Prism: Deconstructing the Blockchain to Approach Physical Limits

CS762: Project proposal

by

Gadde Nitish Samanth - 180050031

Tekuri Sai Akhil - 180050112

Urabavi Revanth Kumar - 180050114

Instructor:

Vinay Joseph Ribeiro

Computer Science and Engineering

INDIAN INSTITUTE OF TECHNOLOGY BOMBAY

2022

Chapter 1

Project Proposal

1.1 Objectives

Prism [1] is a new proof-of-work blockchain protocol, which can achieve high throughput and low latency when compared to standard proof-of-work. Our objectives for this project are as follows

- We would just be doing in depth theoretical study of Prism protocol along with all the theorems mentioned in the paper. We would not be doing any new theoretical study.
- Simulation of Prism protocol in local machines. We are planning to use python to simulate the protocol, there exists no implementation/simulator of the protocol.
- The analysis of the security, throughput and latency of Prism protocol
- Comparing throughput efficiency, latency with standard POW protocol

Tentative Timeline

Date	Progress	Objective
14/02/2022	Progress Report 1	Theoretical study of Prism protocol and understanding proofs of the concept
20/03/2022	Progress Report 2	Simulating the prism consensus protocol
10/04/2022	Final Report and Demo	Analysis of the security, throughput and latency of Prism

Chapter 2

Progress Report 1

2.1 Introduction

In 2008, Satoshi Nakamoto invented the concept of a blockchain. A distributed permissionless setting where people can agree upon the transactions that needs to be in blockchain through POW consensus. In 2015, Ethereum is launched, an another blockchain technology where people not only agree upon transactions but also agree in smart contracts. There are three parameters to measure the performance of blockchain, they are security, transaction throughput, confirmation latency.

Bitcoin and ethereum are 50 % secure. It means they can tolerate an adversary which has up to 50 % of hashing power. So, they are greatly secure. Now coming to transaction throughput, these blockchains support less than 20 txn/sec which is very poor compared to the number of txn/sec supported by the centralized banks. So, Bitcoin and ethereum are quiet poor when it comes to transaction throughput. Conformation latency is at the order of tens of minutes to hours depending on the the conformation error probability. The centralized banks take few seconds to conform the transaction. To sum it up bitcoin and ethereum are highly secure but their throughput and latency are poor.

Now a days the Network capacity(C) (internet speed) is at the order of Mbps. So, it is possible for a node/miner to push 20Mb size worth of transaction per second into blockchain network. But bitcoin blockchain has 1MB every 10 mins on an average. It takes minimum of speed-of-light propagation delay(D) for a confirmation since a transaction needs to reach all the other nodes/miners in the network.

number of transactions which can be confirmed per second : λ

confirmation latency : τ

$$\lambda < C, \tau > D \tag{2.1}$$

So, one solution from increasing transaction throughput and decreasing the latency is to increase the mining rate. But increasing the mining rate increases forks and allow different attacks like double spend attack which decreases the security of the block chain system.

We can approach the physical limit of the network C and D by deconstructing the blockchain. A block in the blockchain serves 3 purposes. One is to store and add transactions to the main chain. Second is to stand in election to be in the longest chain. Third is to vote for ancestor blocks to stay in the longest chain via parent link. Assuming a block B_k. When B_k is added to the blockchain it serves purpose 1 and 2 and server purpose 3 from the point of view of it's ancestor blocks B_k-1, B_k-2 etc, by increasing the depth of it's ancestors blocks in the blockchain. If the depth of a block in the blockchain increases then it becomes more difficult for adversary to do an attack on that block and that block is more secure and stay in the longest chain. In Prism we deconstruct the blockchain into transaction pool, proposal blocks and voter blocks organised in voter trees. The detailed description is given below.

2.2 Prism

In this work we introduce, a new proof-of-work blockchain protocol, which can achieve 1) security against up to 50% adversarial hashing power; 2) optimal throughput up to the capacity C of the network; 3) confirmation latency for honest transactions proportional to the propagation delay D, with confirmation error probability exponentially small in the bandwidth-delay product CD; 4) eventual total ordering of all transactions. Our approach to the design of this protocol is based on *deconstructing* Nakamoto's blockchain into its basic functionalities and systematically scaling up these functionalities to approach their physical limits.

The Prism, , which, in the face of any powerful adversary with power $\beta < 0.5$, can *simultaneously* achieve:

1. **Security:** a total ordering of the transactions, with consistency and liveness guarantees.

