Protocol and staking pool changes that could improve decentralization and reduce consensus overhead

Special thanks to Mike Neuder, Justin Drake and others for feedback and review. See also: earlier posts on similar themes by Mike Neuder, Dankrad Feist and arixon.eth.

The Ethereum status quo can be described as including a large share of emergent **two-tiered staking**. By two-tiered staking, I mean here a staking model where there are two classes of participants:

1. **Node operators**, who run nodes and put up either their reputation or some fixed amount of capital of their own as collateral
2. **Delegators**, who put up some quantity of ETH, with no minimum commitment and no strict *requirement* to participate in any other way beyond bringing their collateral

This emergent two-tiered staking arises through the actions of a large share of stakers who participate in staking pools which offer **liquid staking tokens (LSTs)**, eg. Rocket Pool and Lido.

The status quo has two main flaws:

- **Centralization risk in node operators**. The mechanisms for choosing node operators in existing staking pools are either not very decentralized or have other flaws.
- **Needless consensus layer burden**. The Ethereum L1 verifies ~800,000 signatures per epoch, and with single slot finality that may increase to 800,000 *per slot*. This is a large load. Furthermore, because of the large share of liquid staking, the network is not getting the full benefits of taking on this load. If the network can be made acceptably decentralized and safe without requiring each staker to sign off in each slot, then we could lean on that solution much harder, and reduce the number of per-slot signatures to eg. 10,000.

This post will describe possible solutions to both of these issues. It will particularly take the following angle **suppose that we take it as a given that most capital is held by people who are not willing to personally run staking nodes in their current form, signing messages every slot and locking their deposits and subjecting them to slashing. What other role can they have that meaningfully contributes to the decentralization and security of the network?**

How does two-tiered staking work today?

The two most popular decentralized staking pools today, Lido and RocketPool, are both creating emergent two-tiered staking ecosystems. In Lido's case, the tiers are:

- **Node operators**: these get chosen by a vote in the Lido DAO, so effectively by LDO holders
- **Delegators**: people holding stETH. stETH is created when someone deposits ETH into the Lido smart contract system, which allows node operators to stake it (but, because the withdrawal credentials are bound to a smart contract ETH address, they cannot take it for themselves)

In Rocket Pool's case, the tiers are:

- **Node operators**: anyone can become a node operator by submitting an 8 ETH deposit, plus an amount of RPL tokens
- **Delegators**: people holding rETH. rETH is created when someone deposits ETH into the Rocket Pool smart contract system, which allows node operators to stake it (but not take it for themselves)

The role of delegators

In these systems (or new systems enabled by potential future protocol changes), one key question to ask is: **from the protocol's perspective, what is the point of having delegators at all?**

To see why the question is meaningful, let us consider the following world. The protocol change proposed in this recent post, of limiting slashing penalties to 2 ETH, is implemented. Rocket Pool reduces the node operator deposit to 2 ETH in response. The market share of Rocket Pool increases to 100% (not just among stakers, but also among *ETH holders*: as rETH becomes risk-free, almost all ETH holders become rETH holders or node operators).

Let's suppose that rETH holders get a return of 3% (including both in-protocol rewards and priority fees + MEV) and node operators get a return of 4%. Let's also suppose that the total supply of ETH is 100 million.

Here's how the math works out. To avoid dealing with compounding, we'll look at *daily* returns instead of yearly, so that second-order terms become small enough to be ignorable:

- ETH staked by node operators: $100M * \frac{2}{32} = 6.25M$
- ETH staked in rETH: $100M - 6.25M = 93.75M$
- Node operator daily nominal return: $\frac{0.04}{365.25} = 0.01095\%$
- Delegator nominal returns: $\frac{0.03}{365.25} = 0.00821\%$
- ETH supply growth: $6.25M * 0.0001095 + 93.75M * 0.0000821 \approx 8380$
- Change in % of total ETH supply held by a node operator: $\frac{1.0001095}{1.0000838} - 1 = 0.00257\%$
- Change in % of total ETH supply held by a delegator: $\frac{1.0000821}{1.0000838} - 1 = -0.00017\%$

Now, let us consider a different world. Rocket Pool does not exist. Each staker's minimum deposit is reduced to 2 ETH, and the total quantity of staked ETH is capped at 6.25M. Also, the node operator's return is decreased to 1%. Let's do the math:

