Hubs and Authority based DPoS (HA-DPoS)

Jeet Bhadaniya
jeet.b@ahduni.edu.in
Ahmedabad University
Ahmedabad, Gujarat, India

Smeet Bhadaniya
smeet.b@ahduni.edu.in
Ahmedabad University
Ahmedabad, Gujarat, India

Abstract

The basis of blockchain systems consists of the consensus algorithms and this, directly which affects the performance as well as the functioning of blockchain. Among the most commonly used consensus algorithm is Delegated Proof of Stake (DPoS) that stands out for significantly shortening the time needed for the transaction verification via the selection of the few validators (the nodes of the network) to create the blocks of transactions. Nevertheless, in spite of current algorithms" security flaws like "one ballot, one vote", limited decentralization and bad actors, the recent ones have become more decentralized and much more secure. In this article, we introduce the new Decentralized Proof-of-stake (DPoS) algorithm which integrates the concepts of hubs and authority to determine two reputations: Delegate and Voter. The aim of this algorithm is to solve the global problem of one vote per ballot by designing one ballot, multiple vote system. Nodes are assigned using their reputation as delegates and as voters in the blockchain network on which the reputation they attract. First, there is the concentration of the Reputation that is a measure of the suitability of a node for a delegate election while the voting Reputation assesses its effectiveness. Then this Reputation value and port hubs and authority value can be used to elect delegates which have lower chances of being malicious. Unlike the traditional rule-based reactive security, the system defines penalties and rewards for an appropriate behavior and malicious acts. The new DPoS algorithm being emphasized in the paper aims at boosting the safety, fairness, and efficiency of blockchain consensus mechanisms.

Keywords: DPoS,Hubs and Authority,Consensus,Blockchain, Fairness

SNA Presentation, Ahmedabad University, Ahmedabad,

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

ACM Reference Format:
Jeet Bhadaniya and Smeet Bhadaniya. 2024. Hubs and Authority based DPoS (HA-DPoS). In *Proceedings of April 13, 2024 (SNA Presentation).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

The blockchain technology has pointed to a revolution in different spheres displaying the features of decentralization, transparency and security in data storage and processing. The underlying consensus mechanism at the heart of distributed ledger systems is very essential for it ensures agreement regarding the ledger's state among different distributed nodes.

Consensus algorithms are crucial in decentralised systems to reach agreement without one single authority instating the decision. They function as a medium of exchange, watch the continuity with the transfer of the transactions and supervise the process of the agreement. In general nodes work together for the sake of consensus in the build of such mechanisms as PoW, PoS, and DPoS. Due to those algorithms, participants treatment is truthful, potential double-spending is prevented, and a blockchain's consistency and reliability are kept.

DPoS's usability, mostly, relies on the network/s capabilities of peak throughput, and it is green as will ensure the environment is safe for all living species. DPoS, used by EOS[9] and Tron[6] and introduced by Bitshares[8], is about that people who own a coin, vote between whom the transaction will be checked. The proof-of-stake consensus function provides security equivalent to the PoW without consuming higher energy. DPoS has an impressive architecture for the block production and for governance simply through voting. This feature gives boasts in speed, efficiency, and in the balance of decentralization and central control.

Also in the center are prominent consensus algorithms such as Proof of Work (PoW) and Proof of Stake (PoS). In PoW, like the one that is used in bitcoin, the miners compete in solving the puzzel that ends up in addition of the next block. PoS is one of the consensus algorithms that govern the choice of validators depending on their cryptocurrency holdings, to allow validating of transactions as well as creation of blocks in ethereum[1].

Traditionally, DPoS may focus on a node's voting power ratio for vote casting instead of considerations like reliability or give weight to network members' contributions. In our

proposed HA-DPoS algorithm, we introduce innovative concepts: The reputation of delegates, voters' reputation, and voters and delegates importance should be delegated. The endowments targets at fixed assessment of nodes with the layout of multi-faceted and fair assessment. Our system monitors the value of the voters' vote to automate the process of delegate elections away from conventional methods. This will ensure fairness in delegate elections.

1.1 Problem Definition

The problem at hand is to design a consensus algorithm that leverages the concepts of Hubs and Authority[5] to enhance the efficiency and scalability of blockchain networks. Specifically, we aim to achieve the following objectives:

- **Utilization of Hubs and Authority Metrics**: Incorporate node metrics to refine voting weights.
- **Dynamic Voting Weight Calculation**: Dynamically compute voting weights using Hubs, Authority, and node reputation.
- **Selection of Delegate Nodes**: Choose 'k' delegate nodes based on voting weights for transaction validation and block creation.
- **Efficiency and Scalability Enhancement**: Improve network efficiency and scalability by emphasizing influential nodes in consensus.
- **Maintaining Decentralization**: Ensure network decentralization with rotating delegate nodes, preserving DPoS's democratic essence.