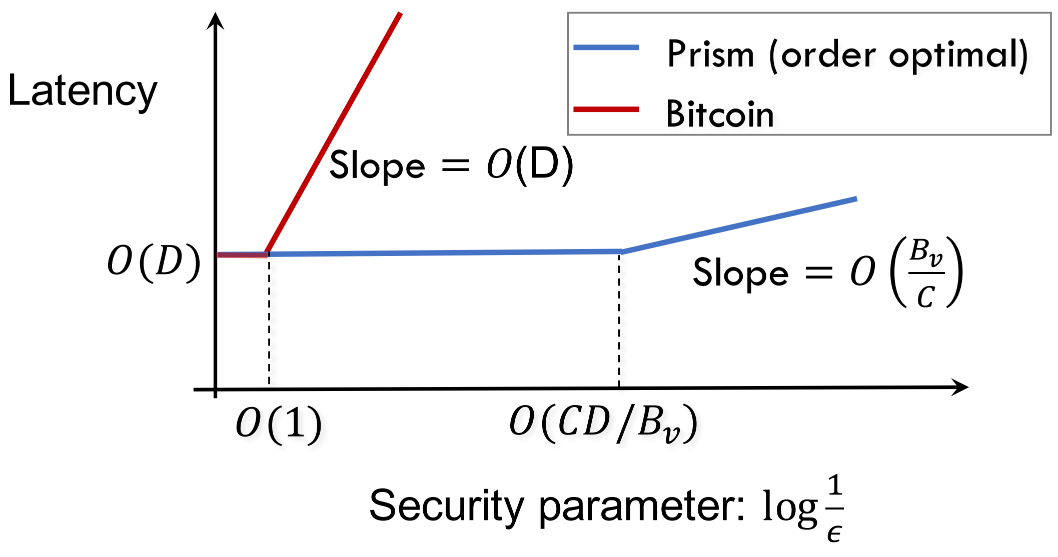


Figure 2.1: Confirmation latency vs. security parameter for . The latency of is independent of the security parameter value up to order CD/B_v and increases very slowly after that (with slope B_v/C). For , latency increases much more rapidly with the security parameter, with slope proportional to D. (Since $CD/B_v \gg 1$, this latter slope is much larger.)

2. **Throughput:** a throughput

$$\lambda = 0.9(1-\beta)C \quad \text{transactions per second.} \tag{2.2}$$

3. **Latency:** confirmation of all honest transactions (without public double spends) with an expected latency of

$$\mathbb{E}[\tau] < \max\left\{c_1(\beta)D, c_2(\beta)\frac{B_v}{C}\log\frac{1}{\varepsilon}\right\} \quad \text{seconds,} \tag{2.3}$$

with confirmation reliability at least $1-\varepsilon$ (Figure 2.1). Here, $c_1(\beta)$ and $c_2(\beta)$ are constants depending only on β

2.2.1 Protocol Description

We describe the content and roles of three types of blocks in the (Π, g) blockchain. We explain the protocol Π and the blockchain data structure C. We then define the *ledger inclusion rule*, g.