- ETH staked: $6.25M$
- Staker nominal daily returns: $\frac{0.01}{365.25} = 0.00274\%$
- ETH supply growth: $6.25M * 0.0000274 \approx 170$
- Change in % of total ETH supply held by a staker: $\frac{1.0000274}{1.0000017} - 1 = 0.00257\%$
- Change in % of total ETH supply held by a non-staker: $\frac{1}{1.0000017} - 1 = -0.00017\%$

Now, let us consider the two situations from a cost-of-attack perspective. In the first case, attackers would not sign up as delegators: delegators have no power, so there is no point. Hence, they would put all of their ETH toward signing up as node operators. To get to 1/3 of all the stake, they would need to put in 2.08M ETH (which, to be fair, is still quite a lot! eg. see this discussion on super-committees, a staking scaling proposal which would have also decreased the cost of attack to a similar value). In the second case, attackers would just stake, and to get to 1/3 of all the stake, they would need to put in... 2.08M ETH.

From both a staking economics perspective, and a cost-of-attack perspective, the final outcome in both cases is exactly the same. The share of the total ETH supply held by a node operator increases by 0.00256% per day, and the share of the total ETH supply held by a non-node-operator decreases by 0.00017% per day. The cost of attack is 2.08 million ETH. Hence, it feels like **in this model delegation becomes a pointless Rube Goldberg machine: we might as well cut out the middleman, and drastically reduce staking rewards and cap total ETH staked to 6.25M**.

The purpose of this argument is not to advocate reducing staking rewards by 4x and capping total staked ETH to 6.25m. Rather, it is to point out a key property that a well-functioning staking system should have: namely, **delegators should be doing *something* that actually matters**. Furthermore, it's okay if delegators are motivated to act correctly to a large degree by community pressure and altruism; after all, that's the main force that is motivating people to stake in decentralized security-increasing (but higher-effort) ways rather than centralized security-threatening (but lower-effort) ways today.

If delegators *can* have a meaningful role, what might that role be?

I see two classes of answers:

- **Delegate selection**: delegates can choose which node operators they delegate their stake to. Node operators would have a "weight" in consensus which is proportional to the total stake delegated to them. Delegate selection already exists in a limited form today, in the sense that rETH or stETH holders can withdraw their ETH and switch to a different pool, but the practical availability of delegate choice could be greatly improved.
- **Consensus participation**: delegators could be given the option to take on a role to play in consensus, which would be "lighter" than full staking and not subject to long withdrawal periods and slashing risk, but would still function as a check on node operators. Many delegators would not want to do this, and would prefer the simplest-of-all interface of holding an ERC20 and doing nothing else, but some would take up the option.

Expanding delegate selection powers

There are three ways to expand delegate selection powers:

- **Better voting tools within pools**
- **More competition between pools**
- **Enshrined delegation**

Voting within pools does not really exist today: in Rocket Pool, anyone can become a node operator, and in Lido, it's LDO

holders that vote, not ETH holders. Lido has a proposal for LDO + stETH dual governance, which would give stETH holders a say in the sense that they could activate a gadget that blocks new votes and hence blocks node operators from being added or removed. That said, this is limited, and could be much stronger.

Cross-pool competition exists today, but is weak. The main challenge is that smaller staking pools' staking tokens are (i) less liquid, (ii) harder to trust, and (iii) less supported by applications.

We can improve on the first two issues by **capping slashing penalties to a smaller amount, eg. 2 or 4 ETH** The remaining (unslashable) ETH could then be safely deposited and withdrawn instantly, making an LST based on that ETH two-way convertible with ETH even for the smallest pools. We could improve on the third issue by creating a central issuing contract for LSTs - somewhat similar to ERC-4337 and ERC-6900 for wallets, such that we can guarantee that any staking token issued through that contract is safe. Applications (eg. a version of RAI that supports staked ETH) could be strongly encouraged to support *all* staking tokens issued through this registry.

Enshrined delegation currently does not exist in-protocol, but could potentially be introduced. It would involve similar logic to the ideas above, but implemented at protocol level. See this post for pros and cons of enshrining things.

