CS620: Assignment2

Clients topology: We have ran the experiments for 5 clients.

Mining Power Utilization: Each client sends connection requests to 2 more clients already in Peer 2 peer system. From previous assignment, we have reduced the connection requests from 4 to 2 to have more forking. We are also giving network delay a uniform value between 0.2 and 0.5 seconds by usig sleep command. Otherwise while running for 5 clients, all clients are one hop away from each other

Selfish Mining: we have reverted back to the previous.

Hashing Power distribution: Uniform distribution for mining power utilization. We used uniform distribution to have more variance in mining power utilization.

And after giving adversary a hashing power, we distribute the remaining uniformly to other clients.

Seed's involvement:

Seed does the work of sending ip addresses of the currently connected clients as in the gossip protocol and also sends "start mining" messages to all the connected clients and then quits after sometime.

Interarrival time variations:

We are plotting mining power utilization for 3 seconds, 8 seconds, 10 seconds, 15 seconds. We can observe that mining power utilization increases with increasing inter arrival time as most of the peers are expected to mine on the longest chain.

Number of blocks(fixed in our case):

Mining power utilisation : we generated 200 blocks on each client for all interarrival times except

20 seconds for which we have generated 120 blocks(to save time).

Selfish mining attack: We generated 75 blocks on each client with 8 second interarrival time

Block structure:

Block format:

(hash_of_prev_block,merkle_root,timestamp)

While sending the block to peers we sent the following information along:

1.hash of the current new block:

To save computation on peer side

2.length of the chain (including this block)

To help peer to decide the longest chain

3.whether generated by honest/adversary client:

To calculate fraction of adversary blocks as a part of total blocks in the longest chain

Longest chain determination:

We have maintained the value of the current longest chain on every peer in a variable 'current_longest_chain'.

As mentioned in the block structure,whenever a peer receives a block it compares his current longest chain variable with the value of length of the chain present in received block.If it is larger than the current_longest_chain we update the current_longest_chain and start mining on the received block(by resetting the timer).

10)Data processing method:

We are maintaining all the validated blocks at run time in memory using a python dictionary data structure.When a peer closes the connection we dump the dictionary in binary format to a file using pickle library.This data base is stored on every client.

Mining power utilisation:

We process all the blocks and tag those which are in longest chain.

Now,we take a random sample of 100 blocks(we should have taken less) from 200 blocks and calculate the fraction of blocks which are in longest chain and we collect such samples from each peer and plot a histogram(frequency(y) vs mining power utilisation(x)).

Forking demonstration:

Pydot python library is used to make a digraph which consist of nodes inscribing hash of the previous block and is length in the chain.

For demonstration purposes we are submitting the plots with 2 seconds interarrival time and 20 blocks to see the forking .

Confidence interval calculation:

It is a Bernoulli distribution in both cases

a) with a block being a part of longest chain or not

b) with a block in longest chain being an adversary block or not.

variance = p * (1-p)

Std dev = sqrt(variance)

Std deviation in sampling = std dev /10

Where 10 = sqrt(100)

90 % CI factor z = 1.645

CI_low = mean - 1.645 * stddev sampling

CI_high = mean + 1.645 * std dev sampling

As we were getting all our values within confidence interval, we have no shown CIs in the diagram.