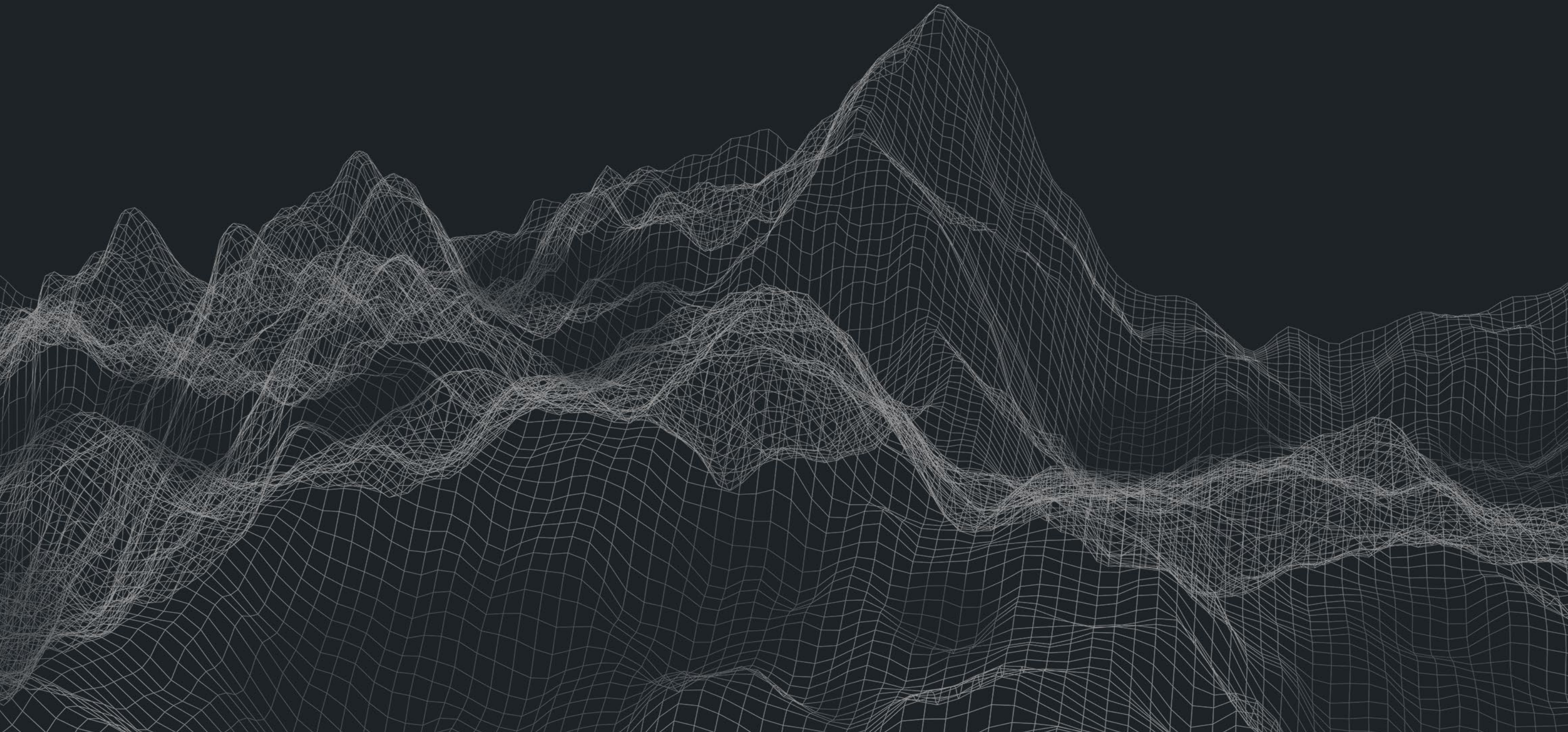
Fundamental Labs

Uncovering Fundamental Insights in Blockchain

Decentralized Storage: A Pillar of Web3

June 2022

By @0xPhillan and @FundamentalLabs



We believe that that Web3, the next iteration of the internet, will be built on decentralized technology across three fundamental pillars: consensus, storage, and computation.

Blockchain technology is what set off a revolution of decentralization and brought about the concept of Web3, representing the idea to not only decentralize consensus, but to use this technology to decentralize the rest of the internet too.

"Web3 is the stack of protocols that enable fully decentralized applications." – Nader Dabit

Just like Web2, Web3 is a complex amalgamation of a wide array of technologies that together form the Web3 ecosystem. Despite its complexity, we can break down the ecosystem into three key infrastructural pillars that need to be developed to achieve full decentralization of the internet: consensus, storage, and computation.

Consensus has matured quickly since Bitcoin's launch in 2009, with dozens of other successful models of decentralized consensus having been brought to life since then. Over time attempts at decentralizing storage and computation have emerged that aim to complement these to build the next pillars of a truly decentralized internet.

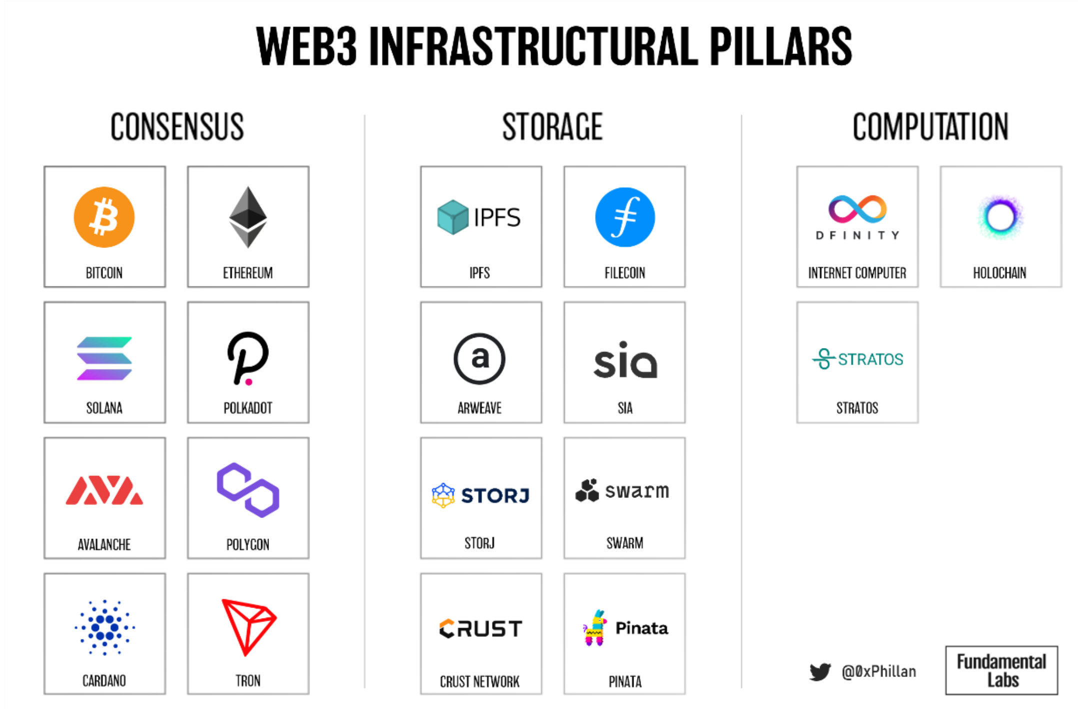


Figure 1: Illustrative slice of projects enabling each of the Web 3 pillars

In this piece we will look at decentralized storage, which describes peer-to-peer networks, in which members combine disk space to create what is essentially a global hard drive that is trustless, immutable, and in some cases permanent and censorship-resistant.

Contents

Section 1: The Need For Decentralized Storage 5
Section 2: Challenges Surrounding The Decentralization of Data 15
Section 3: Overcoming Data Decentralization Challenges 19
Conclusion 38
Section 4: Decentralized Storage Networks Deep-Dive 44
References 84

Section 1

The Need for Decentralized Storage

In this section we take a close look at the question why we even need decentralized storage. First, we look at decentralized storage from a blockchain perspective. Then, we take a closer look at NFTs and dApps from the perspective of decentralization, immutability, and permanence to understand why decentralized storage is preferred over centralized Web2 storage approaches.

The need for decentralized storage from a blockchain perspective can be examined from two primary perspectives:

Economic: storing data on chain is very expensive. Data that does not need to be stored on a blockchain should not be stored on a blockchain.

Technical: storing data on chain is very inefficient, and blocks only have a limited size. To prevent blocks from being filled up with useless data, we need to offload that data elsewhere.

The economic perspective: how expensive is it to store data?

Storing data directly on a blockchain is extremely expensive, which is why blockchains primarily store transactions (or their outcomes as state data), and which is also why smart contracts are reduced to as few lines of code as possible. This is where decentralized storage networks come in: they store data that is too expensive to store on the blockchain, but need to be permanent, immutable, and resistant to censorship.

If we wanted to store the image file of the Bored Aped Yacht Club #3368 NFT On the Bitcoin Network, we would require *at least* 1700 OP_RETURN transactions (conservative estimate) to save the entire file, assuming standard consensus rules and node settings (80 bytes of arbitrary data per OP_RETURN, max one OP_RETURN per transaction). With a transaction fee of 12 sats/vB, that's 0.028 BTC for a single image of the 10,000 piece collection.

Storing the same image data on the Ethereum network's permanent storage would cost roughly 7.9 ETH at ~95 gwei gas fees requiring nearly 23m gas units in a single smart contract deployment. For most applications, such storage costs are just not feasible.

Collection	Bored Ape Yacht Club
Item	#3368
Image size	133,110 bytes

Network	Token	Storage duration	Classification	Total		Price (10th May, 2022)	
Bitcoin	BTC	Permanent	On-chain	0.0284 BTC	892 USD	1 BTC =	31,467
Ethereum	ETH	Permanent	On-chain	7.90 ETH	18,723 USD	1 ETH =	2,369
Filecoin	FIL	200 years	Off-chain	0.0000000079 FIL	0.000000093 USD	1 FIL =	11.80
Arweave	AR	200 years	Off-chain	0.000061 AR	0.0012 USD	1 AR =	18.91
Dfinity	ICP	200 years	Off-chain	0.105 XDR	0.1410 USD	1 XDR =	1.34
Crust	CRU	200 years	Off-chain	0.40 CRU	0.5480 USD	1 CRU =	1.37

@0xPhillan Fundamental Labs

Figure 2: Projects with active Mainnets. 200 years storage duration selected to match Arweave's minimum definition of permanence. Sources: Network Documentations, Arweave Storage Calculator

If we further compare these costs against the cost of storing the same data on a decentralized storage network, we can quickly see that purpose-built storage networks are far more cost-efficient at storing files, while also ensuring permanence, immutability, and censorship resistance – *more on that later*.

The technical perspective: why should we want to avoid storing data directly on public blockchains?

Blockchains, as the name suggests, consist of blocks that are connected to one another forming a chronological sequence of blocks. Each block points to the previous block to ensure that data in past blocks cannot be adjusted. The data that is contained in the blocks are transactions or *state descriptors*. Thousands of nodes globally ensure that nobody cheats the system and consensus between the nodes is maintained.

With every block, a set of transactions are added that change the state of values within the network. Since the size of the blocks is capped, only a certain amount of transactions can be processed per block. This gives an implicit time-value to blocks, that is reflected in the fees that network participants are willing to pay to have their transaction confirmed and included in a block.

When a block is filled, transactions stay within a node's mempool until the block is confirmed and the transactions are added to the next block. If a transaction is not confirmed for an extended period, it may be impacted by slippage or by bots frontrunning the transaction. Storing arbitrary data on blockchains amplifies this issue by occupying blockspace and pushing transactions to be included in later blocks.

The limited supply of blockspace coupled with the demand for transactions to be included in a block thus drives up transaction fees for the entire network, which can dissuade users from interacting with the network.

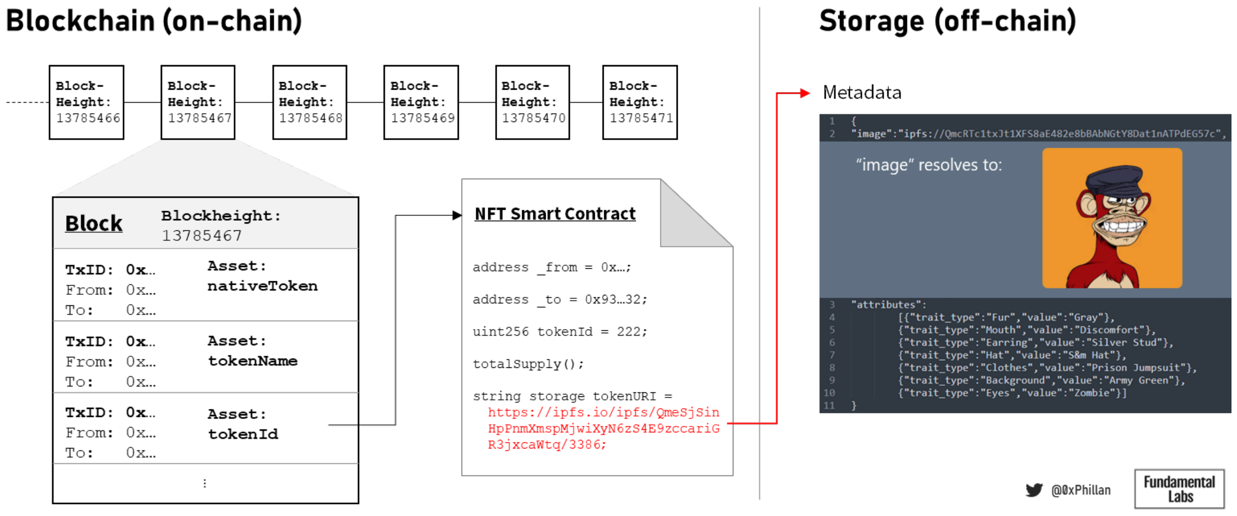
Arbitrary data on blockchains can be reduced through decentralized storage networks, by offloading those data loads while offering similar characteristics to public blockchains.

