One consistently thorny topic in blockchain design is timing. All blockchains so far rely on timestamps provided by nodes in blocks in order to calibrate their blockchain's block time, whether directly (eg. as in PoS schemes that explicitly say "a block at slot 1045 can be created at time 1457284250…1457284259"), or indirectly (eg. like in PoW difficulty adjustment). One possible scheme for going without timestamps is a clock that depends directly on network latency. The blockchain would not even care about timestamps; instead, the clock would be a logical clock whose height keeps incrementing, and every time the height enters a new range (eg. 3570…3579), nodes in the PoS chain would be allowed to create a new block.

However, there is an obvious limit to the possibility of such a clock: if the clock can survive with only portion \alpha

of the network online, then an attacker with an \alpha

share of consensus power can cheat the clock by running the algorithm locally between its own nodes. The good news: it's not that hard to make a near-optimal clock for any \alpha

. Here's the algorithm:

- The genesis 0x000000 is a "signed message", with sequence number 0 (this is an inductive base case)
- Anyone can create an "unsigned message", which points to a (signed message) parent; its sequence number is the parent's sequence number plus 1
- The hash of that unsigned message identifies a random sample of M

validators (the closer M

is to infinity, the better, but the more overhead)

- A signed message can be created by combining an unsigned message with signatures from \alpha * M

of those M

validators

The current clock value is simply the highest seen sequence number. Note that we do not care about the specific chain, only the height, so reversions and "51% attacks" are not an issue.

For efficiency, each random sample signature could include a list of all unsigned messages that it has seen, and inefficiency could be further capped by specifically specifying a smaller sample of validators which are the only ones allowed to make unsigned messages on top of a given parent. This could potentially be made even more efficient and leaderless through some kind of fancy DAG algorithm, though it may not be worth the added complexity.