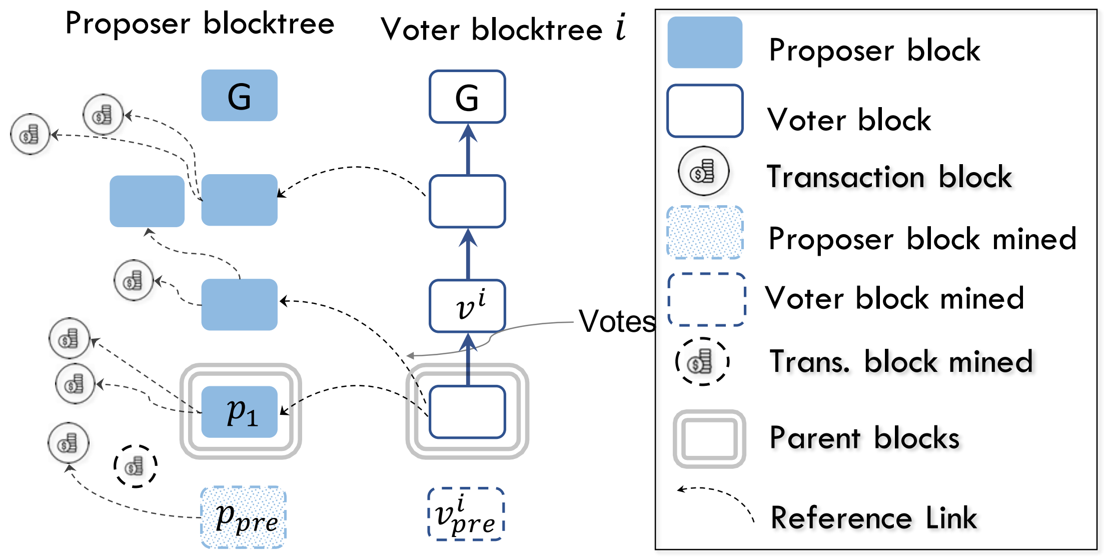


Figure 2.2: Snapshot of a miner's blocktree: The previously mined blocks have solid boundary whereas blocks which are being mined have dotted-boundary. A miner simultaneously mines on p_1, parent on proposer blocktree, v^i, parent on voter block blocktree $i(\forall i \in [m])$.

We use these different blocktrees to maintain three distinct types of blocks:

Proposer blocks: Proposer blocks represent the skeleton of the blockchain and are mined on the proposer blocktree according to the longest-chain rule. The *level* of a proposer block is defined as its distance from the proposer genesis block. The blocktree structure is only utilized in our protocol as a proof of level of a given proposal block. To construct the ledger, our protocol selects proposal block sequences where one block is chosen at each level. Proposer blocks can *refer* to transaction blocks and other proposer blocks by including pointers to referred blocks in their payload. For example, in Fig 2.2, the proposer blocktree has two proposer blocks mined at level 1, and one proposer block mined at levels 2 and 3, and they point to five transaction blocks in total. The proposer block with the most votes on level l is defined as the leader block of level
.

Voter blocks: Voter blocks are mined on m separate voter blocktrees, each with its own genesis block, according to the longest chain rule. We say a voter block *votes* on a proposer block B if it includes a pointer to B in its payload. Note that unlike many BFT consensus protocols, a malicious miner in Prism cannot equivocate when voting because voter blocks are sealed by proof of work. Even if a miner mines conflicting voter blocks and tries to send them to disjoint sets of honest users, all users will receive both blocks within one round. Each longest chain from each voter blocktree can cast at most one vote for each level in the proposer blocktree. More precisely, a voter block votes on all levels in the proposer tree that are unvoted by the voter block's ancestors. Therefore, the voter trees collectively cast at most m votes on a given level of the proposer blocktree. Fig. 2.2 shows voter blocktree i and its votes (dotted arrows) on each level of the proposer blocktree. For each level ℓ on the proposer blocktree, the block with the highest number of votes is defined as the *leader* block of level ℓ. v^i has not voted for any proposer block since the i tree chain has voted on all proposer blocks that were present when v^i is created.

Transaction blocks: Transaction blocks contain transactions and are mined on the proposer blocktree as in Fig.2.2. Although transaction blocks are not considered part of the proposer blocktree, each transaction block has a proposer block as its parent.

The process by which a transaction is included in the ledger is as follows: (1) the transaction is included in a transaction block B_T. (2) B_T is referred by a proposer block B_P. (3) Proposer block B_P is confirmed, either directly (by becoming a leader) or indirectly (e.g., by being referred by a leader).
Prism miners simultaneously mine one proposer block, one transaction block and m voter blocks. This happens via threshold intervals assigned to different blocks. A miner first generates a "superblock" containing the information of m+2 blocks. After mining a block, output of hash is deterministically mapped to either a voter block, transaction block or proposer block. After sortition, the miner discards unncecessary inforamtion and releases the block to the environment.

For transaction block, the content consists of all transactions that have not yet been included.

For proposer block, the content is a list of unreffered transaction blocks and unreffered proposer blocks.

For voter block in ith voter tree , the content is a list of proposer blocks at each level in the proposer blocktree that has not yet received a vote in the longest chain of the ith voter tree.

Chapter 3

Progress Report 2

3.1 Simulation

We have simulated a P2P[1] network using prism protocol. The simulations takes number of nodes, mean time interval between transactions, percentage of slow nodes, percentage of high CPU nodes, mean time interval between blocks, number of voter chains as input.

