



SUBSPACE

NETWORK

Web3 @ Internet Scale
subspace.network

Agenda

Silicon Valley Ethereum Meetup

1. Why do we need Subspace?
2. How does it actually work?
 - Consensus Layer
 - Compute Layer
 - Storage Layer
3. Applications for Ethereum

Background

How to tally votes in Nakamoto Consensus?



Proof of Work

one-CPU-one-vote



Proof of Stake

one-Coin-one-vote









Proof of Capacity

one-Disk-one-vote

Background

Storage required to run a node

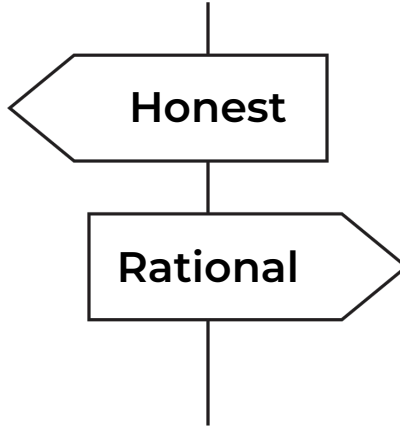
			
 Archival Node <i>full state + history</i>	403 GB	10.5 TB	???
 Full Node <i>full state</i>	4.7 GB	633 GB	123 GB
 Light Client <i>minimal state</i>	58 MB	1 GB	233 KB

Problem

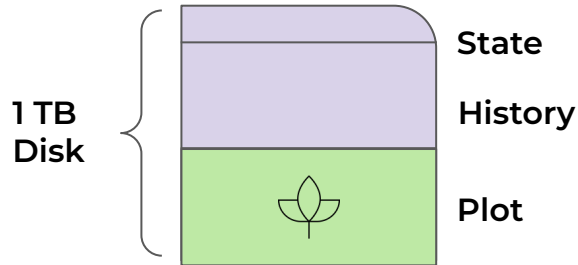
The Farmer's Dilemma



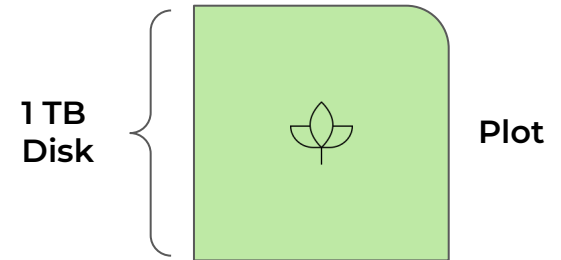
- ✓ Retain the history
- ✓ Maintain the state
- ✓ Verify new blocks



- ✗ Ignore the history
- ✗ Discard the state
- ✗ Trust new blocks



Higher Rol (\$ / TB)



Solution: Subspace

How to make PoC Incentive-Compatible

Subspace: A Solution to the Farmer's Dilemma

Jeremiah Wagstaff
Subspace Labs
Palo Alto, California
jeremiah@subspace.network

Abstract—In an effort to make blockchains more energy-efficient, egalitarian, and decentralized, several new protocols employ consensus based on *Proofs-of-Capacity* (PoC), which replace compute-intensive mining with storage-intensive farming. We observe that PoC consensus introduces a unique mechanism design challenge, referred to as *the farmer's dilemma*, which suggests that existing constructions are not actually incentive compatible. Simply put, farmers must decide whether to allocate scarce storage resources towards *either* maintaining the chain state and history or maximizing the amount of space they pledge towards consensus. Rational farmers will always choose the latter, at best becoming light clients, while at worst encouraging pooled farming under a few trusted operators. To resolve this dilemma, we introduce *Subspace*, a PoC blockchain in which farmers maintain neither the state nor the history, while retaining the security properties and decentralization benefits of a full node. Consensus in Subspace is based on proofs of replicated storage of the history of the blockchain itself. Farmers store the history collectively, many times over, with each farmer storing as many replicas as their disk space allows. Consensus and computation are then decoupled, such that farmers only propose an ordering for transactions, while staked executor nodes maintain the state and compute transitions. This separation of concerns significantly reduces the storage and compute overhead needed to operate a farmer, even in an Ethereum-style execution model, allowing for high levels of participation in consensus by ordinary users with commodity hardware.

1. BACKGROUND

Nakamoto-style blockchains, such as Bitcoin [1] and Ethereum [2], [3], combine the longest-chain fork-choice rule with a *proof-of-work* (PoW) mining puzzle. These systems are provably secure, with respect to safety and liveness, given an honest majority of miners [4]. Unlike legacy Byzantine Fault Tolerant (BFT) consensus algorithms, participation is both permissionless and scalable. These properties are the standard against which all new blockchain consensus protocols are measured. Unfortunately, the security afforded by PoW comes at a massive cost in electricity. Collectively, miners on Bitcoin and Ethereum consume the energy budget of a medium-sized country, with these numbers steadily increasing as more capital flows into the system. This raises the critical question

has access to low-cost electricity. Ethereum mining sought to circumvent this by adopting *one-GPU-one-vote*, but this too has proven susceptible to special purpose hardware and still has the tendency to concentrate in regions with low-cost electricity. This raises another key question of whether or not existing cryptocurrencies are actually decentralized, or if we have simply substituted one trusted third-party (financial institutions) for another (mining pools).

