There are genuine advantages to keeping the current two-layer state trie paradigm (a global trie of accounts, then a storage trie in each account) in the context of a stateless client model:

- It does a good job of keeping Merkle proofs for a transaction short

(in fact, this is so important that we should be seriously considering even putting the code in a Merkle tree).

- In a stateless client model, the fact that disk reads

have the same time complexity between 1 byte and 4096 bytes does not matter

. Hence, a lot of the advantages of a single-byte-slice storage model disappear.

- Witness data becomes closer to first class, and so we can resolve the mispricing issues between small storage tries and big tries

by simply charging for witness data directly.

So it is worth mapping out what that would look like. It is actually fairly simple.

The “account list” data structure in a transaction would be modified as follows. It would be an RLP list of [item, item…], where each item can be either a single item RLP list [address], or a list [address, key1, key2…] consisting of the set of storage keys that may get read. The first case represents the idea that the entire tree could get read; in this case, the witness would need to include the entire tree.

Now, we can add new witness data and gas rules. As before, when starting to process a block we perform a static check to verify that witness data to access everything in the accounts lists of all transactions is present. However, on top of this we add one of two approaches.

1. Add a rule that we charge the miner X gas per byte of witness data. More precisely, the gas limit of a block, as interpreted by a client, becomes BASE_GAS_LIMIT - BYTE_GAS_COST * WITNESS_SIZE. A transaction would also need to specify a set fixed fee that it is willing to pay; this fee represents the amount that the miner gets compensated for including the witness data.
2. Add a rule that the “base gas cost” of a transaction changes from the current “21000 + 68 * (# of nonzero bytes in tx data) + 4 * (# of zero bytes in tx data)” to “21000 + 4 * (# of bytes in tx data) + 4 * (computed witness size)”. The “computed witness size” would be the total length of the Merkle tree nodes that are accessed by the witness validity check of the transaction.

Note that hypothetically (1) or (2) could be used in a not-fully-stateless client network. The basic approach for (1) would be that the witness size would be calculated as the minimum possible witness size according to an optimal algorithm. A block that passes with this check can be turned into a block on the stateless network by simply adding this minimally-sized witness. A block on the stateless network that is considered valid will have a witness size equal or greater to the optimal one, and so its total gas consumption including the witness gas will be equal or lower, and hence also valid. For (2), no special changes are required, except that the algorithm to actually compute witness size would need to be added.

Note also that with this model the storage

tries would NOT need to be “SecureTries” (ie. keys hashed before indexing), as the user pays the cost of getting exploited by attacks on the trie, and if this is an issues it can be overcome by applying the hashing at the higher language level.