By:
Parithosh Jayanthi

Through The Ethereum Looking Glass

Problems & Fixes

Disclaimers

About me

- MSc. Communications Engineering at Technical University of Munich
- Currently working as a DevOps Engineer at the Ethereum Foundation
- Tasks: Maintain, automate and manage testnets and help out with protocol upgrades
- Fun fact: I’ve helped setup a data center deep inside a mountain

parithosh_j

parithosh

Contents

What is Ethereum?

What the core protocol is…

The Ethereum core protocol is what defines Ethereum. Set of rules and specifications that govern the network and their implementation.

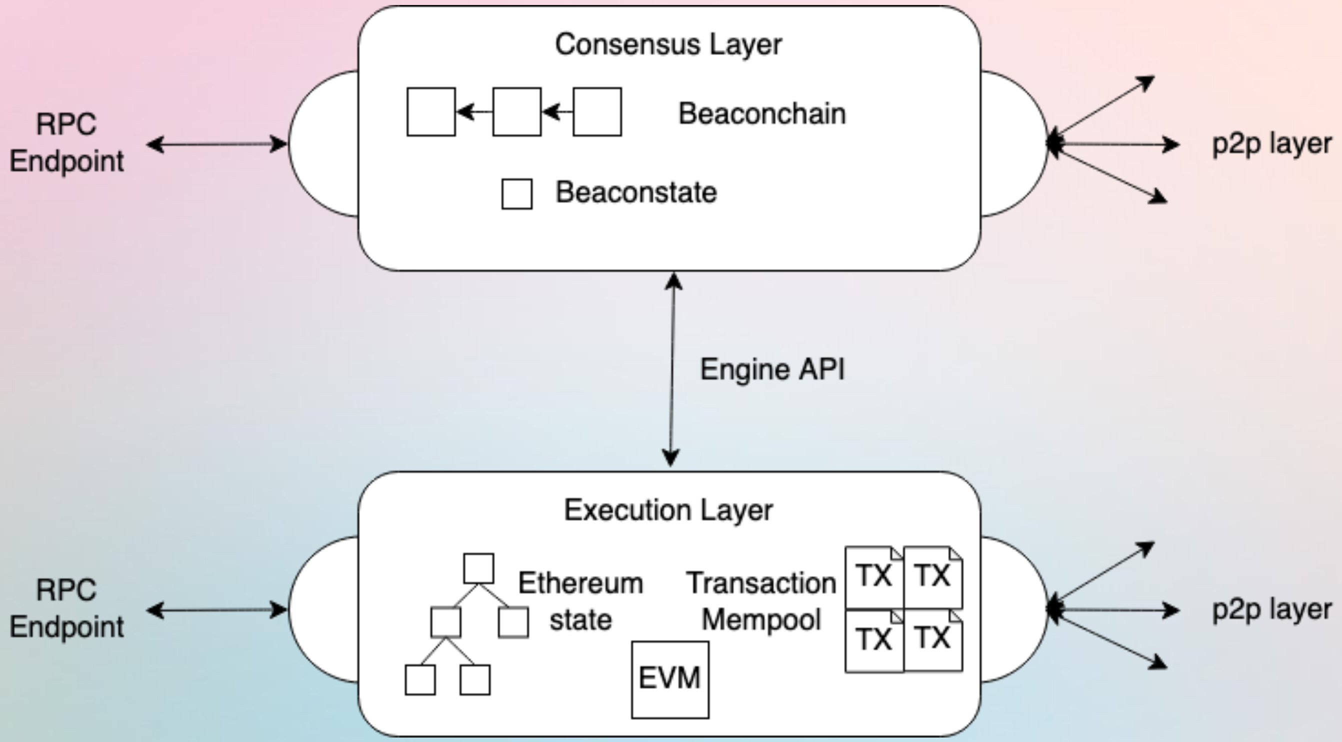
- Research
- Specification
- Execution Clients
- Consensus Clients
- Testing

and isn't.

