Topics Discussed

- New Visualizations
- 3 networks
- Merry-go-round
- 3 networks
- Merry-go-round
- a useful DHT concept from Piper
- Witness Spec updates
- The “Oil” proposal for EVM semantics
- Meta transactions and Account Abstraction
- Meta transactions and Account Abstraction
- Things to work on, research, and think about before the next call

Three Networks, DHT, and merry-go-round

Three Networks

[

Screenshot_2020-05-14_12-48-45

1776×1008 134 KB

](https://ethresear.ch/uploads/default/original/2X/b/b728931409ca6e21654afb66ae25c050bde125b8.jpeg)

This diagram helps to show how different data might propagate through a Stateless Ethereum network, or rather how it would propagate through three related but distinct networks. Currently all information on Ethereum is passed around by the same protocol, which makes optimization difficult.

An improvement would be to split into three complementary networks, and to optimize each for what they are supposed to propagate. Chain data like headers, bodies, and receipts could be delivered by a more responsive DHT network (more on that later), state snapshot could be a network optimized for throughput, while new blocks, transactions, and witnesses could be optimized for low latency.

A client would have a choice to join or leave each of the three networks as needed, with full nodes being always running all three. This would allow clients with partial or zero state much more flexibility while still allowing them to contribute to (rather than leech from) the network.

DHT: it’s how the cool kids get their historical chain data

Piper has been poking around with a novel implementation of DHT+SkipGraph, which you can read aboution the research forums – Although not strictly on ‘the critical path’ this network could handle the information on the left side of the three networks diagram, and would allow clients to start forgetting their historical chain data, which lightens the load, so to speak, on a client trying to keep things lightweight.

Merry-go-round (Alexey’s version)

There are actually two different (and for the moment, incompatible) approaches to Merry-go-round sync-- although they are related. Both proposals require that all participants sync the same part of the state trie at the same time, and continue to do so until the whole trie is covered as a group. To sync the whole state, you ‘jump on the merry go round’ and sync as it iterates over the trie, and stay on until you return to the place you started, at which point you’ll have a nearly

complete state. The difference in approaches comes down to how the syncing schedule is determined.

In Piper’s approach the segments are chosen stochastically, while inAlexey’s, they follow a ‘predetermined order’.

This image illustrate’s Alexey’s concept, though much of it remains valid for the alternate approach.

[

Screenshot_2020-05-14_12-49-20

1882×1048 215 KB

](https://ethresear.ch/uploads/default/original/2X/6/623b715d0c15a89296778365ab45ccd84f4fc78e.png)

In particular one challenging aspect to breaking up the state trie is that it's a bit 'lumpy', with code and storage sub-tries taking up awkward space that might need to be interrupted for uniform segments. Both proposals are relatively fresh and unpolished, so we should expect them to evolve and converge into one depending on what the research shows is the best path forward.

Witness Spec

In a welcome break from "the Alexey and Piper show", Paul jumped in to explain the recent work on the witness specification, which if you recall is one of the main milestones in the Stateless Tech Tree.

In general the work is making the witness much more formal and consistent with some more robust traditions in academic computer science, and more importantly that it become completely un-ambiguous for implementers.

There are actually several implementations for block witnesses already:

GitHub

poemm/eth_witness_experiments

Contribute to poemm/eth_witness_experiments development by creating an account on GitHub.

gist.github.com

https://gist.github.com/carver/382e4e84f461ff99838740e974a25a82

test_witness.py

from abc import ABC, abstractmethod from enum import Enum from typing import (Iterable, Tuple,)

import cbor from eth_typing import Hash32 from eth_utils import (

This file has been truncated. show original

It's shaping up, but Alexey had an idea for an alternative format that might render a lot of work being done currently on the spec as moot. Oh well, progress is progress! This might actually open up a path toward a more formal semantics of the EVM in general, which is not on the critical path, but still quite desirable.

Oil: Same as gas, but different

Suhabe discussed a proposal to introduce a change to the EVM that would decouple two intertwined functions of gas by creating a second 'fuel source' called 'oil'. The change solves a few issues, but was originally conceived to sidestep a problem with charging for witness production.

If the gas schedule is changed (to say, account for witness sizes and such), it has the potential to introduce breaking changes to contracts that hard code or make use of the gas included in a transaction. Lowering the cost of an instruction then might introduce security risks, while raising the cost might break contracts that assume an absolute, hard-coded gas price for calls.

Oil would be an unobservable that prevents smart contracts from hardwiring specific values in call instructions. In the current proposal oil reverts the entire transaction, rather than halting for the caller to inspect.

This is a big deal, because it potentially breaks meta-transactions. There are many projects out there trying to pay for other account's transaction fees, so the change would be quite contentious. Account abstraction, however, might provide an alternative means of achieving this UX for those projects, but it's not known yet, and unwise to rely on that as a solution.

The Quilt team has been working on exactly this and will have some info in just a few weeks that will help to shed light on how a contentious breaking change might be avoided.

Back to work!

One announcement is that a mysterious "team X" out of consensys has shifted to help out with some Eth1.x legwork, and will be working on code merkleization, building on Sina's work, as well as helping to scrounge together some network monitoring tools that will be needed. They are all Australia-based, so will be reading the notes and (hopefully) this digest

Sorry for 2am calls

It looks like the relevant people have a bit of homework in the form of more verbose and polished write-ups about some of the uncertainties discussed in sync as well as EVM changes, but for the most part things are chugging along just fine.

No sync call will be scheduled until there is indication that work has moved forward on that and one is needed, while the next proper Stateless Ethereum call is 4 weeks from now, date TBD.