We used research[2] as a reference which models the degree of connectivity of nodes as a power law distribution. This paper studies the degree distribution of the bitcoin network containing 7025 nodes. From the analysis, it is shown from the data that the out-degree distributions follow power-law. In this distribution, few nodes have high degree of connectivity and more nodes have low degree of connectivity. Two algorithms (first-mover wins and fittest-gets-richer) are given for network generation in the paper which outputs networks in which each node follows power-law. We have implemented the fittest-gets-richer algorithm in our simulation. Since the connectivity of nodes doesn't change from Bitcoin to Prism, we have simulated the connectivity of nodes based on this power law distribution.

We have created abstract Superblock which on the basis of PoW can be a proposer block, voter block or Transaction block. A transaction block contains list of transactions. A proposer block contains transaction block ids. A voter block of chain *i* contains list of (level, proposer block id)votes at each level in the proposer blocktree that has not yet recieved a vote in the longest chain of the *i*th voter tree. During PoW, we have assigned probabilities for generation of block based on the threshold value of various blocks. In the simulation we have assigned equal probabilities for m+2 chains. We can change it since it is intutive to assign more threshold

[1]https://github.com/revanthitsme/Prism

value to the transaction block which means more probability to generate a transaction block.

We have taken 2 classes of nodes: one with high hashing power and one with low hashing power. The hashing power of HIGH CPU nodes is double the hashing power of LOW CPU nodes.Therefore, nodes with high hashing power wait half the time nodes with low hashing power waits on an average to transmit the block generated by it. Generally, we have to set the mean interarrival time of blocks so that there won't be much branching (so high mean is preferrable) and there should not be transactions filling up the transaction queue (so low mean is preferrable). The timedelay of broadcasting of block depends on the content of it. The simulation of prism consensus protocol on honest P2P node network is completed. We are still analysing the simulation results and will compare with standard bitcoin protocol.

Chapter 4

Final Report

4.1 Simulation

To compare PRISM with bitcoin, We have simulated a P2P network for bitcoin protocol. The simulation for bitcoin takes number of nodes, mean time interval between transactions, percentage of slow nodes, percentage of high CPU nodes, mean time interval between blocks as input. Figure 4.1 is the sample network distribution for 10 nodes using paper on Distributed systems.

As we discussed above, we create blocks by assigning different probabilities to various blocks. We have experimented with probabilities of creating different blocks. The probability of creating a transaction block is 4 times the probability of either proposer block or voter block. The probability of creating a transaction block is set to 2/3. The probability of creating a proposer block is set to 1/6. The probability of creating a voter block from m chains is set to 1/6. So, the probability of creating a voter block from a particular chain is 1/6m.

We created a prism tree sample figure 4.2 for number of nodes(n): 10, the percent of slow nodes(z): 20, the mean interarrival time of transactions(Ttx): 2, the percent of High CPU nodes: 80, block interarrival time(in sec): 5, no of voter chains: 10. Here in the figure 4.2, a proposer block points to its children and refers to its transaction blocks and a voter block points to its children and refers to its voted proposer block. In the simulation, each transaction block contain maximum of 20 transactions and each proposer block can refer maximum of 5 transaction blocks.

In the simulation, the proposer block is similar to bitcoin block. In prism, if a block(any) is created for every t secs, then the proposal block is created for every 6t secs on average, since we set the probability of proposer block to 1/6. So, we have to compare the throughput of PRISM

