kyber
network

Ethereum and EVM

Dr. Nguyễn Kim Trọng
Kyber Research

Table of Contents

Ethereum vs Bitcoin

- Account-based vs UTXO-based
- Smart-contract vs Script
- EVM vs Bitcoin's scripting language
- Hard-fork culture vs soft-fork culture

Account-based vs UTXO-based

- **Bank account** as an analogue
- **Account** is the basic unit
- Transactions updates **accounts' state**
- World state includes **accounts**
- More efficient memory usage
- Better fungibility.
- Enable **Smart-contract** and complex logic

- **Cash** as an analogue
- **Unspent Transaction Output** is the basic unit
- Transactions spend old **UTXO** and create new **UTXO**
- World state includes **UTXOs**
- Harder to link transactions

Smart-contract

- Store “in” the **smart-contract account**
- Complex logic
- Divers functionalities
- Storage (state)
- Call among smart-contract

vs

Script

- Store “in” each **UTXO**
- Simple logic
- Use for ownership and spending conditions verification
- Stateless
- No interactions among script

EVM vs Bitcoin's scripting language

- **Computational model.**
- Complex for parallelization (no for actual)
- Access to storage (world state)
- Turing complete

- **Verification model**
- Simple in parallelization
- Stateless
- Turing incomplete

Hard-fork culture

vs

Soft-fork culture

- Non-upgrade nodes **don't** accept any upgraded blocks
- -> Non-upgrade nodes will definitively form separated networks
- A success hard-fork requires the “social consensus” of the majority of **nodes**

- Non-upgrade nodes **accept** every upgraded blocks
- -> Non-upgrade nodes can still stay inside the same network with upgraded nodes
- A success soft-fork requires the “social consensus” of the majority of **miners**. Nodes can gradually upgrade later.

Ethereum overview

- EOA and nonce
- Smart-contract and call
- Block and transaction
- Storage and historical data
- Node and miner/validator

EOA and nonce

EOA (Externally Owned Account)

- Controlled by ECDSA signing key pair (pk,sk).
- sk: signing key known only to account owner

Nonce

- = (#Tx sent) + (#accounts created)
- Anti-replay mechanism

Smart-contract and call

Smart-contract

- Controlled by code
- Code set at account creation time
- Immutable

Call/Message

- Trigger a method in the contract with ETH & data
- A transaction can be a call from an EOA to a smart-contract
- Composability: Smart-contracts can call each other many times in a transaction (internal calls)
  - contract → EOA: contract sends funds to user
  - contract → contract: one program calls another (and sends funds)

Smart-contract vs EOA

Account data	Owned	Contracts
address (computed):	H(pk)	H(CreatorAddr, CreatorNonce
code:	⊥	CodeHash
storage root (state):	⊥	StorageRoot
balance (in Wei):	balance	balance (10^{18} Wei = 1 ETH)
nonce:	nonce	nonce

(#Tx sent) + (#accounts created): anti-replay mechanism

Block and transaction

Transaction types

- EOA → EOA: transfer ETH between users
- EOA → smart-contract: call contract with ETH & data

Transaction's content

- **To**: 32-byte address of target (0 → create new account)
- **From**, [Signature]: initiator's address (calculated from signature on Tx)
- **Value**: # Wei being sent with Tx
- **Tx fees** : gasLimit, maxFee, (maxPriorityFee after EIP 1559)
- If **To** = 0: create new contract code = (init, body)
- If **To** ≠ 0: data (what function to call & arguments)
- **Nonce**: must match current nonce of sender (prevents Tx replay)

Block and transaction

fd8c36118...	Transfer	12268168	774 days 2 hrs ago	0xFdddf8...E916aEf9	→	0x3f5b5A...0a911db9	0.00274119 ETH	0.0021
61984549f...	Transfer	12268168	774 days 2 hrs ago	0x10D450...67144C55	→	0x3f5b5A...0a911db9	0.00274119 ETH	0.0021
e1ec099c8...	Transfer	12268168	774 days 2 hrs ago	0xB620a1...5d1A8165	→	0x3f5b5A...0a911db9	0.00274119 ETH	0.0021
79e8475ca...	Transfer	12268168	774 days 2 hrs ago	0xbACED1...7837c1aF	→	0x3f5b5A...0a911db9	0.00274119 ETH	0.0021
63eea369...	Swap	12268168	774 days 2 hrs ago	0x983376...916f3bC8	→	MEV Bot: 0x000...9d2	0 ETH	0.0044749
5892ac381...	Passive Inco...	12268168	774 days 2 hrs ago	0x21F179...880Cf49e	→	MEV Bot: 0x000...f56	0 ETH	0.00297531
a3788899a...	Vote	12268168	774 days 2 hrs ago	0xb2db7f...3481BD7e	→	Kyber: DAO	0 ETH	0.0256947

