Simulating a Selfish Mining attack using the P2P Cryptocurrency Network developed in the last Assignment

CS 765 Assignment 2

Neel Aryan Gupta	Saurav Garg	Tathagat Verma
180050067	180050093	180050111

Visualization and analysis

We provide the command to run each of them. We follow the key given below in experiments:

```
-n --peers                      Number of peers in the network
-e --edges                      Number of edges in the peer network
-z --slowpeers                  Fraction of slow peers in the network
-t --time_limit                 Run the simulation upto a time limit (in seconds)
-Ttx --txn_interarrival         Mean of exp distribution of interarrival time b/w txns
-Tk --mining_time               Mean of exponential distribution of time to mine a block
-s --seed                       Seed for random number generator
-txn --max_txns                 Run simulation till max transactions are generated
-blk --max_blocks               Run simulation till max blocks are generated
-it --invalid_txn_prob          Probability of generating an invalid transaction
-ib --invalid_block_prob        Probability of generating an invalid block
-zeta --attacker_connection     Fraction of honest nodes adversary is connected to
-a --alpha                      Fraction of hash power belonging to attacker
-adv --adversary_type           Type of adversary, choose from (none, selfish, stubborn)
```

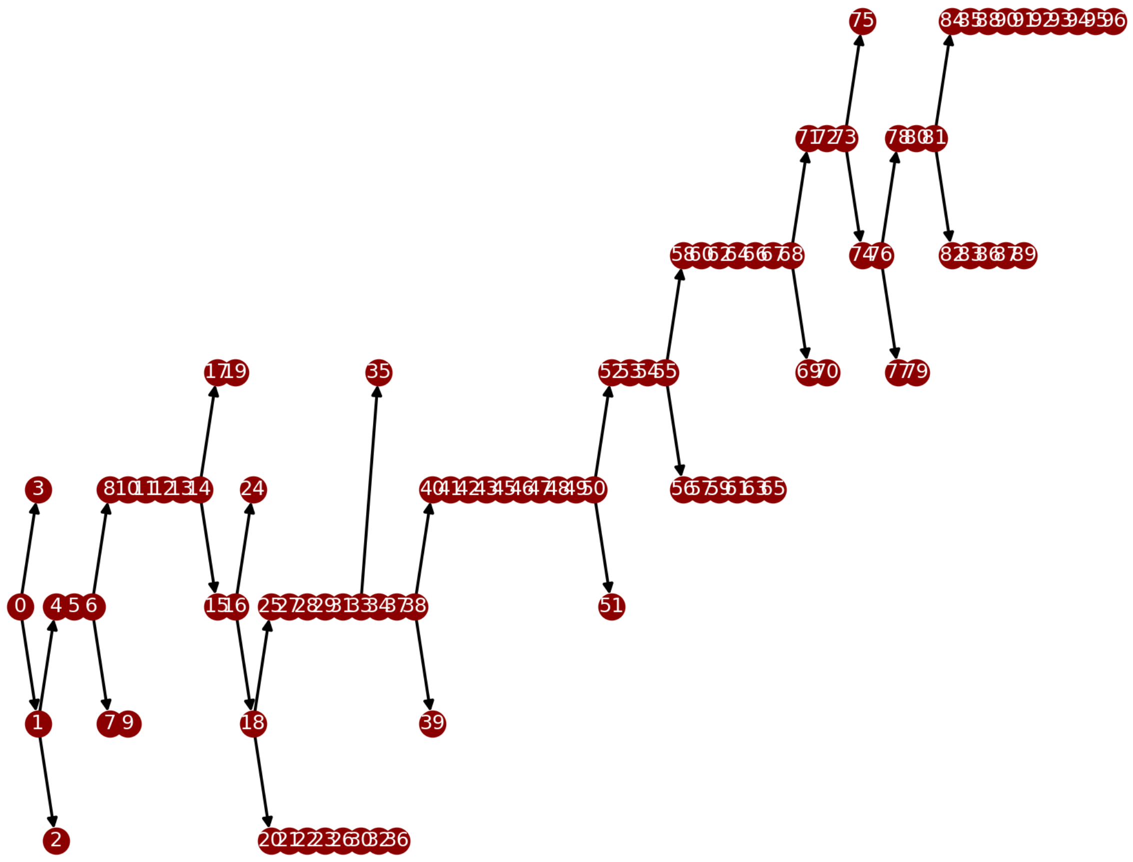
There are three kinds of hash power, either LOW, HIGH and the adversarial hash power. The HIGH hash power is set to be 5 times the LOW hash power, while the hash power of the adversary is the fraction alpha of hash powers of all the peers combined. Therefore, besides the attacker, there are 4 categories of peers. These are based on the hash power (either Low or High) and the network speed (either Fast or Slow) of a peer.

