TLDR

: We detail a simple validator registration scheme with dynamic shard count, fixed-size committees and fluid shuffling. In the discussion we highlight key properties and justify some of the design decisions.

Construction

- Validators

: The Ethereum 2.0 protocol layer has a single type of collateralised participant called validators simultaneously participating in proof-of-stake and sharding. Collateral is fixed-size (32 ETH) and done via a one-way burn contract on the legacy chain (the current Ethereum 1.0 chain).

- Registration states

: Validators can be in one of three registration states: pending_registration

, registered

and pending_deregistration

. A validator is said to be active is it either registered

or pending_deregistration

. Pending registrations and deregistrations are queued in a FIFO.

- Initialisation

: At initialisation there are no validators and no shards. New validators enter the registration queue with state pending_registration

.

- First shard

: When num_pending_registration == 2^10

for the first time one shard is instantiated and those first 2^10 validators become registered

.

- Deregistration

: A registered

validator can request deregistration which triggers a deregistration countdown of 2^21 periods (~4 months assuming 5-second periods). If num_pending_registration > 0

when the deregistration countdown ends, the validator is immediately deregistered and replaced by the top validator in the registration queue. Otherwise num_pending_registration == 0

and the validator enters the deregistration queue. Similarly, a non-empty deregistration queue gets immediately popped when a registration request is made.

- Shard doubling

: Whenever num_pending_registration == num_registered

the number of shards is doubled. For every new shard 2^10 pending_registration

validators become registered

. Notice the two invariants that shard count is a power of two, and that there are 2^10 active validators per shard. We limit the maximum number of shards to 2^9.

- Proposers and notaries

: At every period each shard is assigned two sets of exactly 2^10 validators: a set of proposers and a committee of notaries.

- Shuffling

: Proposers and notaries are shuffled (via pseudo-random permutations) across shards in a staggered fashion and at a constant rate. Proposers are assigned to shards for 2^19 periods (~30 days) and the oldest proposer from each shard are shuffled every 2^(19 - 10) periods. Notaries are assigned to a shard for 2^7 periods (~10 minutes) and the oldest 2^(10 - 7) notaries from each shard are shuffled every period.

Discussion

- Unity

: Sharding and proof-of-stake validators are merged (credit to@vbuterin for pushing in this direction). See benefitshere.

- Homogeneity

: Every shard has the same number of active validators, i.e. the same number of proposers and notaries. The ratio of active validators to shards is a fixed power of two to simplify the design.

- Predictability and fairness

: Every active validator is active on two shards (once as a proposer and once as a notary—possibly the same shard) with permutation-based shuffling (no Poisson distribution).

- Large committee

: A large notary committee of size 2^10 allows for good safety and liveness. Targeting a 2^-40 probability of sampling a bad committee, 2^9-of-2^10 committees only require a 61% honesty assumption.

- Large set of proposers

: Since proposers are infrequently shuffled (every ~30 days) a relatively large set of proposers increases resistance to adaptive attacks (e.g. bribing).

- Reasonable max stake

: With 2^10 active validators per shard, the maximum of 2^9 shards corresponds to 2^24 ETH staked (~32 million ETH). With only 135 active validators per shard the maximum number of shards would balloon to 3,883 for the same amount of stake.

- Fluid shuffling

: Shuffling is homogeneous and fluid. This avoids spikes in global resource usage (notably, bandwidth) to spread load over time. It is a more effective alternative to staggered periods.

- Upsize only

: The shard count can only increase (at least until shard load balancing is implemented).

- Capital burn

: While individual deposits are churned through with new validators joining, the global deposit pool can be considered permanently unavailable, i.e. burnt.

- Overwhelming upsize demand

: We only increase the shard count if there is at least as much demand from new validators as there is existing supply. This allows for more churn liquidity for deregistrations.

- Infrequent upsizing

: Shard doubling combined with the 2^9 maximum number of shards means that there can only ever be nine birthing events, limiting the total number of disruptions. The square-root shard count proposal (see here) achieves something similar in a less extreme way.

- Powers of two

: Shard doubling makes the shard count a power of two which can help simplify crosslinks, and is consistent with the rest of the design.