Kittenless, and statelessness in general, is relatively well defined, save for one very vague and unanswered section. Somehow, nodes need to fetch state!

There are a few categories of state fetch and stateless ethereum needs to be able to support them all:

- Full nodes (and especially new miners) need to be able to fetch all the state
- Stateless clients need to be able to fetch accounts,so they can prove the senders for the transactions they create.
- Clients need to be able to fetch arbitrary accounts and storage items, [so they can run eth_estimateGas

](https://ethresear.ch/t/kittenless-a-concrete-proposal-for-stateless-ethereum/7047/2).

There are two major approaches. The first is incentivized state retrieval, which hasn't been very carefully explored but is widely considered a bit tricky. The second is putting all the data into some sort of DHT, which has the drawbacks that it's vulnerable to sybil attacks, and that we'll have to go to great lengths to ensure that leechers don't bring down the network

I want to propose a third approach, which I argue provides the benefits of both with none of their drawbacks. Under this approach:

- Very little research needs to be done to make sure it works, it could ship within months, leaving us plenty of time to focus on the tricky parts of stateless ethereum, such as the migration to a binary trie
- Each second thousands of mobile clients could join and leech state and then quit without seeding any of it back to the network and this would be fine and not at all a concern.
- Given some kind of bridge they can use to connect to the network the same applies to web browsers.
- eth_call

and eth_estimateGas

not only work, but they run quickly. The only way a stateless clients could run eth_call

faster would be if it became a full node, and even then it might not be able to compete.

- DOS attacks against state retrieval are difficult to perform, and require facing off against teams of people who have strong incentives to make you fail. (This is as opposed to DHTs, where we have to consider all possible attacks beforehand and try to build in countermeasures. In the DHT approach there will be nobody in charge of stopping attacks while they're happening. It's a cat and mouse game with a blind mouse)
- A very natural point is created which client developers can use to fund themselves

The idea behind the State Oligarchy is, let's come up with a standard protocol which nodes can use to fetch state from state providers. The value of using a standard is that state providers which turn out to be evil can easily be replaced with alternatives. In order to retrieve state you send some eth to a state provider. This is part of the protocol, to get around the bootstrapping problem where you need to already be using the protocol in order to fetch the state required to send a transaction and opens a balance you can use to pay for future requests. There's also a second bootstrapping problem here, you need eth in order to talk to a state provider, getting you some eth is something your client can do for you, and they might fund themselves by asking you for a donation during the process.

This solution is obviously a bit heretical, but I'll argue there's no technical reason why it won't work. You might think there are returns to scale or some kind of centralization risk, but they're no stronger than the returns to scale or centralization risk of mining pools! The big unanswered question is how payments work, we don't want state retrieval to fall over when transaction fees rise, but that feels more easily solvable and coming up with a sybil-proof DHT.