FINAL PROJECT

CSCI.652 - Distributed Systems

April 23, 2019

Implementation and Performance Evaluation

Cliffton Fernandes
Manpreet Kaur
Rochester Institute of Technology
MS Computer Science

Contents

IMPLEMENTATION: FULLY DECENTRALIZED VOTING SYSTEM

In this project we have implemented a fully decentralized peer to peer voting system using blockchain technology to achieve security, transparency, accessibility in the voting process. we have implemented the solution using below two approaches:

Simple Block Chain

In this approach we have implemented block chain solution from scratch in python.

System Architecture

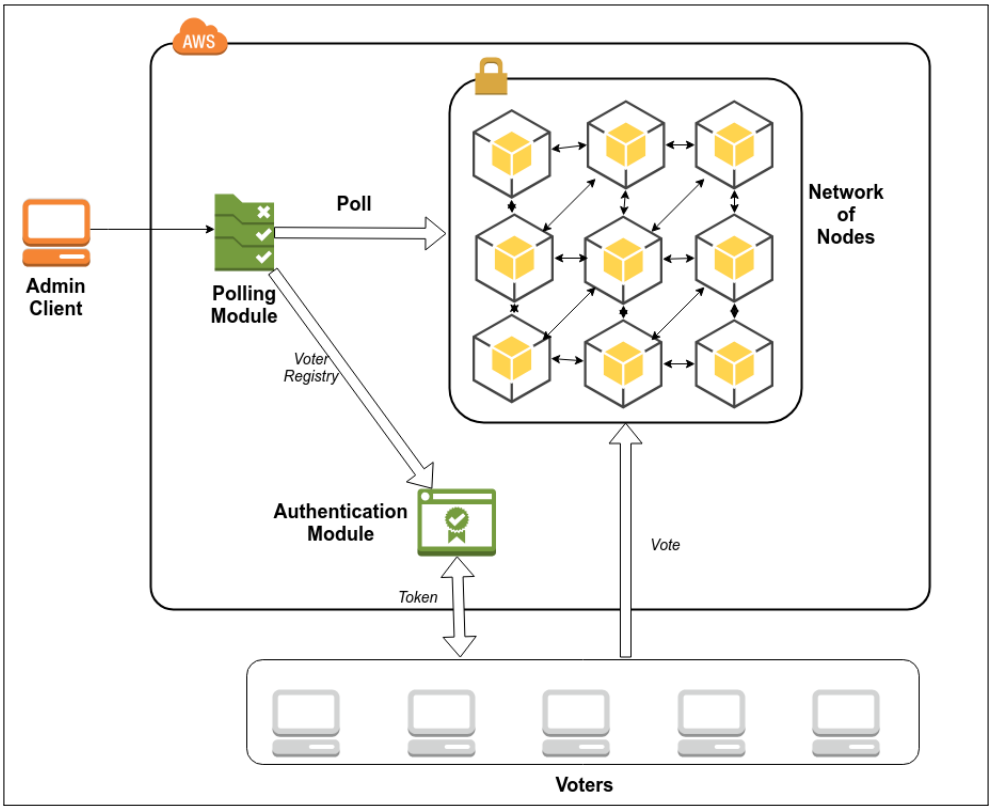

Figure 1: System Architecture

Modules

- **Node:** Nodes are the most important parts of the network they are responsible for registering the transactions, mining the blocks, saving the blockchain and broadcasting the transactions and blocks across the network etc.

- **Blockchain:** The blockchain represents the entire chain of blocks. The blocks are linked together by a cryptographic hash.
- **Block:** Blocks are the individual records that form the blockchain. Each block is linked to the previous block by a cryptographic hash. Each block contains the transaction data, its own proof of work, the hash of the previous block in the chain and the timestamp.

Proof-of-Work

Proof of work in a blockchain network involves the computational problem that needs to be solved by the miners in order to generate the block. The proof-of-work involves scanning for a value that when hashed, such as with SHA-256, the hash begins with a number of zero bits. The average work required is exponential in the number of zero bits required and can be verified by executing a single hash. This ensures that the blocks in the blockchain cannot be changed without redoing all the computational work for not just the current block but for all the blocks till the genesis block.

Network

1. New transactions are broadcast to all nodes.
2. Each node collects new transactions into a block.
3. Each node works on finding a difficult proof-of-work for its block.
4. When a node finds a proof-of-work, it broadcasts the block to all nodes.
5. Nodes accept the block only if all transactions in it are valid and not already spent.
6. Nodes express their acceptance of the block by working on creating the next block in the chain, using the hash of the accepted block as the previous hash.