Etherscan

Block and transaction

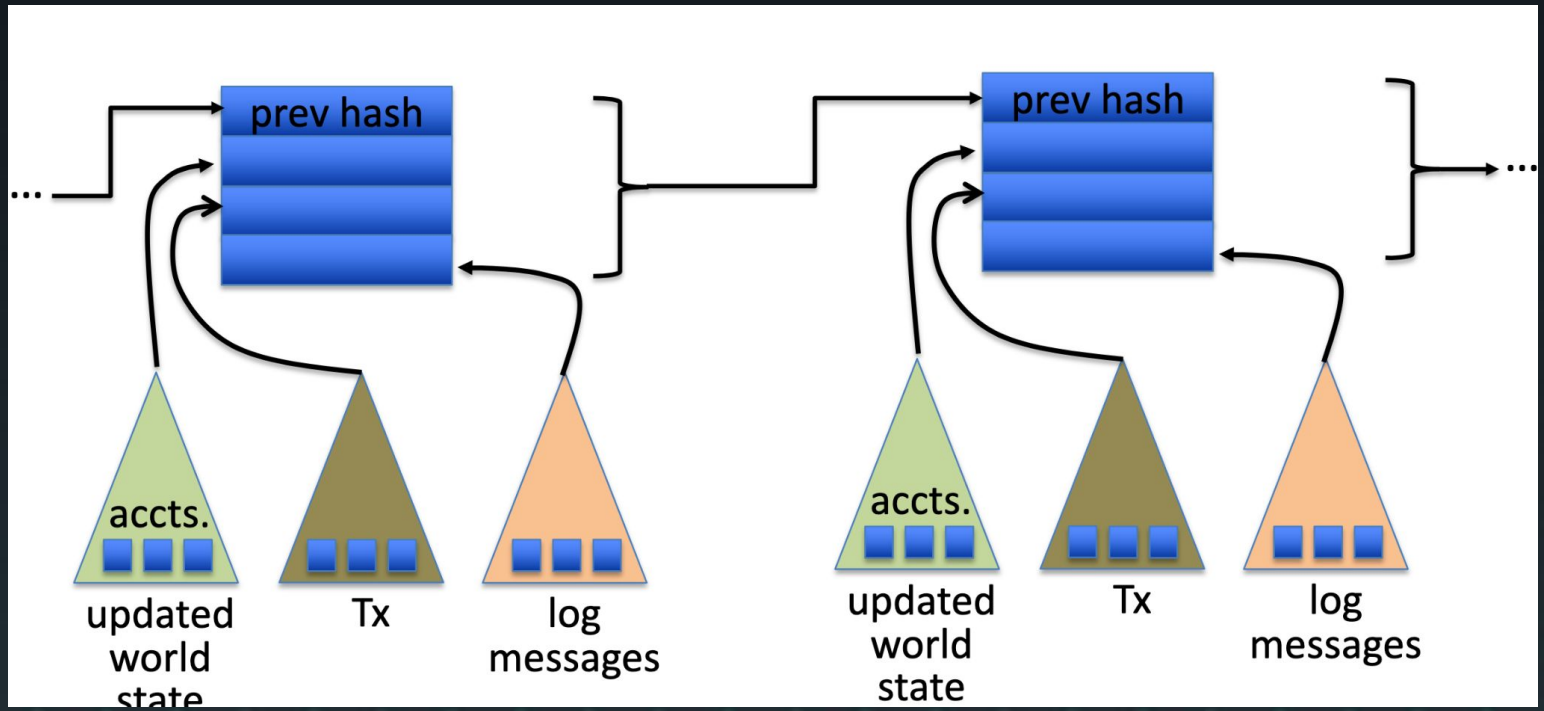
Block

- Includes n Txns
- Execute state change of Txn_i sequentially by EVM
- Record updated world state in block

Block header data

- **Consensus data:** proposer ID, parent hash, votes, etc.
- **address** of gas beneficiary: where Tx fees will go
- **world state root**: updated world state
  - ➢ Merkle Patricia Tree hash of all accounts in the system
- **Tx root**: Merkle hash of all Tx processed in block
- **Tx receipt root**: Merkle hash of log messages generated in block
- **Gas used**: used to adjust gas price (target 15M gas per block)

Storage and historical data



Stanford cs251

Storage and historical data

Storage

- Persistent **storage** that every full-node must maintain and keep forever (pre state/history expiry)
- Accessible to future state calculations (EVM)

Historical data

- Txns' data, receipt logs
- Appear in the blockchain (full-node will keep only the most recent 128 blocks), and are accessible to light clients, but that are not accessible to future state calculations.

Others

- Execution data: execution trace of a transaction, gas profile, ...
- “Off-chain data”: Mempool, smart-contract high level code,...