But why not store files on centralized networks?

The previous perspectives explain why we *shouldn't store data on blockchains*, however, the next question becomes: *why store data on decentralized networks*? Data could just as easily be stored on a Web2 centralized server. The answer to this is quite simple: to ensure immutability and trustlessness, and to enable permanence and censorship-resistance of the data.

The case for NFTs

Let's take a look at NFTs: non-fungible tokens (NFTs) represent a unique (i.e., non-fungible) ownership token that is stored on a blockchain and is controlled by a smart contract. The blockchain records who owns the unique token and points to something called *metadata*, which describes what the token represents. The metadata includes details about the NFT as well as links to other data such as media files – this is what gives the NFT context and meaning.



```
address _from = 0x…;

address _to = 0x93…32;

uint256 tokenId = 222;

totalSupply();

string storage tokenURI =
  https://ipfs.io/ipfs/QmeSjSin
  HpPnmXmspMjwiXyN6zS4E9zccariG
  R3jxcaWtq/3386;
```

```
{
"image":"ipfs://QmcRTc1txJt1XFS8aE482e8bBAbNGtY8Dat1nATPdEG57c",
"attributes":
        [{"trait_type":"Fur","value":"Gray"},
        {"trait_type":"Mouth","value":"Discomfort"},
        {"trait_type":"Earring","value":"Silver Stud"},
        {"trait_type":"Hat","value":"S&m Hat"},
        {"trait_type":"Clothes","value":"Prison Jumpsuit"},
        {"trait_type":"Background","value":"Army Green"},
        {"trait_type":"Eyes","value":"Zombie"}]
}
```

Figure 3: Simplified illustration of a blockchain, blocks, an NFT, and off-chain metadata.

Metadata can be stored anywhere. As long as the data is accessible through the pointer embedded in the NFT smart contract, the contents will be available in the NFT. If metadata is stored on a centralized server the data could be tampered with, the server could be destroyed or access to the data can be restricted – stripping the NFT of its context and meaning. When an NFT collection facilitates hundreds of thousands of ETH in transactions, has a floor price well above US$100k per NFT, and prices of up to *US$70k per kb of image data*, users expect every byte of metadata to be just as immutable and permanent as the on-chain NFT.

Collection	Trade Volume (ETH)	Items	Floor Price (ETH)	NFT#	Image File Size (kb)	Floor price/kb (ETH)	Floor price/kb (USD)
Crypto Punks	902,888	10,000	66.95	#6406	2.25	29.76	70,490.91
BAYC	558,548	10,000	99.79	#3368	129.99	0.77	1,818.62
MAYC	389,015	19,100	22.39	#3870	615.84	0.04	86.13

Figure 4: Crypto Punk Floor Price based on last sale (no floor available at time of writing); Crypto Punk image size based on byte-length of Crypto Punks V2 on-chain byte string. Data as of May 10th, 2022. Sources: OpenSea, on-chain data, IPFS metadata.

Arguably the value of NFTs is not primarily driven by the image data they refer to, but instead it's driven by communities that build a movement and an ecosystem around their collections. Nonetheless, without the metadata the NFTs would have no meaning and without meaning communities could not form.

NFTs are not only limited to art collectibles. The data stored within the metadata could be anything, as long as it can be saved as data: legal documents denoting ownership of tangible assets such as the deeds to a property or ownership certificates of financial instruments could be referenced in an NFT. Such data holds an extrinsic off-chain value, and the preservation of every byte of data is at least as valuable as the entire NFT: if by changing only one byte the entire dataset could be invalidated, immutability of the original data holds even greater importance.

How secure is NFT metadata really?

Apart from being the top three NFT collections on OpenSea in terms of total trade volume at the time of writing this piece, above three collections also each use a different approach to store metadata, each with varying levels of security (i.e., immutability and permanence).

Crypto Punks		
NFT smart contract:	Ethereum Smart Contract	`0xb47e3cd837dDF8e4c57F05d70Ab865de6e193BBB`
Metadata:	Ethereum Smart Contract	`0x16F5A35647D6F03D5D3da7b35409D65ba03aF3B2`
Image data:	Ethereum Smart Contract	

Figure 5: Crypto Punks NFT collection smart contract and metadata storage addresses.

Crypto Punks is the most secure, with all metadata and image data being directly stored on-chain. By parsing the metadata and image data smart contract address, you can directly retrieve NFT attributes and raw image data. Because all data is stored on-chain, the NFT inherits the security attributes of the

Ethereum chain. These NFTs are immutable, will always be on-chain and will always be accessible as long as the Ethereum network exists.

Bored Ape Yacht Club (BAYC)		
NFT smart contract:	Ethereum Smart Contract	`0xBC4CA0EdA7647A8aB7C2061c2E118A18a936f13D`
Metadata:	IPFS CID *(example: #3368)*	`ipfs://QmeSjSinHpPnmXmspMjwiXyN6zS4E9zccariGR3jxcaWtq/3368`
Image data:	IPFS CID *(example: #3368)*	`ipfs://QmcRTc1txJt1XFS8aE482e8bBAbNGtY8Dat1nATPdEG57c`

Figure 6: BAYC NFT collection smart contract and metadata storage addresses.

BAYC stores metadata and image data using the InterPlanetary File System (IFPS), which is a peer-to-peer hypermedia protocol that solves decentralized content addressing. On IPFS once the content receives a content ID (CID), which also acts as a link to the data, it cannot be changed anymore. While IPFS is considered to be censorship-resistant, there are still risks of data being removed from IPFS nodes which would result in the NFT metadata eventually disappearing.

Mutant Ape Yacht Club (MAYC)		
NFT smart contract:	Ethereum Smart Contract	`0x60E4d786628Fea6478F785A6d7e704777c86a7c6`
Metadata:	Webserver *(example: #3870)*	`https://boredapeyachtclub.com/api/mutants/3870`
Image data:	IPFS CID *(example: #3870)*	`ipfs://Qmcg5P6jtJUPf8TdzWx5FLQ3TCQ4gcaDoaBepmGiJAbbLZ`

Figure 7: MAYC NFT collection smart contract and metadata storage addresses.

Finally, MAYC stores all NFT metadata on a centralized webserver, which points to images that are hosted on IPFS. While the images are retrievable through their IPFS CIDs, the metadata stored on the webserver can be changed at any time. This means that all NFTs within the MAYC NFT collection can have their traits and images removed, or have their images replaced with other images.

Out of the top three collections, MAYC is the least secure lacking metadata immutability, permanence, and censorship resistance.

Ultimately how the developers implemented the metadata hosting will determine how secure the NFT metadata is. On-chain is the most secure, but extremely expensive, hence not always a good option. Centralized servers run impermanence and mutability risks. Decentralized storage networks present a middle ground that balance cheaper costs with permanence, immutability, and censorship resistance.

The case for dApps

dApps (decentralized applications) are fundamentally different to NFTs, in that dApps enable services that facilitate interaction with a blockchain. A dApp consists of a front-end user interface and sometimes a back-end to enable and facilitate interaction with smart contracts. A smart contract is a self-executing piece of code on a decentralized blockchain network, that users can interact with. In contrast to dApps, regular apps have their backend code on centralized servers on individual devices.

What is special about smart contracts is that all aspects of the smart contract's operation are written directly into the code and can be publicly reviewed before interacting with the code. Interacting directly with a smart contract, however, requires some technical background and an understanding of how the blockchain and smart contract engine of that blockchain work. To bridge this gap, dApps provide an easy interface for users to interact with a decentralized blockchain network.

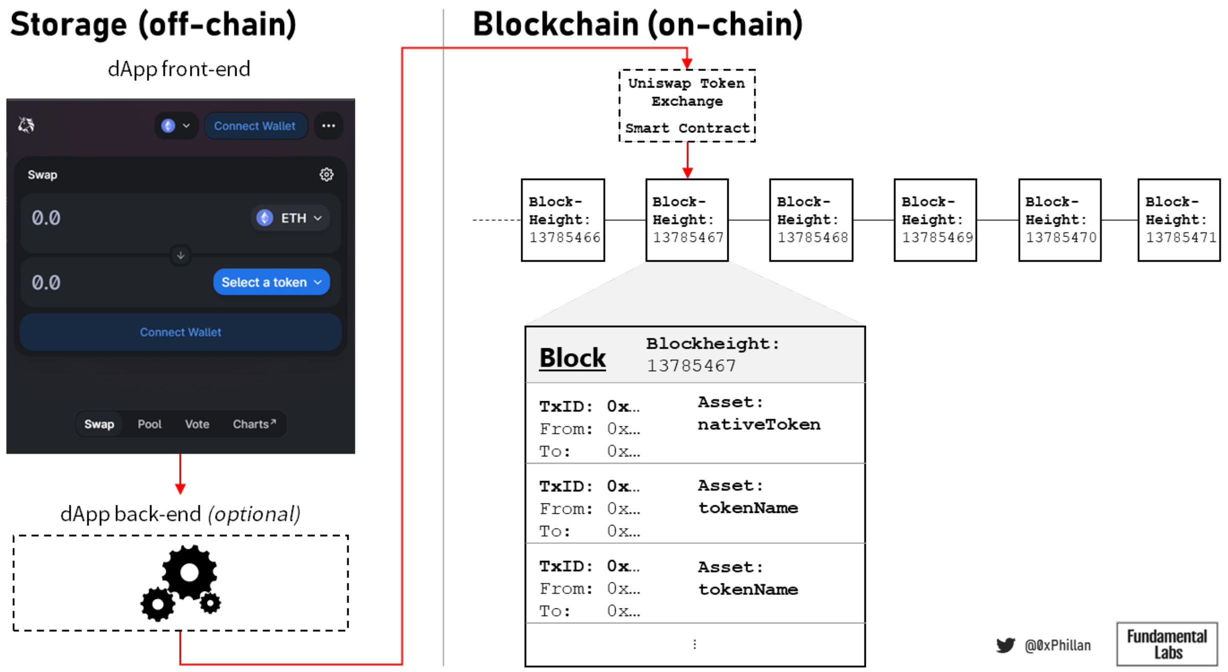
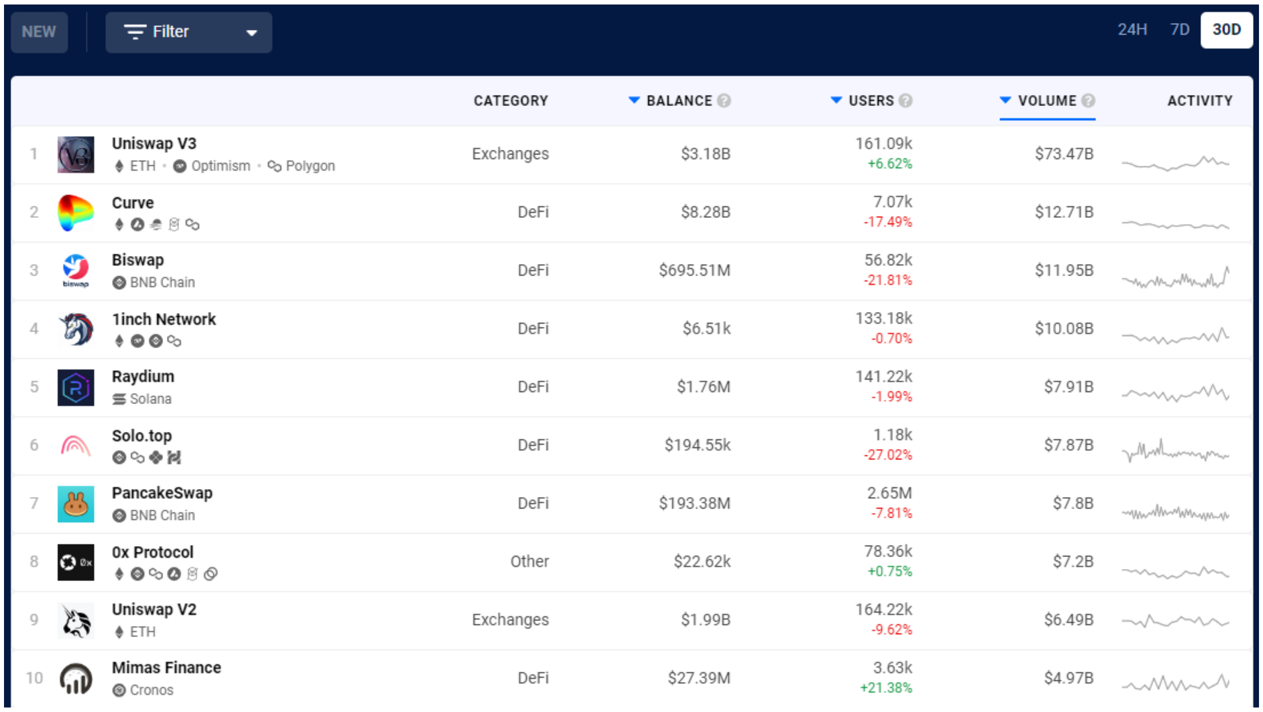


Figure 8: Simplified illustration of dApp interaction with a blockchain.