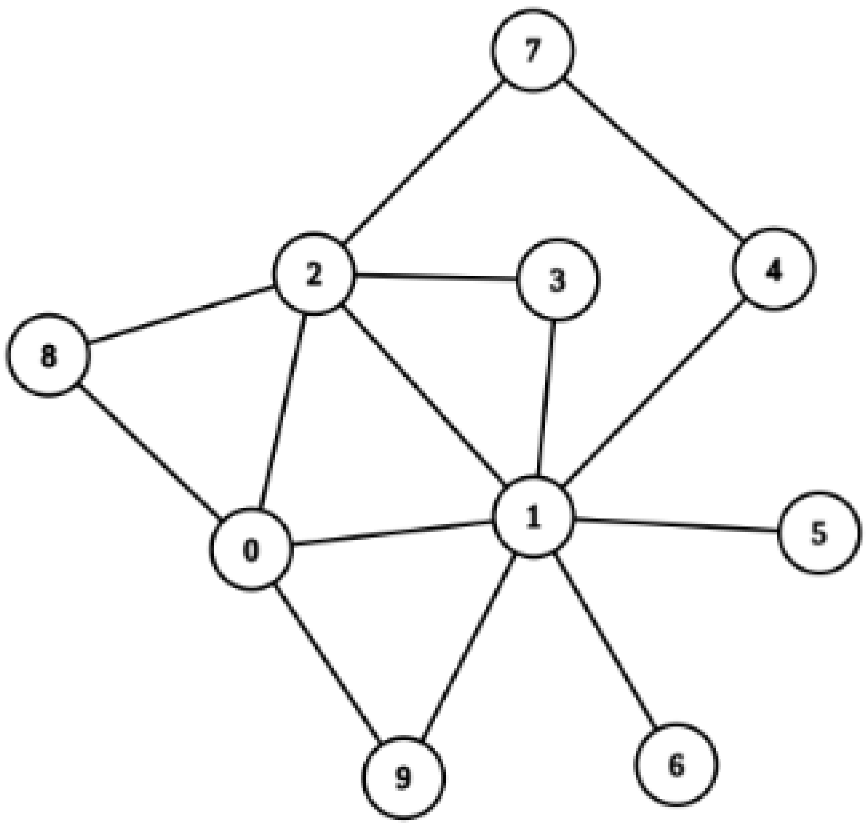


Figure 4.1: Block chain Network with power law distribution[2].

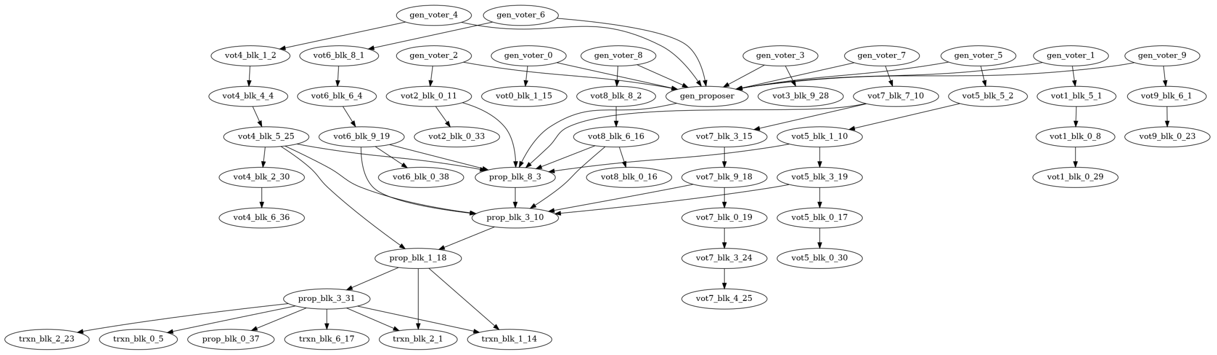
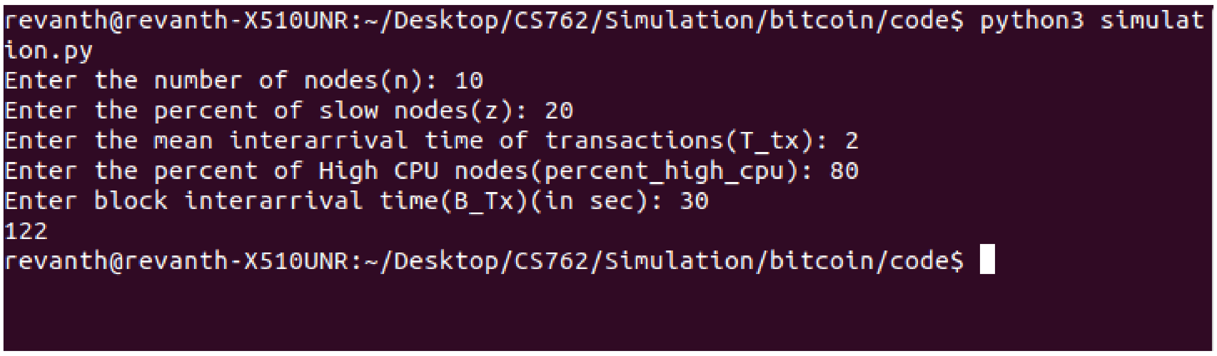
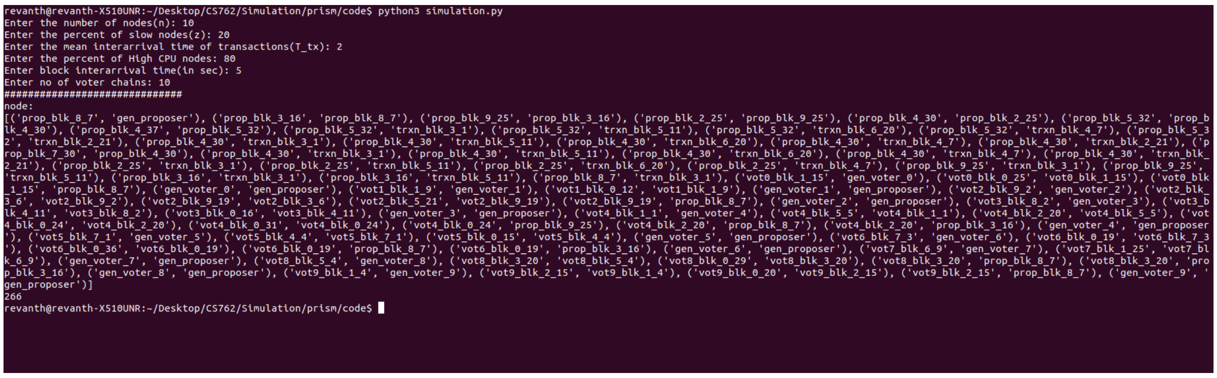