These challenges have served as a rallying cry for a diverse group of hackers, researchers, and engineers who have sought to design a sustainable blockchain that holds true to Nakamoto's vision for a more democratic and decentralized future. The most well-known solution to this problem is *proof-of-stake* (PoS), which employs a system of *virtual mining* based on one's wealth, under the adage *one-coin-one-vote*. While PoS clearly solves the sustainability problem, it does not hold true to Nakamoto's vision. It instead reflects a *permissioned* and *plutocratic* alternative, which also exhibits strong tendencies towards centralization. In fact, PoS systems serve to magnify the existing wealth disparity in cryptocurrencies, which are already significantly larger than historically high disparities in global fiat wealth distribution, effectively serving to make the rich even richer.

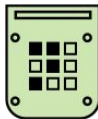
What is instead needed is a cryptographic proof system based on an underlying resource that is already massively distributed and which does not lend itself to special-purpose hardware. Enter *proof-of-capacity* (PoC)', which replaces compute-intensive mining with storage-intensive *farming*, under the maxim *one-disk-one-vote*. Disk-based consensus seems like an obvious choice, as storage hardware has long been commoditized, consumes negligible electricity, and exists in abundance across end-user devices. As it turns out, implementing a PoC such that it does not devolve back into PoW, without resorting to a *permissioned* model, is highly non-trivial, as witnessed by the paucity of live chains to date. Moreover, all existing PoC blockchain designs fail to address a critical mechanism design challenge, to which we turn next.



Incentivize nodes to store the history
Proof-of-Archival-Storage (PoAS) Consensus



Separate consensus and compute
Decoupled Execution (DecEx)



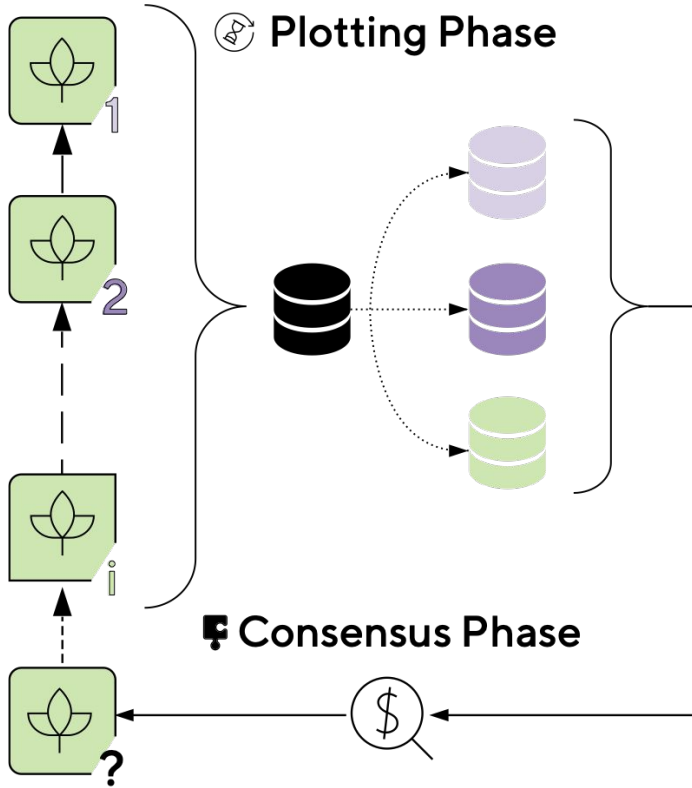
Spread the history across all nodes
Distributed Storage Network (DSN)

Consensus Layer

How to agree on transaction ordering

PoAS Consensus Layer

Proof of Archival Storage



Two Phase Protocol

1. Initial Setup Phase (plotting) – create a unique copy of the history
2. Challenge-Response Phase (Consensus) – audit the history and produce blocks