Node and miner/validator

Node

- Execution client: EVM executes each txns and updates the storage (world state)
- Consensus client: Verify the validity of blocks and their components
- Full node, archive node

Miner/validator (consensus node)

- Need to include/connect to a node for block's data
- **Miner**: Create block by solving the PoW puzzle (from a block's proposal)
- Validator: Participate in block creation by validating a block's proposal for PoS

EVM

- Opcodes and precompiled smart-contracts
- Gas
- Fee
- Execution clients

Overview

Stack machine

- With JUMP -> loop, call
- max stack depth = 1024
- program aborts if stack size exceeded; block proposer keeps gas
- contract can create or call another contract

Two types of zero initialized memory

- Persistent **storage** (on blockchain): SLOAD, SSTORE (expensive)
- Volatile memory (for single Tx): MLOAD, MSTORE (cheap)
- LOG0(data): write data to log (historical data)

Opcodes and precompiled smart-contracts

Opcodes

- Solidity (or Vyper, Yul) -> Bytecode -> Opcode
- Length: 1 byte (256 opcode slots)
- 140 slots used for now

Precompiled smart-contracts

- “Solving the problem of allowing complex cryptographic computations to be usable in the EVM without having to deal with EVM overhead” (Vitalik)
- In the accessed addresses, and no accessed storage
- 9 pre-compiled contracts
- Complex execution, more inputs

Gas

- Txns in Ethereum are different in how much resources they cost -> gas concept
- Every instruction costs gas
- Tx fees (gas) prevents submitting Tx that runs for many steps
- Standard for Txns fee calculation
- Gas used for each opcodes is correspond to
  - ➢ Resources used in average case
  - ➢ Resources used in worst scenario (anti-DoS)
- Txns' **gasLimit** -> control how much gas can be used in the Txns
  - ➢ Quick balance check for node
  - ➢ If txn execution runs out of gas -> txn reverts but gas is still used (anti DoS)
  - ➢ Txns' **gasUsed** -> the actual amount of gas used in a Txn
- Blocks' **size** -> the maximum total of gas that one block can consume
  - ➢ Anti DoS + cap the maximum resource required for a full-node
  - ➢ Fairness for consensus node

Fee

Pre-EIP1559

- Every Tx contains a **gasPrice** ``bid'' (gas → Wei conversion price)
- • Producer chooses Tx with highest gasPrice (max sum(**gasPrice×gasLimit**))
  - ➢ ⟹ not an efficient auction mechanism (first price auction)

EIP-1559

- Every block has a "**baseFee**": the minimum **gasPrice** for all Tx in the block

baseFee is computed from total gas in earlier blocks:

- earlier blocks at gas limit (30M gas) ⟹ base fee goes up 12.5%
- earlier blocks empty ⟹ base fee decreases by 12.5%

} interpolate in between

If earlier blocks at "target size" (15M gas) ⟹ base fee does not change

Fee

EIP1559

- EIP1559 Tx specifies three parameters:
  - ➢ gasLimit: max total gas allowed for Tx
  - ➢ maxFee: maximum allowed gas price (max gas → Wei conversion)
  - ➢ maxPriorityFee: additional “tip” to be paid to block proposer
  - ➢ Computed gasPrice bid: gasPrice ← min(maxFee, baseFee + maxPriorityFee)
- Max Tx fee: gasLimit × gasPrice
- BURN gasUsed× baseFee
- Send gasUsed×(gasPrice – baseFee) to block produce

Execution clients

Client	Language	Operating systems	Networks	Sync strategies	State prunir
Geth ↗	Go	Linux, Windows, macOS	Mainnet, Sepolia, Goerli	Snap, Full	Archive, Pruned
Nethermind ↗	C#, .NET	Linux, Windows, macOS	Mainnet, Sepolia, Goerli, and more	Snap (without serving), Fast, Full	Archive, Pruned
Besu ↗	Java	Linux, Windows, macOS	Mainnet, Sepolia, Goerli, and more	Snap, Fast, Full	Archive, Pruned
Erigon ↗	Go	Linux, Windows, macOS	Mainnet, Sepolia, Goerli, and more	Full	Archive, Pruned

Ethereum.org

From PoW to PoS

- Beacon-chain
- Consensus clients
- The Merge
- Post-Merge

Beacon chain

PoS validator

- To become a validator: stake (lock up) 32 ETH
  - ➢ sign blocks to express correctness (finalized once enough sigs)
  - ➢ occasionally act as block proposer (chosen at random)
  - ➢ correct behavior ⇒ issued new ETH every epoch (32 slots)
  - ➢ incorrect behavior ⇒ slashed

PoS epoch

- 32 slots (can have empty slot) -> 12 *32 = 384s
- Committee of 128 validators
- The first block in each epoch is a checkpoint
- If a pair of checkpoints attracts votes representing at least two-thirds of the total staked ETH, the checkpoints are upgraded.
- Fork choice: Fork that has the greatest weight of attestations in its history

