I'm summarizing one part of Piper's post in order to make it easier to discuss.

The eth subprotocol assumes that all nodes either have all of the state or will eventually have all of the state. Nodes joining the network try to acquire state by firing off GetNodeData

requests to their peers. Some nodes are syncing and will return empty responses for GetNodeData

, but that's okay, because it's quite likely that another of my peers has the state I'm looking for.

If stateless clients were to implement eth

(and they need to fetch the header chain somehow), and if a flood of them were to join the network and outnumber the full nodes (which is the goal), then full nodes trying to join the network will no longer be able to find other nodes to sync off of.

It's possible we could fix this by improving discovery, so that full nodes could find other full nodes to connect to.

However, stateless clients also need to fetch state!

- to a send a transaction you need to know the none of your account
- RPC methods such as [eth_call

](https://github.com/ethereum/wiki/wiki/json-rpc#eth_call) require inspecting arbitrary parts of the state trie

- The current rules for transaction validation require looking up the transaction sender before forwarding transactions along.

If there was some kind of discovery method nodes could use to find full nodes, a flood of stateless clients would consume all the full nodes' peer slots, causing them to reject incoming connections. So, fetching state from the full nodes is not possible, stateless clients will need to fetch state from some other source than the full nodes.

An obvious answer is for stateless clients to talk to other stateless clients, and forward requests to the full nodes, caching as much of the responses as they can. That could

happen with eth

, but it would be a lot easier to use a new protocol better suited for this use-case. Another answer is for stateless clients to talk to a third node type, state providers (mentioned here: (1), (2), (3)), and for that re-using eth

makes even less sense.

I know this seems kind of straightforward, of course we can't just reuse eth

, but I thought it was worth writing down, and there are some parts which aren't entirely straightforward:

- Most of the above applies, even if stateless clients do not need to fetch state

. An MVP which can only verify blocks will still need to fetch the header chain and block bodies, Since we can't use eth

we'll need to find something else for it to use in order to fetch headers and blocks.

- This happens even if we use a better discovery method. If nodes can't choose which kinds of nodes they're talking to, then we're in an even worse spot and definitely can't use eth

.

- A system where stateless clients forward state requests to the full nodes works, but only for as long as there's a certain ratio of full nodes to stateless clients. More and more stateless clients will put a bigger and bigger load upon the full nodes, and eventually some stateless clients are not going to be able to get their requests served.