These are the definitions we use henceforth in our report.

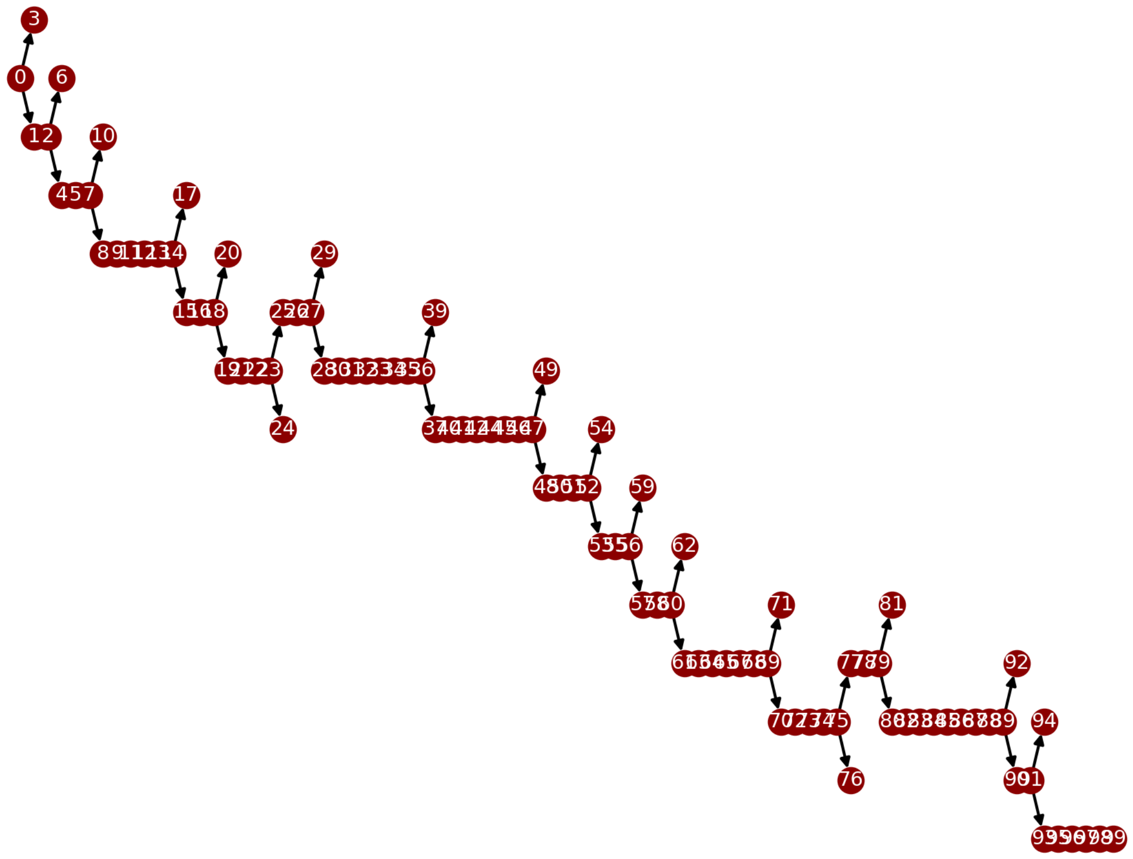
```
MPU_adv     = (# Adversary blocks in main chain) / (# Total blocks mined by adversary)
MPU_overall = (# Blocks in main chain) / (# Total blocks across all peers)
R_pool      = (# Adversary blocks in main chain) / (# Total blocks in main chain)
```

