Algorand Report

Digvijaysingh Gour[1], **Shinjan Mitra**[2]
17305R001[1],17305R005[2]
{digvj,shinjan}@cse.iitb.ac.in
Indian Institute of Technology, Bombay

Abstract

Algorand (Gilad et al. 2017) is a proof-of-stake protocol which confirms transactions on the order of a minute as compared to one hour in the case of Bitcoin. This project aims to build a discrete event simulator for Algorand in Python.

Design

The discrete event simulator has been built using the Python package Simpy. This provides an **environment** in which different **processes** interact using **events**, specifically the timeout event. The various entities and events are described in the following sub-sections.

Entities

- **Environment** : It is a boilerplate for managing simulation time and scheduling and processing of events throughout the Simulation.
- **Nodes**: Nodes are the acting entities in the simulator which is capable of generating different types of messages, processing messages and creating blocks. Nodes are connected by communication links to other nodes with either blocking or non-blocking delays. Nodes participate in different processes like Sortition, Gossip, Block Proposal and BA* for functioning of Algorand Simulation.
- **Blocks**: Blocks are the objects created by Nodes at the end of a single round. A block shall contain previous block hash, 256 bit random string and pointer to previous block.
- **Store**: A pipe like object used as receiving pipe for consumption of objects like Blocks in this case.

Simpy (SimPy) library in Python gives us platform to develop applications using above abstractions directly. It is a library used for developing Discrete Event Simulator using ready made objects like Environment, Events, Process etc.

We have also used NetworkX library in Python to simulate network graph, edge delays and to find out different graph related properties like whether graph is connected or not, whether sum of all degrees is even or not etc. We have also used Numpy, Sympy and Random for distribution generation purpose.

In the remaining section of the report, we describe various experiments that we had conducted, our results and our interpretations corresponding to them.

Experiments

Experiment 1 : Stake Based Cryptographic Sortition

Aim of this experiment is to analyze how selection of sub-users is affected by Stakes owned by Nodes. So we have evaluated mean sub-users which are selected across all steps in all rounds considering following 2 cases :

- $\tau_{step} = 32$: We can observe here that mean value gradually increases as stakes increase. Notice that maximum value attained here is 0.008.

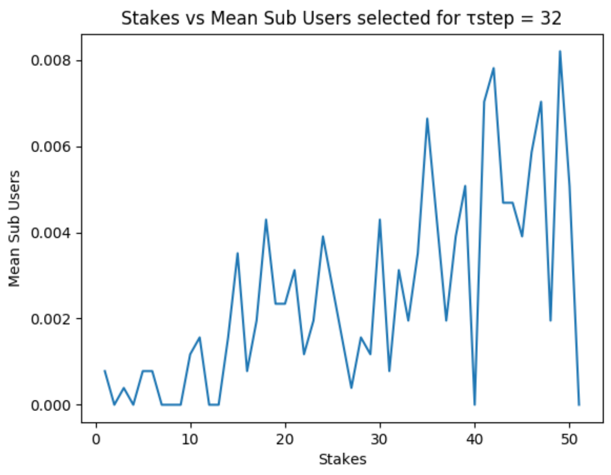

Figure 1: Stake vs Mean Sub Users for $\tau_{step} = 32$

- $\tau_{step} = 64$: We can observe here that mean value gradually increases as stakes increase. We can observe that maximum value attained here is 0.0175 which is roughly double the value attained in previous case.

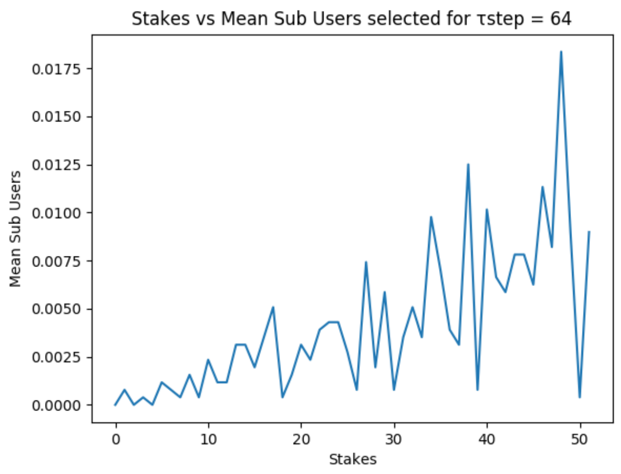

Figure 2: Stake vs Mean Sub Users for $\tau_{step} = 64$

Second part of this experiment was to analyze the fraction of mean sub users across all steps in all rounds. Here are the 2 cases again :

- $\tau_{step} = 32$: We can observe here that mean fraction value gradually increases as stakes increase. Notice that maximum value attained here is $0.12x10^{-6}$. However, maximum value is attained in stakes around 40, most probably majority of stakes were around this value.