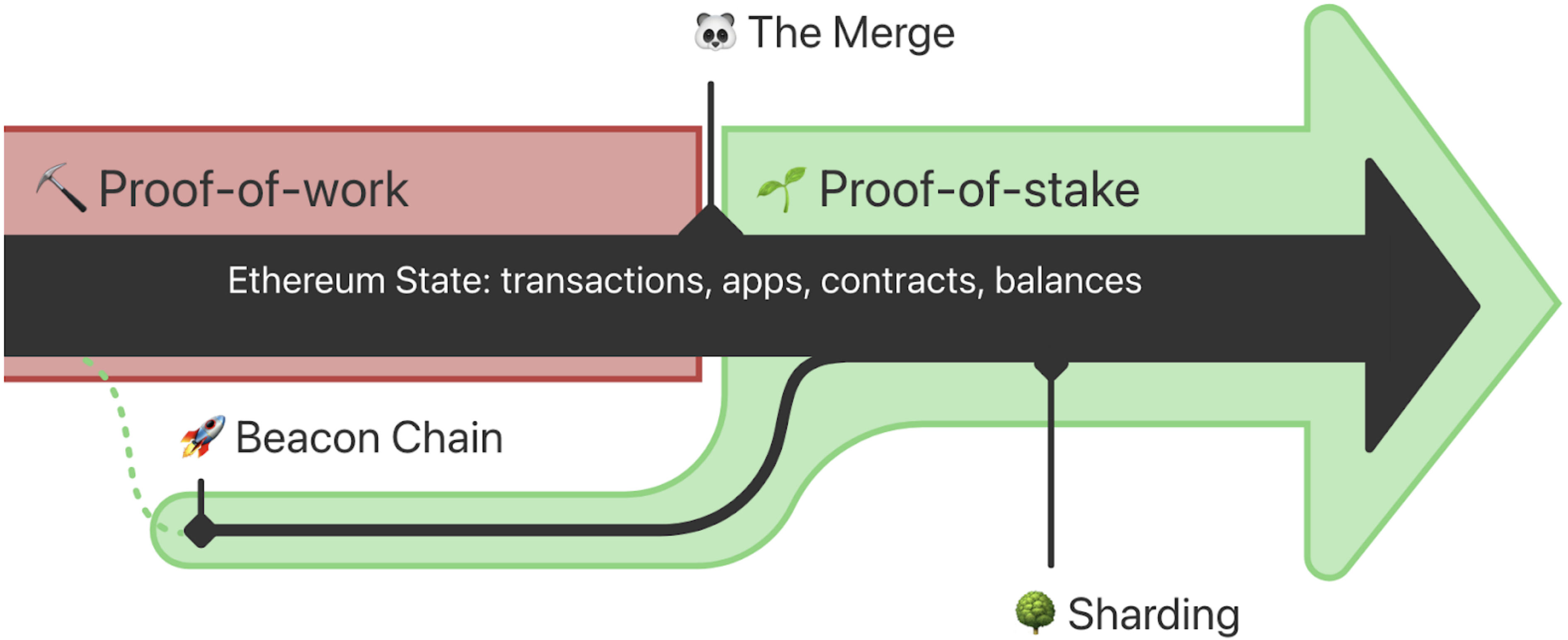
Many tools and programs are built on top of the Ethereum core protocol, but are not part of the core protocol

- A defi application
- An NFT aggregator
- Infrastructure
- Programming language
- Layer 2s

What does Ethereum look like today?



How did I help?