Consensus Algorithm

In cases where a node receives two blocks with the same proof of work the node adds both the blocks to the chain as two separate branches and waits for the next block to arrive. Based on the previous of the next block that arrives the node decides which previous block to discard. Thus the nodes always assume that the longest chain is correct and keep extending the longest chain.

Scalability

The normal blockchain system does not scale well and can process only 3-7 transactions per second therefore a better protocol is required

Block Chain using PHOTON Protocol:

In this approach we allow creation of more blocks per second in order to increase the throughput of the system which results in forking of multiple chains.Later, rather than just maintaining the longest chain and discarding all other blocks we run BlockDAG algorithm to find the blocks that are fully connected to the network of chains. In this way we can accept all the blocks that got created in parallel. We have used the third party implementation of phantom to maintain and manipulate the blockchain.In this protocol chains are maintained in the form of DAG(Directed Acyclic Graph) where each block reveals the full world view. In oppose to simple block chain each block in PHANTOM can have multiple predecessor which basically represents the tips of the DAG when this block got created. The below diagram shows the one possible DAG in PHANTOM and related terminologies:

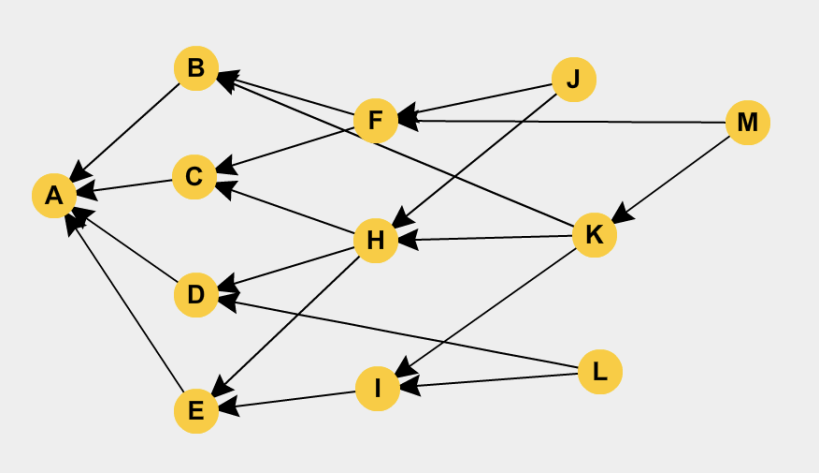

Figure 2: *DAG: Each block reveal it's full world view*

- past(H) = Genesis, C, D, E - blocks which H references directly or indirectly, and which were provably created before H;
- future (H) = J, K, M - blocks which reference H directly or indirectly, and which were provably created after H;

- anticone (H) = B, F, I, L - the order between these blocks and H is ambiguous. Reaching consensus on the order between blocks and other blocks in their anticone is the main challenge that we face.
- tips(G) = J, L, M - leaf-blocks, namely, blocks with in-degree 0; these will be referenced in the header of the next block.

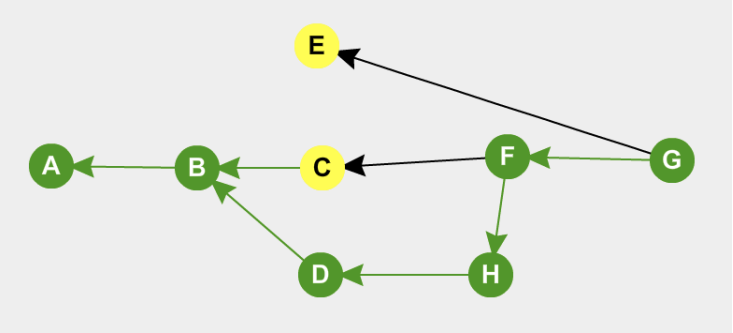

Figure 3: *DAG are more powerful as they can return the longest chain as in simple blockchain protocol. For example the green nodes in the above graph forms the longest chain*

Below steps summarizes how Phantom works:

- Blocks are created using transactions and proof of work.
- Block will point to all the tips of the graphs when it got created.Note that if D represents the propagation delay of the network and t is the time when block got created then block is aware of all the nodes that were in the system at time (t-D) and all the nodes that will be created anytime after (t + D). Thus block is not only verified against other blocks that got created during interval [(t-D),(t+D)]. If the blocks that got created in parallel contains conflicting transaction then it will not be verified.
- Every node will refer to the local copy of DAG and decide if Transaction will be accepted, rejected or marked as pending.