2 Related Work

2.1 State of art algorithms

The introduction of the proof of work (PoW) mechanism by Cynthia Dwork and Moni Naor in 1993[2], therefore, marked one of the first and most important milestones in blockchain technology evolution. Markus Jakobssonor and Ari Juels stated this term in their 1999 paper[3]. Proof of Work (PoW) is the main use of which was developed by Nakamoto in the year 2008 and Bitcoin system was based over the distributed ledger, which aims to maintain the integrity and as result the reliability of the system[7]. PoW is built on a general principle that any node attempting to solve complex SHA256 mathematical problems and the competition between them. The node which gets the problem solved in the first place acquires the right to append a block to the chain; herewith, the Bitcoin award is granted to that node. The random procedure includes finding a number matching the certain conditions like SHA256 hash with a predetermined number of dozens of zeros before it. With hard gaming difficulty progress, so does energy intake, and therefore, the fear of wasted power consumption arises.

In an attempt to correct the problems posed by PoW, holy grails like proof of stake (PoS) were introduced. The former PPcoin success just gave in the ground of Sunny's proposal of PoS to 2012[4]. Demonstrating its egalitarian nature, PoS seeks to counter unjust distribution of the power of mining, which advantage leads to have greater chances of the mined block creation. In a "coin age" basis, a coin's age contributed to the probability of selecting a block during the above process in PoS. While PoS has limitations, including in business areas that require soft consensuses having short time periods, it still offers consumers and businesses many advantages of cryptocurrencies.

The development of Casper protocol, which is just a revised version of PoS, was carried out in refernce to the later stage, namely Serenity, of Ethereum's development. It introduces a mode of economics a denomination based on the deposit system in which one must stake a deposit to be able to participate in mining. Hostile nodes can in some scenarios incur their deposits reductions as a security measure for the network safety. Although it solve vast PoW issues, PoS denotes as a problem due to economic inequality and interminable bifurcation.

2.2 DPoS

DPoS, being a more efficient alternative to PoS, added some additional improvements and nurtures the technology. DPoS (Delegated Proof of Stake) is an extended concept put forward in the year 2014 by Dan Larimer. It looks into solving some of the obstacles faced by traditional PoWs and PoSs such as inadequate scalability. DPoS adopts a more decentralized and precise democracy and also reduces the difficulty in block validation and consensus.

On the decentralized proof of stake (DPoS) system, a pool of the blockchain network users who owns the tokens will vote to elect several delegates. However, in additional to validation of transactions, creation of new blocks, and maintenance of network integrity, the delegates are selected or ranked based on different configurations depending on the protocol of each specific DPoS blockchain. The number of nodes is predefined, this enhances decentralization while also trying to bypass organization obstacles.

Contrary to conventional PoS for which anyone holding a token is eligible to partake in validation, DPoS allows delegation where holders can entrust their voting rights to selected individuals. Token holders can choose delegates and thus the greater the number of tokens the holder owns, the more powerful vote is cast. This mechanism for voting is empowering as holders who trust and believe the delegates act for the best interest of the network, have a right to vote for the candidate of their choice.

One of the remarkable features of DPoS is the possibility to decrease the transaction verification time up to several seconds as it is the case in other distributed systems. Limited elected delegates speed up the transaction process and they will be able to process and confirm them. Conversely, in the case of PoW and PoS modes of validation where all nodes vouch to add blocks ,the transactions time will be increased.

In this case, the efficiency allows DPoS to get up the game for applications that need to endure in high transaction speeds with an enhanced degree of scalability.

3 Our Work

3.1 Dataset

The dynamic graph is generated *num_steps* number of times, with *num_steps* being the timestamp of each edge. The graph is a directed graph, with edges directed from one node to another. Edges are randomly generated between the nodes. The maximum out degree of each node is fixed to *max_out_degree*. The out degree is also determined by generating a random number of degrees for each node. There is no defined maximum degree for each node; it can reach any number. Therefore, the nodes have a random out degree between the range of 0 to *max_out_degree*, and the edges are drawn randomly between the nodes.

Node Attributes:

- *delegate_reputation*:It indicates the trustworthiness and overall reliability of a node.
- *voter_reputation*: Denotes the reputation score of a node as a voter.Mixed with voter value is apt for predicting the weight of the node.
- *delegate_value*: Indicates the delegate value for a particular consensus.It is the authority value of the node.
- *voter_value*: Represents the value of a node when participating as a voter in a consensus. Similar to *delegate_value*, this influences the node's voting weight when combined with it voter reputation.
- *delegate_curr_flag*: A flag indicating whether a node is currently eligible to be elected as a delegate.
- *delegate_max_flag*: A flag indicating whether a node has reached the maximum number of times it has acted malicious as a delegate.
- *voter_curr_flag*: A flag indicating whether a node is currently eligible to vote in the current round.
- *voter_max_flag*: A flag indicating whether a node has reached the maximum number of times it has voted in a single round.