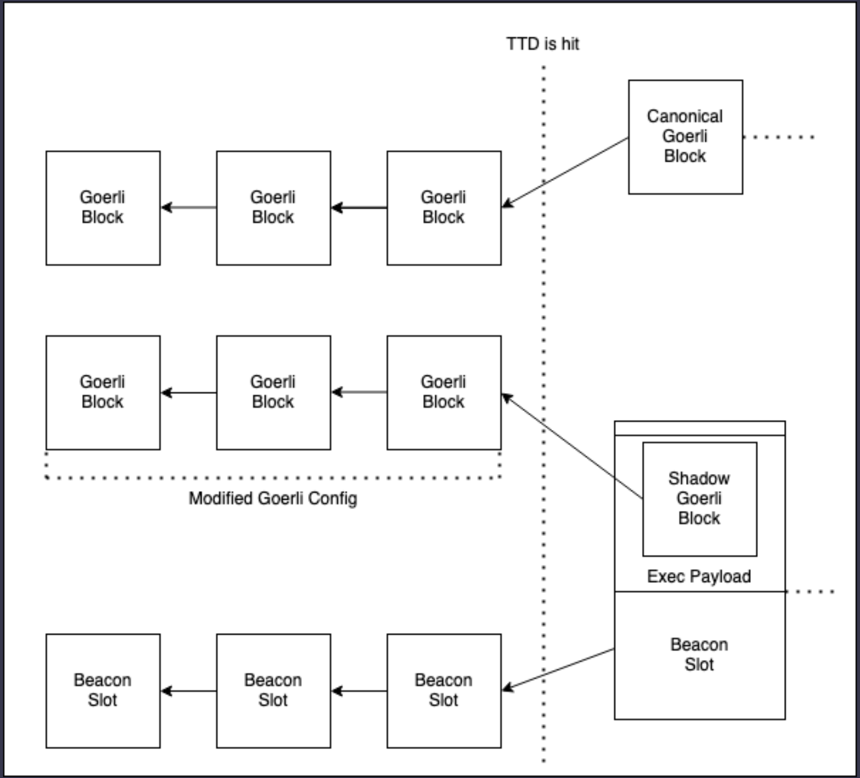

🐼 The Merge
⛏️ Proof-of-work
🌱 Proof-of-stake
Ethereum State: transactions, apps, contracts, balances
🚀 Beacon Chain
🌳 Sharding

What was complicated about it?

- >20 client combinations need to be tested & Regressions can sneak in very easily
- Communicating and debugging various client combinations
- Figuring out how to test this in a reliable manner! - We just had hive tests till now
- All future upgrades will inherit some of the complexity - build once, use many
- Competences for ELs and CLs are quite separate

Enter…Shadow forks

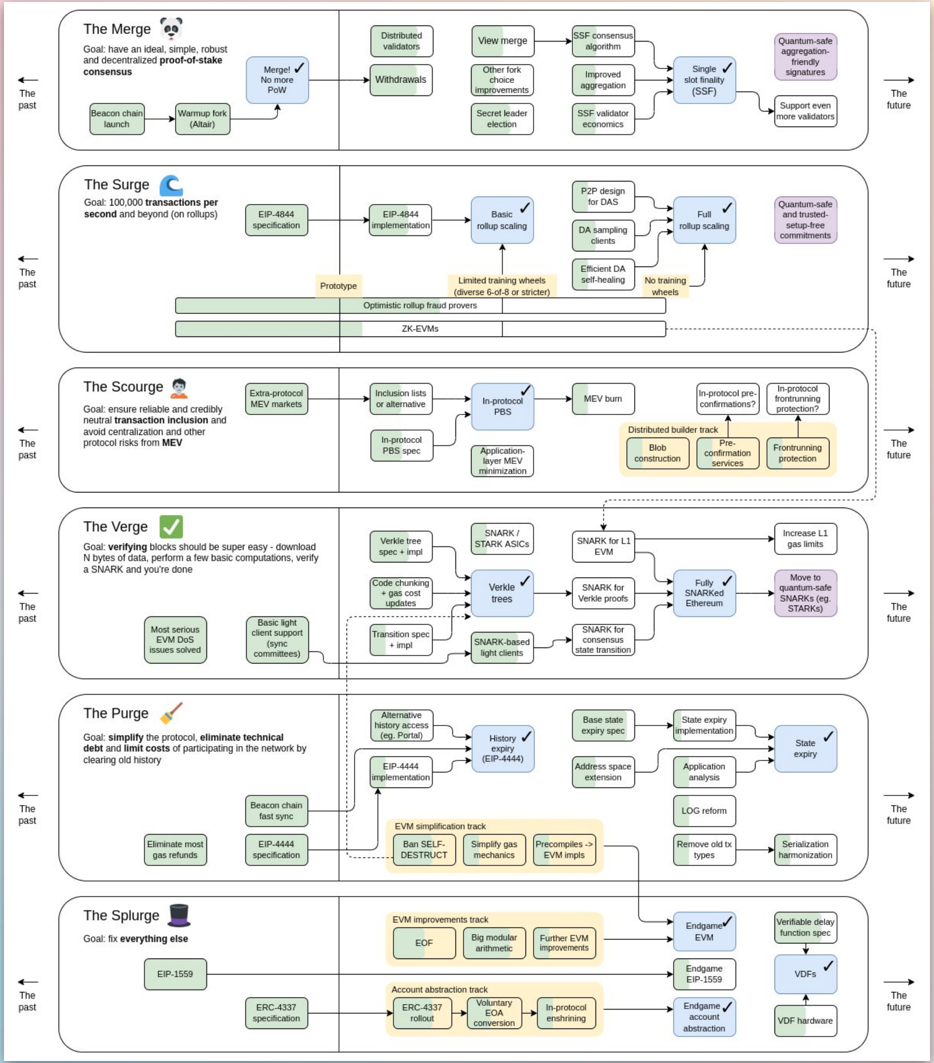
- Allows us to check compatibility across all clients through the entire lifecycle
- Fresh testnets allow us to check assumptions across client pairs without much overhead
- Shadow forks allow us to stress test the clients with real state and transaction load
- We can invite participants in a controller manner to take part in the tests
- Acts as release test which triggers real world edge cases, before we recommend the releases to the general public



Is Ethereum finished?

