We want to have a gas payment mechanism where the base gas charged for a transaction has a component which is approximately linearly proportional to the witness size. In the earlier 1-layer account model, this was accomplished fairly easily by charging a fixed amount of gas (eg. 3000) per account, plus a fixed amount of gas per byte in the account’s code and storage. In the 2-layer model, this is not as good an idea, because the actual witness size corresponds to the logarithm to the size of the storage tree, which is not available to the EVM, and even if it were there is always the possibility of attacks that try to create maximally inefficient trees that have witnesses of size linear in the size of the tree. To avoid all of these issues, charging directly for witness size seems maximally prudent.

The question is, how?

Here are a few alternatives:

Charge the validator

- How it works

: charge the validator for each byte in the witness, or alternatively for the number of state objects in the witness. Allow transaction senders to provide an additional gas stipend of their choice, and leave it to validators to only accept transactions that include a stipend large enough to pay for their access list

- Pros

: simple at consensus layer

- Cons

: requires validators to implement a more complex transaction sorting algorithm than before

Charge gas based on “necessary witness”, calculated in real time

- How it works

: when processing each transaction in a block, calculate the number of state trie node needed to access all of the accounts from the current state (ie. the pre-state of the transaction). Charge N gas per state trie node.

- Pros

: does not require fancy validator transaction sorting logic

- Cons

: requires knowing the pre-state of the transaction, making parallel execution and cross-transaction caching impossible

Charge gas based on “necessary witness”, calculated based on block pre-state

- How it works

: when processing each transaction in a block, calculate the number of state trie node needed to access all of the accounts from the pre-state of the block

. Charge N gas per state trie node.

- Pros

: does not require fancy validator transaction sorting logic, and allows parallel execution. Also, aligns well with actual costs (since the witness in the block must be based on the block pre-state, not the mid-state)

- Cons

: more complex and ugly (though not too ugly; it’s already the case that witness validation for every transaction must be done at the start of block execution; this just adds a step to witness validation that computes a witness data cost for each transaction)

So far I lean slightly toward the first or the third, but there may be other approaches that we have not yet considered.