Figure 4.2: High Level Prism consensus protocol.



```
revanth@revanth-X510UNR:~/Desktop/CS762/Simulation/bitcoin/code$ python3 simulat
ion.py
Enter the number of nodes(n): 10
Enter the percent of slow nodes(z): 20
Enter the mean interarrival time of transactions(T_tx): 2
Enter the percent of High CPU nodes(percent_high_cpu): 80
Enter block interarrival time(B_Tx)(in sec): 30
122
revanth@revanth-X510UNR:~/Desktop/CS762/Simulation/bitcoin/code$
```

Figure 4.3: Bitcoin protocol simulation.



```
revanth@revanth-X510UNR:~/Desktop/CS762/Simulation/prism/code$ python3 simulation.py
Enter the number of nodes(n): 10
Enter the percent of slow nodes(z): 20
Enter the mean interarrival time of transactions(T_tx): 2
Enter the percent of High CPU nodes: 80
Enter block interarrival time(in sec): 5
Enter no of voter chains: 10
##############################
node:
[('prop_blk_8_7', 'gen_proposer'), ('prop_blk_3_16', 'prop_blk_8_7'), ('prop_blk_9_25', 'prop_blk_3_16'), ('prop_blk_2_25', 'prop_blk_9_25'), ('prop_blk_4_30', 'prop_blk_2_25'), ('prop_blk_5_32', 'prop_b
lk_4_30'), ('prop_blk_4_37', 'prop_blk_5_32'), ('prop_blk_5_32', 'trxn_blk_3_1'), ('prop_blk_5_32', 'trxn_blk_5_11'), ('prop_blk_5_32', 'trxn_blk_6_20'), ('prop_blk_5_32', 'trxn_blk_4_7'), ('prop_blk_5_3
2', 'trxn_blk_2_21'), ('prop_blk_4_30', 'trxn_blk_3_1'), ('prop_blk_4_30', 'trxn_blk_5_11'), ('prop_blk_4_30', 'trxn_blk_6_20'), ('prop_blk_4_30', 'trxn_blk_4_7'), ('prop_blk_4_30', 'trxn_blk_2_21'), ('p
rop_blk_7_30', 'prop_blk_4_30'), ('prop_blk_4_30', 'trxn_blk_3_1'), ('prop_blk_4_30', 'trxn_blk_5_11'), ('prop_blk_4_30', 'trxn_blk_6_20'), ('prop_blk_4_30', 'trxn_blk_4_7'), ('prop_blk_4_30', 'trxn_blk_
2_21'), ('prop_blk_2_25', 'trxn_blk_3_1'), ('prop_blk_2_25', 'trxn_blk_5_11'), ('prop_blk_2_25', 'trxn_blk_6_20'), ('prop_blk_2_25', 'trxn_blk_4_7'), ('prop_blk_9_25', 'trxn_blk_3_1'), ('prop_blk_9_25',
'trxn_blk_5_11'), ('prop_blk_3_16', 'trxn_blk_3_1'), ('prop_blk_3_16', 'trxn_blk_5_11'), ('prop_blk_8_7', 'trxn_blk_3_1'), ('vot0_blk_1_15', 'gen_voter_0'), ('vot0_blk_0_25', 'vot0_blk_1_15'), ('vot0_blk
_1_15', 'prop_blk_8_7'), ('gen_voter_0', 'gen_proposer'), ('vot1_blk_1_9', 'gen_voter_1'), ('vot1_blk_0_12', 'vot1_blk_1_9'), ('gen_voter_1', 'gen_proposer'), ('vot2_blk_9_2', 'gen_voter_2'), ('vot2_blk_
3_6', 'vot2_blk_9_2'), ('vot2_blk_9_19', 'vot2_blk_3_6'), ('vot2_blk_5_21', 'vot2_blk_9_19'), ('vot2_blk_9_19', 'prop_blk_8_7'), ('gen_voter_2', 'gen_proposer'), ('vot3_blk_8_2', 'gen_voter_3'), ('vot3_b