On decentralized networks that support smart contract execution, every write operation comes at a cost. Sometimes these operations can be quite complex – this is where dApp back-ends come in. dApp backends are different to smart contracts, as they exist primarily to either convert inputs into smart contract compatible inputs or to shift certain computational loads away from smart contracts to optimize reduce gas costs.

The value proposition for dApps is fundamentally different to that of NFTs, as they provide users with a service instead of having their value locked in an asset. dApps are in a constant state of change: improvements and bug fixes are regularly applied, which causes the underlying data to change over time. As a result, a dApp cannot be measured on the value of the underlying data. Instead, the value of the service needs to be measured in the context of the specific dApp. For DeFi (decentralized finance) dApps, the value can be measured based on the asset transfer volumes facilitated through the dApp or total value locked (TVL), while social dApps may focus more on interaction and user metrics.

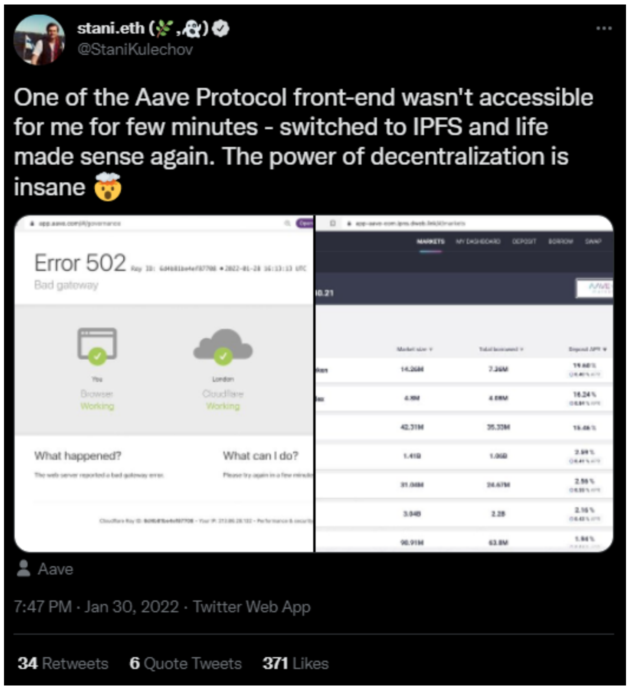


		CATEGORY	BALANCE	USERS	VOLUME	ACTIVITY
1	Uniswap V3 ETH • Optimism • Polygon	Exchanges	$3.18B	161.09k +6.62%	$73.47B	
2	Curve	DeFi	$8.28B	7.07k -17.49%	$12.71B	
3	Biswap BNB Chain	DeFi	$695.51M	56.82k -21.81%	$11.95B	
4	1inch Network	DeFi	$6.51k	133.18k -0.70%	$10.08B	
5	Raydium Solana	DeFi	$1.76M	141.22k -1.99%	$7.91B	
6	Solo.top	DeFi	$194.55k	1.18k -27.02%	$7.87B	
7	PancakeSwap BNB Chain	DeFi	$193.38M	2.65M -7.81%	$7.8B	
8	0x Protocol	Other	$22.62k	78.36k +0.75%	$7.2B	
9	Uniswap V2 ETH	Exchanges	$1.99B	164.22k -9.62%	$6.49B	
10	Mimas Finance Cronos	DeFi	$27.39M	3.63k +21.38%	$4.97B	

Figure 9: The most popular dApps by US$ volume as reported by DappRadar as of May 11th, 2022.

The above list by DappRadar shows the top ten dApps by volume, collectively facilitating transfers of over US$150bn within the last 30 days at the time of writing. While the dApps listed here are primarily DeFi and Exchange dApps, dApps can fulfill any purpose. As long as the application interacts with blockchains by way of smart contracts through some sort of user interface, the application can be considered a dApp. Other popular dApp categories include, games, metaverses, marketplaces, social media and name services.

But why should dApps be decentralized if users can interact with the core mechanics of the dApp through smart contracts on a blockchain? The answer lies in assuring service availability and permanence. With a decentralized storage network that replicates copies of the data to dozens of nodes, dApps reduce the likelihood of going offline due to server malfunctions, improve resistance to DNS hacks and live on, even if development comes to an end. Also, depending on the decentralized storage network, a certain level of censorship-resistance is also introduced in that no single centralized entity can easily remove the data.


stani.eth
@StaniKulechov

One of the Aave Protocol front-end wasn't accessible for me for few minutes - switched to IPFS and life made sense again. The power of decentralization is insane 🤯

Error 502
Bad gateway

What happened?

What can I do?

Aave

7:47 PM · Jan 30, 2022 · Twitter Web App

34 Retweets 6 Quote Tweets 371 Likes

Figure 10: Aave founder Stani Kulechov tweets that the Aave dApp front-end went offline on Jan 20th 2022, but was still accessible through an IPFS-hosted copy of the website. https://twitter.com/StaniKulechov/status/1487754439691845633

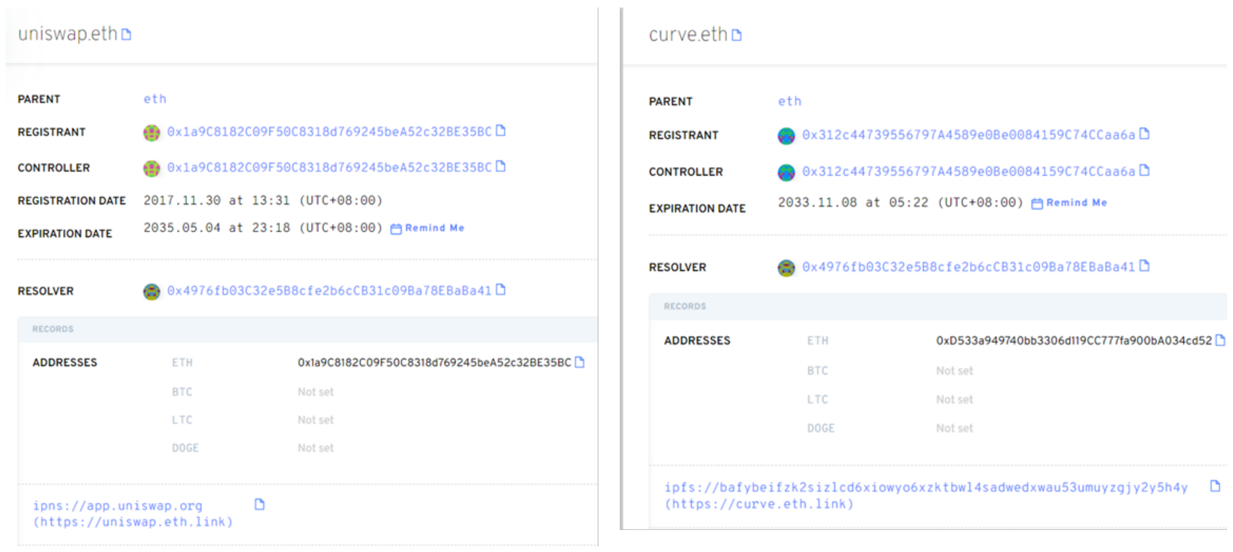
How decentralized are dApps really?

A common misconception is that dApps are decentralized in all aspects (as implied by the word "dApp"). While some dApps, such as Uniswap and Aave, go the extra mile to ensure their dApps can be accessed both from a centralized server and as well through decentralized networks, many dApps opt to only host their services on centralized servers. Still, as long as the applications access and interact with smart contracts on decentralized blockchains, these tools are considered dApps.

When measuring the extent of decentralization of dApps in the context of service accessibility there are a few factors to consider:

1. Is the dApp front-end accessible through decentralized networks?
2. If yes, to what extent is the data on those networks immutable and permanent?
3. Could users locally host the dApp front-end to access the services (i.e., is the dApp source code open source?)

If we look deeper into the above top three dApps in terms of volume, we find that out of the top three dApps, all dApps publish their source code allowing for individuals to deploy the dApp frontends to decentralized storage providers. Furthermore, Uniswap and Curve go a step further and directly provide regularly updated CIDs in their ENS (Ethereum Name Service) records.



uniswap.eth

PARENT	eth
REGISTRANT	0x1a9C8182C09F50C8318d769245beA52c32BE35BC
CONTROLLER	0x1a9C8182C09F50C8318d769245beA52c32BE35BC
REGISTRATION DATE	2017.11.30 at 13:31 (UTC+08:00)
EXPIRATION DATE	2035.05.04 at 23:18 (UTC+08:00) Remind Me
RESOLVER	0x4976fb03C32e5B8cfe2b6cCB31c09Ba78EBaBa41

RECORDS

ADDRESSES	ETH	0x1a9C8182C09F50C8318d769245beA52c32BE35BC
	BTC	Not set
	LTC	Not set
	DOGE	Not set

ipns://app.uniswap.org
(https://uniswap.eth.link)

curve.eth

PARENT	eth
REGISTRANT	0x312c44739556797A4589e0Be0084159C74CCaa6a
CONTROLLER	0x312c44739556797A4589e0Be0084159C74CCaa6a
EXPIRATION DATE	2033.11.08 at 05:22 (UTC+08:00) Remind Me
RESOLVER	0x4976fb03C32e5B8cfe2b6cCB31c09Ba78EBaBa41

RECORDS

ADDRESSES	ETH	0xD533a949740bb3306d119CC777fa900bA034cd52
	BTC	Not set
	LTC	Not set
	DOGE	Not set

ipfs://bafybeifzk2sizlcd6xiowyo6xzktbwl4sadwedxwau53umuyzgjy2y5h4y
(https://curve.eth.link)

Figure 11: Uniswap and Curve deployed on IPFS with latest links updated in their ENS records

While the above is only an illustrative example of best practices, currently only few dApps actively decentralize their user interface. The premise of DeFi, which is often also referred to as Open Finance, is to make away with restrictions on who can access and trade financial assets. While the underutilization of decentralized storage does create a break in that narrative, it also creates opportunities for future growth and adoption.

The next section looks at individual decentralized storage solutions in more detail, covering decentralization mechanics, storage pricing algorithms, and tokenomics.

Section 2:

Challenges Surrounding Decentralization of Data

As we have established, decentralized storage is vital to a web3 future. The decentralized storage landscape is vast considering how young it is. Given the complexity of challenges storage protocols must overcome, each protocol makes certain trade-offs to achieve their vision of decentralization.

Decentralized Storage Landscape

Unlike Layer1 blockchains in which the primary purpose is trustless value transfers, such as Bitcoin and Ethereum, decentralized storage networks need not only record transactions (which are used for storage requests) but must also ensure that data is stored for a specific amount of time as well as overcome other challenges pertaining to storage. As a result, it is not uncommon to see decentralized storage blockchains applying multiple consensus mechanisms that work in tandem to ensure different aspects of the storage and retrieval can work.

In below non-exhaustive list of decentralized storage projects, we can catch a glimpse of the decentralized storage landscape as well as niche data storage use cases, such as P2P filesharing and data markets.

The focus of this research is on storage networks (IPFS and non-IPFS based).

Category	Project	Native Blockchain	Token	Token Type
Storage (IPFS-based)	Filecoin	Filecoin	FIL	Native
Storage (IPFS-based)	CRUST	Crust Network	CRU	Native
Storage (IPFS-based)	Pinata	n/a	n/a	n/a
Storage	arweave	The blockweave	AR	Native
Storage	sia	Sia	SC	Native
Storage	STORJ	n/a	STORJ	ERC-20 (Ethereum)
Storage	swarm	n/a	BZZ	ERC-20 (Ethereum)
Storage & computing	DFINITY	Internet Computer	ICP	Native
Storage & computing	HOLOCHAIN	Holochain	HOL	Native
Storage & computing	STRATOS	Stratos Chain	STOS	Native
P2P filesharing	BitTorrent	n/a	BTT	TRC-20 (Tron)
Data markets	ocean	n/a	OCEAN	ERC-20 (Ethereum)

@0xPhillan Fundamental Labs

Figure 12: Overview of some arbitrarily selected decentralized storage protocols (non-exhaustive)

Decentralized Storage Design Challenges

As has been demonstrated in the first section of this paper, blockchains are not suitable for storing large amounts of data on-chain, due to the cost associated with it and the impact on block space. As a result, decentralized storage networks must apply other techniques to ensure decentralization. Not using the blockchain as the primary storage space, however, leads to a long list of other challenges if the network wants to maintain decentralization.

In essence, a decentralized storage network must be capable of storing data, retrieving data, and maintaining data while ensuring that all actors within the network are incentivized for the work they do, while also upholding the trustless nature of decentralized systems.

