I see that Ethereum community is actively discussing stateless clients.

During Financial Cryptography 2017 we presented a solution for partially stateless clients (https://eprint.iacr.org/2016/994), where only miners need to store the validation state in full in order to generate proofs of its transformations, then other nodes (full nodes actually, which are not losing anything in security but do not store the state) check the proofs.

Now we have solutions for fully stateless clients, for both UTXO- and account-based cryptocurrencies: https://eprint.iacr.org/2018/968 !

There are a lot of open question still if we are talking about not simple payment systems but complex smart contract networks like Ethereum. From the paper: “Recall that in the smart contract setting, the flow of money

will depend of the execution of some contract code on the current contract state, which is updated after

the contract execution. Therefore for Alice to post a contract-triggering transaction she must provide a proof of correctness of the current contract state for EDRAX nodes to execute on. For that, we can again use a Merkle tree whose leaves store hashes of the contract state and have clients provide the respective Merkle proofs. Two challenges that arise in this setting are (i) who is storing the contract state since any client can post transactions that will trigger a contract execution; (ii) how to avoid including the contract state as part of the transaction (the contract state might be too large). For both of these challenges, proof-serving nodes (as introduced above) and SNARKs might help. A complete treatment of these challenges, however, is left as future work.”

We would be happy to hear from the community how stateless clients can affect design of the Ethereum.