Edge Attributes:

- *time*: Each edge is labeled with a timestamp indicating the block number or round in which all blocks and validators reached consensus.

3.2 Hubs and Authotity

In our graph, we intend to apply the hub and authority principles[5] to emphasize the importance of the objects to their interaction in the network. Still, we have selected the nontraditional approach with some changes. On our graph every node gives the 'Hub Score' and 'Authority Score' and they are similar to the "voter_value" and "delegate_value," respectively. The 'Hub Score' is a number representing a node being a "hub" in the network, while the 'Authority Score' is an indicator of the "authority" of the node. The computation of these scores is a repeated process that continues until the In the first zone, each node is partners with both the hub and authority scores. On the other hand, during each iteration, the authority score of a node is retrieved. This is calculated by adding the authority scores of all the other nodes that voted for the leading node, which reflects how important this node and its voters are. Then the hub scores are recalculated individually considering the sum of authority scores of the nodes each node voted for, which help position those nodes as a "hub" in the network. Individually, hubs and authorities have different strengths, one of which is trusting one's own expertise and the other's being a source of information. However, this is a cyclical process as hubs update their authority scores and vice versa, which eventually become reinforcing of both types of nodes – hubs and authorities. This approach therefore builds a more complex and deep awareness about what authority score is about for nodes: high authority scores(delegate_value) are translating into the more powerful delegate node candidates for that particular consensus and high hub scores(voter_value) are coming along with frequent connections to higher voting weight.

3.2.1 Our algorithm. In this developed algorithm, it uses the concept of DPoS consensus algorithm and tries to make it 'one ballot, multiple vote' making it more decentralized and reduce the energy wastage. The algorithm is described below:

Each node needs to be eligible to vote in the consensus. The node is eligible to be the voter node if the node has attribute *voter_curr_flag* equal to 0, *voter_max_curr_flag* less than 3, and *voter_reputation* greater than 60.

Each node needs to be eligible to receive votes in the consensus. The node is eligible to be the node to receive votes or to be the delegate node, if the node has attribute *delegate_curr_flag* equal to 0, *delegate_max_curr_flag* less than 3, and *delegate_reputation* greater than 60. In this developed algorithm, it uses the concept of DPoS consensus algorithm and tries to make it 'one ballot, multiple vote' making it more decentralized and reduce the energy wastage. The algorithm is described below:

Each node needs to be eligible to vote in the consensus. The node is eligible to be the voter node if the node has attribute *voter_curr_flag* equal to 0, *voter_max_curr_flag* less than 3, and *voter_reputation* greater than 60.

Each node needs to be eligible to receive votes in the consensus. The node is eligible to be the node to receive votes or to be the delegate node, if the node has attribute *delegate_curr_flag* equal to 0, *delegate_max_curr_flag* less than 3, and *delegate_reputation* greater than 60.

Each eligible voter node can vote to any eligible delegate node active in the blockchain and cast at maximum *max_votes* number of times. Ideally, each node is expected to

cast *max_votes* number of votes, but that is not always the case. The graph generated will be a directed graph, with the edge going from the node that cast the vote to the node that received the vote. After each voting, the Hubs and Authorities algorithm is implemented on the graph with 5 iterations. The hubs score of each node after the 5 iterations is stored in the attribute of the node under the name of *delegate_value*, and the authorities score of each node after the 5 iterations is stored in the attribute of the node under the name of *voter_value*.

The calculations are as follows:
If delegate node does not act malicious:

Delegate reputation and of delegate node will incremented

$$delegate\_reputation = delegate\_reputation + \left[(100 - delegate\_reputation) \times \left(\frac{delegate\_value}{totatldelegate\_value}\right)\right]$$

Voter reputation of node voted delegate node will incremented

$$voter\_reputation = voter\_reputation + \left[(100 - voter\_reputation) \times \left(\frac{voter\_value}{delegate\_valueoddelegatenode}\right)\right]$$

If delegate node act malicious:

Delegate reputation of delegate node will decreased and flags will be incremented

$$delegate\_reputation = delegate\_reputation + \left[(100 - delegate\_reputation) \times \left(\frac{delegate\_value}{totatldelegate\_value}\right)\right]$$

$$max\_delegate\_flag++$$

$$delegate\_curr\_flag += 2^{max\_delegate\_flag}$$

Voter reputation of node voted malicious delegate node will decreased and flags will be incremented

$$voter\_reputation = voter\_reputation + \left[(100 - voter\_reputation) \times \left(\frac{voter\_value}{delegate\_valueoddelegatenode}\right)\right]$$

$$max\_voter\_flag++$$

$$voter\_curr\_flag += 2^{max\_voter\_flag}$$

After the calculation of *delegate_value* of each node, the top *num_of_delegates* number of nodes with respect to *delegate_value* will be selected as delegate nodes.