Protocol Roadmap

“Ethereum is still only about 55% finished”



The Merge
Goal: have an ideal, simple, robust and decentralized proof-of-stake consensus
Beacon chain launch
Warmup fork (Altair)
Merge! No more PoW
Distributed validators
Withdrawals
View merge
Other fork choice improvements
Secret leader election
SSF consensus algorithm
Improved aggregation
SSF validator economics
Single slot finality (SSF)
Quantum-safe aggregation-friendly signatures
Support even more validators
The Surge
Goal: 100,000 transactions per second and beyond (on rollups)
EIP-4844 specification
EIP-4844 implementation
Basic rollup scaling
P2P design for DAS
DA sampling clients
Efficient DA self-healing
Full rollup scaling
Quantum-safe and trusted-setup-free commitments
Prototype
Limited training wheels (diverse 6-of-8 or stricter)
No training wheels
Optimistic rollup fraud provers
ZK-EVMs
The Scourge
Goal: ensure reliable and credibly neutral transaction inclusion and avoid centralization and other protocol risks from MEV
Extra-protocol MEV markets
Inclusion lists or alternative
In-protocol PBS spec
In-protocol PBS
Application-layer MEV minimization
MEV burn
In-protocol pre-confirmations?
In-protocol frontrunning protection?
Distributed builder track
Blob construction
Pre-confirmation services
Frontrunning protection
The Verge
Goal: verifying blocks should be super easy - download N bytes of data, perform a few basic computations, verify a SNARK and you're done
Most serious EVM DoS issues solved
Basic light client support (sync committees)
Verkle tree spec + impl
Code chunking + gas cost updates
Transition spec + impl
SNARK / STARK ASICs
Verkle trees
SNARK-based light clients
SNARK for L1 EVM
SNARK for Verkle proofs
SNARK for consensus state transition
Fully SNARKed Ethereum
Increase L1 gas limits
Move to quantum-safe SNARKs (eg. STARKs)
The Purge
Goal: simplify the protocol, eliminate technical debt and limit costs of participating in the network by clearing old history
Eliminate most gas refunds
Beacon chain fast sync
EIP-4444 specification
Alternative history access (eg. Portal)
EIP-4444 implementation
History expiry (EIP-4444)
EVM simplification track
Ban SELF-DESTRUCT
Simplify gas mechanics
Precompiles -> EVM impls
Base state expiry spec
Address space extension
State expiry implementation
Application analysis
LOG reform
Remove old tx types
State expiry
Serialization harmonization
The Splurge
Goal: fix everything else
EIP-1559
ERC-4337 specification
EVM improvements track
EOF
Big modular arithmetic
Further EVM improvements
Account abstraction track
ERC-4337 rollout
Voluntary EOA conversion
In-protocol enshrining
Endgame EVM
Endgame EIP-1559
Endgame account abstraction
Verifiable delay function spec
VDFs
VDF hardware
The past
The future

Exploring the problem space

Scalability

Design/drawbacks:

Current Ethereum design is monolithic:

- One chain
- One state
- State exists forever: State growth

Potential approaches to fix it:

- Increase block size (Gas limit): Talk on “Where TPS meets Physics(Pèter Szilági)”
- Sharding: Communication overhead between shards, upgrades
- Layer 2s: Held back from full potential due to block space costs

What is a Layer-2?