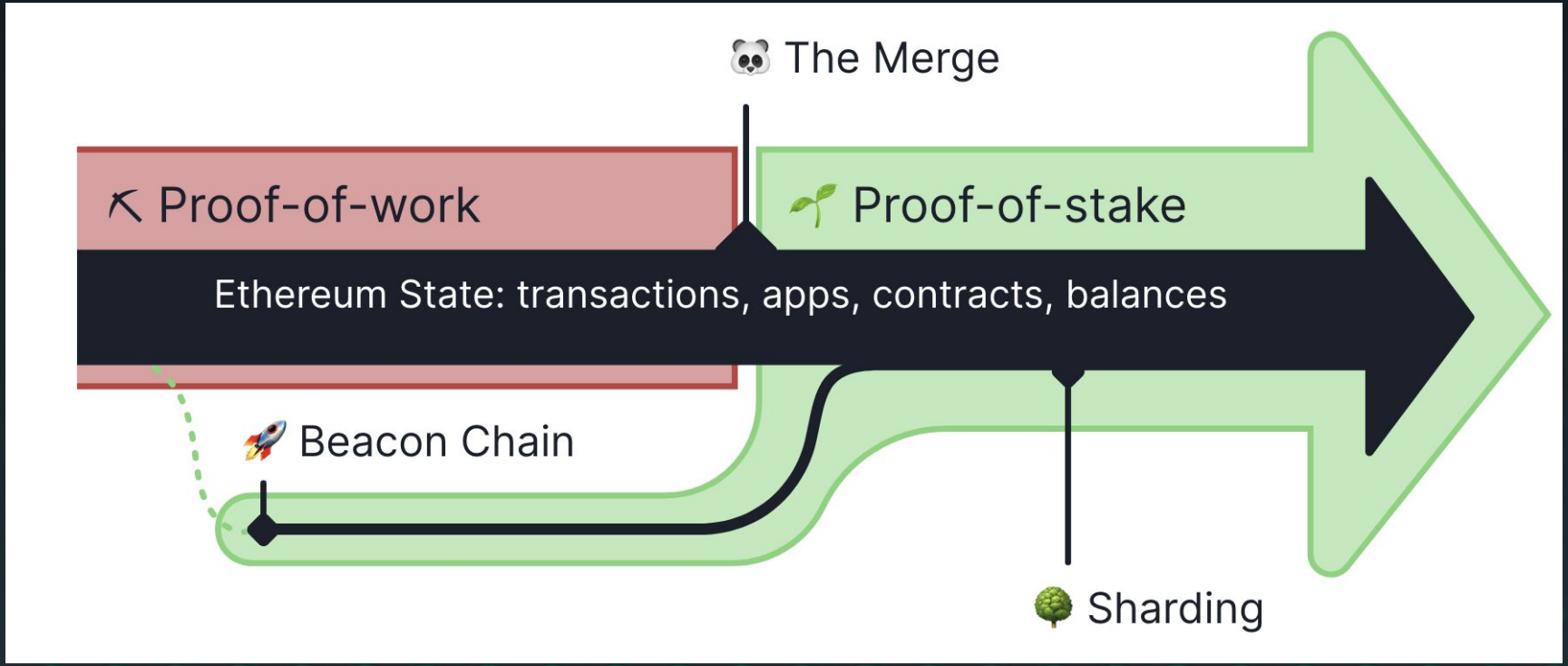
Beacon chain

Beaconcha.in

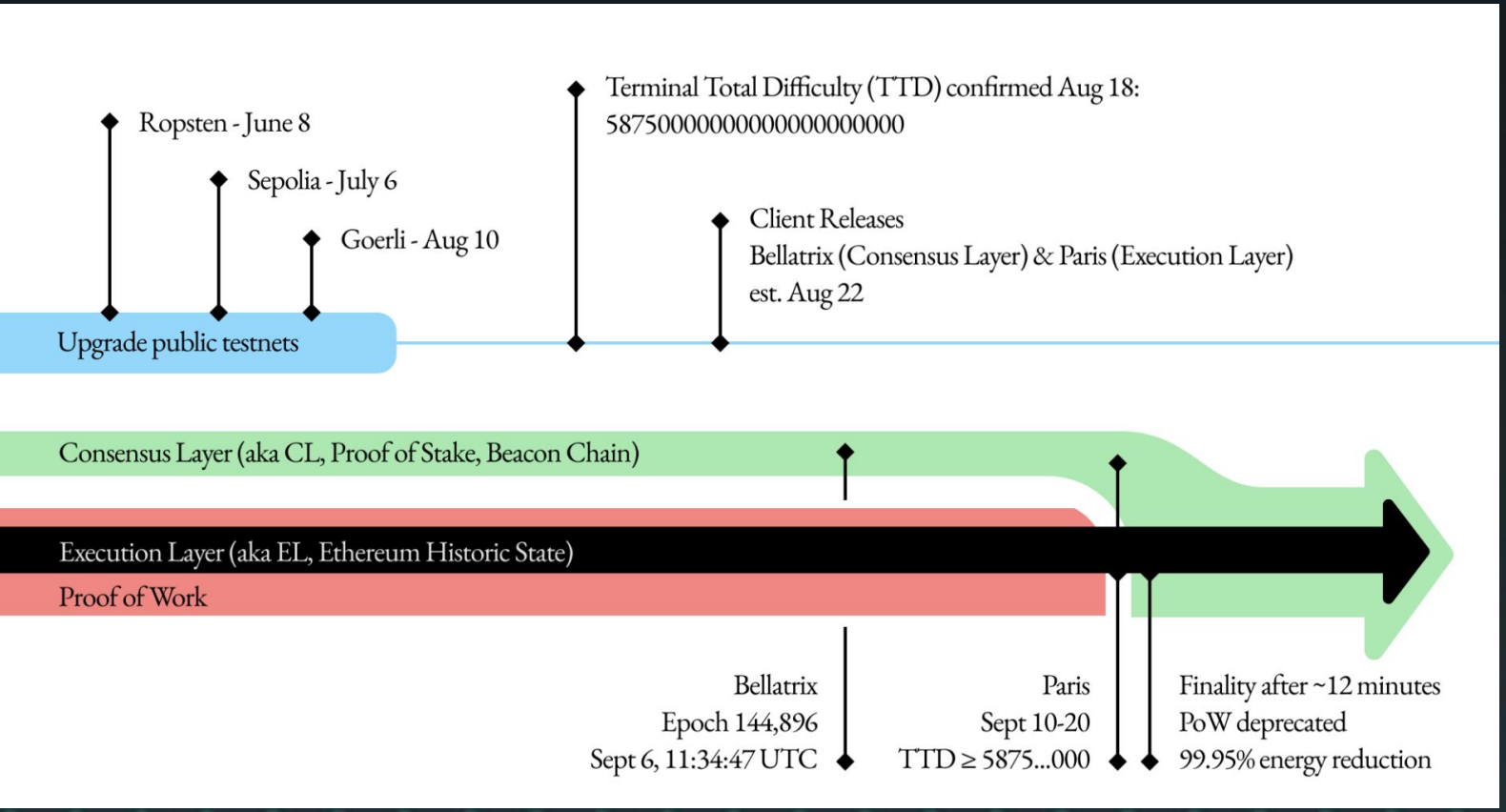
Consensus clients

Client	Language	Operating systems	Networks
Lighthouse ↗	Rust	Linux, Windows, macOS	Beacon Chain, Goerli, Pyrmont, Sepolia, Ropsten, and more
Lodestar ↗	TypeScript	Linux, Windows, macOS	Beacon Chain, Goerli, Sepolia, Ropsten, and more
Nimbus ↗	Nim	Linux, Windows, macOS	Beacon Chain, Goerli, Sepolia, Ropsten, and more
Prysm ↗	Go	Linux, Windows, macOS	Beacon Chain, Gnosis, Goerli, Pyrmont, Sepolia, Ropsten, and more
Teku ↗	Java	Linux, Windows, macOS	Beacon Chain, Gnosis, Goerli, Sepolia, Ropsten, and more