How it Compares

A permissionless proof-of-useful-storage

Proof-of-Space

useless data



Permissionless
dynamic availability

Proof-of-Storage

useful data



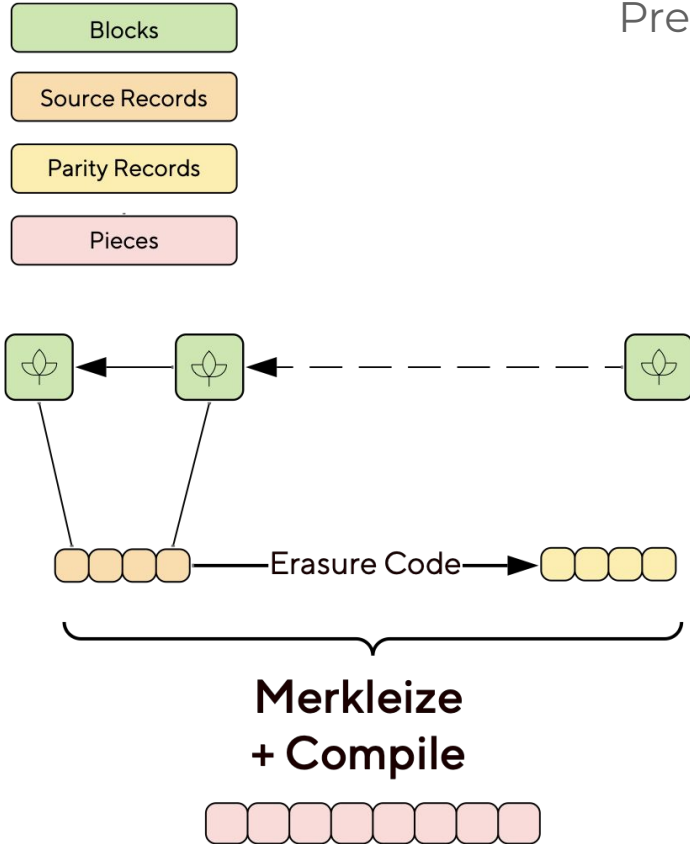
Permissioned
farmer registration

⌘ spacemesh



Archiving

Preparing the history

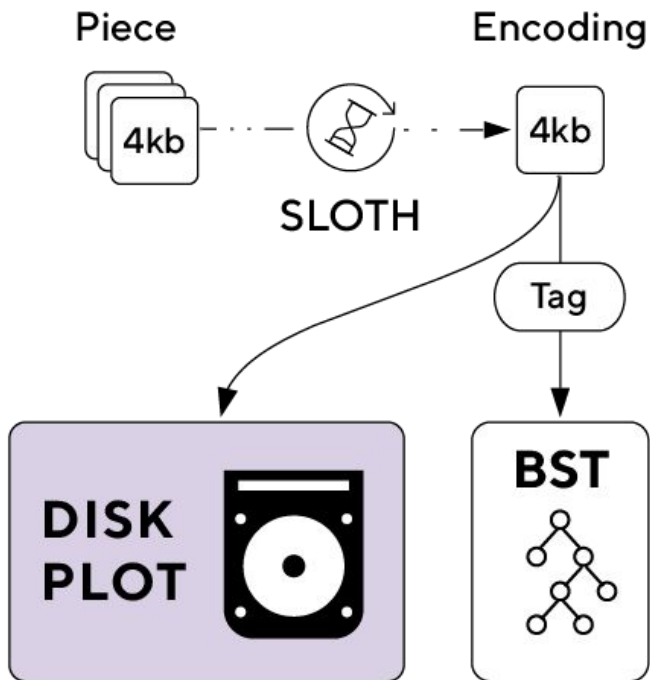


For each new block

1. Append to a buffer of some size
2. When buffer is full
 - Slice into a set of source records
 - Erasure code a set of parity records with some rate
 - Merkelize the entire record set
 - Append a merkle proof to each record, yielding a piece set
3. Commit to a root chain block

Plotting

Initial Setup Phase



For each 4096 byte piece of history

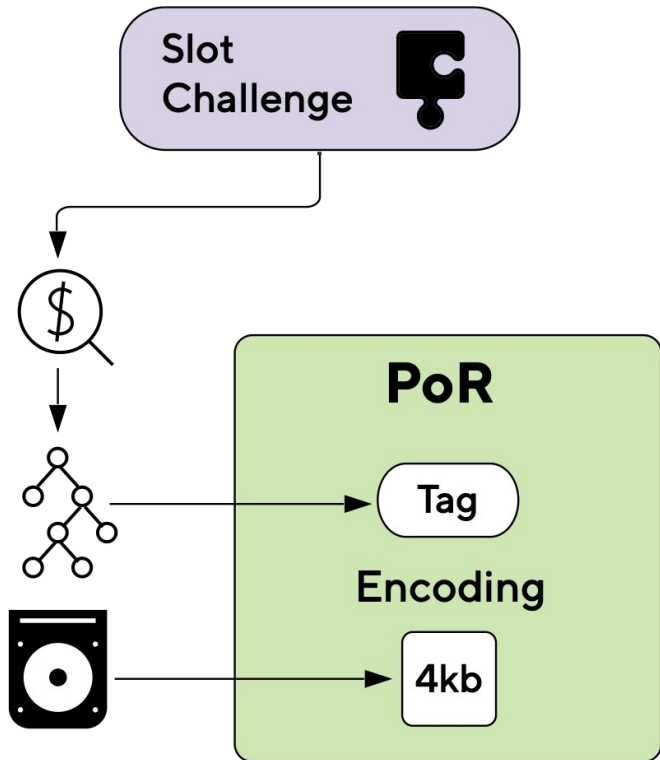
1. Encode using SLOTH-256-189 where key is hash of public key
2. Create a tag (commitment) to the encoding
3. Write the encoding to disk (plot)
4. Store the tag prefix within a Binary Search Tree (BST)