Comparisons of Two attacks (Along with Blockchain Tree)

Stubborn Mining Attack



Selfish Mining Attack



The idea behind Selfish Mining Attack is for the selfish miner (Adversary) to keep her discovered blocks private, thereby intentionally forking the chain. The honest nodes continue to mine on the public chain, while the adversary mines on its own private branch. If the adversary mines more blocks, she develops a longer lead over the public chain and continues to keep these new blocks private. When the public branch approaches the adversary's private branch in length, she reveals blocks from its private chain to the public.

Contrary to Selfish Mining, Stubborn Mining Attack involves maintaining the competition with the honest chain. Instead of revealing her entire private chain, the adversary reveals the next block on her private chain only to match the length of the public chain, thereby keeping the competition for the longest chain instead of completely killing off the honest chain.

As observed in the above blockchain trees, in case of stubborn mining attack, most of the forks are composed of more than one block due to the fact that a stubborn miner allows the honest chain to grow as well, as discussed above. On the contrary, most of the forks in selfish mining attack are composed of a single block, which happens because the selfish miner kills off the honest chain everytime it catches up to the private chain, resulting in a small fork.

In general, as per the graphs given in following experiments, we observe that selfish mining attack is preferred for low mining power values while stubborn mining is more beneficial for mining power of adversary close to 0.5. This might be due to the fact that stubborn is never killing the chain, and in case of less hashing power, it is possible that it wastes its work done on mining if honest chain wins. For large hashing power, stubborn miner is successful in wasting the effort of honest miners, thereby giving higher return than selfish mining attack.

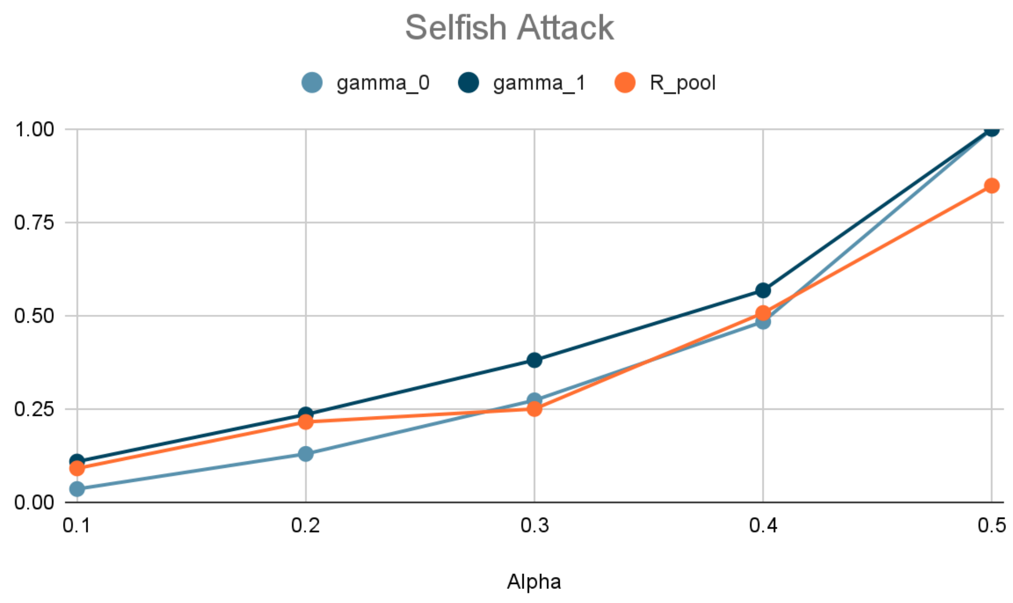
The comparison between the two attacks for different zeta and alpha values is discussed below.