Hence, from a design perspective, we can summarize the primary challenges in below illustrative paragraphs:

Data Storage Format – First the network must decide *how* to store the data: should the data be encrypted and should the data be saved as a full set or split into smaller pieces.

Replication of Data – Then the network needs to decide *where* to store the data: how many nodes should the data be stored on, and whether all data is to be replicated to all nodes or whether each node should get different fragments to further protect data privacy. The data storage format and the network diffusion of data will determine the probability of data being available on the network with regards to devices failing over time (durability).

Storage Tracking – From here, the network needs a mechanism to *track where* the data is stored. This is important, because the network needs to know which network locations to ask to retrieve specific data.

Proof of Data Stored – Not only does the network need to know where data is stored, but storage nodes also need to be able to prove that they are indeed storing the data they are intended to be storing.

Data Availability over Time – Networks also need to ensure data is where it is meant to be *when* it is meant to be there. This means mechanisms must be designed to ensure nodes do not remove old data over time.

Storage Price Discovery – And nodes expect to be paid for the on-going storage of files.

Persistent Data Redundancy – While networks need to know where data is located, by nature of public open networks nodes will continually leave the network and new nodes will continually join the network. Hence, apart from ensuring a single node is storing what they are meant to be storing when they are meant to be storing it, the network needs to ensure that when a node leaves and its data disappears, sufficient *copies of the data or data fragments are maintained* across the network.

Data Transmission – Then, when the network connects to nodes to retrieve data that is requested (by a user or for data maintenance workloads), the nodes that are storing the data must be *willing to transmit* the data, as bandwidth also comes at a cost.

Network Tokenomics – Finally, apart from ensuring that data resides within a network, the network must ensure that the network itself will be around for a long time. If the network were to disappear, it would take all data with it – hence strong *tokenomics* are necessary to ensure network permanence, and thus data availability, over the long term.

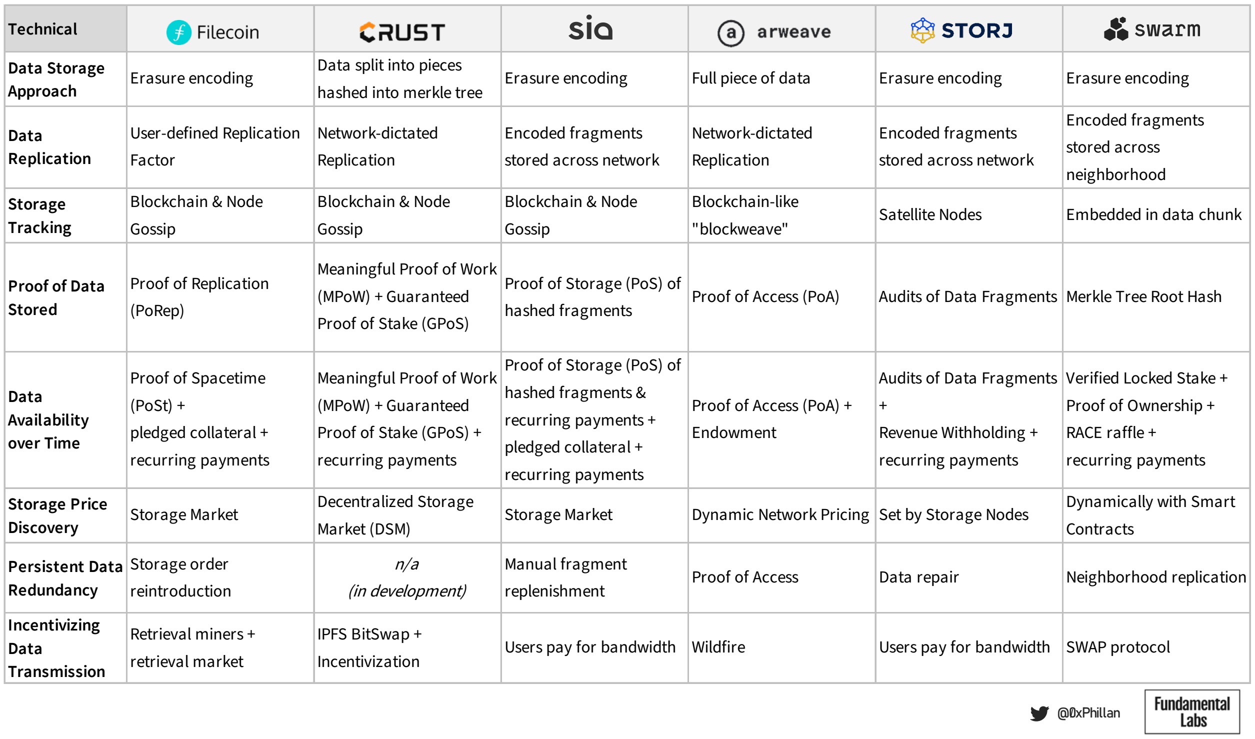
Section 3:

Overcoming Challenges of Data Decentralization

Note: For a detailed technical introduction to each of the below projects, please consult section 4.

In this section, I compare and contrast various aspects of decentralized storage network design of IPFS, Filecoin, Crust Network, Arweave, Sia, Storj, and Swarm and how they overcome the aforementioned challenges. This reflects well-established as well as up-and-coming decentralized storage networks that use a wide range of technologies to achieve decentralization.

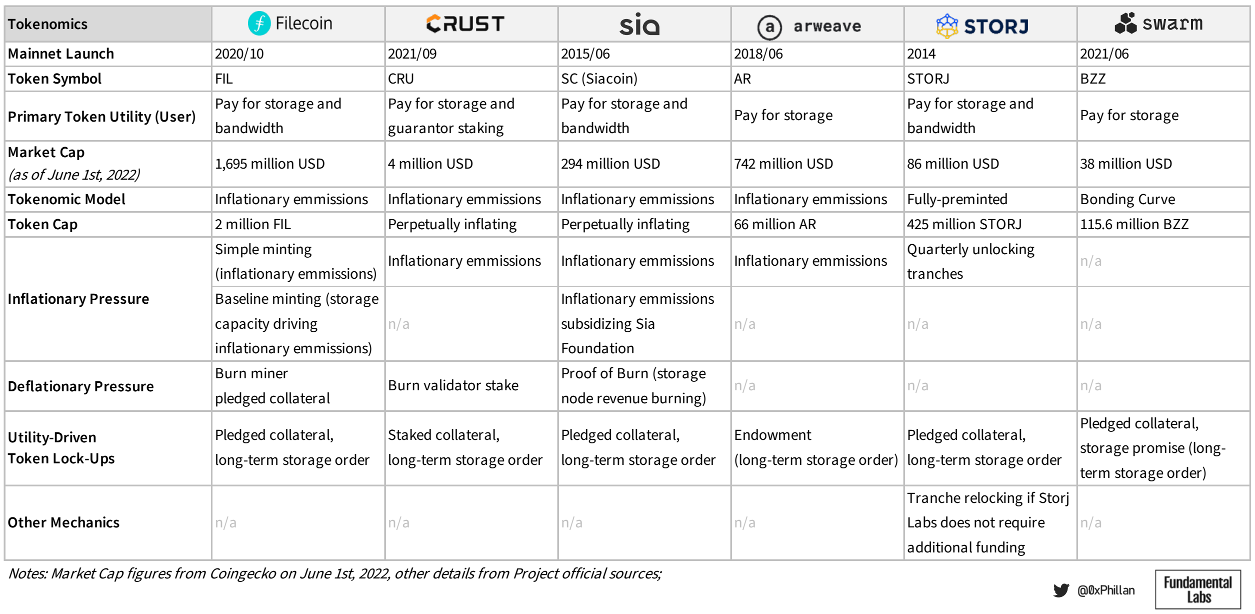
Below tables summarize both technical aspects and tokenomics of each network, that will be covered in greater detail throughout this section, as well as what the writer believes are strong use cases of these chains following their various design-elements.



Technical	Filecoin	CRUST	sia	arweave	STORJ	swarm
Data Storage Approach	Erasure encoding	Data split into pieces hashed into merkle tree	Erasure encoding	Full piece of data	Erasure encoding	Erasure encoding
Data Replication	User-defined Replication Factor	Network-dictated Replication	Encoded fragments stored across network	Network-dictated Replication	Encoded fragments stored across network	Encoded fragments stored across neighborhood
Storage Tracking	Blockchain & Node Gossip	Blockchain & Node Gossip	Blockchain & Node Gossip	Blockchain-like "blockweave"	Satellite Nodes	Embedded in data chunk
Proof of Data Stored	Proof of Replication (PoRep)	Meaningful Proof of Work (MPoW) + Guaranteed Proof of Stake (GPoS)	Proof of Storage (PoS) of hashed fragments	Proof of Access (PoA)	Audits of Data Fragments	Merkle Tree Root Hash
Data Availability over Time	Proof of Spacetime (PoSt) + pledged collateral + recurring payments	Meaningful Proof of Work (MPoW) + Guaranteed Proof of Stake (GPoS) + recurring payments	Proof of Storage (PoS) of hashed fragments & recurring payments + pledged collateral + recurring payments	Proof of Access (PoA) + Endowment	Audits of Data Fragments + Revenue Withholding + recurring payments	Verified Locked Stake + Proof of Ownership + RACE raffle + recurring payments
Storage Price Discovery	Storage Market	Decentralized Storage Market (DSM)	Storage Market	Dynamic Network Pricing	Set by Storage Nodes	Dynamically with Smart Contracts
Persistent Data Redundancy	Storage order reintroduction	*n/a (in development)*	Manual fragment replenishment	Proof of Access	Data repair	Neighborhood replication
Incentivizing Data Transmission	Retrieval miners + retrieval market	IPFS BitSwap + Incentivization	Users pay for bandwidth	Wildfire	Users pay for bandwidth	SWAP protocol

@0xPhillan Fundamental Labs

Figure 13: Summary of technical design decisions of reviewed storage networks

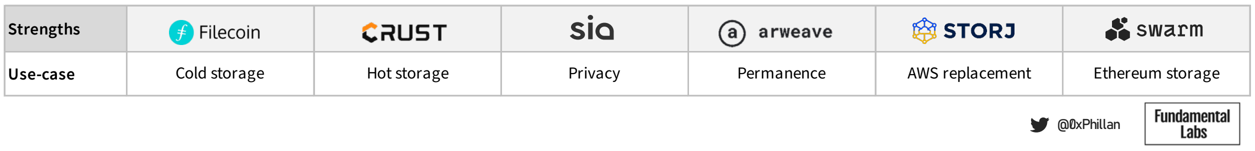


Tokenomics	Filecoin	CRUST	sia	arweave	STORJ	swarm
Mainnet Launch	2020/10	2021/09	2015/06	2018/06	2014	2021/06
Token Symbol	FIL	CRU	SC (Siacoin)	AR	STORJ	BZZ
Primary Token Utility (User)	Pay for storage and bandwidth	Pay for storage and guarantor staking	Pay for storage and bandwidth	Pay for storage	Pay for storage and bandwidth	Pay for storage
Market Cap *(as of June 1st, 2022)*	1,695 million USD	4 million USD	294 million USD	742 million USD	86 million USD	38 million USD
Tokenomic Model	Inflationary emissions	Inflationary emissions	Inflationary emissions	Inflationary emissions	Fully-preminted	Bonding Curve
Token Cap	2 million FIL	Perpetually inflating	Perpetually inflating	66 million AR	425 million STORJ	115.6 million BZZ
Inflationary Pressure	Simple minting (inflationary emissions)	Inflationary emissions	Inflationary emissions	Inflationary emissions	Quarterly unlocking tranches	n/a
	Baseline minting (storage capacity driving inflationary emissions)	n/a	Inflationary emissions subsidizing Sia Foundation	n/a	n/a	n/a
Deflationary Pressure	Burn miner pledged collateral	Burn validator stake	Proof of Burn (storage node revenue burning)	n/a	n/a	n/a
Utility-Driven Token Lock-Ups	Pledged collateral, long-term storage order	Staked collateral, long-term storage order	Pledged collateral, long-term storage order	Endowment (long-term storage order)	Pledged collateral, long-term storage order	Pledged collateral, storage promise (long-term storage order)
Other Mechanics	n/a	n/a	n/a	n/a	Tranche relocking if Storj Labs does not require additional funding	n/a

Notes: Market Cap figures from Coingecko on June 1st, 2022, other details from Project official sources;

@0xPhillan Fundamental Labs

Figure 14: Summary of tokenomics design decisions of reviewed storage networks



Strengths	Filecoin	CRUST	sia	arweave	STORJ	swarm
Use-case	Cold storage	Hot storage	Privacy	Permanence	AWS replacement	Ethereum storage

@0xPhillan Fundamental Labs

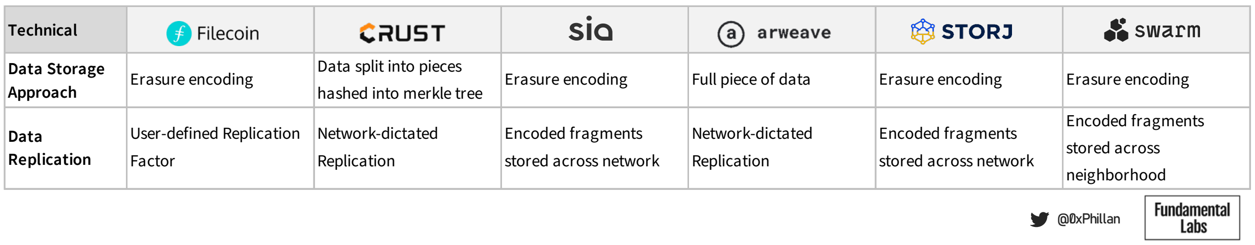
Figure 15: Summary of strong uses-cases for reviewed storage networks

Due to many concepts being closely interlinked within each protocol design, it is not possible to clearly split out each challenge, hence there is some overlap between sub-sections.