- Layer 2(L2) is a collective term for a specific type of scaling solution

- An L2 is a **separate blockchain** that extends Ethereum and inherits its security guarantees

- Examples: Arbitrum/Optimism on Ethereum and Lightning network on Bitcoin

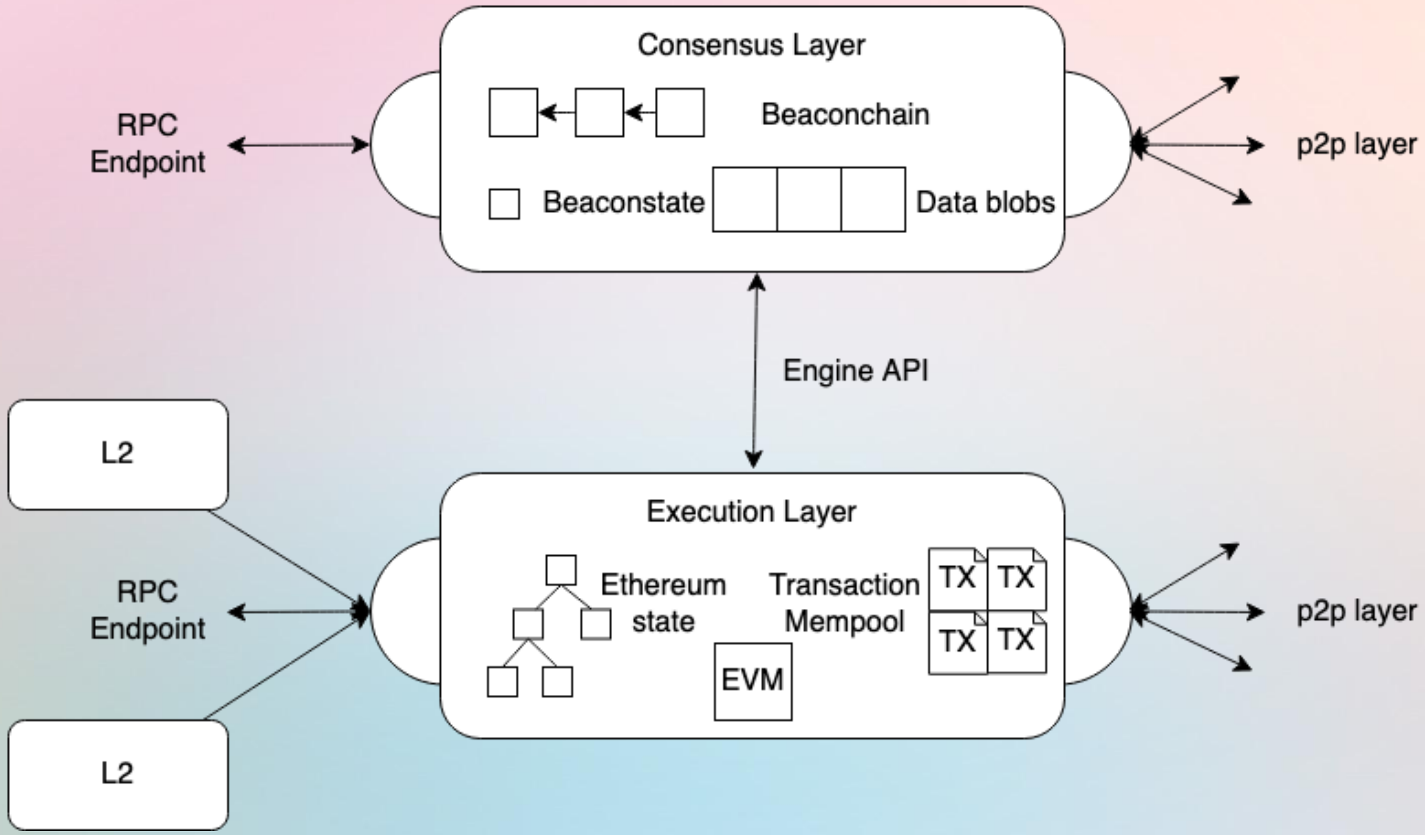
- Rollups bundle transactions on L1, distributing gas costs across its users: Optimistic and Zero knowledge rollups

Data Sidecars
Data Availability Sampling(DAS)

What is Proto-Danksharding?

- Also known as EIP-4844,
- Introduces a concept of data blobs
- Data blobs are not accessible via the EVM: allows for cheap costs
- They are deleted after ~2 weeks, limiting state bloat
- They can be ~512kb-2MB large: pending decision
- L2s ~100x cheaper!

How would Ethereum look post EIP-4844?



Statelessness

Design/Drawbacks

Current Ethereum design is difficult to build proofs for:

- Merkle tree needs large proofs
- Larger state => Larger Merkle proofs

Potential approaches to fix it:

- State expiry approach: Delete data that hasn't been accessed in x time
- Weak statelessness: Block producers need to store state, the rest can verify statelessly. Needs a switch to Verkle trees.