PERFORMANCE EVALUATION

Simple Block Chain

In this approach we have implemented block chain solution from scratch in python where the block creation is regulated by the network propagation delay.However, we observed the below problems with this approach:

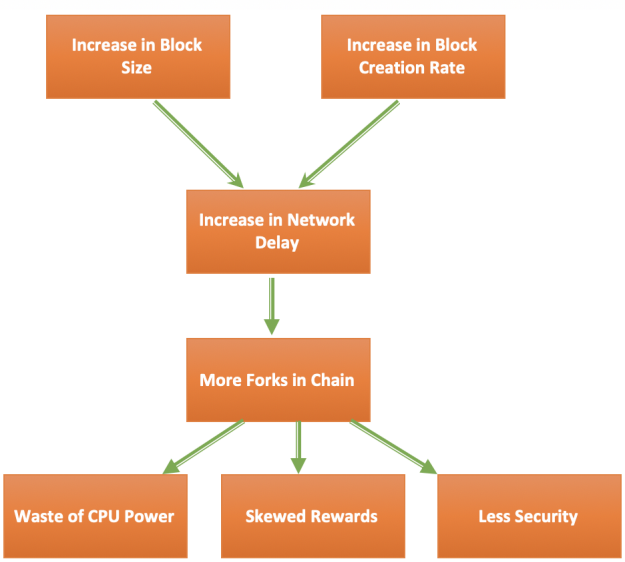

Figure 4: Problems in Simple Block Chain

- Since, block creation is limited by the propagation delay in the network, throughput of the system is 3-7 transactions per second in a large system.
- Increased throughput can lead to decreased security and waste of CPU power.
- System needs to wait for a long time for a transaction to get confirmed.
- Size of the block cannot be increased above a certain limit to avoid network propagation delay.

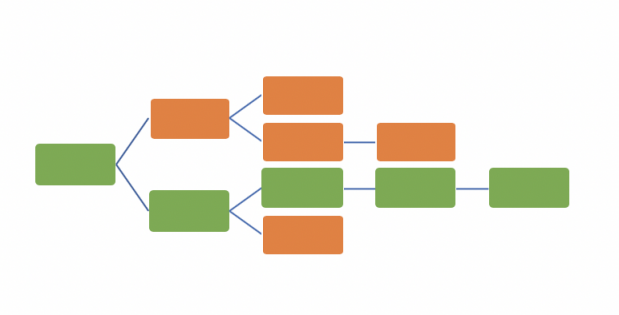

Figure 5: *Simple Block Chain: In This Case blocks labelled green will be part of the system as they belong to the longest chain. The one Marked in orange will be discarded as they do not belong to the longest chain.*

Results

We ran the simulation with below parameters:

- For each run we tried modifying the number of blocks created per second.
- we ran the simulation on 4 nodes where each node has the same computation power.
- for each run we have recorded the length of the chain, Number of Blocks created and number of blocks dropped.

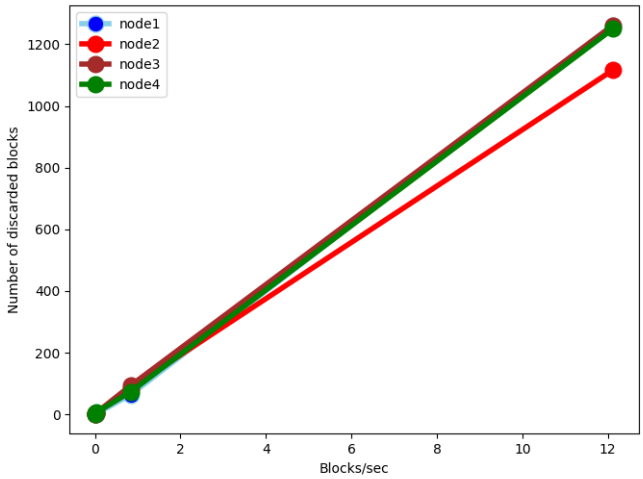

Figure 6: *Blocks Dropped vs Number of Blocks per Second*

The above results clearly shows that with the increase in the number of blocks per second, simple blockchain solution is dropping more blocks and thus most of the computation power is getting wasted.In order to simulate the above environment that is to vary the number of b locks generated per second, we modified the parameter for the hash function that generates the nonce. We generated the above results by varying the length of the nonce from 4 to 8. we could not test it with the nonce length more than 8 as the complexity for the algorithm increase exponentially and with value greater than 9 our program was not terminating on our machine.

We didn't test the propagation time by varying the block size as our blocks were of the same size. But Figure 7 graph from the paper clearly shows the increase in propagation time as the block size increases.

