Project Report
on
Algorand Simulator

CS 620: New Trends in Information Technology

Under the guidance
Of

Prof. Umesh Bellur and Prof. Vinay Ribeiro

Presented By : Vaishali Jhalani (173050033)
Kantikumar Lahoti (173050011)

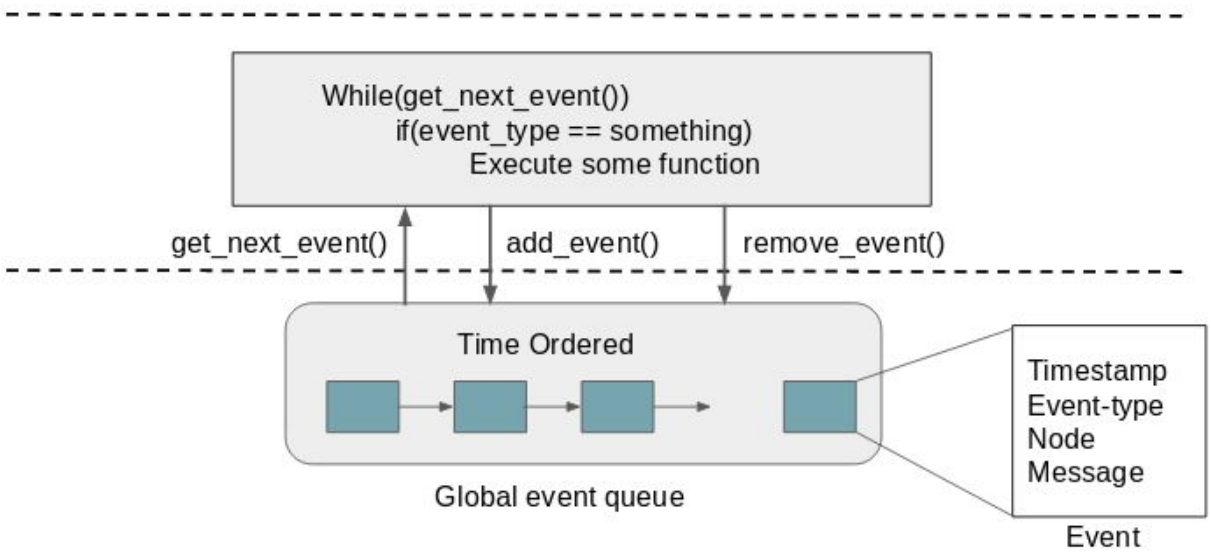
Introduction:

Algorand is a cryptocurrency which is scalable and adds transactions to the blockchain with latency in minutes. Algorand uses Byzantine agreement protocol to reach consensus. We have built a discrete event simulator for Algorand protocol. Algorand consist of multiple steps:

1. Committee formation
2. Block proposal
3. Reduction
4. Binary BA*

Simulator Design:

Our simulator consists of a global queue which has multiple events, each with different event type and timestamp. As soon as the simulation starts, events are dequeued from the queue and based on the event type, some action is taking place.



There are many parameter values that we have considered during simulation:
t_proposer = 5

t_step = 32

t_final = 32

MAX_STEPS = 10

DIFFERENT ENTITIES OF EVENT SIMULATOR:

1. EVENT_SIMULATOR
2. NETWORK
3. NODE
4. MESSAGE
5. BLOCKCHAIN

In the main loop, we have many different cases based on the event type of the new dequeued event. Here we are listing all the event types that we have used:

1. PRIORITY_MESSAGE
2. RECEIVE_PRIORITY_MESSAGE
3. CREATE_BLOCK_PROPOSAL
4. RECEIVE_BLOCK_PROPOSAL
5. CAST_VOTE
6. RECEIVE_VOTE_MESSAGE
7. COUNT_VOTES
8. BINARY_BA*
9. CAST_VOTE_BA*
10. COUNT_VOTE_BA*
11. COMMON_COIN
12. BA*_LOOP
13. FINAL_COUNT_VOTES
14. ADD_BLOCK

We will discuss this one by one:

1. **PRIORITY_MESSAGE**

As soon as the simulation starts, we add this event to every node in the network. When this event dequeued from the queue a function is called **create_priority_message()**. This function does the sortition for the first time and for all the nodes who are committee members, generate the priority for that member.

