See also v1: Alternative bounded-state-friendly address scheme

This document describes a scheme for how accounts and storage slots can be stored in a way that allows them to be pruned over time (see this doc on state expiry schemes) and minimizes issues around resurrection conflicts. The mechanism is epoch-based and so can be viewed as a form of “regenesis”.

- Let an epoch

be a period of roughly 1 year in length.

- We introduce an extended address scheme

: an address can be viewed as a tuple (e, s)

where e

is an epoch and s

is a subcoordinate (a 20-byte value that contains the same information that addresses contain today). An account with address (e, s)

can only be touched when the chain is in epoch \ge e

- The state consists of an ever-growing list of state trees

, S_1

, S_2

, S_3

…, where each S_i

is the state corresponding to the current epoch. During epoch e

: * Only tree S_e

can be modified

- Old trees (all trees S_d

where d < e

) are frozen and can only be accessed with witnesses. Full nodes do not need to store anything from these trees except the roots. Block producer nodes are expected to store S_{e-1}

to assist with producing witnesses (so users only need to produce witnesses for epochs more than 2 in the past).

- Future trees (all trees S_f

where f > e

) are empty (because there was not yet any legal way to modify them)

- Only tree S_e

can be modified

- Old trees (all trees S_d

where d < e

) are frozen and can only be accessed with witnesses. Full nodes do not need to store anything from these trees except the roots. Block producer nodes are expected to store S_{e-1}

to assist with producing witnesses (so users only need to produce witnesses for epochs more than 2 in the past).

- Future trees (all trees S_f

where f > e

) are empty (because there was not yet any legal way to modify them)

- An account (e, s)

can be accessed in any epoch f \ge e

, and is always stored in the tree at position hash(e, s)

- Account editing rules are as follows:
- If an account (e, s)

is modified during epoch e

, this can be done directly with a modification to tree S_e

- If an account (e, s)

is modified during epoch f > e

, and this account is already part of tree S_f

, then this can be done directly with a modification to tree S_f

- If an account (e, s)

is first created during epoch f > e

and was never before touched, then the sender of the transaction creating this account must provide a witness showing the account’s absence in all trees S_{e}, S_{e+1} ... S_{f-1}

- If an account (e, s)

is modified during epoch f > e

, and this account is not yet part of tree S_f

, and the account was most recently part of tree S_{e'}

with e \le e' < f

, then the sender of the transaction creating this account must provide a witness showing the state of the account in tree S_{e'}

and its absence in all trees S_{e'+1}, S_{e'+2} ... S_{f-1}

- If an account (e, s)

is modified during epoch e

, this can be done directly with a modification to tree S_e

- If an account (e, s)

is modified during epoch f > e

, and this account is already part of tree S_f

, then this can be done directly with a modification to tree S_f

- If an account (e, s)

is first created during epoch f > e

and was never before touched, then the sender of the transaction creating this account must provide a witness showing the account’s absence in all trees S_{e}, S_{e+1} ... S_{f-1}

- If an account (e, s)

is modified during epoch f > e

, and this account is not yet part of tree S_f

, and the account was most recently part of tree S_{e'}

with e \le e' < f

, then the sender of the transaction creating this account must provide a witness showing the state of the account in tree S_{e'}

and its absence in all trees S_{e'+1}, S_{e'+2} ... S_{f-1}

Example

Epoch 7

- Alice publishes a smart contract in epoch 7. Alice generates an address (7, 0x14b647f2)

and sends a transaction to initialize that contract. The contract is saved in the state tree S_7

at position hash(7, 0x14b647f2)

. This is unchanged from status-quo Ethereum.

- Later in epoch 7, Alice interacts with her contract. The record at position hash(7, 0x14b647f2)

is modified. This is still unchanged from status-quo Ethereum.

Epoch 8

- Time passes and it is now epoch 8. To interact with her contract again, Alice's transaction in the block would need to provide the witness from S_7

(which can no longer be modified) to prove to the chain the most recent state of that block from the last epoch. Fortunately, as a convenience for Alice, block producers are expected to store S_7

, so Alice simply sends her transaction as is, and the block producer will add the witness. The record of Alice's contract is now saved in the state tree S_8

at position hash(7, 0x14b647f2)

. Alice's experience is unchanged from status-quo Ethereum, but block producers do have the new requirement that they need to generate the witness. S_8

stores the updated state of Alice's contract, S_7

forever stores the state that it had at the end of epoch 7.

- Alice interacts with her contract again in epoch 8. Because the position hash(7, 0x14b647f2)

in S_8

already contains a state object, there is no need for even the block producer to provide a witness.

Epoch 13

- Alice's plane crashes on an island, and Alice has no contact with the world for five years. Fortunately, Alice is eventually discovered, and she makes it back to civilization.
- As soon as she gets home, she immediately wants to get back to playing with the most important thing in the world for her: her smart contract. It is now epoch 13. To recover her smart contract, she needs to find someone who has a witness to show the value at position hash(7, 0x14b647f2)

in S_8

, and show that that position is empty in S_9

, S_{10}

and S_{11}

(the block producer can add the witness for S_{12}

).

- Bob, overjoyed at Alice's return, sends her a nyancat NFT as a welcome-back present. However, he foolishly assigns the nyancat's ownership to another address that Alice generated and gave him all the way back in epoch 5 (the address is (5, 0x2718bfa3)

).

- Alice never actually used that address. So she finds some third party (eg. etherscan) who has the witnesses proving that (5, 0x2718bfa3)

was not in S_5

, S_6

or any other tree up to S_{11}

. She sends a transaction containing those witnesses, verifying the state of her account (her account might be stateful because it's a social recovery wallet in which case simply proving the code from the address and adding an epoch-aware anti-replay scheme is not sufficient), and transfers her nyancat NFT to a different account.

Epoch 16

- Alice hibernates for a few years, and wakes up. It is now epoch 16. To use her smart contract or her nyancat-holding account (as well as the nyancat itself), she must provide witnesses from S_{13}

and S_{14}

.

Nice properties

- Full protection against resurrection conflicts
- No situation is unrecoverable; even when Alice was on an island or hibernating, she was always later able to recover her account by asking a third party for witnesses
- Note: if we want to support recovery from multi-hundred-year absences [eg. Alice is cryogenically frozen and then revived with advanced technology in the 2300s], then we may need ZK-SNARKs to make the set of all witnesses for the intervening epochs verifiable within the gas limit, but we don't need to actually think about adding support for this for at least a hundred years
- Note: if we want to support recovery from multi-hundred-year absences [eg. Alice is cryogenically frozen and then revived with advanced technology in the 2300s], then we may need ZK-SNARKs to make the set of all witnesses for the intervening epochs verifiable within the gas limit, but we don't need to actually think about adding support for this for at least a hundred years
- Does not require any fancy properties from trees; trees can simply be key-value stores

Adding storage slots

The simplest way to add storage slots to this construction is to move toward a one-layer state scheme, so storage slots are treated equivalently to independent accounts. The SSTORE and SLOAD opcodes would

need to be modified to add an epoch parameter (so each account would de-facto have a separate storage tree for each epoch), and for maximum efficiency contracts would need to be rewritten to take advantage of this. But even if they do not, contracts would not break; rather, existing SSTORE/SLOAD operations would be mapped to epoch 0, so reading/writing to new storage slots would simply start to become more and more expensive due to witness costs, and performance would degrade gracefully.