Farming

Continuous Challenge-Response Phase

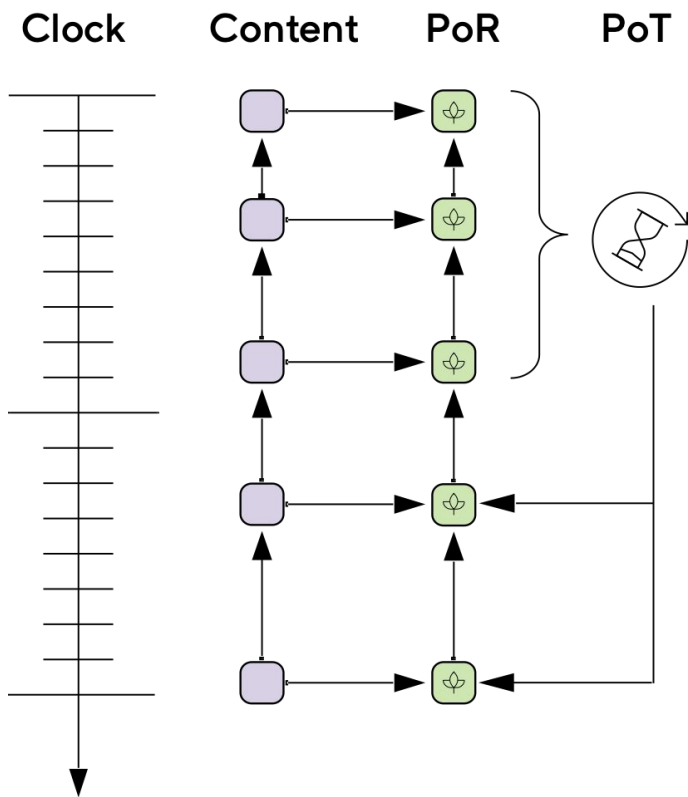
For each challenge

1. Query BST for nearest tag to the challenge by XOR distance
2. If within dynamic solution range: compile a Proof-of-Replication (PoR)
 - a. Sign the tag and challenge
 - b. Attach encoding and public key
 - c. Broadcast to the network
3. All nodes verify the PoR
 - a. Ensure tag w/in solution range
 - b. Decode and verify witness
 - c. Check the signature



Secure Farming

Rational Security Model



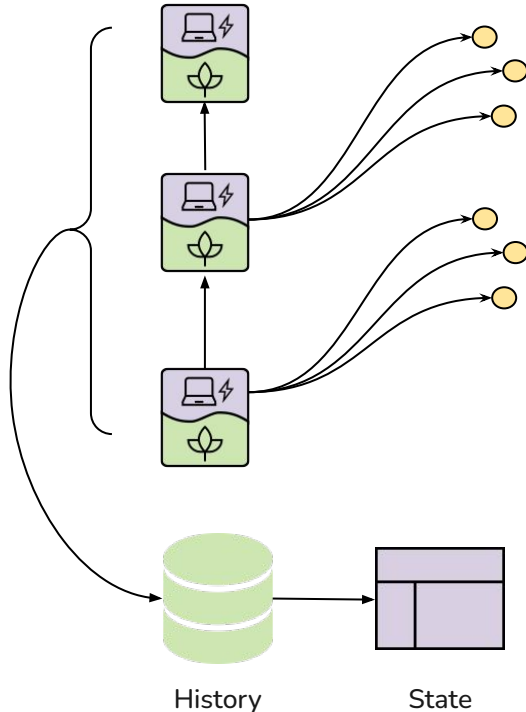
1. Prevent grinding w/block segregation
2. Discourage sybil farming w/max plot size and locally derived challenges
3. Prevent simulation with c-correlation
4. Does not rely on NTP, uses the chain as a source of relative time
5. Discourage space-time trade-off attacks with the encoding delay
6. Discourage compression farming with salted binary search trees
7. Prevent long-range attacks, mitigate bribery and space-time trade-off attacks with a proof-of-time

Compute Layer

How to agree on the global state

Decoupled Execution

Separation of Concerns



In a standard blockchain, each full node will...

- 1 Propose new blocks
- 2 Verify new transactions
- 3 Maintain chain history
- 4 Maintain chain state

