Decentralized Validator Sampling

Motivation

All BandChain validators have an additional responsibility other than voting and proposing new blocks. That is submitting reports for incoming oracle requests. However, asking all validators to retrieve results and waiting for their reports for every request is time inefficient, since this would become a single queue for requests bottlenecking the whole system of BandChain. Since there are many validators on BandChain, we can process multiple requests simultaneously and mitigate the bottleneck effect by distributing each oracle request to a random subset of the validators.

The randomized selection is based on the voting power of validators. The underlying reason is the same as that of Proof-of-Stake concept. Validators with higher voting powers are those that have staked more BAND token. Due to the higher commitment to risks, they are incentivized to be good actors and to deliver the correct oracle results within the expected time interval.

Those with lower voting power have less risk exposure and thus are generally less incentivized to be a good actor as they do not have much to lose. Nevertheless, we still want every validator to have a chance in participating in the oracle retrieving result task. Otherwise, the platform would be centralized around few validators with high voting powers, undermining the very goal of any blockchain -- decentralization.

Therefore, when determining which algorithm to use in selecting the validators to respond to an oracle data request, there are two main characteristics that we have incorporated:

1. A validator with a higher voting power should have a higher chance of being chosen than one with a lower voting power
2. Every validator should still have a chance to be selected

Validators

From the pool of all validators, we have to select a specific set of validators from that pool. To do this, we make use of a random number generator.

Random Number and Seed Generation

As with most random number generator, our number generation proces require the use of a seed. In this specific case, our seed comprises of:

- A list ofblockHashes
- TherequestID
- of the request the validators are being chosen for
- chainID
- of BandChain

In the case of the list ofblockHashes , we will use the blockHashes of the previous n blocks (currently we use 32). We then take\frac{32}{n} bytes from each of the hashes and concatenate them. The purpose of this is to prevent any potentially malicious validators from influencing the entire seed in the turn they have to propose the block. Even if they attempt to intentionally construct certainblockHashes in their proposed block, they can only control \frac{n}{32} of the seed. Finally, we concatenate the result fromblockHashes withrequestID andchainID to create the seed.

Manipulation Resistant Seed Generation

After we have used our concatenated seed to generate a random number, we then use that value to select the validator for that round.

To do so, we again assume that the validators in the selection space is sorted in descending order of voting power (as shown below from validator with 100 voting power, to 85, then to 70 and so on.)

Then, we imagine that we have a cumulative scale running across that list, with the values being the validator's voting power. For example:

where k > l > m

{1, k} \hspace{90pt} is assigned to the first validator in line with k voting power {k+1, k+l} \hspace{57pt} to the second in line with l power {(k+l)+1, (k+l)+m} \hspace{8pt}to the third in line with m power . . . and so forth.

With that, the specific range in which our random number falls in along that cumulative scale determine which validator is ultimately chosen for that round. A visual representation of this method is shown below.

Setup

Selection using a Random Number

Previous Cosmos IBC Integration Next Lite Client Protocol