TL;DR

This is a proposal to split witnesses into 2 complementary parts: data

and proofs

, allowing for stateless clients to process just the necessary data without the merkle proofs (and thus reducing network traffic) and without proving the stateRoot

, while retaining a high degree of certainty for the data being processed thanks to a new witnessHash

field in the block header (a hash of the hashes of the data and the merkle proofs) secured by the consensus layer, with the option to retrieve the proofs as well and fully validate the state transition.

The biggest motivation for this is the reduction of witness sizes, which are dominated by merkle proofs. An additional benefit might be faster stateless client execution and faster beam sync (a new-ish take on it) if the stateRoot

check is postponed (compared to constantly re-merkelizing the state trie all the way up to the top after every block).

Detailed description

The data

part of a witness

It should contain the absolute minimum for transaction execution:

- the state trie nodes referenced by the transactions in the associated block
- the code chunks (but without their merkle proofs), as envisioned in thecurrent code-merkelization efforts
- positional information (bitmaps) for the code chunks - which parts of the bytecode of a contract are present (for a 340-byte contract with 32-byte chunking that would be 11 bits - truly negligible)

Currently, (not in this proposal) witnesses contain the necessary trie nodes, the merkle multiproof, and some structural information (masks) required to position the nodes in the right places in the state trie in order for the stateRoot

hash to actually match. If we decide to also add that structural information to the data

part of this split witness proposal we could enable a new kind of beam sync - one which gradually reconstructs the state trie without merkelizing all the way up to the stateRoot

every time (and thus requiring smaller witnesses), but merkelizing the branches that are already entirely available (more on this a bit further below). This structural information (the “masks”) is really small (compared to the hashes in the merkle multiproof), as can be seen in this graph (taken from this article) and it can be easily bundled in the data

part instead of the proofs

part.

The proofs

part of a witness

It should contain everything else necessary to fully validate the stateRoot

for the given block:

- the merkle proofs for the state trie nodes
- the merkle proofs for the code chunks within each contract
- the merkle proofs for the state trie for all contracts used in the block (actually a subset of the first point here)

Witness construction and the witnessHash

in the block headers

If we hash the data

and the proofs

parts we can then hash those 2 hashes and construct the witnessHash

, which should go into the block header (which would require a protocol change). When distributing just the data

part of the witness we would need to also ship the hash of the proofs

part in order for clients to validate the witnessHash

without requiring the proofs

part as well.

This way, even though the stateRoot

isn't validated, we still rely on the PoW or PoS consensus for the validity of the witnessHash

hash (because constructing bogus blocks would be costly to attackers).

A witnessHash

has already been mentioned in articles such as this (in the context for witness indexing). Given that the average block size is 40Kb, the addition of 32 extra bytes for the witnessHash

would lead to an increase of 0.08% of the main blockchain which is negligible.

Construction, encoding, and decoding

The current witness specification requires that witnesses are efficient (decoded, validated, and executed in a single pass with minimum dynamic memory allocation) and streamable (stream-as-you-encode - without intermediate dynamic buffers). I'm not familiar with merkle multiproofs, but it should be possible to achieve the same goals with 2 separate binary blobs, which ought to be created and consumed together and in parallel (lockstep), except for when just the data

part is being used by a stateless client.

Size savings

The only addition of new data in this 2-part witness proposal is the bitmaps for the contract chunks, which as we saw will be just a few bytes for each contract - all other data is already present in the current witness spec. All the savings would come from separating the proofs. The best resource regarding the data

/proofs

ratio I've found is this article and especially this graph (if I'm reading it correctly…) from which I conclude that the ratio is something along the lines of 0.3 (~220 / (~250 + ~450)), so that means that the data

part is about 24% of the witness size and thus we would be saving ~75%. Would appreciate any numbers regarding this.

Security

In fast sync, you basically trust all the state transitions until the point at which you joined and you trust the consensus layer and just how economically infeasible it would be for someone to have cheated before that. Not validating the stateRoot

by using just the data

part of witnesses is just an extreme continuation of that - by constantly pushing this trust boundary to the current head block (in contrast to fully validating the chain from some point onwards). Stateless clients that don't validate the stateRoot

are just a new point on the edge of the trust spectrum (how much assumptions are being made). In the case that a block producer cheats and changes the stateRoot

to something which doesn't match reality - all stateless clients who don't use the proofs

part will be oblivious to that.

However, in case of a broken block, the rest of the full nodes (and stateless clients who use the proofs

part) won't agree with it - they will simply not propagate the bad blocks, and witnesses and miners will refuse to build upon the invalid chain. In addition to that, stateless clients can at any point in time request the proofs

part and validate if things are OK - the frequency of requesting/processing it can be a tunable parameter - let's say once an hour (for 1 out of every 240 blocks).

Beam sync

If the structural information (masks) for the state trie nodes is included in the data

part, then stateless clients can gradually build a local copy of the state trie. The difference with how beam sync currently works is that the stateRoot

won't be getting validated for quite some time - this will save on computation and networking, which should speed up the sync process, but at some point, the node should request a multiproof for all gaps (branches) of the state trie which prevent full merkelization all the way up to the top, after which it could switch to fast sync for the remaining state.

Advantages

- This solves the need for witness indexing and has already been mentionedhere (look for witnessHash

).

- It also solves the problem of incentivizing full nodes to produce witnesses.
- Blocks should be propagated only when their witness comes along- a good constraint.
- The witnessHash

for a block can get validated before executing the block ==> less room for a DoS attack with invalid witnesses.

- The less impact witnesses have on networking, the less need there is for gas scheduling and moving from a hexary to a binary trie for the state (which is a complex engineering challenge).

Disadvantages

- Harder to implement (especially the part around beam sync and not fully merkelizing up to the top), more complexity around which parts of witnesses are propagated through the network (and when).
- A little less trustworthy - not "true" block validation.

Other notes

- I'm not familiar with Kate commitments, if there's consensus for using them and if the idea of moving to a binary state trie has been abandoned - perhaps this entire proposal is obsolete…?