Variation with Alpha along with theoretical limits of Eyal and Sirer

Selfish Mining Attack					
Alpha	0.1	0.2	0.3	0.4	0.5
Zeta	0.5				
MPU_adv	0.66667	0.70833	0.80769	0.87805	0.97500
MPU_overall	0.91667	0.84946	0.85714	0.73196	0.56790
gamma_0	0.03564	0.12967	0.27313	0.48372	1.00000
gamma_1	0.10919	0.23516	0.38062	0.56744	1.00000
R_pool	0.09091	0.21519	0.25000	0.50704	0.84783

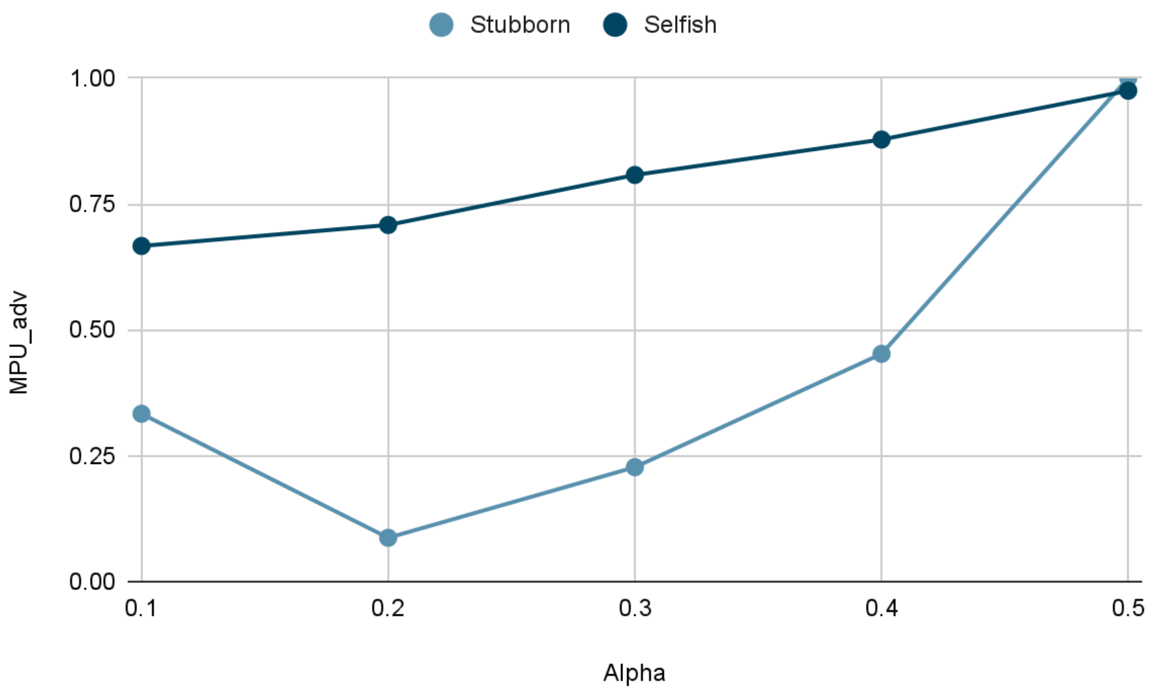
Gamma_0 and Gamma_1 are the theoretical limits as per Eyal and Sirer paper.