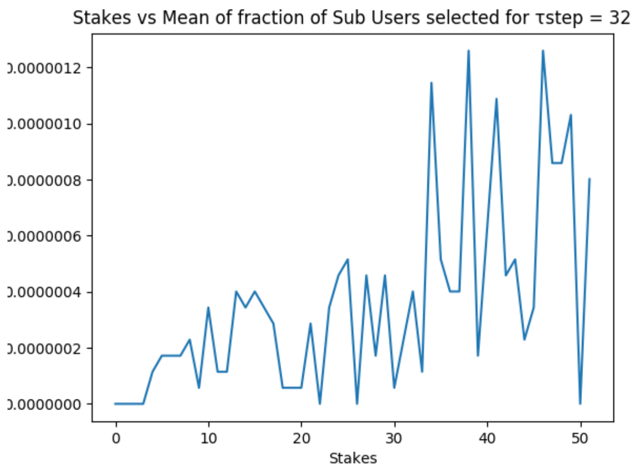

Figure 3: Stake vs Mean Sub Users for $\tau_{step} = 32$

- $\tau_{step} = 64$: We can observe here that mean fraction value gradually increases as stakes increase. We can observe that maximum value attained here is $0.2x10^{-6}$ which is roughly double the value attained in previous case and max value is attained again around 40.

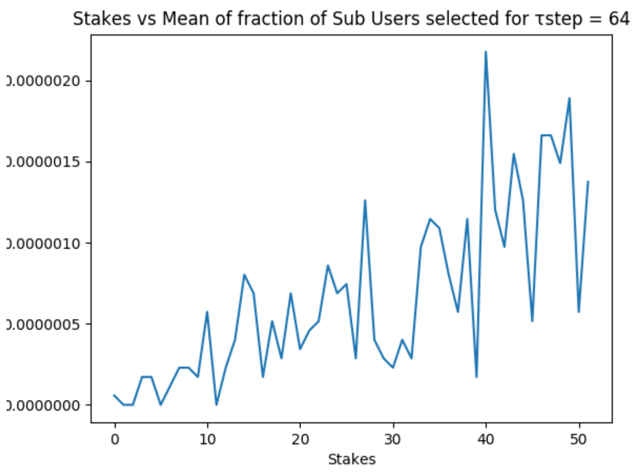

Figure 4: Stake vs Mean Sub Users for $\tau_{step} = 64$

Experiment 2 : Highest Proposer and Network Delay

Aim of this experiment is to analyze how view of each proposer changes regarding highest proposer in a given around at the end of λ_{step}. So we have evaluated highest block proposer value for each proposer in following cases with $\tau_{proposer} = 5$:

- **Message delay follows N(40,64) distribution** : We can observe here that there is only one Node ID with value 5 since $\tau_{proposer} = 5$ and delays are sufficient.

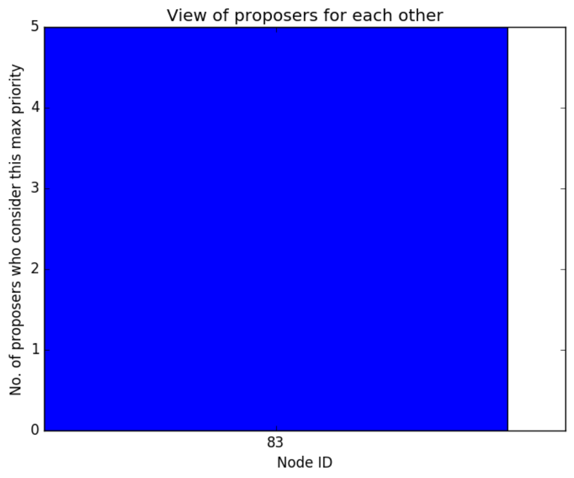

Figure 5: Node ID vs No. of Proposers considering this as highest proposer following N(40,64) distribution

- **Message delay follows N(60,64) distribution** : We can observe here that there is only one Node ID with value 8 since $\tau_{proposer} = 5$ and delays are sufficient.

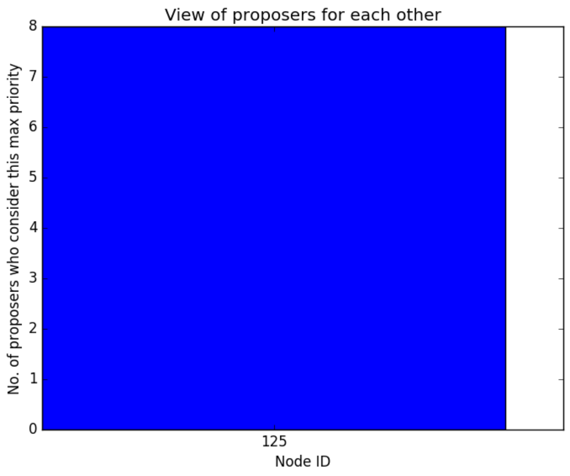

Figure 6: Node ID vs No. of Proposers considering this as highest proposer following N(60,64) distribution

- **Message delay follows N(100,64) distribution** : We can observe here that there is only one Node ID with value 7 since $\tau_{proposer} = 5$ and delays are sufficient.