***We separate these roles
between two types of nodes***



**Storage
Farmers**

Prove they are storing the
actual blockchain history

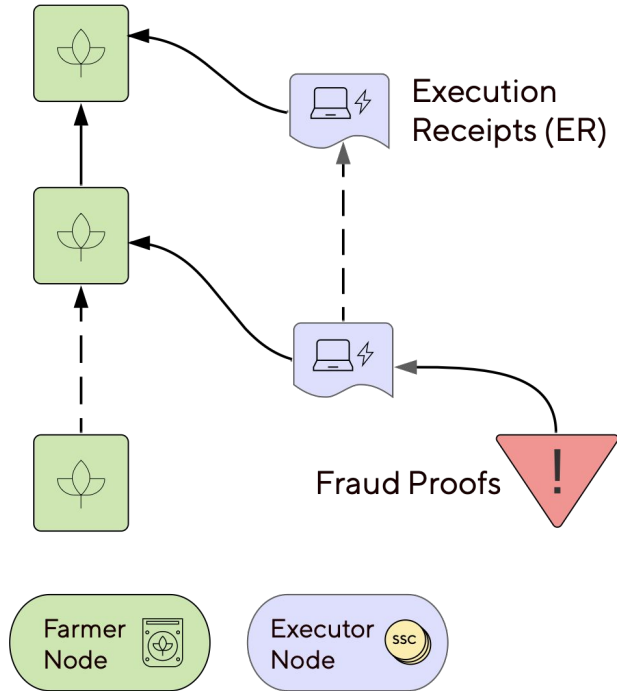


**Staked
Executors**

Prove they are holding coins
and tracking the latest state

Decoupled Execution

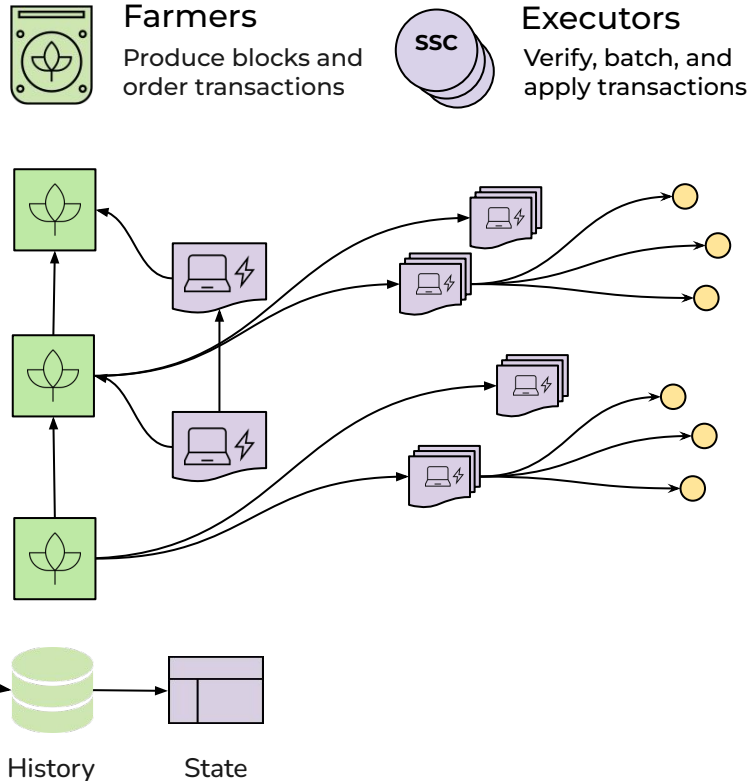
Maintaining Security



1. Farmers produce blocks
2. Executors apply blocks
3. Executors produce ERs proportional to their staked credits
4. Invalid ERs will lead to fraud proofs
5. Farmers can verify fraud proofs, leading to executors being slashed
6. Assumes at least one honest full node that is not under eclipse attack

Decoupled Execution

Vertical Scaling



1. User's generate transactions and broadcast to executors
2. Executors verify the user has funds to cover the transaction fee
3. Executors produce transaction bundles proportional to their stake
4. Farmers include include bundles into blocks, providing an ordering
5. Executors apply the bundles and generate a new state root
6. Block size not limited by network delay, but farmer bandwidth
7. Data Availability Sampling (DAS) will allow scaling to executor bandwidth

