The “goal” of “Stateless Ethereum” is to modify the protocol such that we can have stateless clients which do not store the state

and instead, use witnesses to execute and verify new blocks.

The current “Stateless Ethereum” roadmap can be loosely summarized as:

1. Use binary trie and code merklization to reduce witness sizes.
2. Modify the core protocol such that block witnesses are created and gossiped
3. Stateless clients use block witnesses for stateless block execution.

I am going to make a case that this roadmap is fundamentally flawed and that we’re unlikely to deliver on our goal of having clients that do stateless block execution.

## Useless stateless clients

If we accomplished the above, a client could be built that executes blocks in a stateless manner. The problem is, that client would be capable of doing very little else.

- Without the state it would not be able to participate in transaction gossip since transaction validation depends on having access to the state to check nonces and account balances.
- Without the state it would not be able to serve some of the most important JSON-RPC endpoints (eth_call

, eth_getBalance

, eth_estimateGas

)

- Without the state the client couldn’t be part of the DevP2P eth

network as it would not be able to serve GetNodeData

requests.

Our roadmap gets us the ability to do stateless block execution, but none of the clients are going to build out that functionality because the client would be useless.

## Training the monkey instead of building the pedestal

https://blog.x.company/tackle-the-monkey-first-90fd6223e04d

It is my assertion that we’re not working on the right problems. We know that we can reduce witnesses to a manageable size. Producing block level witnesses and gossiping them isn’t a fully solved problem but we have ideas for how to go about solving it and the problem space is well understood. These are not the hard problems.

The hard problems are:

- Stateless clients need to be able to expose and serve the JSON-RPC API.
- Stateless clients need to be able to participate in transaction gossip even when they don’t have any of the state.

## Stateless clients and Full nodes are going to be fundamentally different

The other major change in thinking I think we need to make is that we should not expect existing clients to be the first ones to adopt stateless block execution. The architecture of a stateless client is going to be fundamentally different from existing full node architecture. Because they won’t have the state they will be required to use these different mechanisms for serving JSON-RPC requests and doing transaction gossip (and likely other things as well). Current client development teams already are taxed to their limits simply building and maintaining their clients.

It is my assertion that we should expect the first stateless clients to be greenfield projects that operate only in the stateless paradigm. Thus, we should not expect the majority of the work to be done by existing client development teams, but rather by newly formed teams who are specifically focused on delivering this new type of client.

## What about witness sizes…

This isn’t to say that we shouldn’t do the binary trie conversion or code merklization. We still need to do these. They are going to be necessary. Luckily they are already underway. We need to continue to support the people working on these

efforts, but the research part should be largely done, and the implementation part is going to rest primarily on the existing mainnet client teams.

### What about witness production…

Without consumers of witnesses, witness production is a useless activity. Without stateless clients, we are going to have no consumers of witnesses. Focusing on witness production before we have a reasonable handle on the clients that will consume them is doing things in the wrong order. We need to focus on the clients

### How can a stateless client do stateless block execution if they don't have witnesses

They can't, but this doesn't actually matter. If we build the functionality for stateless clients to gossip transactions, serve JSON-RPC, etc, then they don't actually need

to execute blocks in order to be useful. Once we have a solid handle on filling out all of the other functionality needed to make stateless clients useful, we've built everything needed to create really nice ultra light clients.

Stateless block execution is simply an extra layer of verification and security that can be enabled/added once witnesses are being produced and made available.

### How does chain history come into play

It technically is orthogonal, but we should do it anyways. It will be useful to all types of clients and there's little reason not to build it if we have the resources to do so.

### Additional reading and information

A three part blog post I wrote on how we solve these additional problems.

- https://snakecharmers.ethereum.org/the-winding-road-to-functional-light-clients/
- https://snakecharmers.ethereum.org/the-winding-road-to-functional-light-clients-part-2/
- https://snakecharmers.ethereum.org/the-winding-road-to-functional-light-clients-part-3/

A talk that I gave on the same subject:https://www.youtube.com/watch?v=MZxqRs_tLNs

## Summary

- We keep going with the binary trie conversion and code merklization
- We need new networks to serve on demand state and on demand chain history
- We need new gossip mechanics to allow stateless transaction gossip.
- We need new client teams that are dedicated to build stateless clients since we should not

expect current client teams to do this.

This will require multiple independent teams working concurrently on these problems. I'm already working to secure funding. It's very likely that ESP may play a roll in this as well providing grants.

## Questions and Critical Feedback

What questions do you have.

Do you see something you think I got wrong, or something I missed?