A major bottleneck of statelessness is block witness size Below are proposals to reduce block witness size. The first five proposals are already planned to be used.

(1) Hexary to binary tree overlay. Over 3x savings in the number of merkle hashes.

(2) Multiproofs to deduplicate merkle hashes. ~1.5x savings in number of merkle hashes near the root.

(3) Maximize overlapping merkle paths. Related to (2), but worth mentioning separately. We need fast algorithms to build blocks which maximize overlapping merkle paths. Unfortunately, it is undecidable a priori

which merkle paths will be used by general transactions. But it may be decidable forsome transactions. Users may send overlapping transactions together. Perhaps 2x savings in number of merkle hashes is within reach – open problem.

(4) Witness encoding. Tree structure encoding is a small fraction of the witness size, which is dominated by hashes and code, but maybe this fraction can be further reduced. A few percent savings in witness size.

(5) Code merkleization. Gives 2x code size reduction. Also noteworthy is that code compression gives3x-4x code size reduction.

(6) Deposit-as-rent. Power-users can deposit 1 Eth per byte to store their account in a witness-free way. The total of these bytes will currently be at most 110 MB (plus some overhead). Savings from this is an open question.

(7) Cache. Experiments by Alexey and Igor show that a cache of recent block witnesses can give a ~10x (!!!) savings in witness size. Unfortunately, consensus caches are complicated, so caches may be at the networking-layer until we become desperate for consensus witness size reductions. If consensus caching is considered, a related option is a consensus transaction pool (a two-step process (i) transactions with access lists but no witnesses are included in blocks and put in a consensus transaction pool, and (ii) their execution is delayed until a reasonable amount of time, say 100 blocks, for their witnesses to propagate).

(8) 20 bytes per merkle hash. We already depend on 20 byte hashes for addresses. For security, the system can be adaptive: when a hash collision is detected, it triggers a tree remerklization to add two extra bytes per hash. This gives a 1.6x savings in hash size.

(9) New stateless-friendly dapps. Stateless-friendly patterns are needed. Savings from this is an open question.

Any block witness size reduction proposals missing? Any feedback on the above proposals?