[1] **Input:** *num_nodes*, *max_out_degree*, *num_steps*
Output: *G*, a directed graph
G ← empty directed graph *attributes* ← dictionary with default attributes *attributes* ← { "delegate_reputation": 60, "voter_reputation": 60, "delegate_value": 1, "voter_value" : 1, "delegate_curr_flag" : 0, "delegate_max_flag" : 0 }
for *i in range(num_nodes)* **do**
end
G.*add_node*(i, ∗∗attributes)**for** *step in range(num_steps)* **do**
end
new_edges ← empty list **for** *node in G.nodes()* **do**
end
out_degree ← random integer between 0 and *max_out_degree*
possible_targets ← list(set(range(*num_nodes*)) - {*node*})
Avoid self-loops *targets* ← random sample from *possible_targets* of size *out_degree* **for** *target in targets* **do**
end
G.nodes[*target*].get('delegate_curr_flag') == 0 *new_edges*.append((*node*, *target*, {'time': *step*})) Add edge with time
G.add_edges_from(*new_edges*)
print("Number of new edges added in step *step*: ", **len**(*new_edges*)) **return** *G*

Algorithm 1: Generate Dynamic Random Directed Graph

3.3 Analysis: Non Malicious Nodes

In an ideal scenario resembling a well-distributed plot, every node, assumed to be non-malicious, should have equal opportunities within a secure and fair blockchain consensus algorithm. Our algorithm achieves this balance. The following bar chart illustrates the frequency of each node being elected as a delegate node and the other bar chart shows the delegate reputation after the 1000th consensus with 450 nodes and all the nodes are non malicious.

3.3.1 Result and Inferences. The described algorithm and seeing the results we can safely say that the algorithm proposed is fair as all the nodes are given equal opportunity to become the delegates when every node is a non malicous node but that is an ideal condition to have, but it serves as ground truth for our algorithm.

3.4 Analysis: Effect of Hubs and Authority

3.4.1 Introduction. In the sphere of network analysis and blockchain consensus algorithms, the hub and authority principles have been used for a long period of time to show that

Algorithm 2: Compute Delegate and Voter Values (Part 1)

```
Input  : Graph G, Number of Steps num_steps
Output : Updated graph G with delegate and voter values
Function ComputeValues(G, num_steps):
    for step in range(num_steps) do
        delegates ← empty list;
        total_delegate_value ← 0.01;
        for i in range(5) do
            for node in G.nodes do
                votes_received = node.delegateValue;
                for predecessor in G.predecessors(node) do
                    if G.has_edge(predecessor, node) and G[predecessor][node]['time'] = step then
                        votes_received+ = G.predecessor. ('delegate_value');
                    end
                end
                G.nodes[node]['delegate_value'] = votes_received;
            end
            for node in G.nodes do
                votes_casted ← G.nodes[node].get('voter_value');
                for successor in G.successors(node) do
                    if G.nodes[node].get('voter_curr_flag') > 0 then
                        votes_casted+ = 0;
                    end
                    else if G.edges[node, successor].('time') = step then
                        if G.successor.('delegate_curr_flag') = 0 then
                            votes_casted+ = G.successor.('delegate_value');
                        end
                    end
                end
                G.nodes[node]['voter_value'] = votes_casted;
            end
        end
```

Algorithm 3: Compute Delegate and Voter Values (Part 2)

```
        end return G
        delegate_values ← {node : G.nodes[node].get('delegate_value') for node in G.nodes()};

        sorted_nodes ← sort nodes based on delegate_values;
        top_5_nodes ← top 5 nodes in sorted_nodes;
        delegates.extend(top_5_nodes);
        malicious_node_id ← [45];
        if step mod 2 = 0 then
            malicious_node_id.extend([179, 178, 180, 181, 182]);
        end
        for delegate in delegates do
            total_delegate_value+ = G.nodes[delegate].get('delegate_value');
        end
        for delegate in delegates do
            if delegate in malicious_node_id then
                G.nodes[delegate]['delegate_max_flag'] ← G.nodes[delegate].get('delegate_max_flag', 0)+ 1;
                G.nodes[delegate]['delegate_curr_flag'] ← 2^{G.nodes[delegate].get('delegate_max_flag')} + 1;
                G.nodes[delegate]['delegate_reputation'] ← G.nodes[delegate].get('delegate_reputation', 1)− (G.nodes[delegate].get('delegate_value') / total_delegate_value × 5);
                for predecessor in G.predecessors(delegate) do
                    if G.[delegate].('delegate_value') ('delegate_value') ≠ 0 then
                        G.predecessor['voter_reputation'] ← G.predecessor.get('voter_reputation', 1)− G.predecessor.('voter_value') / G.delegate.('delegate_value');
                    end
                end
            end
            else
                G.nodes[delegate]['delegate_reputation'] ← G.nodes[delegate].get('delegate_reputation', 1)+ (100 − G.nodes[delegate].get('delegate_reputation', 1))× (G.nodes[delegate].get('delegate_value') / total_delegate_value);
                for predecessor in G.predecessors(delegate) do
                    if G.nodes[delegate].get('delegate_value') ≠ 0 then
                        G.nodes[predecessor]['voter_reputation'] ← G.nodes[predecessor].get('voter_reputation', 1)+ (100 − G.nodes[predecessor].get('voter_reputation', G.nodes[predecessor].get('voter_value') / G.nodes[delegate].get('delegate_value');
                    end
                end
            end
        end
```

