The majority of last week's All Core Devs call was spent discussing a topic that has been near and dear to my heard for years. I'd like to attack this by applying The Five Whys so that we can understand the root cause, and hopefully avoid wasting effort focusing on the wrong problems.

We'll start with the following problem statement

Problem Statement #1

Geth represents a super-majority of the Ethereum mainnet and is maintained by a very dedicated, but small team. The current conditions under which the client is developed are not sustainable and lead to client burn out.

Alright, so there's the first try at capturing the problem. We've got at least two things going on here. The correctness

requirements for the project are extreme, since the health of the network rests effectively entirely on their shoulders, resulting in the job being stressful. The small team makes this a lottery/bus problem since one or two of the devs burning out and moving onto something else would likely significantly impact the velocity of development on the project that might take years to recover from.

So we can maybe

agree on part of the problem here. Having a single client that represents a super majority of the network results in a situation where if

the client has a consensus bug which would cause a fork, the entire network will follow the wrong

fork since the majority of the nodes on the network will have the bug. Said differently, the minority clients on the network play near zero role in choosing the correct

fork, since they will always be out-voted by the client holding the super majority. We'll refer to this going forward as "client diversity"

.

The other facet of this problem, the team itself being small and subject to burnout, is a real problem, but I'm going to set it aside for now. We can't fix the problems associated with poor client diversity by focusing on the go-ethereum project alone. Maybe we could reduce the stress on the team, and build out a larger team to get to a safer place, but we can't address the rest of the problem via focusing on a single client.

With this established, we can now take another stab at the problem statement.

Problem Statement #2

The Ethereum mainnet does not have adequate client diversity. Because of this, a critical consensus bug, or other denial of service attack effecting the Geth client could severly impact the smooth operation of the network.

With this new problem statement we get closer to the real problem. What happens when the Geth client breaks. Software is notoriously difficult to get right, and even were we to try and apply the most stringent practices to focus on achieving correctness in the Geth client, it's still likely that something would eventually go wrong.

Having established that this problem can't be solved by focusing on the Geth client itself, we expand our focus to the minority clients. Here is what the network looks like today according to Etherscan data

- Geth: 75%
- Parity: 15%
- OpenEthereum: 5%
- Nethermind: 1%

None of the other clients on the network have a meaningful market share.

To the best of my current knowledge, there are four minority clients actively working towards being viable mainnet clients for production use.

- Nethermind
- Besu
- Turbo Geth

- Trinity

I won't go into specifics about any of these. Instead, I want to look at the broader trend. With the exception of maybe Nethermind and Besu very recently, there has not been a new client introduced to the network which could sync and operate reliably since Parity joined the party years ago. Why haven't we seen more clients join the network? Why aren't the current minority clients progessing faster and being used more broadly?

I have spent almost three years working on the Trinity Ethereum client. I'm going to gloss over the details, but my conclusion after working for a long time to try and introduce a new client to the network is that it is prohibitively difficult.

Problem Statement #3

Creating an Ethereum client is too difficult. This has lead to poor client diversity, which leaves the network in a vulnerable position.

Now we're getting somewhere. A naive approach to our current client diversity problem would be to try and provide better support for minority clients. One problem with this approach is that Geth has years of head start. We cannot reasonably expect new clients to catch up anytime soon.

So, if we cannot create new clients quickly due to the underlying difficulty in doing so, can we attack the problem from a different direction and make it easier to create clients?

Problem Statement #4

The current structure of the Ethereum network results in it being prohibitively difficult to create a client for the network. This results in the network having very few clients, and thus, poor client diversity.

Here we re-focus the problem from the clients, back to the protocol. Here are some things that are difficult in the current protocol.

1. The Ethereum "state" is very large
2. Managing the local "state" requires high end hardware and complex software
3. Syncing the state in a reasonable amount of time is complex
4. The network stack is a "monolith". Being part of the DevP2P ETH

network requires you to support: 1. On demand access to any part of the Ethereum state and state trie

1. On demand access to the full chain history
2. On demand access to any part of the Ethereum state and state trie
3. On demand access to the full chain history

This results in a high barrier to entry for new clients since accomplishing these things, amoung the many other things an Ethereum client must do is a complex engineering task.

Problem Statement #5

I don't think there is necessarily a 5th layer to this, but theFive Whys inspiration for this post requires me to have five things.

Conclusion

If we accept that there is a problem with client diversity, then I believe there is a strong case to be made that we can fix the problem by focusing on the network and protocols themselves. Specifically, we need to make it easier for clients to manage and sync the state, and we need to loosen the requirements placed on clients to join the DevP2P network.

I've intentionally stayed away from the "solution" space but I believe that the current "Stateless Ethereum" roadmap aims to address a part of this problem. Alexey's "three networks" model likely solves the "monolith" issue. The re-genesis suggestion by Alexey would also temporarily address some of the problems as well.