Data Storage Format and Replication of Data

Data format and the replication of data refers to how data is stored on a single node instance, and how data is spread across multiple nodes when a user or application requests to store a file (henceforth users and applications will be collectively referred to as users or clients). This is an important distinction, as data can also be stored on a node as a result of data maintenance procedures initiated by the network or other network actors.

In below table we can see a brief overview of how the protocols store data:



Technical	Filecoin	CRUST	sia	arweave	STORJ	swarm
Data Storage Approach	Erasure encoding	Data split into pieces hashed into merkle tree	Erasure encoding	Full piece of data	Erasure encoding	Erasure encoding
Data Replication	User-defined Replication Factor	Network-dictated Replication	Encoded fragments stored across network	Network-dictated Replication	Encoded fragments stored across network	Encoded fragments stored across neighborhood

@0xPhillan Fundamental Labs

Figure 16: Data storage approach and data replication of reviewed storage networks

From the above projects, Filecoin and Crust use the Interplanetary File System (IPFS) as the network coordination and communication layer for transmitting files between peers and storing files on nodes. IPFS and Filecoin were both developed by protocol labs.

When new data is to be stored on the Filecoin network, a storage user must connect to a storage provider through the Filecoin storage market and negotiate storage terms, before placing a storage order. The user must then decide which type of erasure encoding (EC) is to be used and the replication factor thereof. With EC, data is broken down into constant-size fragments, which are each expanded and encoded with redundant data, so that only a subset of the fragments are required to reconstruct the original file. The replication factor refers to how often the data should be replicated to more storage sectors of the storage miner. Once the storage miner and the user agree on the terms, the data is transmitted to a storage miner and is stored in a storage miner's storage sector.

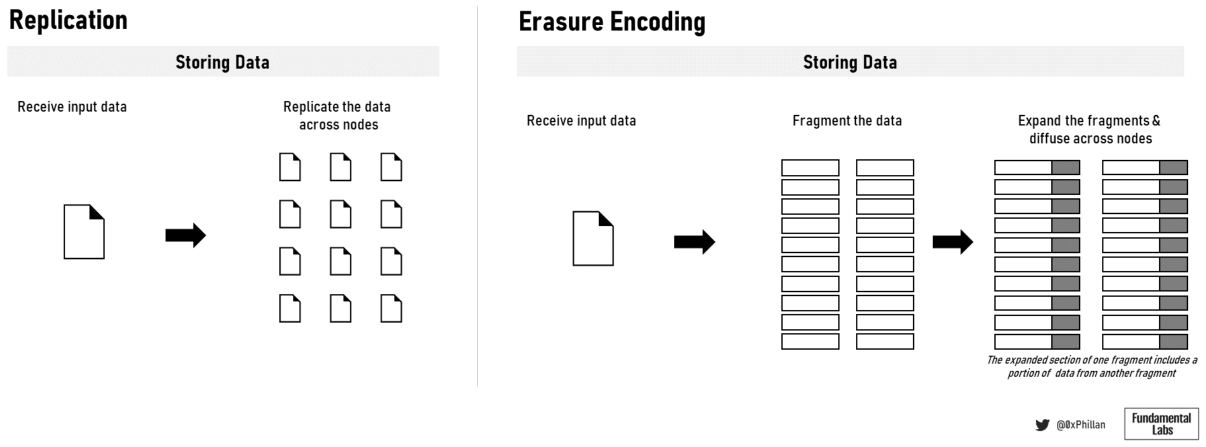

Replication
Storing Data
Receive input data
Replicate the data across nodes
Erasure Encoding
Storing Data
Receive input data
Fragment the data
Expand the fragments & diffuse across nodes
The expanded section of one fragment includes a portion of data from another fragment
@0xPhillan
Fundamental Labs

Figure 17: Data replication and erasure encoding of data

If users want to further increase redundancy, they need to engage in additional storage deals with additional storage providers, as there still exists the risk that one storage miner goes offline and with it all of their pledged storage sectors. Applications such as NFT.Storage and Web3.Storage built by Filecoin on the Filecoin protocol solve this by storing files with multiple storage miners, however at a protocol level users must manually engage with multiple storage miners.

In contrast, Crust replicates data to a fixed number of nodes: when a storage order is submitted the data is encrypted and sent to at least 20 Crust IPFS nodes (the number of nodes can be adjusted). On each node the data is split into many smaller fragments which are hashed into a Merkle tree. Each node keeps all fragments that make up the full file. While Arweave also uses replication of full files, Arweave takes a somewhat different approach. After a transaction is submitted to the Arweave network, first one single node will store the data as a block on the blockweave (Arweave's manifestation of a blockchain). From there a very aggressive algorithm called Wildfire ensures the data is replicated to rapidly across the network, because in order for any node to mine the next block, they must prove they have access to the previous block.

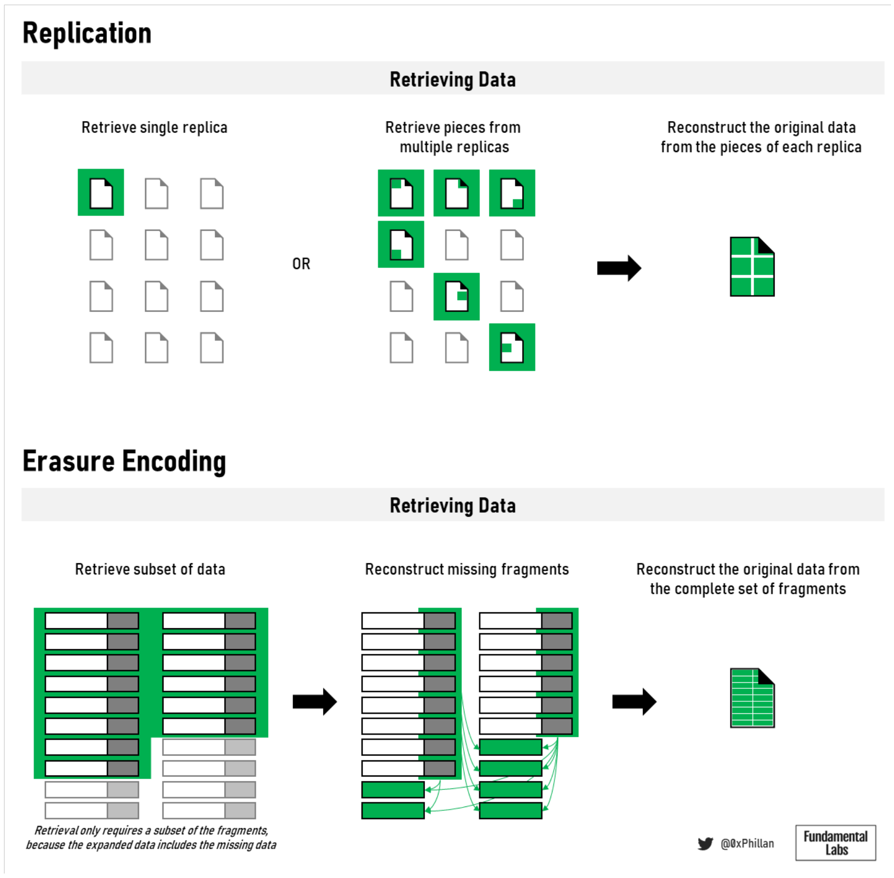

Replication
Retrieving Data
Retrieve single replica
OR
Retrieve pieces from multiple replicas
Reconstruct the original data from the pieces of each replica
Erasure Encoding
Retrieving Data
Retrieve subset of data
Reconstruct missing fragments
Reconstruct the original data from the complete set of fragments
Retrieval only requires a subset of the fragments, because the expanded data includes the missing data
@0xPhillan
Fundamental Labs

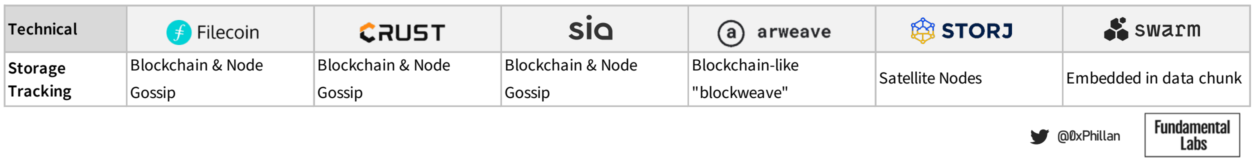
Figure 18: The data storage format will impact retrieval and reconstruction

Sia, Storj, and Swarm use erasure encoding (EC) to store files. With Crust's implementation, 20 full data sets are stored across 20 nodes. While this is extremely redundant and makes the data highly durable, this is highly inefficient from a bandwidth perspective. Erasure encoding presents a much more efficient means of achieving redundancy, by improving data durability without a large bandwidth impact.

Sia and Storj directly propagate EC fragments to a specific number of nodes to meet certain durability requirements. Swarm, on the other hand, manages nodes in a way that nodes that are in closer proximity form a neighborhood, and those nodes proactively share data chunks (the specific fragment format used in Swarm) among each other. If there is popular data that is being recalled from the network often, other nodes are incentivized to store the popular chunks as well – this is called opportunistic caching. As a result, in Swarm it's possible to have a far greater number of data fragments in the network than what is considered a minimum "healthy" amount. While this does have bandwidth implications, this can be considered front-loading future retrieval requests by reducing the distance to the requesting node.

Storage Tracking

After data has been stored, either to one or many nodes, the network needs to know where the data is stored. This way when users request to retrieve their data, the network knows where to look.



Technical	Filecoin	CRUST	sia	arweave	STORJ	swarm
Storage Tracking	Blockchain & Node Gossip	Blockchain & Node Gossip	Blockchain & Node Gossip	Blockchain-like "blockweave"	Satellite Nodes	Embedded in data chunk

@0xPhillan
Fundamental Labs

Figure 19: Storage tracking of reviewed storage networks

Filecoin, Crust, Sia, and Arweave all use a blockchain or a blockchain-life structure to manage storage orders and keep a record of every storage request placed on the network. In Filecoin, Crust and Sia storage proofs (i.e., proof that files have been stored by a miner, are stored on chain). This allows these networks to know where which data is located at any point in time. With Arweave, the network incentivizes all nodes to store as much data as possible, however, nodes are not required to store every single piece of data. Because Arweave stores data as blocks on their blockchain and nodes are not required to store all data, nodes can be missing some data which can be retrieved at a later time. Hence why Arweave's blockweave is a "blockchain-like" structure.

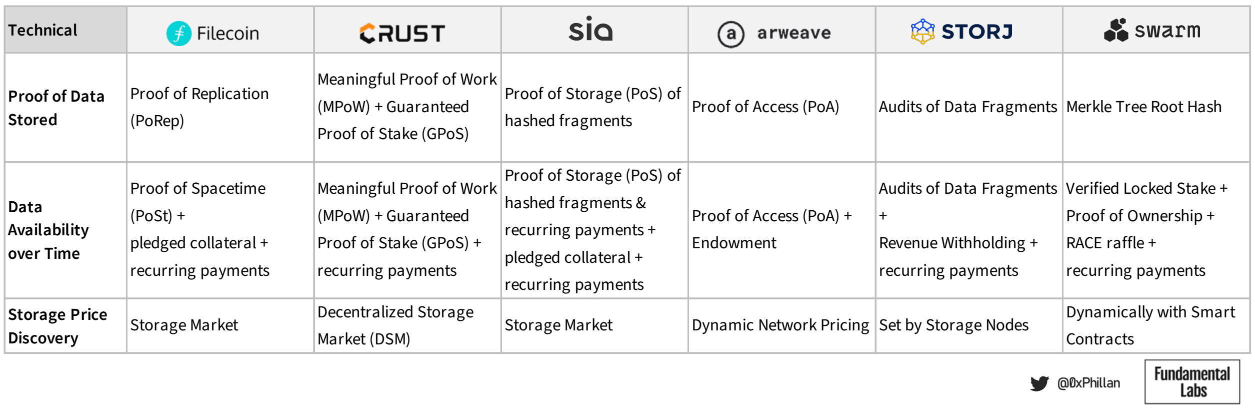
On Filecoin, Crust and Sia, storage nodes all maintain a local table with details of which storage nodes store which data. This data is regularly updated amongst nodes by gossiping amongst each other. For Arweave, however, when content is requested nodes are requested opportunistically instead of reaching out to specific nodes that are known to save the content.