Algorithm 4: Update Node Reputations

Input : Graph G, List of delegates *delegates*
Output: Updated graph G with updated reputations
Function `UpdateReputations`(G, *delegates*):
  for *delegate in delegates* **do**
    if *delegate acts malicious* **then**
      Update *delegate_reputation* and *voter_reputation* for *delegate* and voters;
    end
    else
      Update *delegate_reputation* and *voter_reputation* for *delegate* and voters;
    end
  end
  return G;

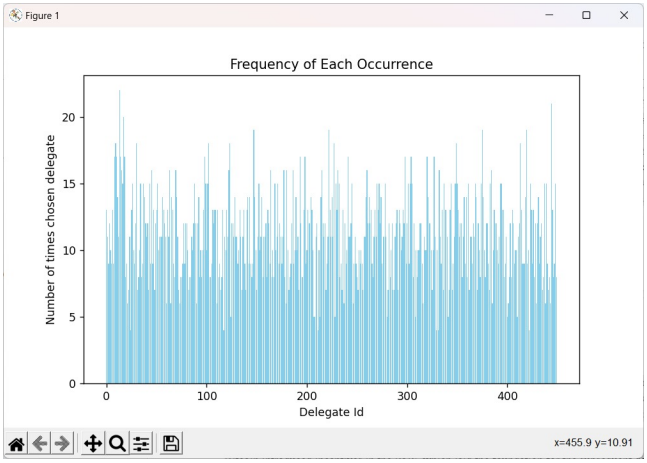

Figure 1. Number of times chosen as Delegate Vs. Delegate Id

the nodes of a network are significant with the numbers of interactions that are taking place in the network. Alternatively, trial and error is my usual modus operandi, acknowledged and expressed as my own metaphor. I am inspired by the famous hubs and authorities theory by Kleinberg [5] and based on the modifications made as per my graph. Inside our network, every node does contribute to both their ' Hub Score' as well as ' Authority Score' which is in the same line with " Voter_Value" and " Delegate_Value" respectively. Incoming links from many different nodes count toward the 'Hub Score', the measure of a node's role as a "hub" in the network. The 'Authority Score' reflects the number of links from nodes that have a high 'Hub Score' and is a measure of the "authority" of the node.

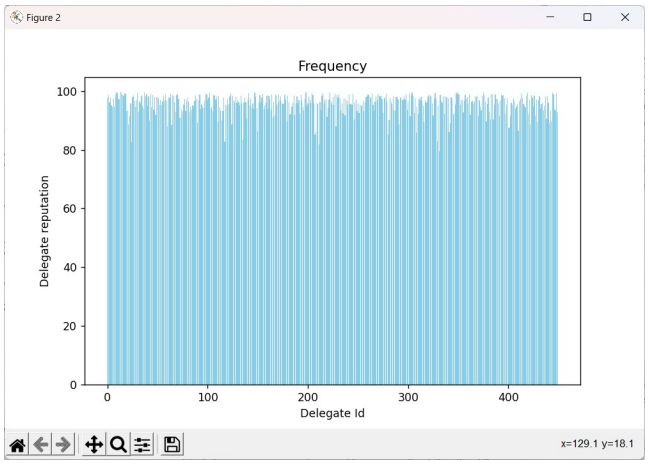

Figure 2. Delegate Reputation Vs. Delegate Id

3.4.2 Methodology. Our mechanism to get these numbers are through a cyclic process that continues until the iterations converge. Firstly, each node gets by default equal Scoring Hubs and Authority Scores. Then, as part of every iteration, the Authority Score of a node is calculated by the count of all votes of all nodes that voted for it. Here influence of a node is represented by the centrality of the node which emphasizes that role of the node and its supporters within the group. This is another factor because the scores attached to the Hubs are summed up based on the sum of the Authority Scores for the nodes that each node voted for. This "being a 'hub" position, within the network hierarchy, shows a node's ability to link different parts of the network together.
The cyclical nature of the algorithm ensures that hubs take turns in updating their authority scores. As authorities adjust the scores of the hubs, hubs on the other hand also take turns in stabilizing both types of nodes. This approach provides a nuanced understanding of the Authority Score for nodes: high authority scores indicate more influential candidates as delegate nodes in each node consensus, while high hub scores indicate node voting links that are more frequent to higher weight voting nodes.