2. **RECEIVE_PRIORITY_MESSAGE**

All the committee members will receive the priority message broadcasted by other members. Function **receive_prio_msg()** is used for that. This gossip will be done for 3 seconds.

3. **CREATE_BLOCK_PROPOSAL**

After 3 seconds, this event will be dequeued. **Propose_block()** function will be called which calculate the highest priority node.

This node will generate a block containing payload as a random string and sign the block with its private key. Also at this particular timestamp all the events of RECEIVE_PRIORITY_BLOCK.

4. **RECEIVE_BLOCK_PROPOSAL**

For more 30 seconds, every committee member will wait for the block proposal. When this event dequeues from the queue, **receive_block_proposal()** is called. This function verifies the block and also validate the sortition of the block proposer.

5. **CAST_VOTE**

On successful validation of the block proposer, this event takes place and **committee_vote()** function is called. This function does the sortition with new step number and generate a vote block. Also, all events of RECEIVE_BLOCK_PROPOSAL removed in this step. This vote message is broadcasted to all the neighbours of that node.

6. **RECEIVE_VOTE_MESSAGE**

This event is to broadcast the vote message. A function **receive_vote()** is called and store the incoming vote message in the node buffer after verifying the message.

7. **COUNT_VOTES**

For 3 seconds the above gossip happens and then this event generated. A function **count_votes()** is called. This function checks all the vote messages came from all

other nodes and if for any block the number of votes is higher than the threshold value then that block is returned.

Now if the step value is less than 2 then we are in the reduction procedure and CAST_VOTE event is again added to the queue otherwise BINARY_BA* event is added. As after the second case_vote, we will get the final output of the reduction step and we can proceed to the binary_ba*.

8. **CAST_VOTE_BA*, COUNT_VOTE_BA* COMMON_COIN, BA*_LOOP**. All these events are related to BINARY_BA* but they are using the same functions which we have used in the above events. After the BA*, final or tentative blocks can be found. After the final consensus, ADD_BLOCK event is called which adds the final block to the blockchain.

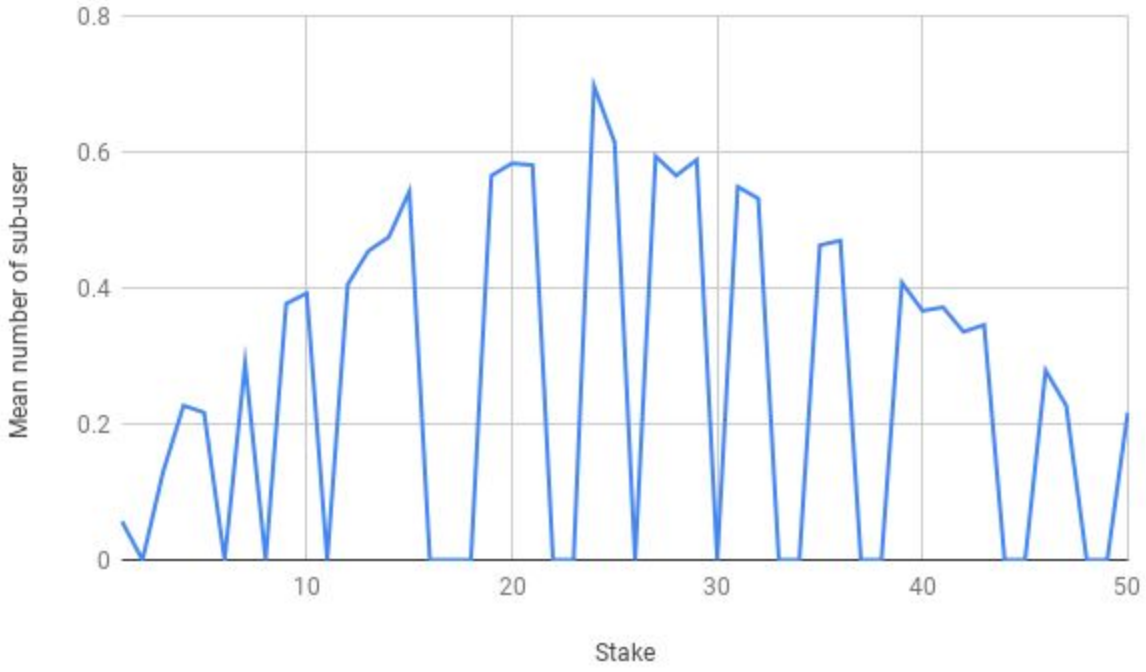
EXPERIMENTS:

1. Stake based Cryptographic Sortition:

   1.a t_step = 32

   X-axis = stake value

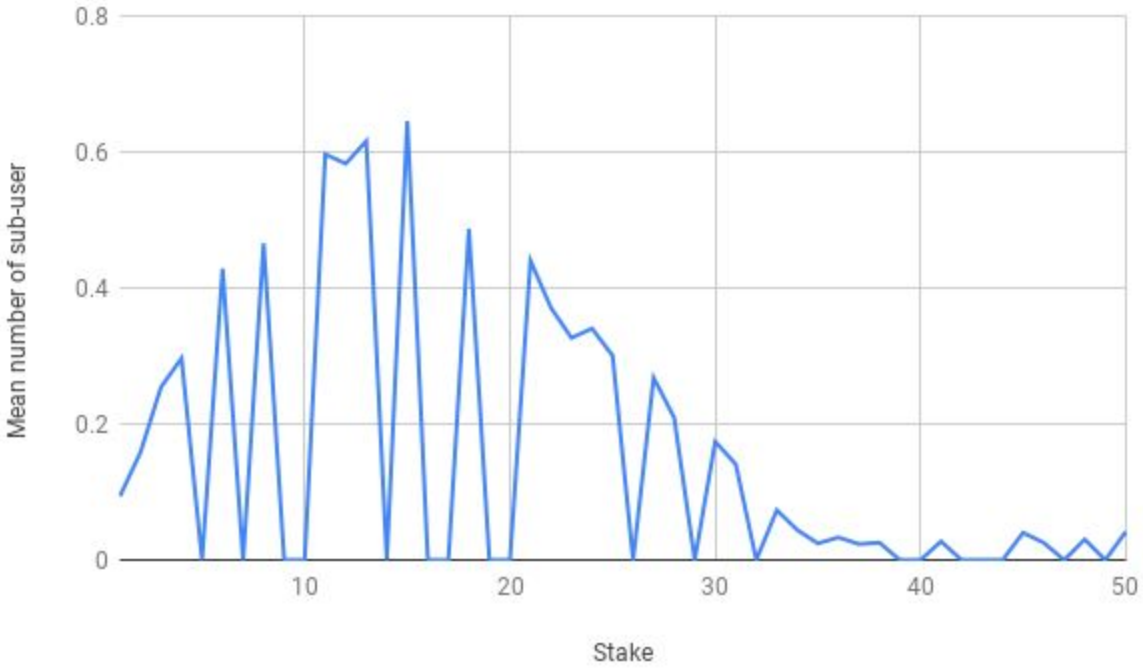
   Y-axis = mean number of sub-user



t_step = 64

X-axis = stake value