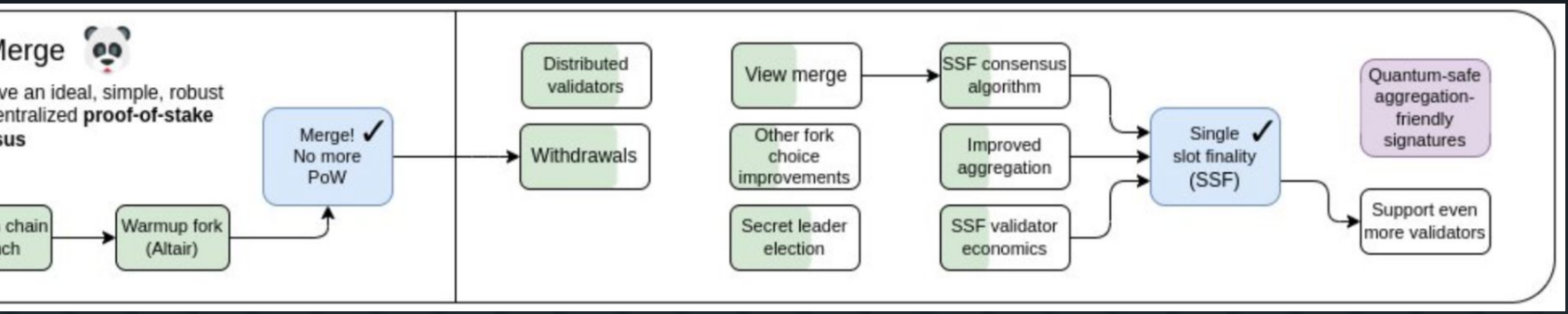
The Merge


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

The Merge


Ropsten - June 8
Sepolia - July 6
Goerli - Aug 10
Upgrade public testnets
Terminal Total Difficulty (TTD) confirmed Aug 18:
58750000000000000000
Client Releases
Bellatrix (Consensus Layer) & Paris (Execution Layer)
est. Aug 22
Consensus Layer (aka CL, Proof of Stake, Beacon Chain)
Execution Layer (aka EL, Ethereum Historic State)
Proof of Work
Bellatrix
Epoch 144,896
Sept 6, 11:34:47 UTC
Paris
Sept 10-20
TTD ≥ 5875...000
Finality after ~12 minutes
PoW deprecated
99.95% energy reduction