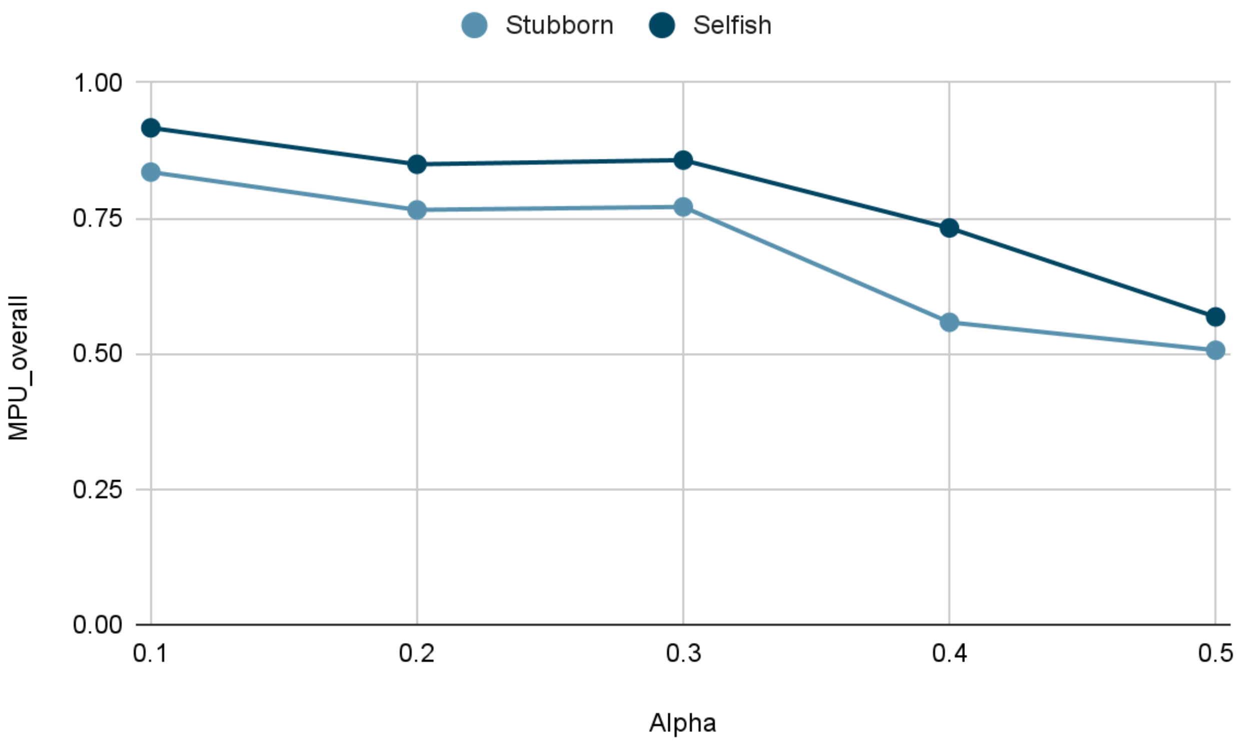
Stubborn Mining Attack					
Alpha	0.1	0.2	0.3	0.4	0.5
Zeta	0.5				
MPU_adv	0.33333	0.08696	0.22727	0.45238	1.00000
MPU_overall	0.83505	0.76531	0.77083	0.55789	0.50633
R_pool	0.06173	0.02667	0.06757	0.35849	0.97500



The experimental graph R_pool values are sandwiched between the gamma=0 and gamma=1 values. Here gamma_0 and gamma_1 are the theoretical estimates given in the Eyal and Sirer paper. In the experimental scenario, the gamma value is between 0 and 1, hence it's expected that the R_pool values will lie in between the gamma_0 and gamma_1 band.

In the next section, we study the effect of alpha (adversary hash power) on both attacks. Alpha is varied over 0.1, 0.2, 0.3, 0.4 and 0.5.





In the selfish mining attack, we make the following observations; MPU_adv and MPU_overall trend are of opposite trends. MPU_adv increases with an increase in adversary mining power alpha. This is expected since as the mining power of the attacker increases, it has more and more fraction of blocks entering the main chain as it can kill the honest blocks easily. The MPU_overall will consequently decrease since more blocks generated by honest miners get wasted this decreases the no of honest blocks in the main chain.