3.4.3 Results. In our simulation experiment, we conducted 1000 consensus rounds on graphs with varying node counts: 150, 400, 800, 1000, and 1200. An intriguing observation emerged as we increased the number of nodes: the sum of delegate values exhibited increasing randomness. This phenomenon can be explained by the fact that in a smaller graph with fewer nodes, the randomness is constrained, resulting in multiple nodes having similar delegate values (authority).

To mitigate this effect and gain more precise information on optimal delegate nodes, we would need to increase the iterations of the Hubs and Authorities algorithm. However,

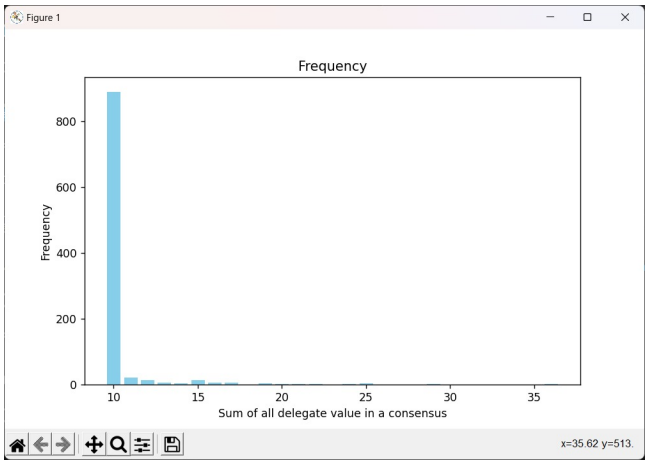

Figure 3. Sum of delegate values of all delgates(150 nodes) Vs. Delegate Id

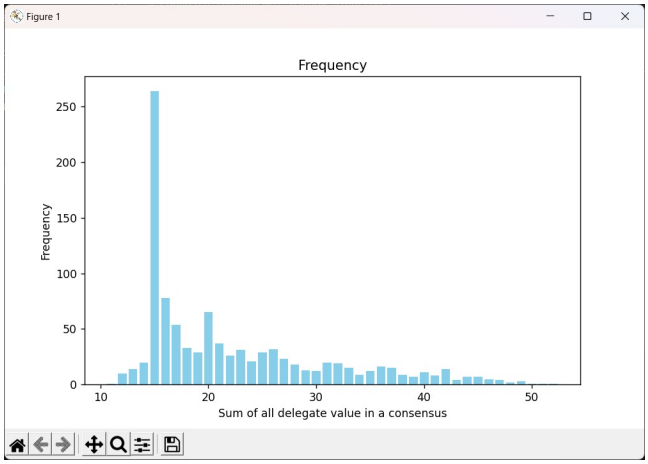

Figure 4. Sum of delegate values of all delgates(400 nodes) Vs. Delegate Id

this escalation comes at the expense of increased algorithmic complexity. It underscores the delicate balance between accuracy and computational efficiency in determining the most suitable delegate nodes for a blockchain consensus.

3.5 Analysis: Comparison between unintentional malicious node and intentional malicious node

3.5.1 Introduction. In DPoS (Delegated Proof of Stake), sometimes, delegates may be penalized for putting forward misconduct, however, such punishment does not always apply to wrongdoing of this kind. Here, deceitful actions might result from different reasons that include arising from network related problems or limited computing ability. In order to tackle this, we also allocate vote integrity as one of our attributes of nodes. Being that the probability of the node failing to get elected is reduced if they have a bad habit even by accident, the power of their votes is not being affected.