Storj and Swarm both do not have their own layer 1 blockchain and hence track storage in a different way. In Storj, storage order management and the storage of files are split between two different types of nodes, namely satellites and storage nodes. Satellites, which can be a single server or a redundant collection of servers, only tracks data which users have submitted to them for storage, and only store it on storage nodes that have entered an agreement with them. Storage nodes can work with multiple satellites and store data from multiple satellites. This architecture means that in Storj to store files, no network-wide consensus is needed, which means increased efficiency and fewer computing resources are required to store data. However, this also means that if a satellite goes offline, the data managed by that satellite will be inaccessible. Hence it is recommended that a satellite be comprised of multiple redundant servers.

In Swarm, the address of data stored is directly recorded in the hash of each chunk in the data to chunk conversion process. Since chunks are stored across nodes in the same address space (i.e. neighborhood), the neighborhood of a file can be identified simply by the chunk hash itself. This means that a separate tracking mechanism of where which files are stored is not necessary, as the storage location is implied by the chunk itself.

Proof of Data Stored, Availability Over Time, and Storage Price Discovery

Apart from the network knowing where data is stored, the network must have a way to verify that the data that is meant to be stored on a specific node is indeed stored on that specific node. Only after that validation has occurred can the network employ other mechanisms to ensure that the data remains stored over time (i.e., that storage nodes do not remove the data after the initial storage operation). Such mechanisms include algorithms that prove data is stored over certain period of time, financial incentivization for successfully concluding the duration of storage requests, and disincentivization of non-completion thereof, amongst others. It should be noted here that data availability over time does not equate to permanence, although permanent storage is a form of long-term data availability. Finally, nodes expect to be paid for their storage efforts, which is reflected in the aforementioned incentivization mechanisms.



Technical	Filecoin	CRUST	sia	arweave	STORJ	swarm
Proof of Data Stored	Proof of Replication (PoRep)	Meaningful Proof of Work (MPoW) + Guaranteed Proof of Stake (GPoS)	Proof of Storage (PoS) of hashed fragments	Proof of Access (PoA)	Audits of Data Fragments	Merkle Tree Root Hash
Data Availability over Time	Proof of Spacetime (PoSt) + pledged collateral + recurring payments	Meaningful Proof of Work (MPoW) + Guaranteed Proof of Stake (GPoS) + recurring payments	Proof of Storage (PoS) of hashed fragments & recurring payments + pledged collateral + recurring payments	Proof of Access (PoA) + Endowment	Audits of Data Fragments + Revenue Withholding + recurring payments	Verified Locked Stake + Proof of Ownership + RACE raffle + recurring payments
Storage Price Discovery	Storage Market	Decentralized Storage Market (DSM)	Storage Market	Dynamic Network Pricing	Set by Storage Nodes	Dynamically with Smart Contracts

@0xPhillan Fundamental Labs

Figure 20: Proof of data stored, availability over time, and pricing mechanisms of reviewed storage networks

To illustrate both proof of data stored and how data availability is ensured over time, this section will look at the full storage process per protocol.

Filecoin

On Filecoin, before a storage miner can receive any storage requests, they must deposit collateral to the network which acts as a pledge to provide storage to the network. Once completed, miners can offer their storage on the Storage Market and set a price for their services. Users who want to store data on Filecoin can set their storage requirements (e.g., storage space required, storage duration, redundancy, and replication factor) and place an ask.

The Storage Market then matches the client and the storage miner. The client then sends their data to the miner, who stores the data in a sector. The sector is then sealed, which is a process that transforms the data into a unique copy of the data called a replica that is associated with the public key of the miner. This sealing process ensures that every replica is a physically unique copy and forms the basis of Filecoin's Proof of Replication algorithm. This algorithm verifies the validity of storage proofs provided using the Merkle tree root of the replica and the hash of the original data.

Over time storage miners are required to consistently prove their ownership of the stored data by regularly running this algorithm. However, consistent checks like this require a lot of bandwidth. The novelty in Filecoin is that to prove data is stored over time and reduce bandwidth usage, the miner generates proofs of replication in sequence, using the output of the previous proof as the input for the current proof. This is executed over a number of iterations that represent the duration that the data is meant to be stored for.

Crust Network

In Crust Network nodes must also first deposit collateral before they can take storage orders on the network. The amount of storage space a node provides to the network determines the maximum amount of collateral, which is staked and allows the node to participate in creating blocks on the network. This algorithm is called Guaranteed Proof of Stake, which guarantees that only nodes that have a stake in the network can provide storage space.

Nodes and users are automatically connected to the Decentralized Storage Market (DSM) that automatically chooses which nodes to store the user's data on. Storage prices are determined based both on user requirements (e.g. storage duration, storage space, replication factor) and network factors (e.g. congestion). When a user submits a storage order, the data is sent to a number of nodes across the network, which split the data and hash the fragments using the machine's Trusted Execution Environment (TEE). Since the TEE is a sealed off hardware component that even the hardware owner cannot access, there is no way the node owner can reconstruct the files on their own.

After the file is stored on the node a work report which includes the hashes of the files is published to the Crust blockchain, together with the remaining storage of the node. From here to ensure data is stored over time the network regularly requests random data checks: within the TEE a random Merkle tree hash is retrieved together with the relevant file fragment, which is decrypted and re-hashed. The new hash is then compared against the expected hash. This implementation of storage proofs is called Meaningful Proof of Work (MPoW).

Sia

As is the case in Filecoin and Crust, storage nodes must deposit collateral to be able to offer storage services. On Sia, the node must decide how much collateral to post: the collateral directly impacts the storage prices for users, but at the same time posting a low collateral means the node has nothing to lose if they disappear from the network. These forces drive nodes towards an equilibrium collateral.

Users are connected to storage nodes through an automatic storage marketplace, which functions in a similar way to that of Filecoin: nodes set their storage prices, and users set their expected prices based on their target price and expected storage duration. Users and nodes are then automatically connected with each other.

After a user and nodes agree on a storage contract, the funds are locked in the contract and the data is split into segments using erasure encoding, each segment is hashed individually using different encryption keys and each piece is then replicated on several different nodes. The storage contract which is recorded on Sia's blockchain records the agreed terms as well as the data's Merkle tree hash. From there to ensure that data is stored for the expected storage duration, storage proofs are regularly submitted to the network. These storage proofs are created based on a randomly selected portion of the originally stored file and a list of hashes from the file's Merkle tree that is recorded on the blockchain. Nodes are rewarded for every storage proof they can submit over time, and finally upon completion of the contract.

On Sia, storage contracts can last a maximum of 90 days. To store files beyond 90 days, users must connect to the network manually using the Sia client software to extend contracts by another 90 days. Skynet, another layer on top of Sia similar to Filecoins Web3.Storage or NFT.Storage platforms, automates this process for the user by having Skynet's own instances of the client software execute contract renewals for users. While this is a workaround, it is not a Sia protocol level solution.

Arweave

Arweave uses a very different pricing model compared to the previous solutions, as Arweave does not allow for temporary storage: on Arweave, all data stored is permanent. On Arweave, the storage price is determined by the cost of storing data on the network for 200 years, assuming that annually those costs reduce by -0.5%. If the cost of storage reduces more than -0.5% in a year, the savings are used to append additional years of storage to the end of the storage duration. In Arweave's own estimates, the -0.5% annual reduction in storage costs is very conservative. If reductions in storage costs are perpetually greater than Arweave's assumption, then the storage duration will continue to grow infinitely, making the storage permanent.

The price of storing files on Arweave is dynamically determined by the network, based on the previously mentioned 200-year storage cost estimate and the difficulty of the network. Arweave is a Proof-of-Work (PoW) blockchain, meaning that nodes must solve a cryptographic hash puzzle to mine the next block. If more nodes join the network, solving the hash puzzle becomes more difficult, thus more computational resources are needed to solve the puzzle. The dynamic price-difficulty adjustments reflect the cost of the additional computational power to ensure nodes remain motivated to stay on the network to mine new blocks.

If a user accepts the price to store files on the network, nodes accept the data and write it to a block. This is where Arweave's Proof-of-Access algorithm comes into play. The Proof-of-Access algorithm works in two phases: first, the node must prove they have access to the previous block in the blockchain, then they must prove access to another block that is selected at random called the recall block. If the node can prove access to both blocks, they enter the PoW phase. In the PoW phase only the miners that could prove access to both blocks start to try to solve the cryptographic hash puzzle. When a miner successfully solves the puzzle, they write the block – and thus the data – to the blockchain. From here, for nodes on the network to be able to mine the next block, they must include the freshly mined block. As a result, the new block and it's data is rapidly permeated across the network.

The miner then receives transaction fees for including the data and block rewards from network token emissions. Apart from the transaction fees, the rest of the price paid by the user is stored in an endowment, which is paid out to miners that hold the data over time. This is only paid out when the network deems that transaction fees and block rewards are not enough to make mining operations profitable. This creates a float of tokens in the endowment, which further extends the 200-year minimum storage duration.

In Arweave's model there is no tracking of storage locations. As a result, if a node does not have access to the data that is being requested, it will ask nodes in a peer list that it maintains locally for the block data.

Storj

In the Storj decentralized storage network, there is no blockchain or block-chain like structure. Not having a blockchain also means that this network does not have network-wide consensus about its state. Instead, tracking data storage locations is handled by satellite nodes, and the storage of data is handled by storage nodes. Satellite nodes can decide which storage nodes to work with to store data, and storage nodes can decide which satellite nodes to accept storage requests from.

Apart from handling the tracking of data locations across storage nodes, satellites are also responsible for billing and payment of storage and bandwidth usage to storage nodes. In this arrangement, storage nodes set their own prices and as long as users are willing to pay those prices, satellites connect them with each other.

When a user wants to store data on Storj, the user must select a satellite node to connect to and share their specific storage requirements. The satellite node will then pick out storage nodes that meet the storage requirements and connect the storage node with the user. The user then directly transmits the files to the storage nodes, while making payment to the satellite. The satellite then pays out storage nodes every month for the files held and the bandwidth used.

To ensure storage nodes are continuously storing the data fragments they are meant to be storing, satellites run regular audits on the storage nodes. The satellite, which does not store any data, selects a random piece of a file fragment before erasure encoding is applied and asks all nodes that store an erasure encoded fragment to validate the data. When sufficient nodes return data, the satellite can identify nodes that report faulty data.

To prevent nodes from disappearing and taking data offline as well as ensuring they consistently verify file fragments through audits, Storj satellites withhold large portions of storage node revenue making it financially unviable to leave the network early or to fail audits. As nodes stay in the network for longer, the proportion of withheld revenue is released. Only when a storage node determines that they want to leave the network after at least 15 months of operation and the storage node signals to the network that they want to leave the network allowing the network to move all data, does the network return remaining withheld funds.

Swarm

While Swarm does not have a layer 1 blockchain for tracking storage requests, storing files on Swarm is handled through Smart Contracts on Ethereum. As a result, storage orders with some details about the files can be tracked. And because in Swarm the address of each chunk is included in the chunk, the neighborhood of the chunk can also be identified. So, when data is requested, nodes within a neighborhood communicate with each other to return the chunks requested by the user.

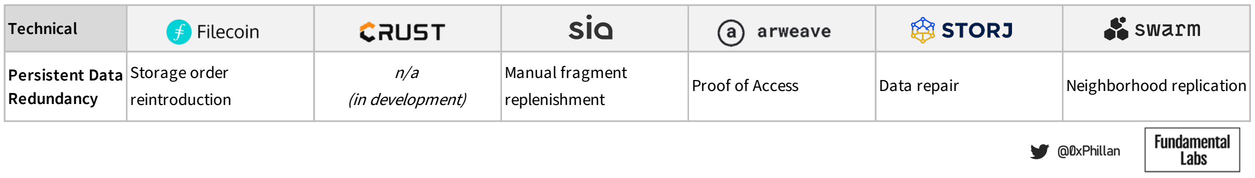
Through client software, Swarm lets users determine the amount of data and duration that data is meant to be stored on Swarm and is calculated using smart contracts. When data is stored on Swarm, the chunks get stored on a node and are then replicated to other nodes in the same neighborhood as the uploading node. When the data is stored on the nodes it gets split into chunks, which map the data to a chunk tree, which builds up a Merkle tree, for which the root hash of the tree is the address that is used to retrieve the file. Hence, the root hash of the tree is proof that the file was properly chunked and stored. Every chunk in the tree further has an inclusion proof embedded, which proves that a chunk is part of a specific chunk tree and can be used as proof of custody if evidence needs to be provided that a node owns a specific chunk of uploaded data.

Nodes that want to sell long-term storage (aka promissory storage) must have a stake verified and locked-in with an Ethereum-based smart contract at the time of making their promise – essentially a security deposit. If, during the promise period, a node fails to prove ownership of the data they promised to store, they lose their entire security deposit.

Finally, to further ensure data isn't removed over time, Swarm employs a random lottery, where nodes are rewarded for holding a random piece of data that is picked through Swarm's RACE lottery system.

Persistent Data Redundancy

If data is stored on a certain number of nodes, it can be assumed that in the long-term as nodes leave and join the network, this data will eventually disappear. To combat this, nodes must ensure that data, in whatever form it is stored, is regularly replicated to consistently maintain a minimum level of redundancy over the user-defined storage duration.