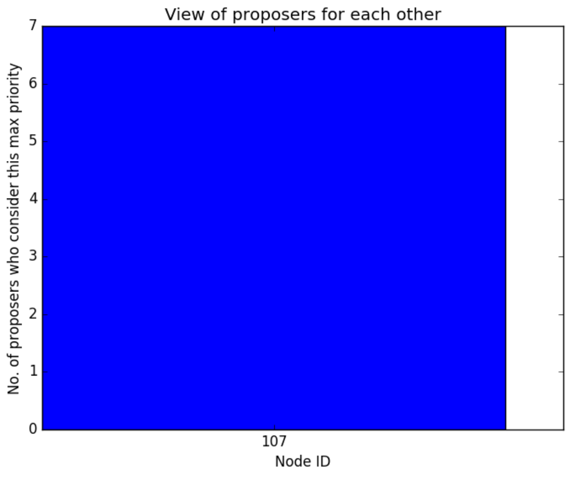

Figure 7: Node ID vs No. of Proposers considering this as highest proposer following N(100,64) distribution

Second part of this experiment was to analyze how changing value of $\tau_{proposer}$ affects scenario. Here are the 2 cases again following N(40,64):

- $\tau_{proposer} = 10$: We can observe here that there is only one Node ID with value 11 since $\tau_{proposer} = 10$ and delays are sufficient.

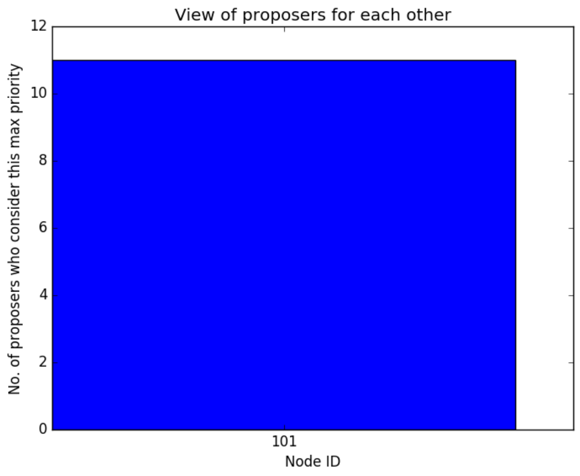

Figure 8: Node ID vs No. of Proposers considering this as highest proposer for $\tau_{proposer} = 10$

- $\tau_{proposer} = 15$: We can observe here that there is only one Node ID with value 17 since $\tau_{proposer} = 15$ and delays are sufficient.

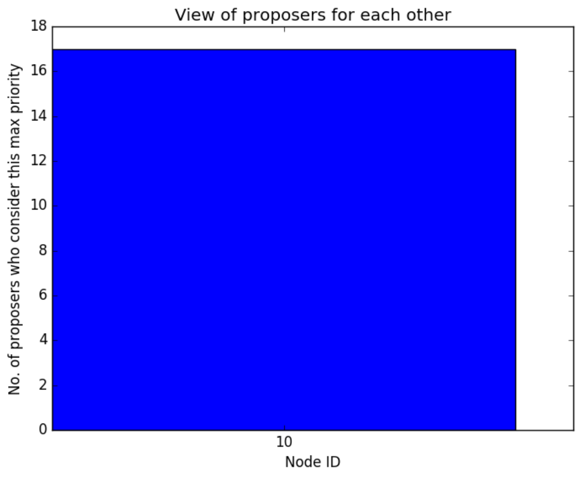

Figure 9: Node ID vs No. of Proposers considering this as highest proposer for $\tau_{proposer} = 15$

However, when we decrease λ_{step} to 0.1 or 100 ms we get following result.

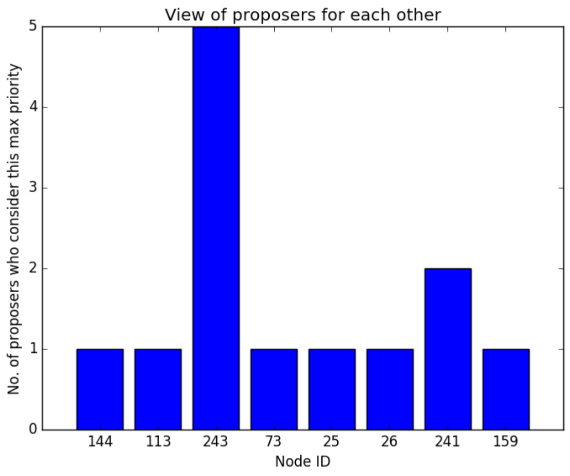

Figure 10: Node ID vs No. of Proposers considering this as highest proposer for $\tau_{proposer} = 15$ with $\lambda_{step} = 0.1$

Since now $\lambda_{step} = 0.1$ delays are quite less thus entire network is not covered under gossip, we are now able to see more number of proposers here.

References

Gilad, Y.; Hemo, R.; Micali, S.; Vlachos, G.; and Zeldovich, N. 2017. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles*, 51–68. ACM.

SimPy. Events - simpy 3.0.11 documentation.