Delegation V2

Date: 27-02-2023

Auditor: Shard Labs

Hash: 3bfd49cacce135f88f97bd7ee47b68cfa2c8c5e8

Summary

This report examines how the current delegation algorithm selects node operators before delegating. There are 2 edge cases where the selection doesn't return an optimized solution. The re-balance feature still works but could be improved.

Case 1

Summary

The algorithm can ignore that the protocol is unbalanced and do a balanced distribution if the node operators' stakes are close to the average stake. As a result, the delegated tokens will be distributed equally between the node operators.

Example

```
// the distance between the big node stake and the small node to say the protocol is balanced
DistanceThresholdPercentage = 120
// the nodes' stake
validators_stake = [1000, 1200, 1250, 950]
// the amount that we want to delegate to the nodes
delegation = 200
// calculate the average stake
average_stake = (∑ validators_stake[i] + delegation) / number_of_validators
average_stake = (1000 + 1200 + 1220 + 980 + 200) / 4
average_stake = 1150
// check if the protocol is balanced
// (validator index 2 stake = 1250)
// (validator index 3 stake = 980)
currentDistanceThreshold = 1250 * 100 / 980 = 127
isBalanced = currentDistanceThreshold > DistanceThresholdPercentage
isBalanced = false // the protocol is not balanced

// Calculate the validators' ratio to distribute the delegations to the nodes with less stake
// case1: if the `validators_ratios` is an array of ZEROs it means the protocol is balanced
//        and we have to distribute the delegations equally between the nodes.
//        example: validators_ratios = [0, 0, 0, 0]
// case2: if the `validators_ratios` is an array with percentages it means the protocol
//        is not balanced and we have to distribute the delegations using the ratios.
//        example: validators_ratios = [20%, 0, 10%, 70%]
```

```
validators_ratios = [0, 0, 0, 0]
// The protocol is not balanced, so we have to calculate the validators' delegation ratio.
// if the validator's stake is greater than the **average_stake** it get's ignored.
// The validator at index 1 and 2 are ignored because **average_stake < validator_stake**.
// we will focus only on the validators at index 0 and 3
// before we check if the validator is close to the balance state.
distance_threshold_to_average_stake[i] = average_stake * 100 / validators_stake[i]
V1_distance_threshold_to_average_stake[1] = 1150 * 100 / 1000 = 115
V4_distance_threshold_to_average_stake[4] = 1150 * 100 / 980 = 117
// the validator v1 and v2 will be considered balanced because the
// **distance_threshold_to_average_stake < DistanceThresholdPercentage**
V1_distance_threshold_to_average_stake[1] = 115 < 120
V1_distance_threshold_to_average_stake[1] = 117 < 120

// As a result we ignored all the validators and we return validators_ratios = [0, 0, 0, 0]
validators_ratios = [0, 0, 0, 0]
```

Conclusion

Even if the protocol is not balanced the output ratios return a balanced protocol. As result, the delegation will be split between all the validators (balanced).

Case 2

```

// the distance between the big node's stake and the small node's stake to say the protocol is balanced
DistanceThresholdPercentage = 120
// the nodes' stake
validators_stake = [1000, 1200, 1300, 900]
// the amount that we want to delegate to the nodes
delegation = 200
// calculate the average stake
average_stake = (∑ validators_stake[i] + delegation) / number_of_validators
average_stake = (1000 + 1200 + 1300 + 900 + 200) / 4
average_stake = 1150
// check if the protocol is balanced
// (validator index 2 stake = 1300)
// (validator index 3 stake = 900)
currentDistanceThreshold = 1250 * 100 / 950 = 144
// Check if the protocol is balanced.
isBalanced = currentDistanceThreshold > DistanceThresholdPercentage
isBalanced = false // the protocol is not balanced

// Calculate the validators' ratio to distribute the delegations to the nodes with less stake
// case1: if the `validators_ratios` is an array of ZEROs it means the protocol is balanced
//        and we have to distribute the delegations equally between the nodes.
//        example: validators_ratios = [0, 0, 0, 0]
// case2: if the `validators_ratios` is an array with percentages it means the protocol
//        is not balanced and we have to distribute the delegations using the ratios.
//        example: validators_ratios = [20%, 0, 10%, 70%]

// The protocol is not balanced, so we have to calculate the validators' delegation ratio.
// if the validator's stake is great than the **average_stake** it get's ignored.
// The validator at index 1 and 2 are ignored.
// we will delegate only to the validators at index 0 and 3
// before checking if the validator is close to the balance state.
validators_ratios = [0, 0, 0, 0]

distance_threshold_to_average_stake[i] = average_stake * 100 / validators_stake[i]
V1_distance_threshold_to_average_stake[1] = 1150 * 100 / 1000 = 115
V4_distance_threshold_to_average_stake[4] = 1150 * 100 / 900 = 127
// the validator v1 will be considered balanced because the
// **distance_threshold_to_average_stake <= DistanceThresholdPercentage**

// As a result we will delegate to one node operator validators_ratios = [0, 0, 0, 100%]
validator_ratios = [0, 0, 0, 100%]
// Example:
// a similar case happened on mainnet
// https://etherscan.io/tx/0xa78af256b1f2eb264a4e43a1edd57b9864133eedb3eb92105a372073f7fd3684
```