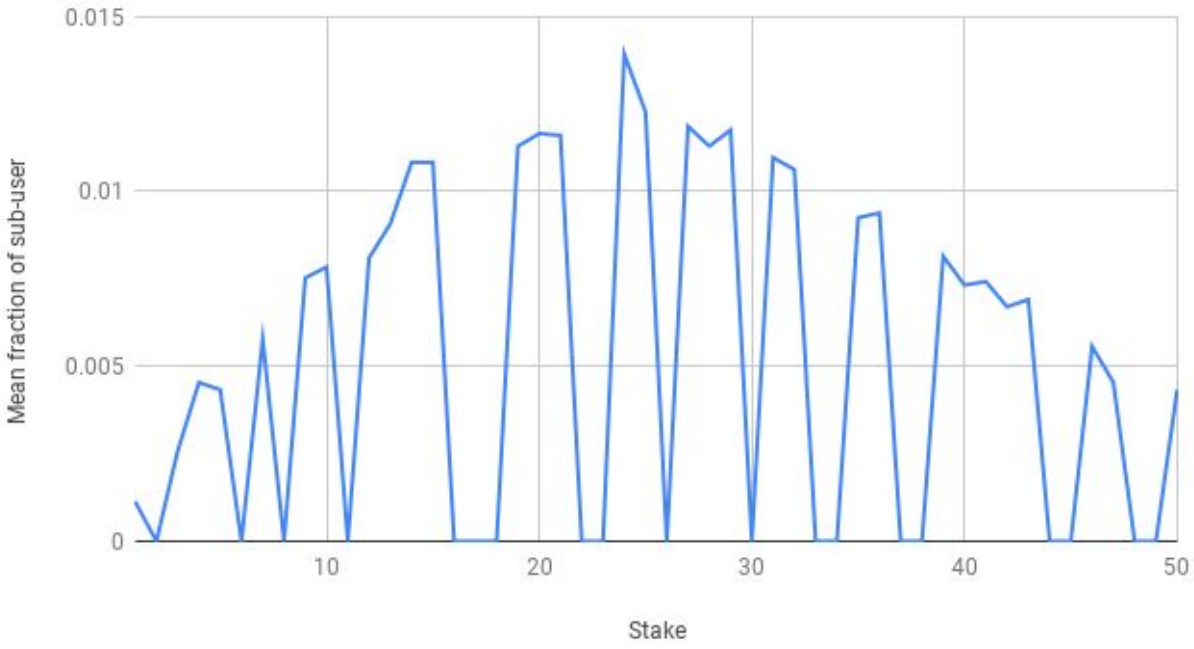
Y-axis = mean number of sub-user



1.b t_step = 32

X-axis = stake value

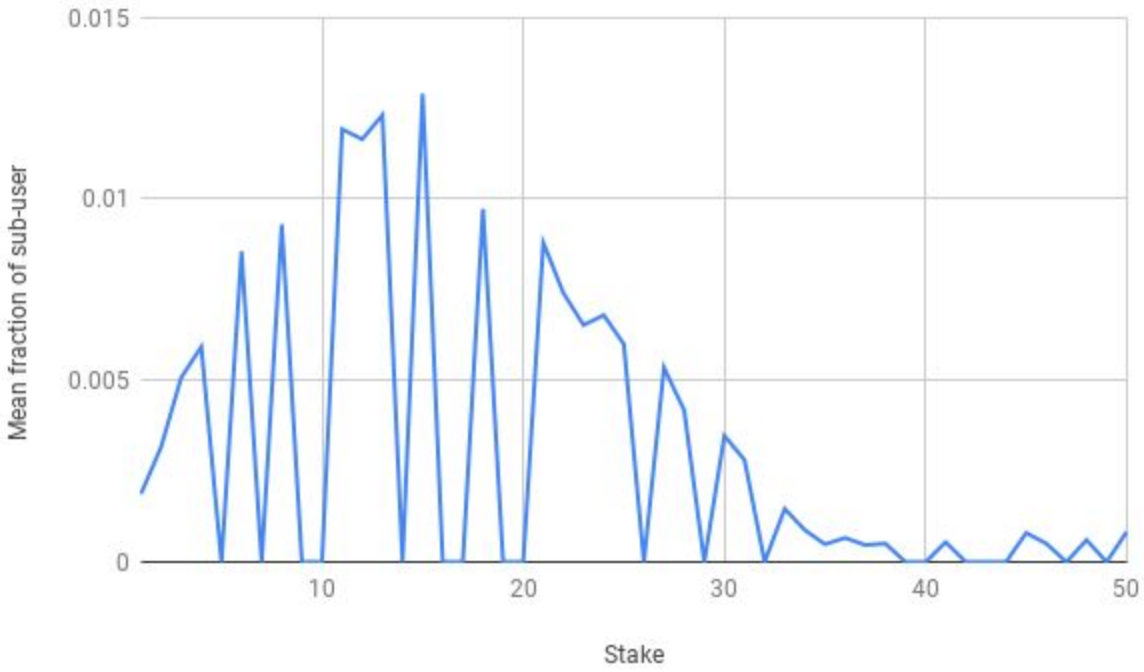
Y-axis = mean fraction of sub-user



t_step = 64

X-axis = stake value