Technical	Filecoin	CRUST	sia	arweave	STORJ	swarm
Persistent Data Redundancy	Storage order reintroduction	*n/a (in development)*	Manual fragment replenishment	Proof of Access	Data repair	Neighborhood replication

@0xPhillan
Fundamental Labs

Figure 21: Data persistence mechanisms of reviewed storage networks

At every block mined on the Filecoin network, the network checks that required proofs for stored data are present and that they are valid. If a certain failure threshold is crossed, the network considers the storage miner faulty, marks the storage order as failed, and reintroduces a new order for the same data on the storage market. If the data is deemed to be unrecoverable, the data is lost and the user gets refunded.

Curst Network, being the youngest network of those reviewed with a Mainnet launch in September 2021, does not yet have a mechanism to replenish file redundancy over time, but this mechanism is currently in development.

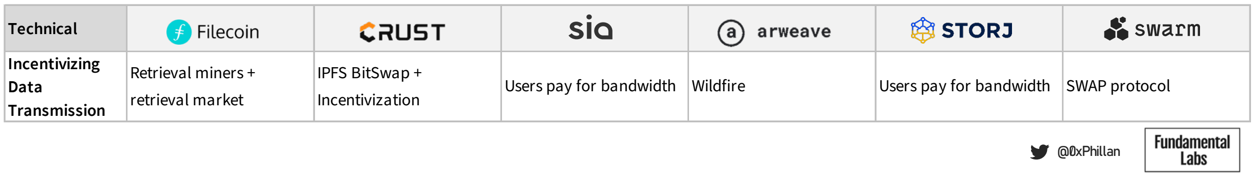
On Sia, the number of erasure encoded fragments available on the network is converted into a health indicator. As nodes and thus erasure encoded fragments disappear over time, the health of a piece of data reduces. To ensure that health remains high, users must manually open the Sia client, which checks the health status and if it is not at 100%, the client replicates the data fragments to other nodes on the network. Sia recommends opening the Sia client once a month to run this data repair process to avoid data falling below an unrecoverable threshold of fragments and the data ultimately disappearing from the network.

Storj follows a similar approach to that of Sia, but instead of having the user take action to ensure sufficient erasure encoded file fragments are on the network, satellite nodes take over this job. Satellite nodes regularly execute data audits on the fragments stored on storage nodes. If an audit returns faulty fragments, the network will reconstruct the file, regenerate the missing pieces and store them back on the network.

For Arweave, consistent data redundancy is achieved through the Proof of Access algorithm that requires nodes to store older data to be able to mine newer data. This requirement means that nodes are incentivized to search and keep older and “rare” blocks, to increase their chances of being allowed to mine the next block and receive mining rewards.

Finally, Swarm ensures persistent redundancy by neighborhood replication as the key measure against data disappearing over time. Swam requires each set of nearest neighbors of a node to hold replicas of that nodes data chunks. Over time as nodes leave or join the network, these neighborhoods reorganize and each node’s nearest neighbors gets updated, requiring them to re-sync the data on their node. This leads to eventual data consistency. This is a continually ongoing process that is executed entirely off-chain.

Incentivizing Data Transmission



Technical	Filecoin	CRUST	sia	arweave	STORJ	swarm
Incentivizing Data Transmission	Retrieval miners + retrieval market	IPFS BitSwap + Incentivization	Users pay for bandwidth	Wildfire	Users pay for bandwidth	SWAP protocol

@0xPhillan
Fundamental Labs

Figure 22: Mechanisms to promote data transmission of reviewed storage networks

After users store data on a network, the data must also be retrievable when a user, another node, or a network process requests access to the data. After nodes receive and store data, they must be willing to transmit it when it is requested.

Filecoin achieves this through a separate type of miner called a retrieval miner. A retrieval miner is a miner that specializes in serving pieces of data and is rewarded in FIL tokens for doing so. Any user in the network can become a retrieval miner (including storage nodes), and retrieval orders are handled through the retrieval market. When a user wants to retrieve data, they place an order on the retrieval market, and retrieval nodes serve it. Although Filecoin is built on the same underlying stack as IPFS, Filecoin does not use IPFS's Bitswap exchange protocol for transmitting user data. Instead, the Bitswap protocol is used to request and receive blocks for the Filecoin blockchain.

Crust directly uses IPFS's Bitswap mechanism to retrieve data and motivate nodes to be willing to transmit data. In Bitswap, every node maintains credit and debt scores of nodes it communicates with. Nodes that only ask for data (for example when a user submits a data retrieval request) eventually have high enough debt that other nodes will stop reacting to its retrieval requests until it also starts to fulfill sufficient retrieval requests itself. Adding to this, in Crust Network the first four nodes that can provide proof of storage for data storage requests are awarded a proportion of the storage fees by the user that initiated the order, meaning that nodes benefit from being able to quickly receive data, which is contingent on how active they are in providing data. As a result, nodes are motivated to continuously fulfill data retrieval requests.

Swarm's SWAP protocol (Swarm Account Protocol) works in the same fashion as IPFS's Bitswap mechanism, with additional functionality integrated. Here also nodes maintain local databases of other nodes' bandwidth credits and debts, creating a service-for-service relationship between nodes. However, SWAP assumes that sometimes there just isn't data needed from one of the nodes to re-balance credit to debt ratio in the short term. To solve this, nodes can pay other nodes cheques to repay their debt. A cheque is an off-chain voucher that a node commits to pay another node, that can be redeemed for BZZ tokens through a smart contract on the Ethereum blockchain.

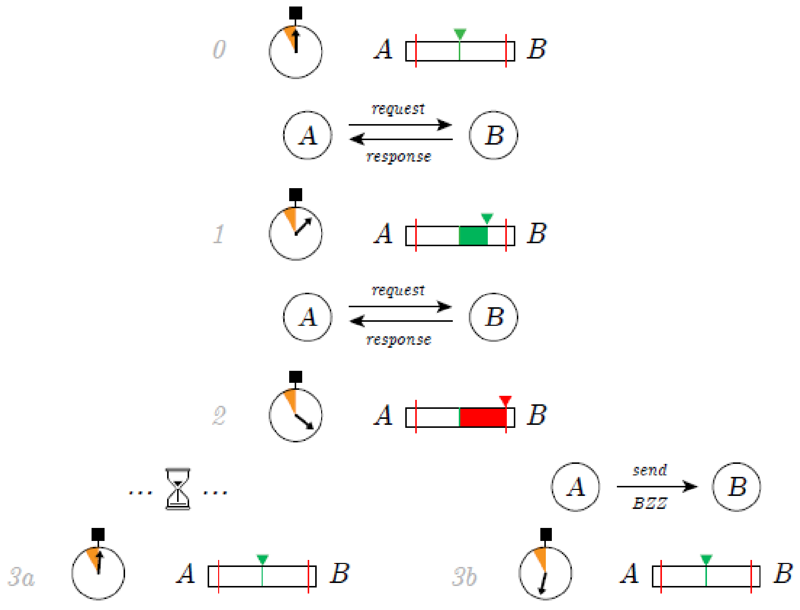


Figure 23: Swarm Accounting Protocol. Source: Swarm whitepaper.

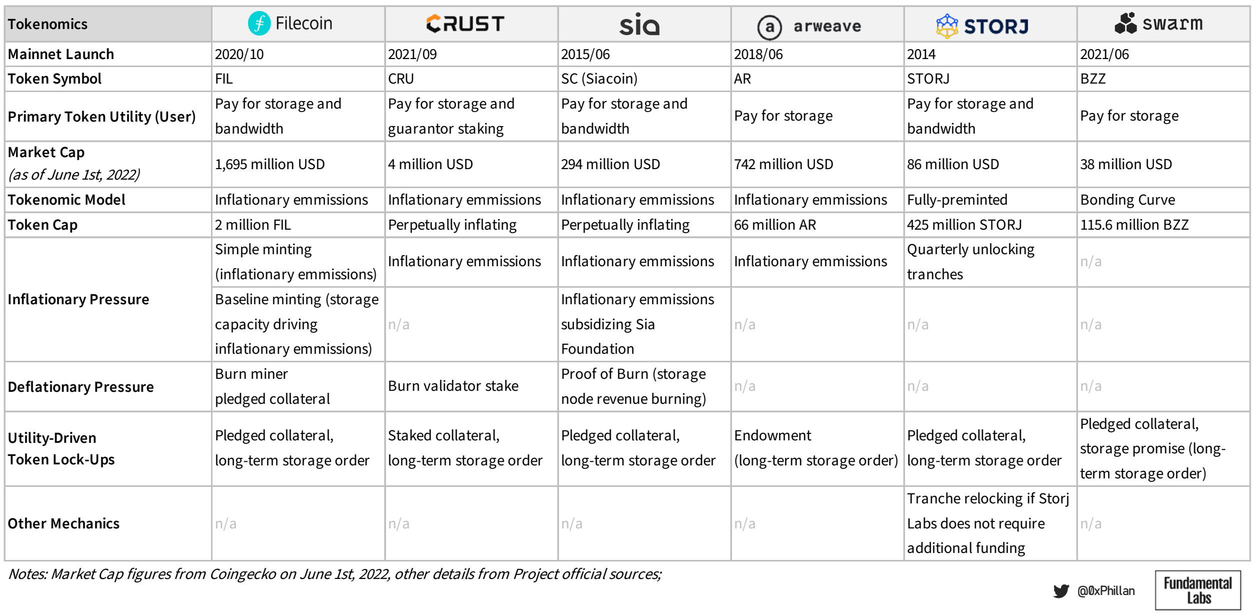
In both Sia and Storj, users pay for bandwidth that is used. In Sia, upload, download and repair bandwidth are paid by the user, while in Storj bandwidth required for upload is covered by the storage node. In Storj, this is meant to discourage nodes from deleting data immediately after it has been received. Due this set up, nodes have no reason to avoid using bandwidth, as bandwidth is paid for at a price they dictate before accepting storage orders.

Finally, in Arweave, nodes rationalize their bandwidth allocation based on how reliably a peer node shares transactions and blocks and how reliably it responds to requests. The node then keeps track of these factors for all peer nodes it interacts with, and preferably communicates with peer nodes that score higher. This promotes willingness for nodes to transfer data and share information, as receiving blocks in a slower fashion means they have less time compared to other nodes to solve the cryptographic hash puzzle of Arweave's PoA consensus algorithm.

Tokenomics

Finally, networks must decide on a tokenomics design. While the above ensures data will be available whenever it should be available, tokenomics design ensures that the network will be around in the future. As without the network, there would be no underlying data for users and hosts to interact with. Here we will take a closer look at what tokens are used for and the factors that impact token supply.

Note: while all the above sections affect tokenomics design, here we primarily focus on token utility and token emissions design



Tokenomics	Filecoin	CRUST	sia	arweave	STORJ	swarm
Mainnet Launch	2020/10	2021/09	2015/06	2018/06	2014	2021/06
Token Symbol	FIL	CRU	SC (Siacoin)	AR	STORJ	BZZ
Primary Token Utility (User)	Pay for storage and bandwidth	Pay for storage and guarantor staking	Pay for storage and bandwidth	Pay for storage	Pay for storage and bandwidth	Pay for storage
Market Cap *(as of June 1st, 2022)*	1,695 million USD	4 million USD	294 million USD	742 million USD	86 million USD	38 million USD
Tokenomic Model	Inflationary emissions	Inflationary emissions	Inflationary emissions	Inflationary emissions	Fully-preminted	Bonding Curve
Token Cap	2 million FIL	Perpetually inflating	Perpetually inflating	66 million AR	425 million STORJ	115.6 million BZZ
Inflationary Pressure	Simple minting (inflationary emissions)	Inflationary emissions	Inflationary emissions	Inflationary emissions	Quarterly unlocking tranches	n/a
	Baseline minting (storage capacity driving inflationary emissions)	n/a	Inflationary emissions subsidizing Sia Foundation	n/a	n/a	n/a
Deflationary Pressure	Burn miner pledged collateral	Burn validator stake	Proof of Burn (storage node revenue burning)	n/a	n/a	n/a
Utility-Driven Token Lock-Ups	Pledged collateral, long-term storage order	Staked collateral, long-term storage order	Pledged collateral, long-term storage order	Endowment (long-term storage order)	Pledged collateral, long-term storage order	Pledged collateral, storage promise (long-term storage order)
Other Mechanics	n/a	n/a	n/a	n/a	Tranche relocking if Storj Labs does not require additional funding	n/a

Notes: Market Cap figures from Coingecko on June 1st, 2022, other details from Project official sources;

@0xPhillan Fundamental Labs

Figure 24: Tokenomics design decisions of reviewed storage networks

In the Filecoin network, the FIL token is used to pay for storage orders and retrieval bandwidth. The Filecoin network has an inflationary token emissions model using two types of minting: Simple minting, which emits new tokens as block rewards on a 6-year halving schedule (compared to Bitcoin's 4 years) and baseline minting, which creates additional token emissions if the network reaches total storage space milestones (*see figure 23*). This means that storage miners on the network are incentivized to provide as much storage to the network as possible.

There are two ways that circulating supply of FIL on the market can be reduced. If miners fail to live up to their commitments, their pledged collateral is burned and permanently removed from the network (30.5 million FIL at time of writing). Finally, time-locked storage orders temporarily remove FIL from circulation and are paid out to miners over time. This means that the more storage is used, the fewer coins are in circulation in the short-term creating deflationary price pressure on the token value.