How it Compares

Eventually Eager Execution



CELESTIA

Eager Execution
Standard Model

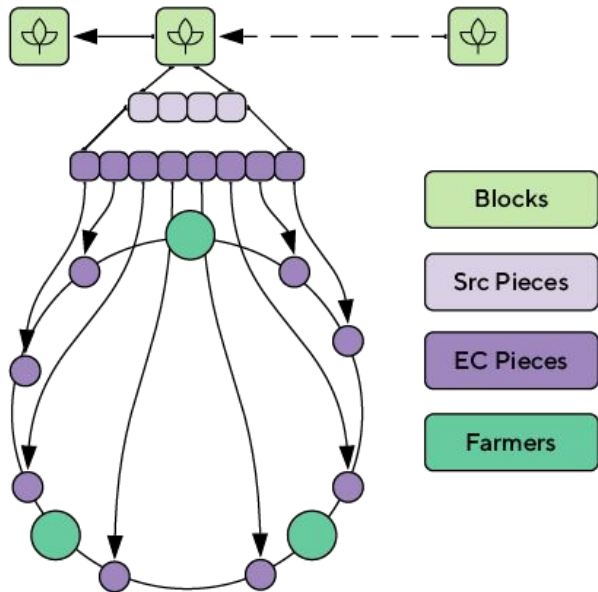
Lazy Execution
Client Side

Storage Layer

How to ensure data persists

Distributed Storage Of Subspace

How to store and retrieve our own history



Assuming a very large history, we must ensure it remains

- Durable
- Load Balanced
- Retrievable
- Efficient Sync

To achieve this we employ

- Erasure coding
- Consistent Hashing
- Light Kademlia DHT
- Super Light Client

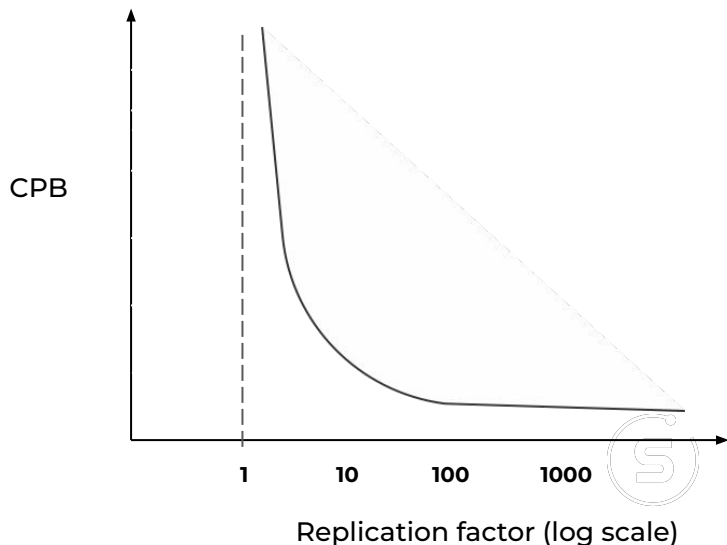
Storage Fee Pricing

Incentivizing Permanent Storage

Cost of Storage (CoS)
Subspace Credits / byte (CPB)

→

$$\frac{\sum \text{Circulating Credit Supply}}{\sum \text{Space Pledged} - \sum \text{Space Reserved}}$$



CoS is normally a constant

- op_return → satoshis / byte
- call_data → gwei / byte

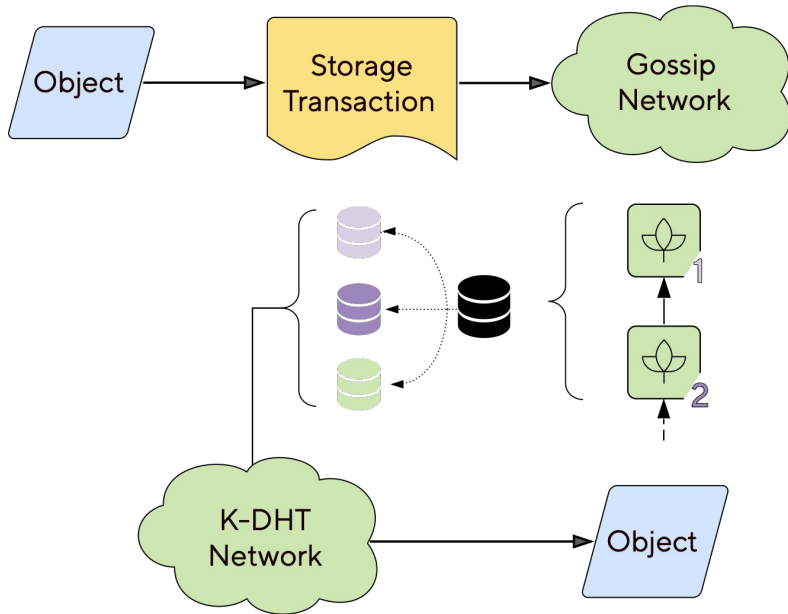
Our CoS (storage fee) is dynamic

- RF increases, fees get lower
- As RF decreases, fees get higher

Portion of fees are placed in an endowment and paid out gradually

Storage API

How to store and retrieve any data



Subspace.js – Developer SDK

put(object) → object_id

- wrap in subspace transaction
- include within a block
- store mapping on DHT
- pieces are spread across plots

get(object_id) → object

- retrieve mapping from DHT
- retrieve pieces from plots
- reconstruct object and verify

How it compares?

