TL;DR

: An EE that automatically relocates contracts between shards, allowing many cross-shard transactions to be included without any additional complexity (“for free”) if one is prepared to wait for a short time. Also allow atomicity between certain contracts, even if they don’t allow user-initiated yanking (which would probably be default for some applications). Slightly more complex shuffling functions could extent this to all contracts.

Construction

Since shards, from the perspective of validators are fully stateless (state is kept by state providers, which could be implemented by some more powerful validators), moving a deterministic part of the state from one shard to another comes at no additional cost at the consensus layer. Let’s use the 32 bytes of on-shard state of an EE to store the root of two sub-states: The “Hotel state” and the “Train state”.

When a new block is proposed, there are two possibilities:

1. The previous block was not fully crosslinked – then all shards keep their current Train and Hotel states from the end of the previous execution block;
2. The previous block was fully crosslinked on the beacon chain. Then we do the following:
3. all “Hotel states” stay at their current shard
4. Train states of shard i

move to shard i+1

for i<n-1

; state n

moves to shard 0

1. all “Hotel states” stay at their current shard
2. Train states of shard i

move to shard i+1

for i<n-1

; state n

moves to shard 0

When two parts of hotel and train state are colocated in the same shard, synchronous transactions are possible between them. An account or contract can also change between hotel and train at any point and thus move between shards.

In the proverbial “train and hotel” problem, the booking of a train and a hotel atomically would be trivial if there was an agreement that all hotel providers would add their hotel contracts to hotel states, and all train providers their trains to train states.

In essence this represents a form of “automatic yanking” between shards, with the intention that many “yanks” would never have to be formally introduced as cross-shard transactions would just wait for the right kinds of state to be colocated. Applications with lower latency requirements would still aim to be colocated on the same slice of state.

There are of course many more possible designs that allow larger parts of the to be mixed up in ways that mean that almost all contracts/accounts will be colocated regularly.