Currently, the shard block time is 6 seconds, with active proposals to decrease this to 3-4 seconds. But we would ideally like to have transactions get a first-confirmation that comes even faster than that, perhaps close to the network latency delay. We could try to do this by decreasing block times to <1 second, but there is an alternative that is more in line with a “conservative layer 1, do crazy stuff on layer 2” ethos.

Shard block time staggering

First, we stagger the “intended” block production times: we specify in the protocol that, if the slot interval is d

and the genesis slot time is G

, a block on shard i

at slot h

is “supposed” to appear at time G + d * (h + \frac{i}{SHARD_COUNT})

. That is, blocks on lower shards get produced earlier in the slot, and blocks on higher shards get produced later in the slot.

Getting expected block publication times

Second, to account for clock disparities, we add a p2p network message GET_TIME_UNTIL_BLOCK_PUBLICATION

that allows clients to ask a validator for the number of milliseconds until it plans to produce a block (to avoid burdening block proposers, we could also have them broadcast such signed

messages every 200 milliseconds and then have them get constantly rebroadcasted throughout the network).

We hope that the majority of block proposers will be honest; note that honesty can be further incentivized because clients could be aware of that validator’s past behavior (assuming 10M ETH staked, so 312500 validators totally and so ~305 per shard, a validator will on average produce a block once every eg. 4 * 305 seconds = 20 minutes, and at 100M staked that goes up to only ~3.3 hours, so the info will be there) and determine which validator is likely to be honest next time.

Broadcasting the tx on the shard that will publish sooner

We then create mechanisms by which users are able to carry out any given action from multiple shards (potentially any shard) and so they are free to query multiple shards’ next proposers, and send a transaction to the shard that is planning to propose sooner.

Here are two possibilities:

- If you have ETH on multiple shards, and are asked to send ETH to an address on shard N, you can pick the shard that will propose sooner, and send a transaction that sends the ETH (ie. generates the cross-shard transfer receipt) on that shard. This applies more generally to any token payment, or to any asynchronous operation that does not have race condition issues.
- A Plasma-like architecture except where data is published to chain to remove dependence on operators. The Plasma-like architecture uses an exit game where if a piece of data authorizing a transaction is published in any shard

, it can later be referenced in the exit process to enforce the state transition.

Consequences

This implies that there is less need to try very hard to reduce block times of individual shards, as layer 2 mechanisms can give fast confirmation times most of the time (for even more guarantee at the cost of efficiency, one could try publishing to two shards) even if underlying block times are relatively slow. This is arguably further evidence in favor of a minimal and abstraction-focused layer 1, so that in a world where such layer 2s where the mapping between layer 2 state and layer 1 state is often quite complex, there is no “wasted complexity” on the layer 1 side.

An extreme version of this position would even go so far as to argue that inter-epoch shard blocks are not necessary, and all

that we need are crosslinks (as with 1024 shards and 384-second epochs there would one shard block per 375 milliseconds), though that may be going too far, especially since one dishonest validator could force a message to be delayed by 6 minutes.