What are Verkle trees? What do they solve?

- Constructed similarly to Merkle Trees, but using Vector Commitments(special hash function) rather than regular cryptographic hash function
- In a Merkle Tree, a parent node is the hash of its children. In a Verkle Tree, a parent node is the Vector Commitment of its children
- Merkle tree of n leaves has O(log2 n)-sized proofs
- Verkle trees are of constant proof size but construction time is O(n2)
- Verkle tree offers constant proof size => At the cost of construction time
- Constant proof size => Easier to validate
- Higher construction time => Harder to build blocks

Merkle -> Verkle, How even?

We considered a few approaches:

- Local bulk: Every node converts locally, 2x state for a short time
- Conversion node: Some special nodes convert state, rest sync once done
- State expiry method: Only carry “hot” data, let rest expire
- Overlay method: Convert and delete x% of the state every block

Current winner: Overlay method

- It might be fast enough to finish the fork in a reasonable amount of time
- Network is live while it happens
- Easier to test and configure sync for as its deterministic

https://notes.ethereum.org/@parithosh/verkle-transition for deep dive!

UX
Account Abstraction

Design/Drawbacks

Current Ethereum design is that Exernally Owned Accounts(EOA) trigger events:

- Hard to batch transactions
- User always needs Eth to cover gas
- User loosing private key => looses access to EOA

Potential approaches to fix it:

- Upgrade smart contracts to allow triggers: nonce gets hard to maintain
- Add a out of protocol user call: Only opt-in, would not fix the issue for all accounts
- Allow EOAs to delegate to a contract: Bug in contract would be horrible

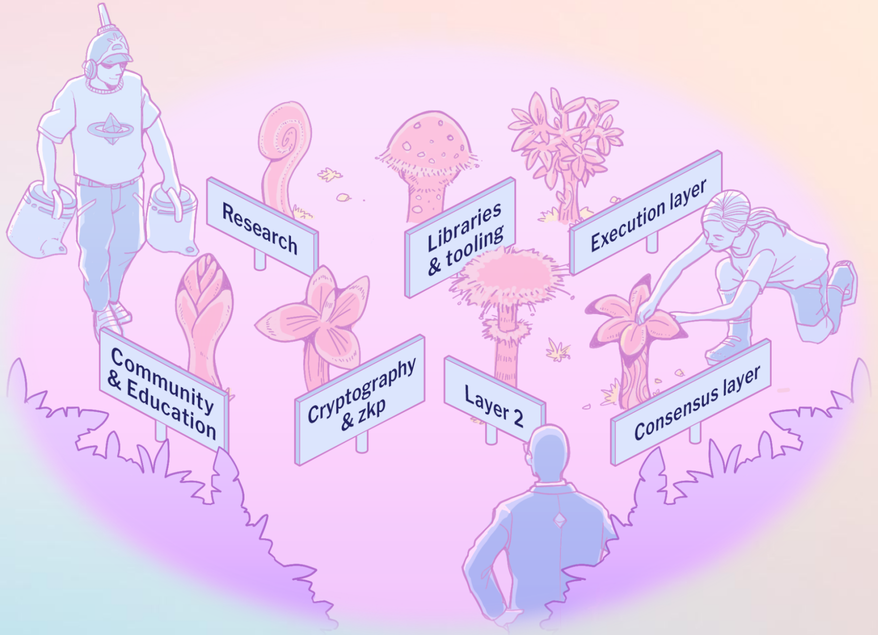
No current winner, but slight preference for out of protocol approach (EIP-4337)

How can you take part?

I DONT KNOW HOW TO CONTRIBUTE TO THE CORE PROTOCOL
JOIN THE EPF!!
imgflip.com

Some projects from the third cohort

- Teku light client
- Verkle tree research
- MEV games
- Ethereum monitor
- Relay trust research
- Prysm validator for Beacon API
- Portal Network Ultralight Client
- Account abstraction bundler
- Reward API



Run a node!

https://esp.ethereum.foundation/run-a-node-grants

Join an internship program at a client team
https://nethermind.io/internship-program

Join the public discussions!

Ethereum R&D Discord

Ethereum Privacy and Scaling Discord

https://ethresear.ch/

https://www.youtube.com/@EthereumProtocol

Thank you!

Parithosh Jayanthi
DevOps Engineer, Ethereum Foundation
parithosh@ethereum.org

@parithosh_j