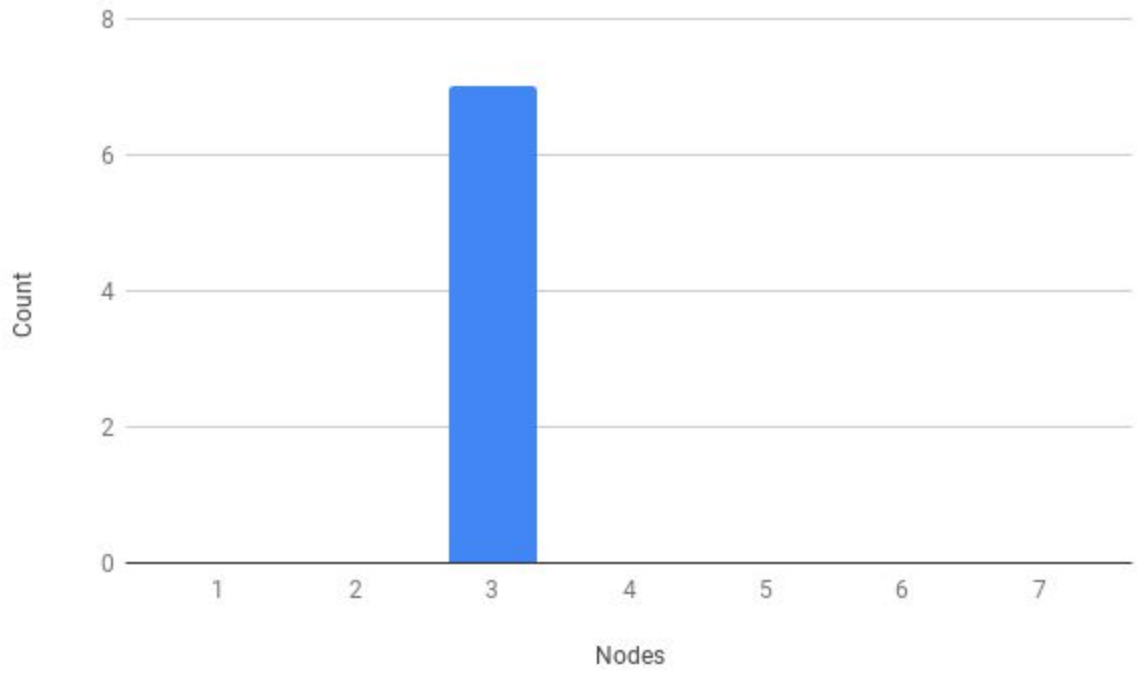
Y-axis = mean fraction of sub-user



2. Highest Proposer and Network Delay
    1. T_proposer = 5

       Non-block delay = (40, 64)ms

       T_priority = 500ms



Public Key:

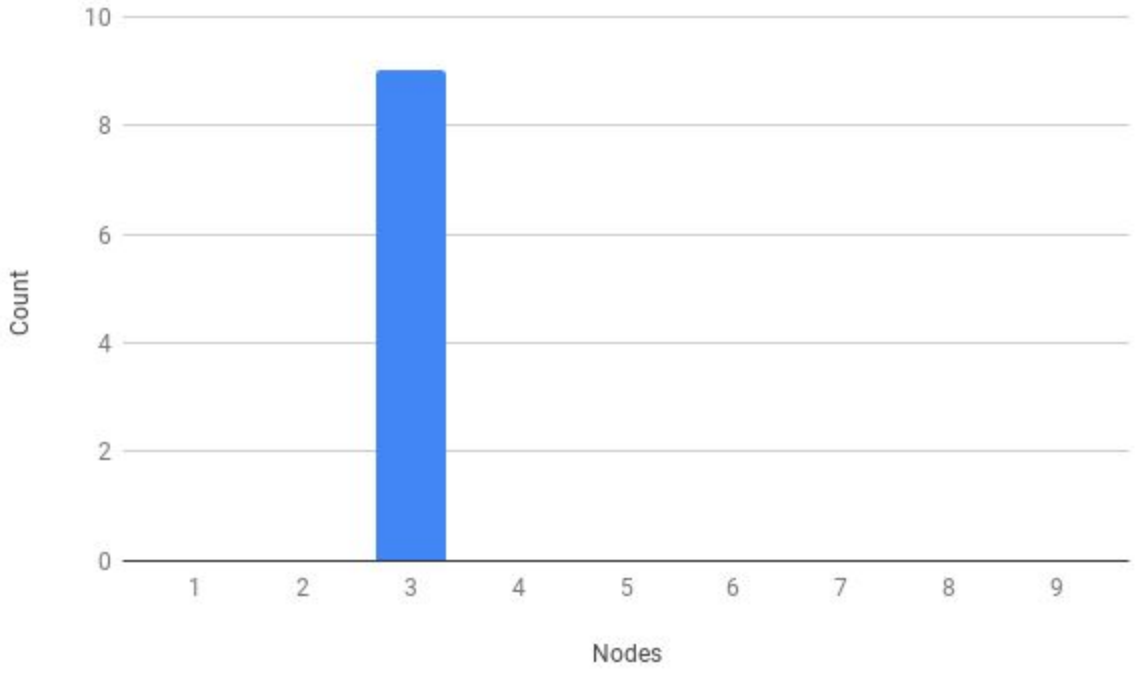
Node 3

‘Y\xb2c\xe1w\xf9@\xc0p"\x96\xca\x00\x01l\xc17/\xefwRs?\xd9\x97H\xea\xd65\x0cf\x80\x0b\xa0\xbb\x90\x0fR\x1f\x18r\xe6\xce\xb6vz\xc7\xe5’

2. T_proposer = 5

   Non-block delay = (60, 64)ms

   T_priority = 500ms



Public key:

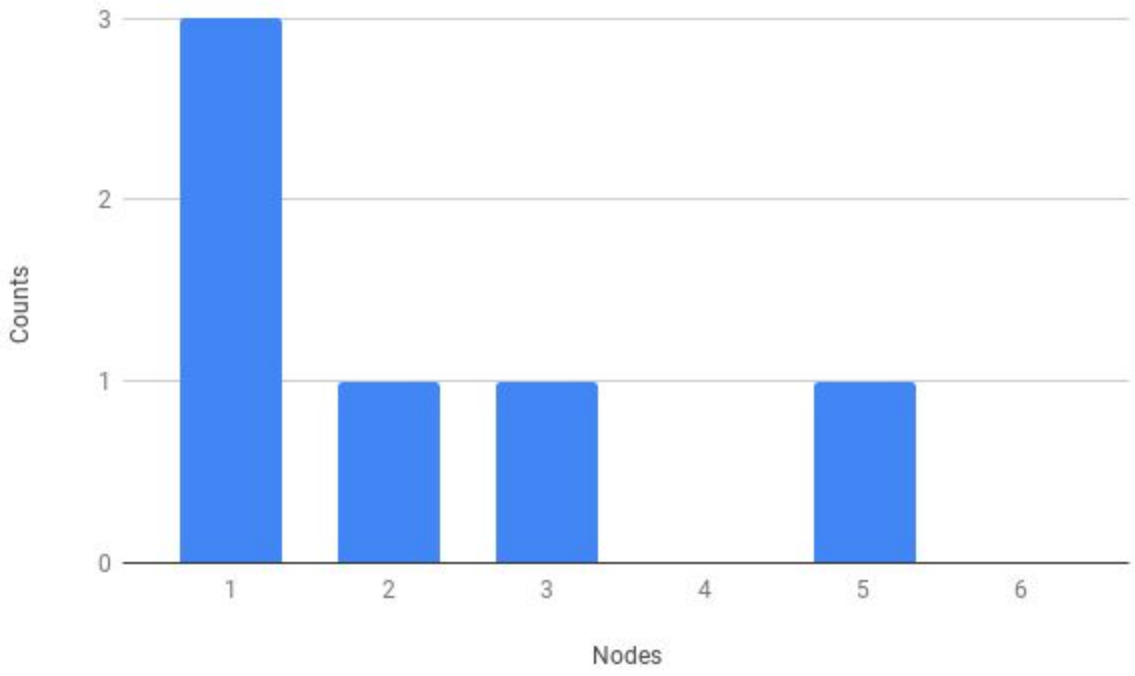
Node 3:

h\x1a\x07H\xeeN\xc7\x1b\xcb)\x1c\t5\xfc\xec\x82\x11\xe2\x8d\xb8>V,\xa58\x02l\xc8\x10\xb9\xdd#\xf1+J\xc7a\x00\xd63\n2.\xad(H\xef\xa3'