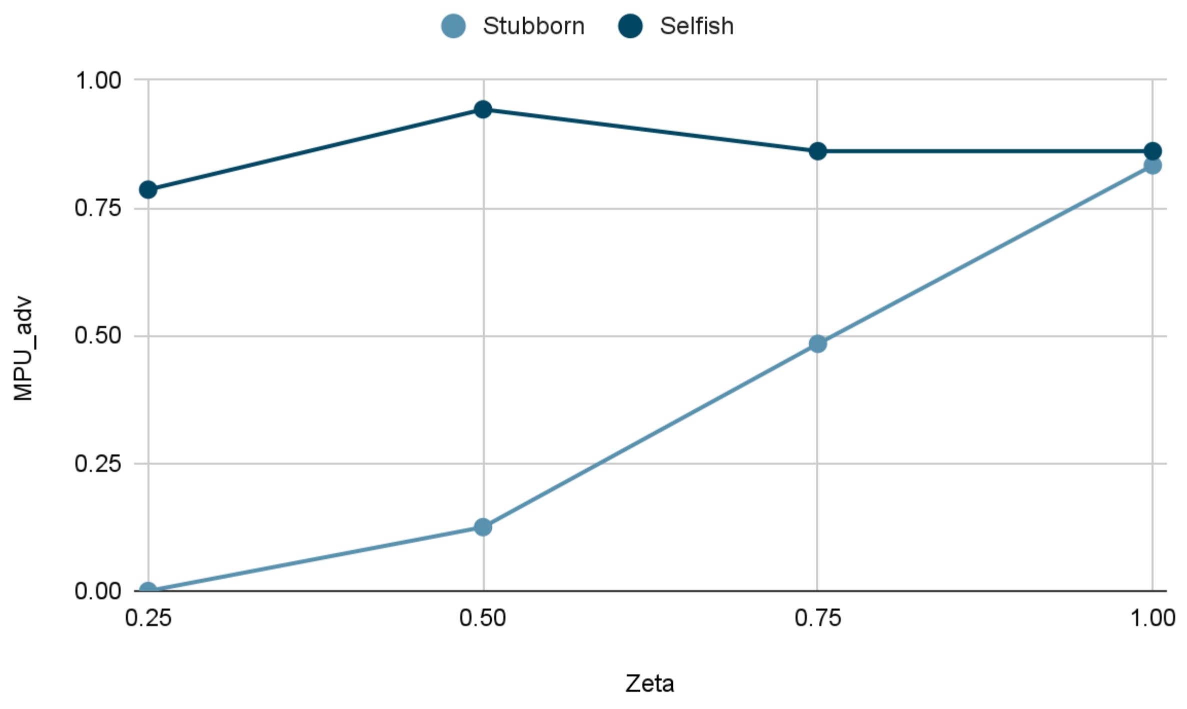
For the stubborn mining attack, initially the ratio decreases with increasing hashing power since there are more chances of stubborn blocks being rejected as it is not killing the honest chain directly. So, when hashing power increases, the adversary mines more blocks but most of them are rejected (as is confirmed by the R_pool ratio) which means an even less fraction of blocks in main chain. After alpha=0.2, the mining power helps the adversary and we observe MPU_adv increases with the alpha. For MPU_overall, the trend is similar to the selfish mining attack. The overall ratio decreases as the honest miners' hashing power fraction decreases. So, the adversary can get more of the honest blocks to be wasted.