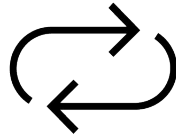
AMM for On-Chain Storage



SUPPLY

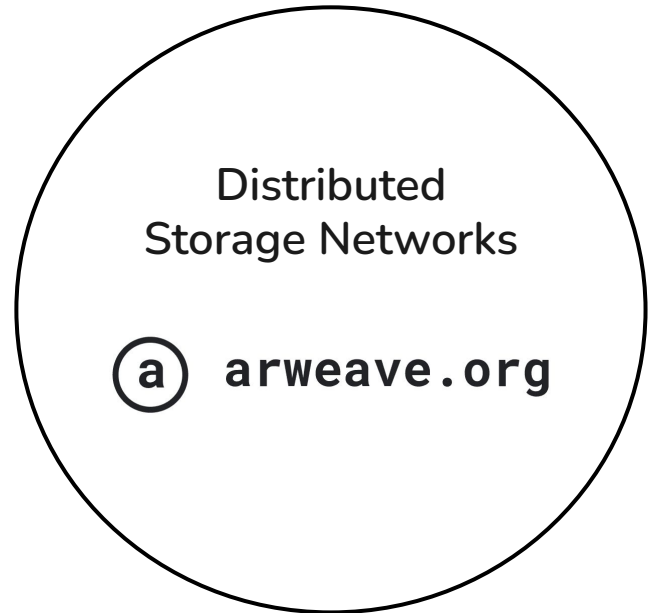


*Permanent
Storage*



Filecoin

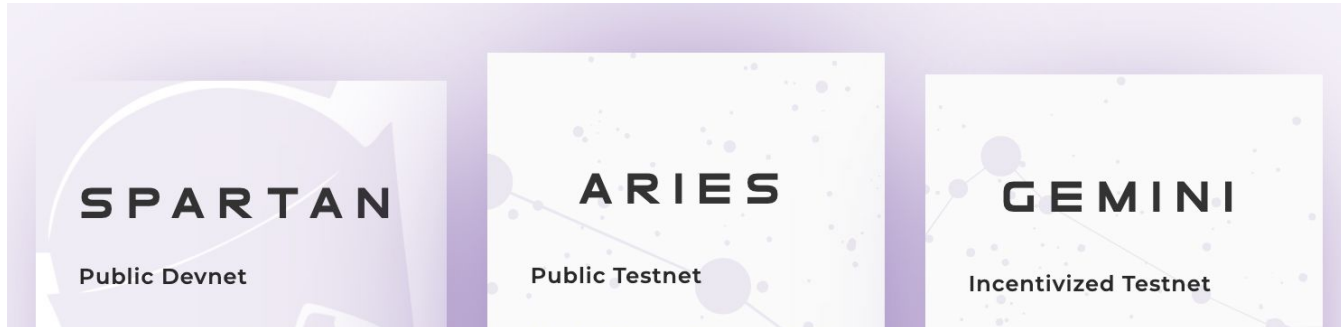
*Temporary
Storage*



DEMAND

Public Farmnet

Initial Supply



Polkadot 1734

Kusama 3295

Subspace testnet 1000

Subspace testnet 21640



BEST BLOCK

#10,165,542



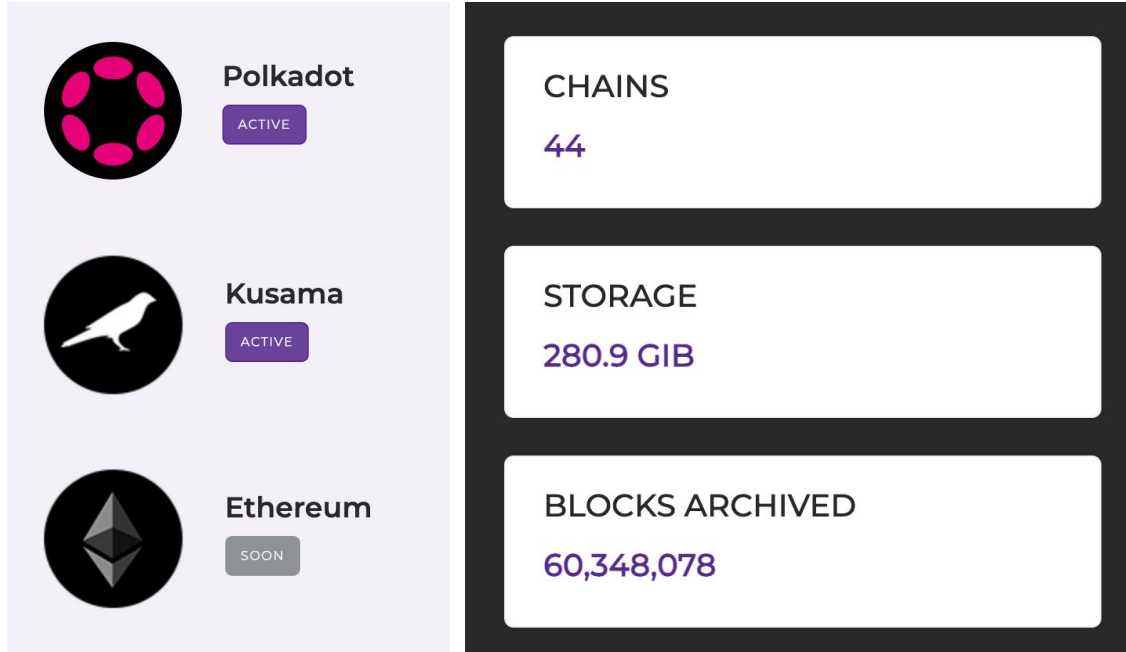
BEST BLOCK

#860,205

<https://telemetry.subspace.network/>

Subspace Relay

Initial Demand



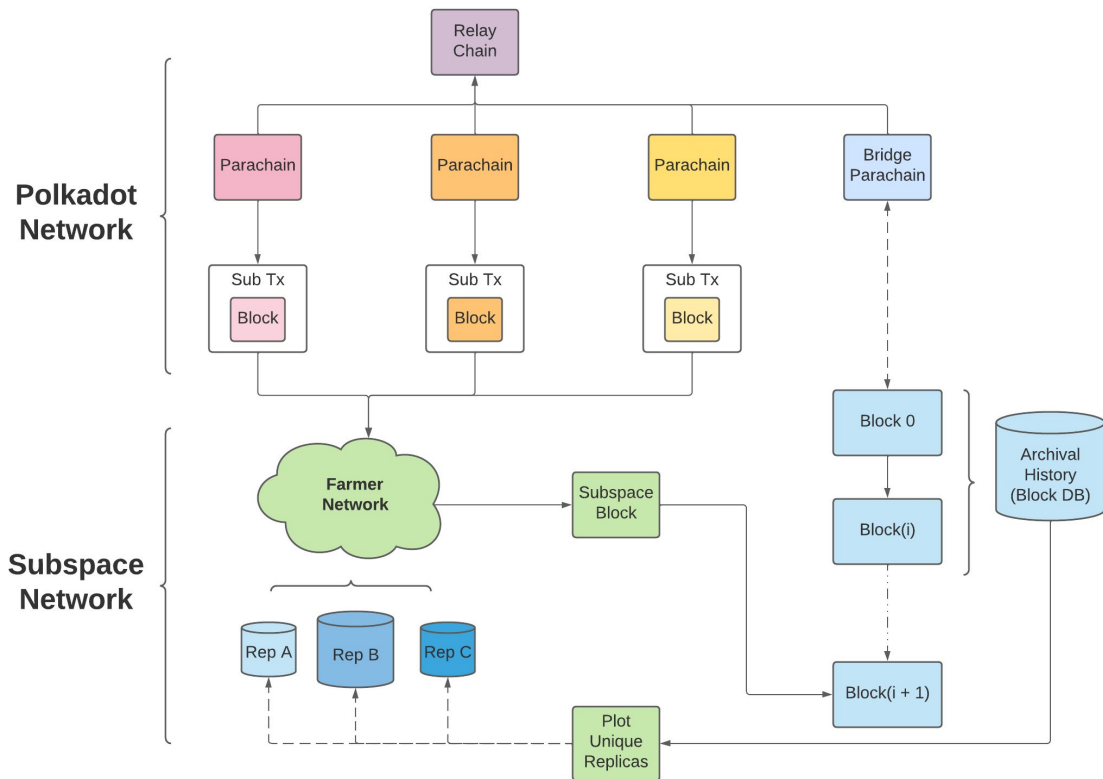
<https://testnet-relayer.subspace.network/>

Application Layer

How Ethereum can benefit from Subspace

10k Foot Overview

Subspace 🤝 Dotsama



Archival storage
Relay & Parachain Blocks

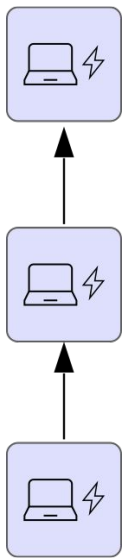
Off-chain storage
dApp Assets & Metadata

Cross-chain storage
Offsetting State Bloat

Ethereum Ecosystem