All of these ideas are an improvement over the status quo, but there is a limit to how much benefit they can provide. **Token voting governance sucks, and ultimately any form of unincentivized delegate selection is just a type of token voting; this has been my main source of discomfort with delegated proof of stake since the beginning**. Hence, it seems valuable to *also* think about enabling stronger forms of consensus participation.

Consensus participation

There are limits to the current approach to solo staking, even without taking the current issues around liquid staking into account. Assuming single-slot finality, our best estimates suggest a limit of ~100k - 1M BLS signatures that could be processed per slot, and that's assuming a significant increase in slot time. Even if we use recursive SNARKs to aggregate signatures, **signing accountability (for slashing purposes) requires a bitfield of who participated to be available for each signature**. If Ethereum becomes a global-scale network, then even somehow using full danksharding to store the bitfields would not be enough: 16 MB per slot would only support ~64 million stakers.

Here, from that perspective as well, there is value in splitting up staking into a higher-complexity slashable tier, which acts every slot but perhaps only has 10,000 participants, and a lower-complexity tier that only gets called up to participate occasionally. The lower-complexity tier could be exempt from slashing entirely, or it could randomly give its participants opportunities to temporarily (ie. for a few slots) deposit and become subject to slashing.

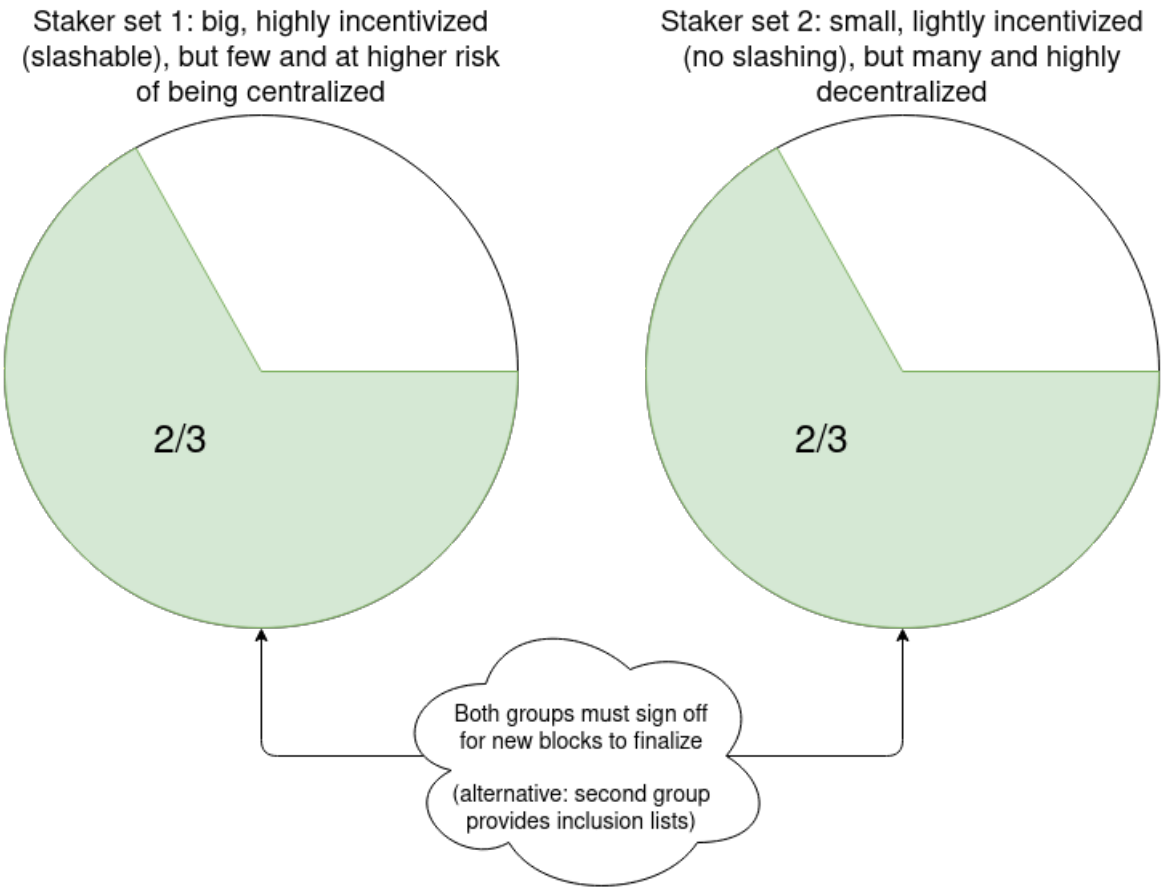
In practice, this could be implemented by raising the validator balance cap, and later implementing a balance threshold (eg. 2048 ETH) to determine which existing validators enter the higher vs lower complexity tier.