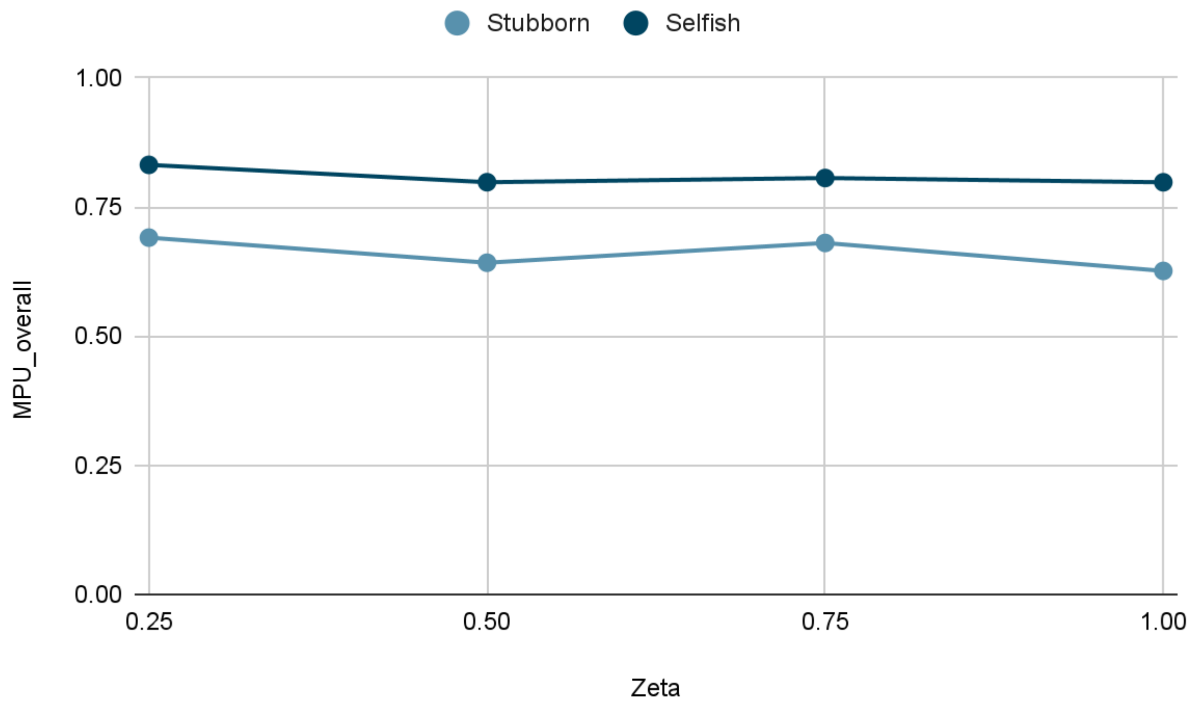
Variation with Zeta

Selfish Mining Attack				
Zeta	0.25	0.50	0.75	1.0
Alpha	0.35			
MPU_adv	0.78571	0.94286	0.86111	0.86111
MPU_overall	0.83158	0.79798	0.80612	0.79787
R_pool	0.27848	0.41772	0.39241	0.41333
gamma_0	0.36651			
gamma_1	0.46556			

Stubborn Mining Attack				
Zeta	0.25	0.50	0.75	1.0
Alpha	0.35			

MPU_adv	0.00000	0.12500	0.48387	0.83333
MPU_overall	0.69072	0.64211	0.68041	0.62626
R_pool	0.00000	0.06557	0.22727	0.48387