Conclusion

In some cases, one validator can receive the total delegation. It could be a bad case if the delegation is large.

Solutions

- The solution to fix the above edge cases is to remove the following block code https://github.com/lidofinance/polygon-contracts/blob/main/contracts/NodeOperatorRegistry.sol#L543-L548

Why did we decide to add this block?

When designing the delegation algorithm we tried to solve the issue of gas cost for executing the delegate transaction due to estimated size of the node operator set. So, if those node operators are not balanced this block code can help to reduce their number by ignoring the nodes which are close to getting balanced and avoid going out of gas. This algorithm has some edge cases where in some situations it will consume more gas than expected for example, in case one the protocol is not balanced but this block code ignores the unbalanced nodes because they are close to getting balanced, then the delegation is distributed to all the nodes (more gas).

Apply this fix to both cases:

```
// the distance between the big node stake and the small node to say the protocol is balanced
DistanceThresholdPercentage = 120
// the nodes' stake
validators_stake = [1000, 1200, 1250, 950]
// the amount that we want to delegate to the nodes
delegation = 200
// calculate the average stake
```

```
average_stake = (∑ validators_stake[i] + delegation) / number_of_validators
average_stake = (1000 + 1200 + 1220 + 980 + 200) / 4
average_stake = 1150
// check if the protocol is balanced
// (validator index 2 stake = 1250)
// (validator index 3 stake = 980)
currentDistanceThreshold = 1250 * 100 / 980 = 127
isBalanced = currentDistanceThreshold > DistanceThresholdPercentage
isBalanced = false // the protocol is not balanced

// Calculate the validators' ratio to distribute the delegations to the nodes with less stake
// case1: if the `validators_ratios` is an array of ZEROs it means the protocol is balanced
//        and we have to distribute the delegations equally between the nodes.
//        example: validators_ratios = [0, 0, 0, 0]
// case2: if the `validators_ratios` is an array with percentages it means the protocol
//        is not balanced and we have to distribute the delegations using the ratios.
//        example: validators_ratios = [20%, 0, 10%, 70%]

validators_ratios = [0, 0, 0, 0]
// The protocol is not balanced, so we have to calculate the validators' delegation ratio.
// if the validator's stake is great than the **average_stake** it get's ignored.
// The validator at indexes 1 and 2 are ignored because **average_stake < validator_stake**.
// we will focus only on the validators at index 0 and 3

// before we check if the validator is close to the balance state.
distance_threshold_to_average_stake[i] = average_stake * 100 / validators_stake[i]
V1_distance_threshold_to_average_stake[1] = 1150 * 100 / 1000 = 115
V4_distance_threshold_to_average_stake[4] = 1150 * 100 / 980 = 117
// the validator v1 and v2 will be considered balanced because the
// **distance_threshold_to_average_stake < DistanceThresholdPercentage**
V1_distance_threshold_to_average_stake[1] = 115 < 120
V1_distance_threshold_to_average_stake[1] = 117 < 120

// As a result the selected nodes at indexes 0 and 3
validators_ratios = [37.5, 0, 0, 62.5%]
```

Conclusion

With this solution, we improved the delegation and we have an optimized delegation.