Post Merge

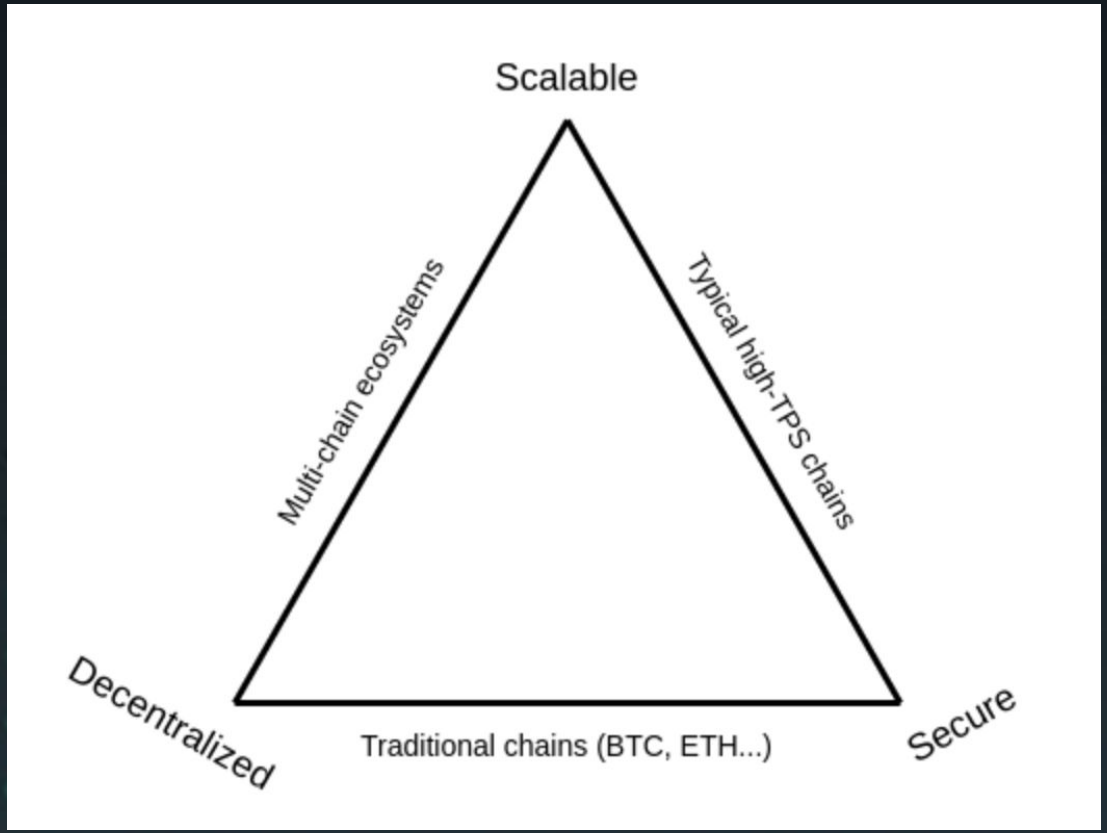


Vitalik

Ethereum's vision

- Blockchain’s trilemma
- Decentralization end-game

Blockchain’s trilemma

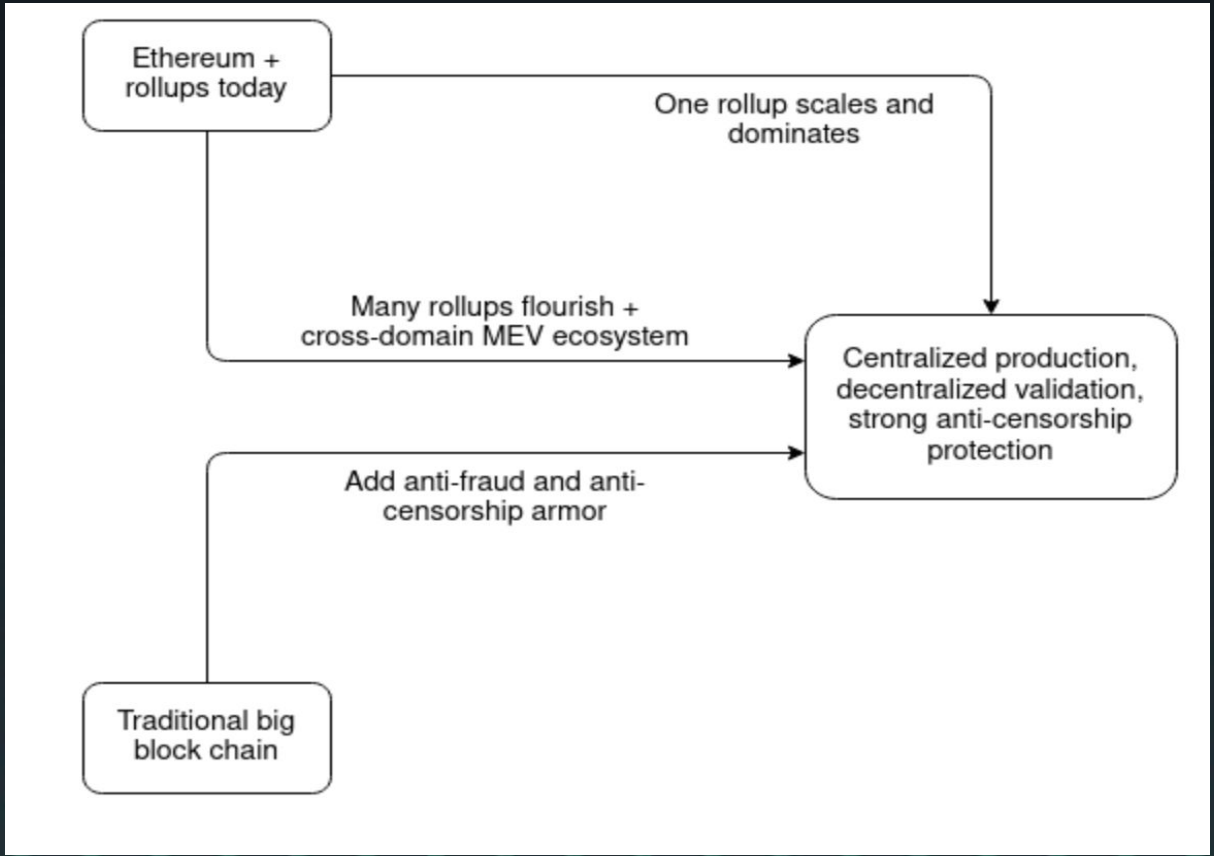


Blockchain’s trilemma

What is “decentralized”

- ⚡ Block production -> Focus on consensus nodes
- ⚡ Block verification -> Focus on general Nodes
- ⚡ Anti-censorship