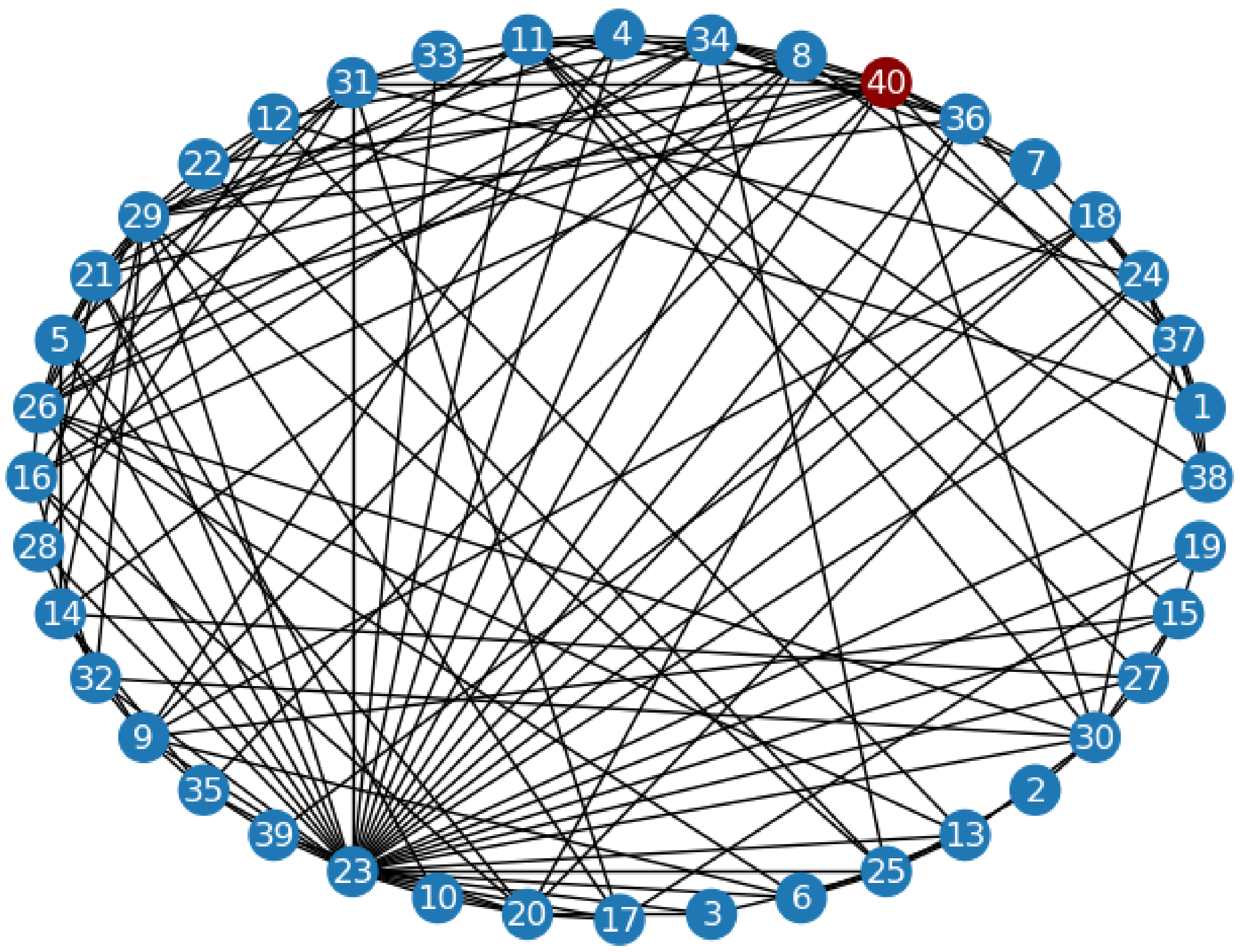
We run experiments to study the effect of zeta (fraction of honest nodes to which the adversary is connected) on both attacks in this section. The adversary’s hash power is set to 35% of the total hash power. We vary zeta over values 0.25, 0.5, 0.75 and 1.

The parameter zeta plays an important role in stubborn mining attack as is clear from the MPU_adv graph. This is due to the fact that stubborn mining is highly dependent on winning the competition at every block. So, more the direct connections of adversary, higher are the chances of adversary's block winning the competition (since more honest blocks will mine on adversary's block as soon as they receive it). The zeta parameter doesn't affect the selfish mining attack by that extent but the slight maxima at 0.5 is explained below.

In the selfish mining attack, we observe from the table and plots that the MPU_adv value has a slight maxima at zeta 0.5. This is justified as follows - on increasing the number of connections with honest peers, the adversary keeps getting faster updates of any new blocks that have been created. Now, whenever the adversary loses the race to create a block earlier, it will start mining on the new honest block. When zeta is low, then even when an honest block is created, the adversary will take more time to get the block and will keep mining on the previously existing block. Due to this, for lower zeta values, the adversary will make less switches onto the honest blockchain and hence will generate a lower number of blocks overall (denominator of MPU_adv) and will have higher MPU_adv values. On increasing zeta, the benefit that an attacker gets is that its blocks will be received faster by the honest peers and will hence start mining on the attacker’s blocks with a higher probability, this favors the attacker in getting a higher fraction of its blocks into the blockchain. This results in a trade-off between the effects of low and high zeta values. Hence, the optimal zeta should be achieved somewhere in the middle, which is exactly observed in the maxima at zeta 0.5.

The MPU_overall in both the attacks remains more or less constant and is not affected much by varying the zeta parameter. Although we see a slight decrease in MPU_overall ratio which is expected since MPU_adv is increasing and as MPU_adv increases, honest miners' blocks will struggle to get into the chain, thus decreasing the MPU_overall ratio.

Sample Peer Network



The blue nodes correspond to honest miners, whereas the red node is the adversary. We have adopted the convention that the adversary will always be the last peer (numerically the highest).