I did a simulation of partial slashing with agent-based modeling methodology, and I found that partial slashing may not work well as it is intended.

Condition for joining/leaving a pool

As far as I understood, partial slashing is designed to make people join smaller pools. In my simulation, I designed the agents to join a pool if a possible slashing fraction

of the pool is smaller than their risk taking trait

, and to leave the pool if the possible slashing fraction

became larger than the risk taking trait

. When the next formula is satisfied, the agent joins and stays at the pool.

risk_taking_trait > fault_possibility * partial_slashing_factor * pool's_deposit / system_total_deposits

Assumptions behind the parameters are like below:

- risk_taking_trait

: It is agent's unique trait that explains how much this agent can endure the loss (how much share of their asset). It's in inverse proportion to the agent's balance, and I added some random number with the range of [0, 0.1)

- fault_possibility

: A probability that an error(fault) can occur for the pool that the agent is in. I set this as a constant value, and every agent knows this value. This variable can be seen as every agent's assumption about the possibility of the fault occurrence.

- partial_slashing_factor

: In reality, the partial slashing fraction gets affected by other validator who got slashed recently. However, I assumed that the agent's pool gets slashed alone, for the simplicity of the simulation model. It might be reasonable if the purpose of the partial slashing is make staking pool smaller, because we cannot do any special things if a catastrophical event or a coordinated attack happens to large numbers of validators at the same time. Therefore, according to the current implementation, partial_slashing_factor

is 3.

From the last formula, we can replace pool's_deposit / system_total_deposits

with share_of_pool

, put 3

for partial_slashing_factor

, and we get the simpler formula.

share_of_pool < risk_taking_trait / (3 * fault_possibility)

risk_taking_trait

would be close to 0.05 in my assumption stated above, and I assumed fault_possibility

as 1%. With this assumption, the formula changes like below.

share_of_pool < 1.666…

As this formula is always true, so none of the agents wants to leave the pool in any cases!

How to solve this

I know that the assumption can be problematic, especially for the fault_possibility

. However, 1% of the possibility can be a reasonable assumption in the whole ecosystem's aspect(how many attacks have we seen in the timeline of the crypto space? How about for each of ourselves?). Furthermore, we can think about the cases of crypto exchanges. Many investors just put their tokens in the wallet of the exchanges, even though they read news about attacks on exchanges and exit scam. Also, even if we decided to set the fault_possibility

higher for the simulation, the risk_taking_trait

can also be higher in reality. We saw 40% price changes for a few days period in crypto space quiet frequently.

To solve this situation, we might take three approaches (as we cannot change risk_taking_trait

. Oh, can we do that by educating some investment strategy?).

- Set partial_slashing_factor

higher

- Make fault_possibility

recognized by each agent higher

- Develop and promote decentralized pool

Let's see one by one.

1. Set partial_slashing_factor

higher

According to Vitalik's explanation, it is changed from 6 to 3 as the slashing interval doubled. I think this 3 and 6 is related with byzantine fault tolerance rate, but anyway I think we can use 6 again while we using doubled interval. It is because the centralization can happen, with the deposit share higher than 16.666%.

When the pools start to appear, each pool will have a kind of reputation. There would be a first-mover advantage, longer running time without any faults, and so on. We may see some big companies (like bank) start to run staking pools using their reputations in the real world. Tech people can run their own node, but most of the people are not familiar with tech. In the future that ethereum got mass adoption, many non-tech people may want to stake their ethers. In that case, will they choose smaller pools as it is "decentralized"? I think they will choose a pool backed by a reliable company with a solid reputation, and we may see higher deposit share than we expected. Therefore, I think it would be better to put a higher hurdle in advance to suppress that kind of centralization.

2. Make fault_possibility

recognized by each agent higher

In my simulation, I tried to see the effect of knowing correct information about the risk. The difference between Model 3 and Model 3' is choosing the pool to join. In Model 3, each agent watches only three pools randomly and calculate its possible slashing fraction. In comparison with this, In Model 3', each agent choose the pool with the lowest risk (pool with the smallest deposit). As a result, Model 3' showed more evenly distributed deposits among the pools than Model 3.

It can be interpreted as we need a service that shows the risk of each pool, helping people to choose the pool with the least risk. Some UX research related to a function for wallets/clients that help people choose a pool could be needed.

3. Develop and promote decentralized pool

We all know that decentralized pool is better. But it might not to unfamiliar-with-tech people. Therefore, we need decent implementations of decentralized pool and we have to make decentralized pool popular. I know that casper team is implementing a staking pool mvp, and it will be better if we can get more people involved to build more implementations of the decentralized pool.

Add to this, we may consider next points to get better result.

- Cost to change the pool(financial, physical, psychological - people have inertia)
- Compensation policy of centralized pools when they got slashed
- Policy of pools about leaving the pool (delay, fee, etc.)
- Fee taken by each pool

Thank you for reading!