3. T_proposer = 5

Non-block delay = (100, 64)ms

T_priority = 500ms



Public Key:

Node 1:

'\xf8\xd3\x87-]\x91\xd0\xe2\x86b\x12\xff\x19\xed\xa7\xaf\xea\xd5\xa6\xadKI\xb6\xd4\xff\t\xd80.\xa6$\x8cA`B.\xbb\xd1\xec\xc2\xf7\xa62\xd0>P\xc7='

Node2:

‘I\xbdO\x15\xd7\x9a]\xee\xd3\xe6#\xfe\xfd\xba\xfb\xed-\x17_$\xe8\x15qf\x8f`?*\x95NZz\x0e["\xe7Vg\xce'\x91\x99@E\x15\xa7\xea\x98’

Node3:

‘\x1d\xef\xdc\xcb\x8ed\xfd\xf8\xbd6\xde|\xaa\xbc\xb3\xc5\x8b\xa9(\xb3\x89\xdch\xa0\xfe5\xc0\x1e'\xa9\xdd\xd4{_\xf6\x1c\xf4\x8fh\xdf!\xb3\xea1\xc1V>\xe3’

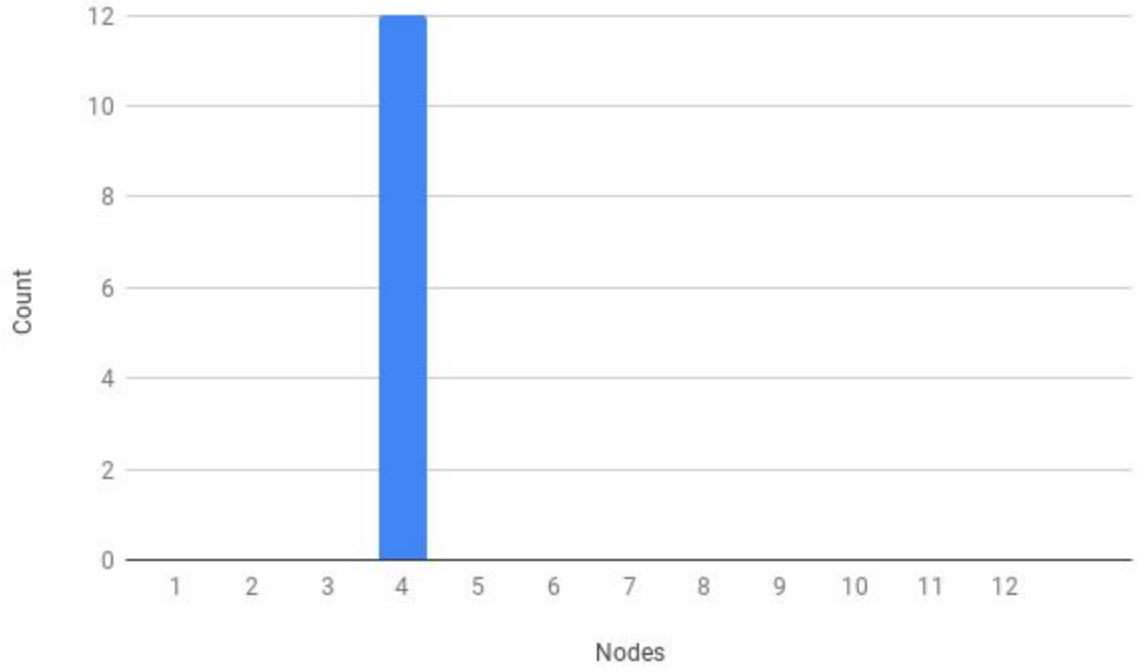
Node5:

'L&v\x8f\x92%\xc0\xb3\x9a\x1b0\xa1l\xc0\xe0\xa2\x7f\x08\xba\xd8\xb3!!\xac.c\xe5\x05\xc4\x19xzX[\x92\xb9-\xa4n\xa92g~\xc7\x1e"a\x1d'

4. T_proposer = 10

   Non-block delay = (40,64)ms

   T_priority = 500ms



Public Key:

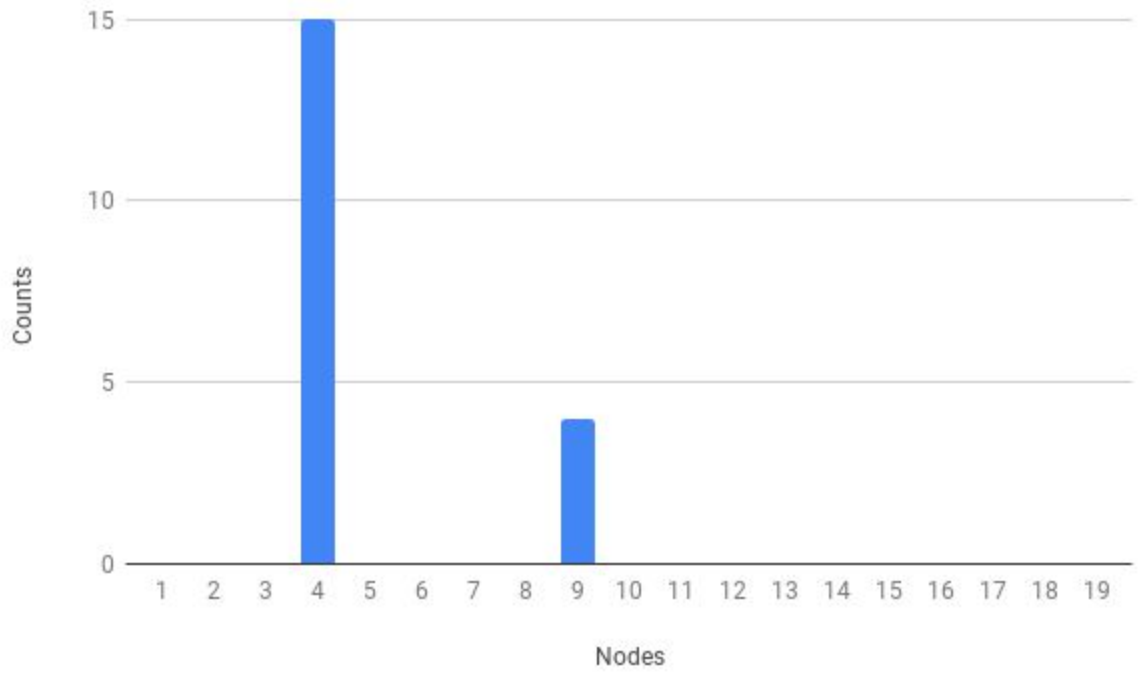
Node 4

'y\x9f\xcf\x0e\xfczwu\xbc\xf8}\x1f\xedl\x95wp4&1*\xf9\x9c\x01\xf4\t\xd0\xf4;\xf4\x91%\x9b\x8el\xc6\xeea\xd2M\xc8\x9e\xa4h)\xf6\xd8o'

5. T_proposer = 15

Non-block delay = (40,64)ms

T_priority = 500ms



Public Key:

Node4

'\xcb\xfd(\x01\x9a\x7f#\xef?\x1e\x87\xbe\xc3=qDQ$\\\xa0\x9bZ\x7f\xf0\xc1\x19\xbco\xb7\xfahkH^\x06-\xc3F-\x17\x0f\x9e)\xfe\xdb"\xfa\x05'

Node9

'cm\x8eI\x99\xddKH\x89\xcb\xa99cp\xbb\x83\xf4\xe9\x8c7U\x9e\x19\\k\xbb\x05\xa2\x8f\xf1Z9\x11\xbe\x06\x91-\x9d\x07\x93\x8d\x82\tf\xfb)/'

CodeBase Hash:

be6a019b54da8bdc9e7d62dfa46ba2322d67cda0ec9f8717d9fafd7b8d0a755c