lk_4_11', 'vot3_blk_8_2'), ('vot3_blk_0_16', 'vot3_blk_4_11'), ('gen_voter_3', 'gen_proposer'), ('vot4_blk_1_1', 'gen_voter_4'), ('vot4_blk_5_5', 'vot4_blk_1_1'), ('vot4_blk_2_20', 'vot4_blk_5_5'), ('vot
4_blk_0_24', 'vot4_blk_2_20'), ('vot4_blk_0_31', 'vot4_blk_0_24'), ('vot4_blk_0_24', 'prop_blk_9_25'), ('vot4_blk_2_20', 'prop_blk_8_7'), ('vot4_blk_2_20', 'prop_blk_3_16'), ('gen_voter_4', 'gen_proposer
'), ('vot5_blk_7_1', 'gen_voter_5'), ('vot5_blk_4_4', 'vot5_blk_7_1'), ('vot5_blk_0_15', 'vot5_blk_4_4'), ('gen_voter_5', 'gen_proposer'), ('vot6_blk_7_3', 'gen_voter_6'), ('vot6_blk_0_19', 'vot6_blk_7_3
'), ('vot6_blk_0_36', 'vot6_blk_0_19'), ('vot6_blk_0_19', 'prop_blk_8_7'), ('vot6_blk_0_19', 'prop_blk_3_16'), ('gen_voter_6', 'gen_proposer'), ('vot7_blk_6_9', 'gen_voter_7'), ('vot7_blk_1_25', 'vot7_bl
k_6_9'), ('gen_voter_7', 'gen_proposer'), ('vot8_blk_5_4', 'gen_voter_8'), ('vot8_blk_3_20', 'vot8_blk_5_4'), ('vot8_blk_0_29', 'vot8_blk_3_20'), ('vot8_blk_3_20', 'prop_blk_8_7'), ('vot8_blk_3_20', 'pro
p_blk_3_16'), ('gen_voter_8', 'gen_proposer'), ('vot9_blk_1_4', 'gen_voter_9'), ('vot9_blk_2_15', 'vot9_blk_1_4'), ('vot9_blk_0_20', 'vot9_blk_2_15'), ('vot9_blk_2_15', 'prop_blk_8_7'), ('gen_voter_9', '
gen_proposer')]
266
revanth@revanth-X510UNR:~/Desktop/CS762/Simulation/prism/code$
```

Figure 4.4: PRISM protocol simulation.

and bitcoin-blockchains by setting the block interarrival time as t and 6t respectively. So, from fig 4.3 and fig 4.4 we can see the interarrival time for bitcoin is set to 30 where as for PRISM, it is set to 5. The number of transactions that entered into the bitcoin ledger are 122. The number of transactions that entered into the PRISM ledger are 266. The values can change when run at different time. The throughput for PRISM is higher than that of bitcoin from the simulations.

4.2 Video link

`https://drive.google.com/drive/folders/1GpQkZjS4FwQxX3gUO5IEBQMcAYYiROQZ?usp=sharing`

4.3 Github Code link

`https://github.com/revanthitsme/Prism`

References

[1] Vivek Bagaria, Sreeram Kannan, David Tse, Giulia Fanti, and Pramod Viswanath. Prism: Deconstructing the blockchain to approach physical limits. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 585–602, 2019.

[2] Marco Alberto Javarone and Craig Steven Wright. From bitcoin to bitcoin cash: a network analysis. In *Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems*, pages 77–81, 2018.