Here are a few ideas for how these **small-staking roles** could work:

- Each slot, 10000 small-stakers are randomly chosen, and they can sign off on what they think is the head of that slot. An LMD GHOST fork choice rule is run using the small-stakers as input. If the **small-staker-driven fork choice** and the **node operator-driven fork choice** ever diverge, the user's client does *not* accept any block as finalized, and displays an error. This forces the community to mediate the situation.
- A delegator can send a transaction declaring to the network that they are online and are willing to serve as a small-staker for the next hour. For a message (block or attestation) from a node to count, **both the node and a randomly selected delegator must sign off**.
- A delegator can send a transaction declaring to the network that they are online and are willing to serve as a small-staker for the next hour. Each epoch, 10 random delegators are chosen as **inclusion list providers**, and 10000 more are chosen as voters. These are chosen k slots in advance, and given a k-slot window to publish a message on chain confirming that they are online. Each confirmed chosen inclusion list provider can publish an inclusion list, and a block is invalid unless for each inclusion list, it either (i) it contains the transactions in that inclusion list, or (ii) it contains votes from 1/2 of chosen voters showing that the inclusion list is unavailable.

These small-staking roles all have in common that they do not involve actively participating in each slot, are not slashable (and so have very low key management risk), and are very "light" in the sense that **they do not even require a full node to run**. Verifying the consensus layer *only* would be sufficient. Hence, they could be implemented through apps or browser plugins that are mostly passive and have very low computational overhead, hardware requirements or technical know-how requirements, all *without* even assuming technical advances like ZK-EVMs.

These small-staking roles also all have a common goal: **they prevent a 51% majority of node operators from engaging in transaction censorship**. The first and second also prevent a majority from engaging in finality reversion. The third focuses more directly on censorship, though it's more vulnerable to the possibility that a node operator majority will also choose to censor inclusion list provider confirmation messages.



These ideas were written from the perspective of an enshrined two-tiered staking solution implemented in protocol, but they could also be implemented as staking pool features. Here are some concrete implementation ideas:

- At protocol level, each validator is allowed to specify *two* staking keys: a ***persistent* staking key P**, and an *Ethereum address* which, when called, outputs a ***quick* staking key Q**. Nodes track the fork choice of messages signed by P-keys *and* messages signed by Q-keys; if the two disagree, they do not accept any block as finalized. Staking pools are responsible for randomly selecting delegators as the Q-key holders for the current slot.
- Alternatively, the protocol could stay mostly-the-same but the staking public key for that validator for that slot would be set to $P + Q$. Note that for slashing, two slashable messages could have different Q keys, but they would have the same P key; the slashing design would need to handle this.
- Alternatively, the Q-keys could *only* be used in-protocol to sign and verify the inclusion list in the block. In this case, Q could be a smart contract rather than a single key, and so pools could use it to implement more complex voting logic that accepts either an inclusion list from a randomly chosen provider or a sufficiently large number of votes saying that the inclusion list is not available.

Conclusions

If done right, tweaks to the staking design could solve two birds with one stone:

1. Give people who do not have the resources or capability to solo-stake today an opportunity to participate in staking that keeps more power in their hands: both **(i) power to choose which nodes** they are supporting and **(ii) power to actively participate in consensus** in some way that's lighter than full staking-node operation but still meaningful. Not all participants would take either or both options, but any that do would significantly improve things relative to the status quo.
2. **Reduce the number of signatures** that the Ethereum consensus layer needs to process in each slot, even in a single-slot-finality regime, to a smaller number like ~10,000. This would also aid decentralization, by making it much easier for everyone to run a validating node.

For a lot of these solutions, there are different layers of abstraction where the solution to the problem could live: powers granted to users within a staking pool protocol, user choice between staking pool protocols, and in-protocol enshrinement. This choice should be considered carefully, and [minimal viable enshrinement], minimizing both protocol complexity and level of change to protocol economics while still achieving the desired goal, is generally best.