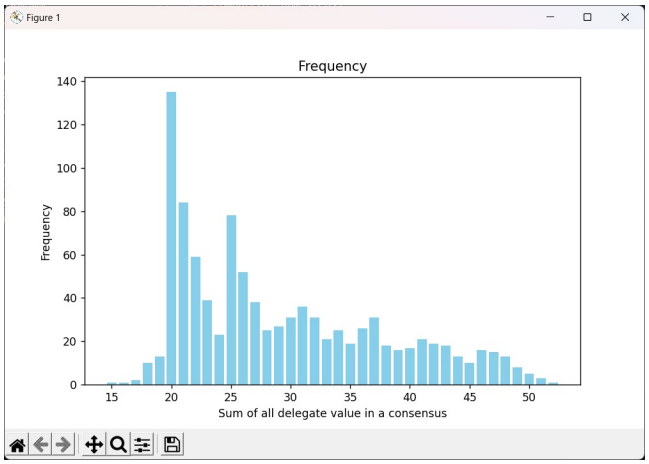

Figure 5. Sum of delegate values of all delgates(800 nodes) Vs. Delegate Id

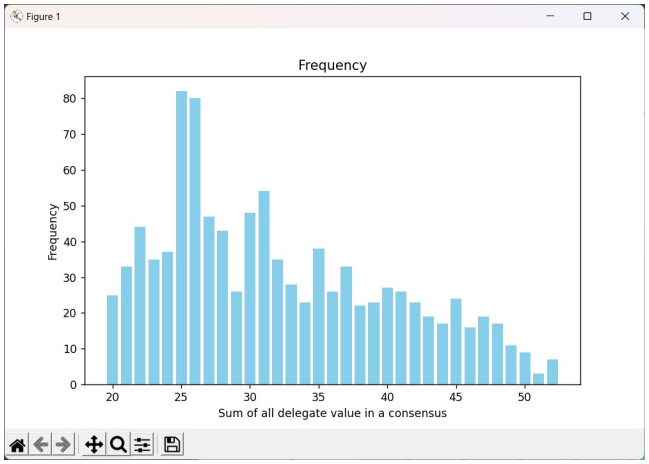

Figure 6. Sum of delegate values of all delgates(1000 nodes) Vs. Delegate Id

3.5.2 Methodology. To illustrate this, let's consider an example with 3 nodes: A, B, and other loyal nodes. Node A is a malicious node, while B unintentionally behaves maliciously. All other nodes are loyal and true. We conducted a simulation and plotted bar charts to visualize the delegate reputations. To do this programmatically, we give out a condition to satisfy which the elected node wont be able to do it every time, thus making it malicious sometimes.Here, A is the node with ID 45, and B, which is unintentionally malicious, is a node with ID 31.

3.5.3 Results. The results show that node A has a delegate reputation lower than 60. In DPoS, if a node's reputation falls below 60 or if it acts maliciously 3 times in a row, it is removed from the blockchain. On the other hand, node B has a delegate reputation lower than the truly loyal nodes but higher than the truly malicious node A.

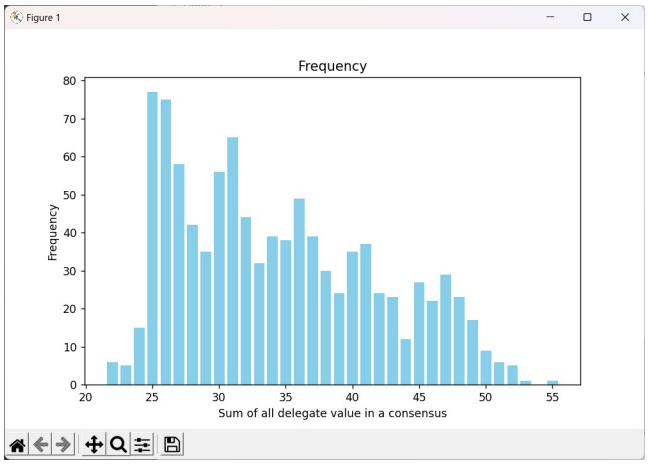

Figure 7. Sum of delegate values of all delgates(1200 nodes) Vs. Delegate Id

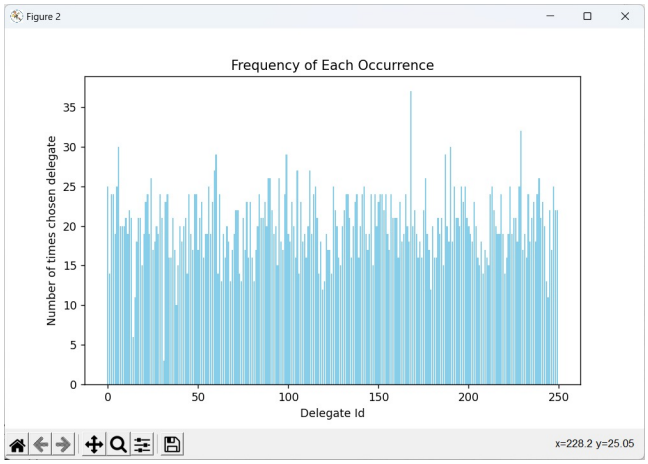

Figure 8. Number of times chosen as Delegate Vs. Delegate Id

We can also analyze the number of times nodes A and B are elected as delegates. Node A, being kicked out of the blockchain due to its low reputation, will not be elected more than 3 times in our simulation. Node B, on the other hand, falls between node A and the other loyal nodes in terms of delegate reputation and thus is likely to be elected more than A but less than the truly loyal nodes.

Now if we want to see the increment of delegate reputation of a loyal node with the consensus rounds proceed.