Decentralization end-game

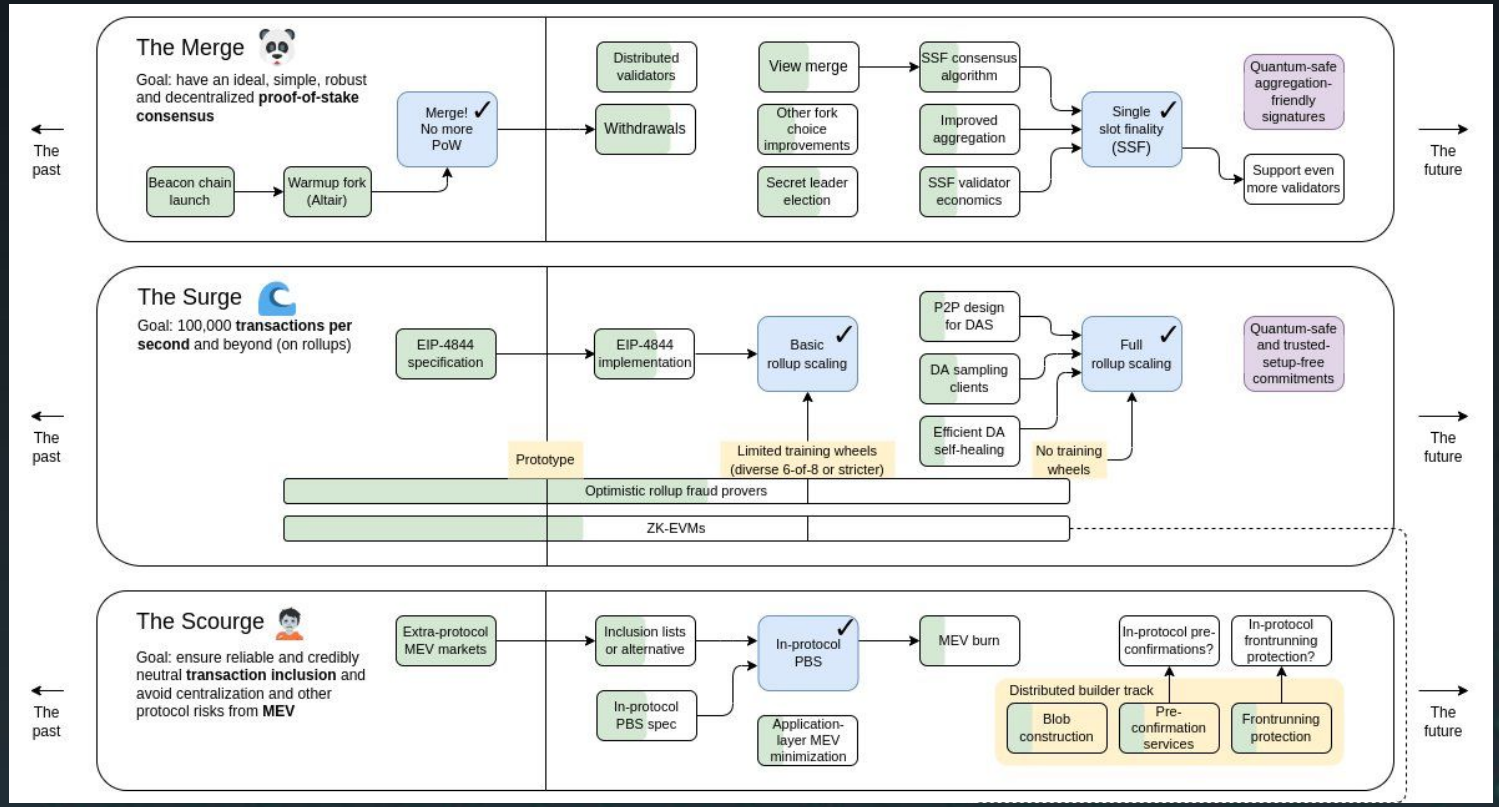


Decentralization end-game

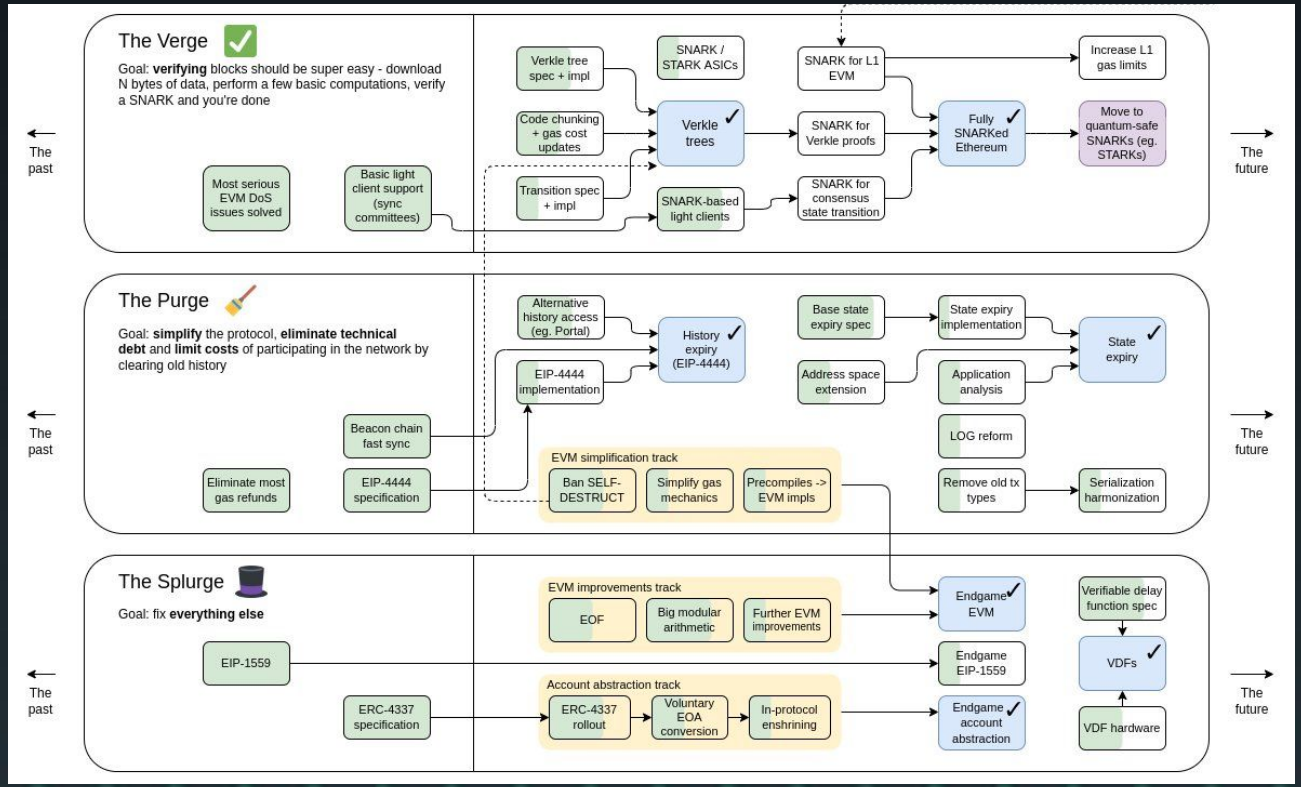
At the end of Ethereum Roadmap

- Weak decentralized in Block production
- Strong decentralized block verification
- Strong Anti-censorship

Ethereum roadmap



Ethereum roadmap



Q&A