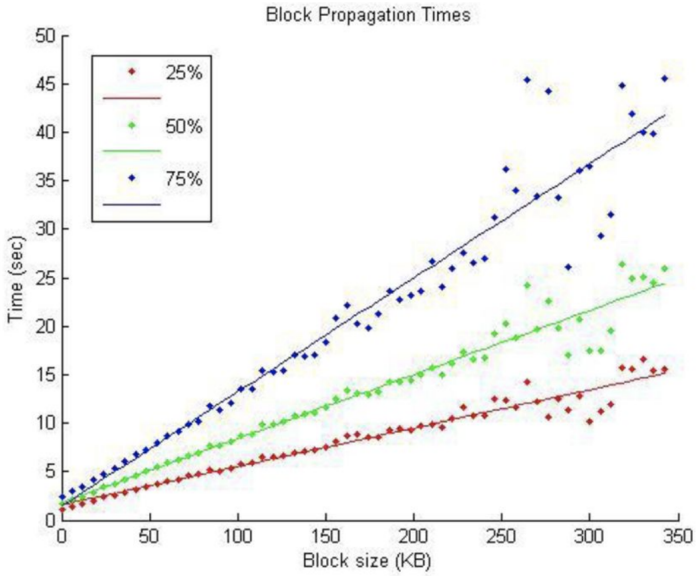

Figure 7: *Propagation Time VS Block Size*

PHOTON Results

Figure 8 shows the PHOTON results by varying the size of K. where k is the paramater that decides the number of blocks that are allowed to create in parallel. Results clearly shows that we need large confirmation depth for k > 0.

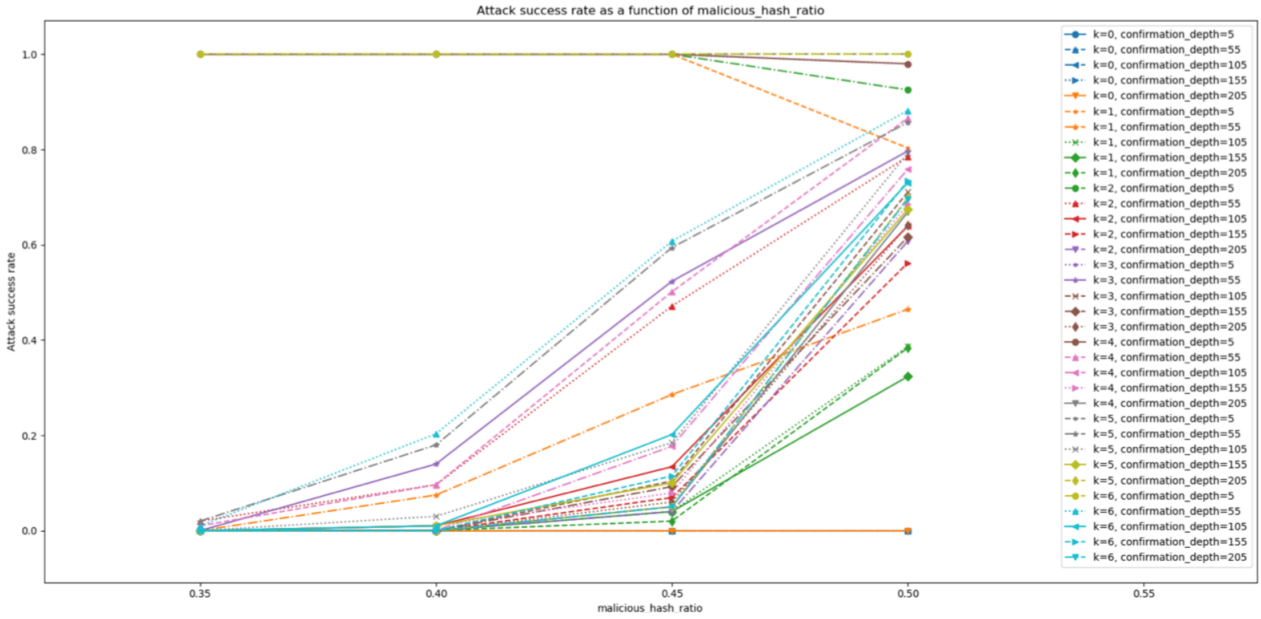

Figure 8: *K vs Confirmation Depth*

REFERENCES

- https://eprint.iacr.org/2018/104.pdf
- https://bitcoin.org/bitcoin.pdf
- https://eprint.iacr.org/2013/881.pdf
- https://github.com/AvivYaish/PHANTOM