4 Future Work

Despite modifying the DPoS, some points remain untouched which in real life can be observed, which this dataset is not able to help us analyze, hence there remain several fields or ideas for future research and exploration. Some potential areas for future work include:

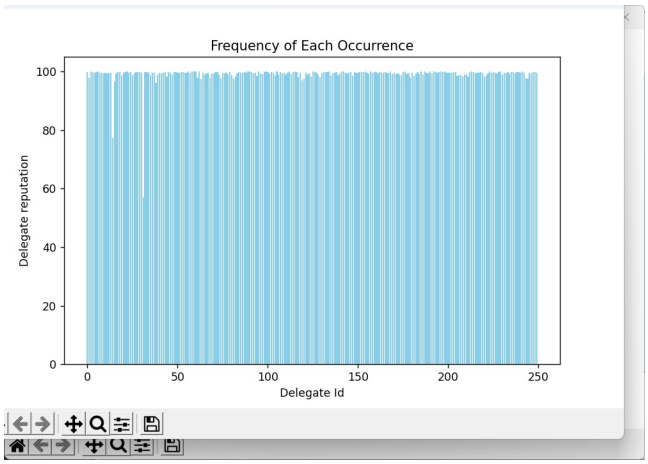

Figure 9. Delegate Reputation Vs. Delegate Id

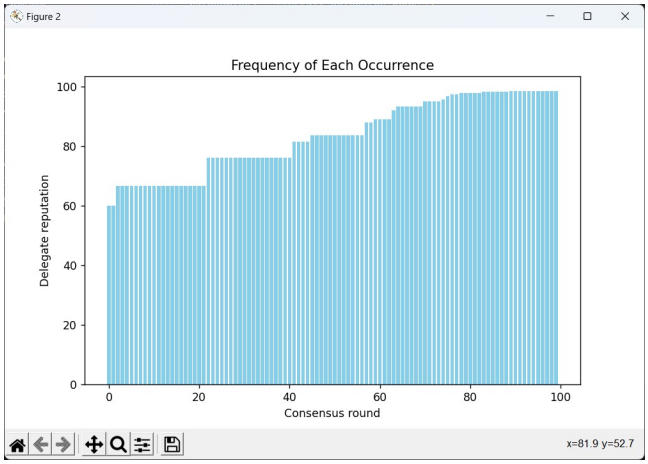

Figure 10. Delegate Reputation of non malicious node Vs. Consensus Round

Furthermore, this dataset can be made more dynamic by writing much more complicated codes. The further work on the dataset can be done by forming the general expression which provides us with the safety from the 'n' number of malicious nodes colluding to attack the blockchain. This expression must reveal the values of the percentage by which the reputation of malicious nodes and the nodes voting malicious nodes be reduced such that the attack of 'n' nodes can be prevented beforehand. Also, here we need to find what the value of curr and max flags be set to further make the algorithm more safer from attacks. Additionally, we can also find out how many nodes were in the cluster of malicious activity and putting the required values in the equation obtained can dynamically change the flags and increment and decrement pattern of reputation on each consensus.

The other functionality that can be added to the algorithm is to identify how or up to what extent the malicious activity was committed, and accordingly penalize the nodes. This

gives the node a better chance to be in consensus without impacting much on its reputations as its intensity of malicious behaviour was low. While it effectively reduces the reputations or bans the node carrying out the malicious activity of high intensity more faster.

By pursuing these ideas for future work, we can contribute to a deeper understanding of social network dynamics and consensus algorithms in blockchain, and foster a more inclusive and trustworthy blockchain environment giving incentives to nodes for electing the right nodes for block generation.

References

[1] Bin Cao, Zhenghui Zhang, Daquan Feng, Shengli Zhang, Lei Zhang, Mugen Peng, and Yun Li. 2020. Performance analysis and comparison of PoW, PoS and DAG based blockchains. *Digital Communications and Networks* 6, 4 (2020), 480–485.

[2] Cynthia Dwork and Moni Naor. 1993. Pricing via Processing or Combatting Junk Mail. In *Advances in Cryptology — CRYPTO' 92*, Ernest F. Brickell (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 139–147.

[3] Markus Jakobsson and Ari Juels. 1999. Proofs of work and bread pudding protocols. In *Secure Information Networks: Communications and Multimedia Security IFIP TC6/TC11 Joint Working Conference on Communications and Multimedia Security (CMS'99) September 20–21, 1999, Leuven, Belgium*. Springer, 258–272.

[4] Sunny King and Scott Nadal. 2012. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *self-published paper, August* 19, 1 (2012).

[5] Jon M Kleinberg. 1999. Hubs, authorities, and communities. *ACM computing surveys (CSUR)* 31, 4es (1999), 5–es.

[6] Chao Li, Balaji Palanisamy, Runhua Xu, Li Duan, Jiqiang Liu, and Wei Wang. 2023. How hard is takeover in dpos blockchains? understanding the security of coin-based voting governance. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*. 150–164.

[7] Satoshi Nakamoto and A Bitcoin. 2008. A peer-to-peer electronic cash system. *Bitcoin.–URL: https://bitcoin. org/bitcoin. pdf* 4, 2 (2008), 15.

[8] Fabian Schuh and Daniel Larimer. 2017. Bitshares 2.0: general overview. *accessed June-2017.[Online]. Available: http://docs. bitshares. org/downloads/bitshares-general. pdf* (2017).

[9] Brent Xu, Dhruv Luthra, Zak Cole, and Nate Blakely. 2018. EOS: An architectural, performance, and economic analysis. *Retrieved June* 11, 2019 (2018), 41.