Where will all of this data live?

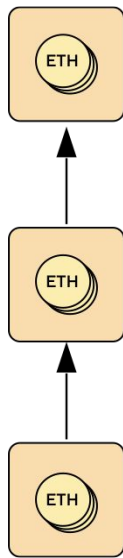
Mainnet



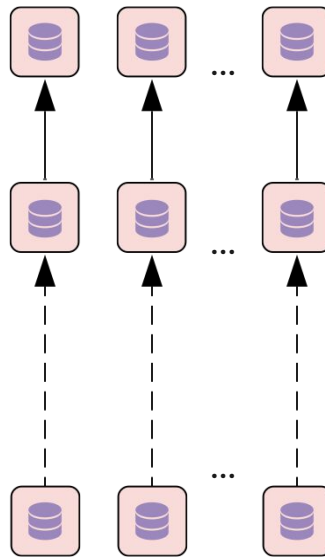
Rollups



Beacon Chain



Shard Chains

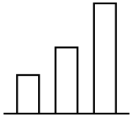


dApps



Integration Benefits

Subspace  Ethereum



Greater Scalability

A release valve for blockchain bloat

Off-Chain Storage

State Management

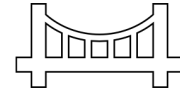


Higher Decentralization

Reduced reliance on traditional infra

Node Synchronization

Distributed Archival Nodes



Better Interoperability

Validated archiving allows for trustless bridging

Common Query API

Cross-Chain Messaging


Thanks!

And please ask questions :-)

Resources

Join / follow Subspace Network!

 discord.gg/JnFs5fFi

 medium.com/subspace-network

 t.me/subspacelabs

 github.com/subspace

 twitter.com/NetworkSubspace

subspace.network