when IAT>>ND hence we see that at higher values of IAT, the adversary fraction is almost same as 33/(100-fp) as expected. Hence, the adversary fraction decreases on increasing IAT due to reduction of this selfish mining effect.