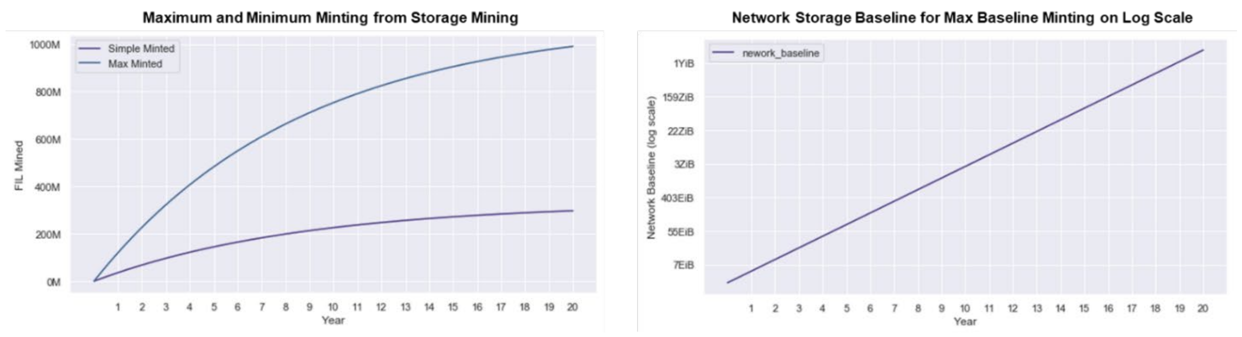
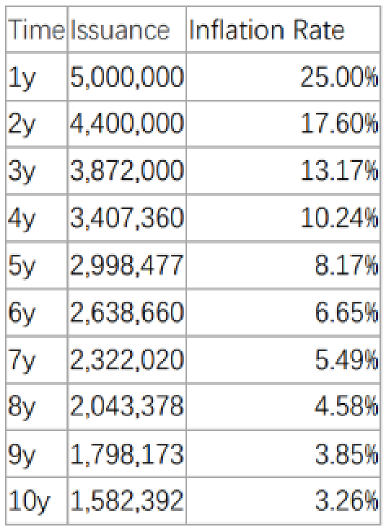


Figure 25: Max and Min Minting from Storage Mining and Max Baseline Minting. Source: https://filecoin.io/blog/filecoin-circulating-supply/

In Crust Network, the CRU token is used to pay for storage orders and used for staking as part of Crust Network's Guaranteed Proof of Stake (GPoS) consensus mechanism. In this model, network token emissions are also inflationary and used as block rewards. Crust Network, however, does not have a token cap – for 12 years inflation is reduced YoY, after which token inflation continues perpetually at 2.8%.

In Crust, the stake a validator and its guarantors lock also acts as pledged collateral. If it is detected that a validator acts maliciously or is unable to provide the required proofs, their stake is slashed and burned. Finally, staked collateral and time-locked storage orders temporarily remove tokens from circulation. Since miner network storage capacity determines a miner’s staking limit, miners are incentivized to provide more storage capacity to maximize their staking income proportional to other miners. Staked tokens and tokens locked in time-locked storage orders create deflationary price pressure on the token value.



Time	Issuance	Inflation Rate
1y	5,000,000	25.00%
2y	4,400,000	17.60%
3y	3,872,000	13.17%
4y	3,407,360	10.24%
5y	2,998,477	8.17%
6y	2,638,660	6.65%
7y	2,322,020	5.49%
8y	2,043,378	4.58%
9y	1,798,173	3.85%
10y	1,582,392	3.26%

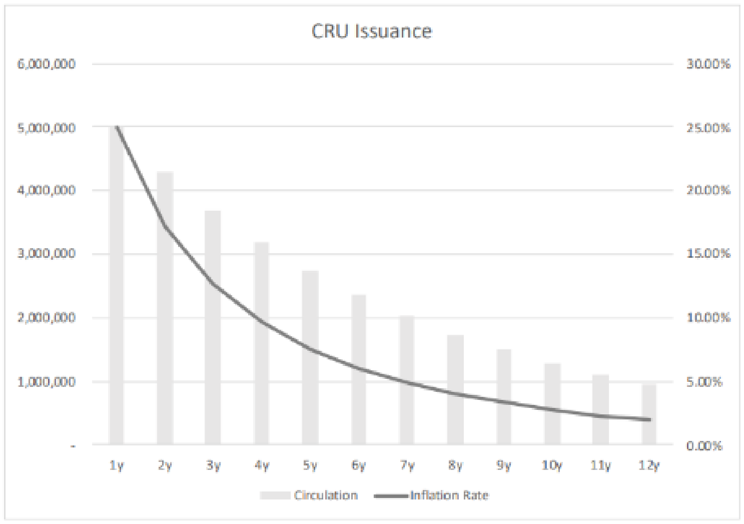


Figure 26: Crust Network token emissions. Source: Crust Economy Whitepaper (https://gw.crustapps.net/ipfs/QmRYJN6V5BzwnXp7A2Avcp5WXkgzyunQwqP3Es2Q789phF)

Sia has two coins that are used in the network; one is the utility coin Siacoin and the other is a revenue-generating coin called Siafunds. Siafunds were sold to the public when the network first went live, and are mostly held by the Sia Foundation. Siafunds entitle holders to a certain % of revenue for every storage order placed on the network. Siafunds do not have a substantial impact on the tokenomics of Sia and are hence not covered further here.

Siacoin has an inflationary token emissions model that act as block rewards, with no token cap. The block rewards perpetually decreased in a linear fashion per block until block height 270,000 (roughly 5 years of operation; reached in 2020). From then onwards, every block includes a fixed block reward of 30,000 SC. In 2021 the Sia Foundation hard-forked the Sia network to include an additional 30,000 SC subsidy per block to fund the Sia Foundation, a non-profit entity meant to support, develop, and promote the Sia network.

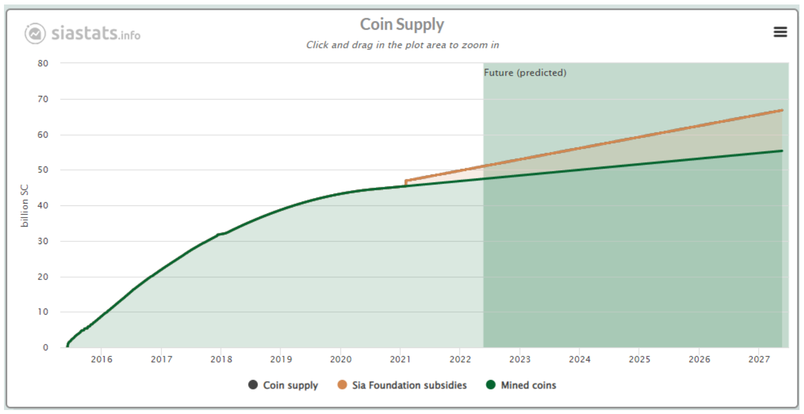


Figure 27: Annual growth of Siacoin supply and Foundation coin minting. Source: https://siastats.info/macroeconomics

Sia also uses a Proof of Burn mechanism, that requires miners to burn 0.5-2.5% of their revenue to prove there are legitimate nodes on the network. This creates downward pressure on token supply, although annual burns only reflect roughly 500k SC compared to 3.14 billion SC of token emissions. Finally, pledged collateral and long-term storage orders also temporarily remove tokens from circulation in Sia.

The Arweave network's native token is the AR token, which is used to pay for perpetual and theoretically permanent storage on the Arweave network. Arweave also uses an inflationary token model, with a maximum supply cap of 66 million AR tokens. In Arweave, the primary deflationary impact is driven by Arweave's endowments, which is Arweave's implementation of a long-term storage contract. When a user wants to store files on Arweave, only a small amount of the storage fee goes to the miner – the rest is placed into an endowment which covers at least 200 years of storage time using Arweave's highly conservative assumptions. That means, any storage order that is placed locks tokens away for at least 200 years, and is slowly paid out over this 200 year duration.

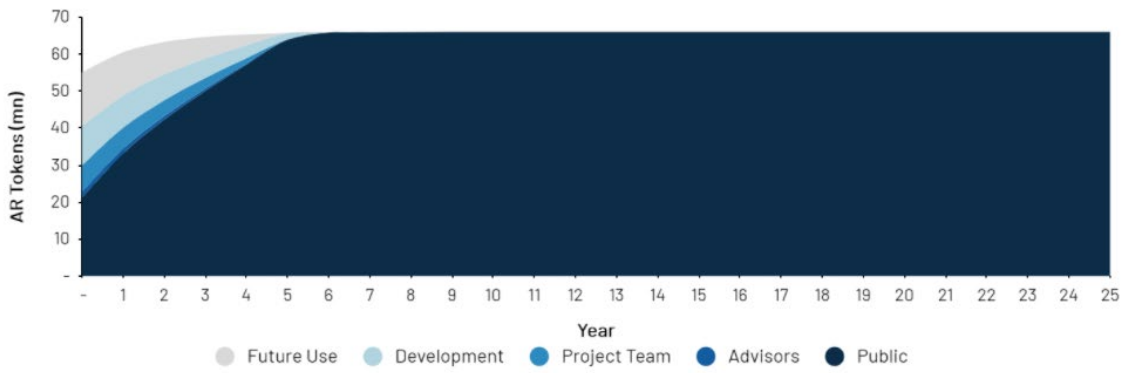
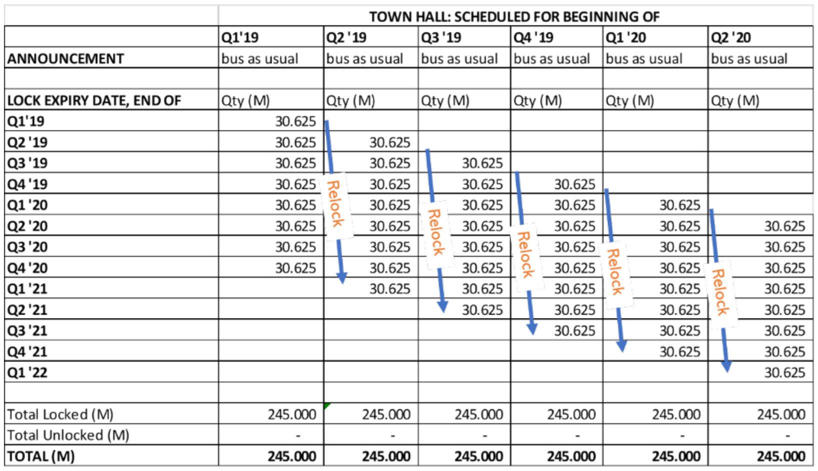


Figure 28: AR token inflation and team allocation. Source: https://medium.com/amber-group/arweave-enabling-the-permaweb-870ade28998b

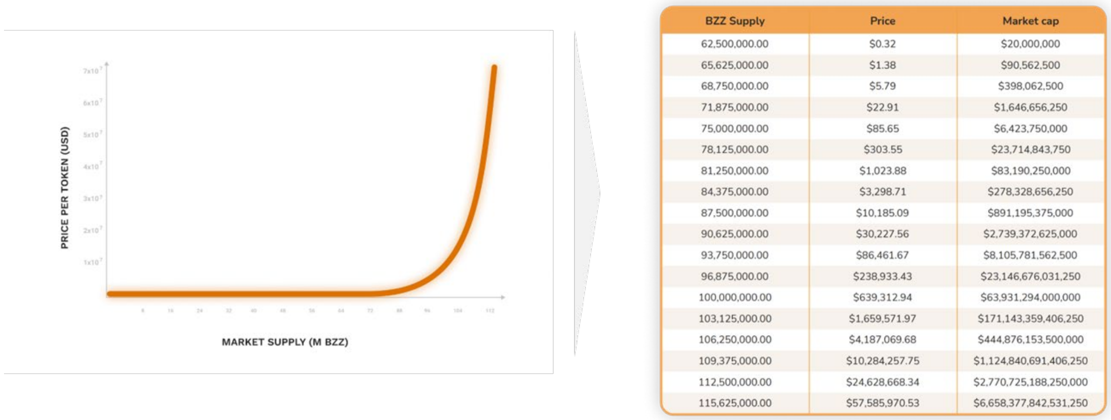
In Storj, the STORJ token is used to pay for storage and bandwidth. All 425 million STORJ tokens are pre-minted as ERC20 tokens on the Ethereum network. Previously, the Bitcoin-based SJCX token was used, however, in 2017 Storj Labs converted moved their tokens to Ethereum and renamed the ticker to STORJ. Of the STORJ tokens, currently 190.8 million STORJ tokens are locked up in six smart-contract controlled tranches in Storj Labs' custody, while 234.1 million STORJ tokens are unlocked. Every quarter a tranche is unlocked, and when Storj Labs deems they do not need the funds to finance operations, they re-lock a tranche. This means that nearly half of STORJ supply is in direct control of Storj Labs, however, if they wanted to cash out they would have to wait 6 quarters, due to the funds being locked behind smart contracts. In Storj, pledged collateral by storage nodes and long-term storage orders also drive down circulating supply, as these tokens get temporarily locked-up.


Relock
Relock
Relock
Relock
Relock

	TOWN HALL: SCHEDULED FOR BEGINNING OF					
	Q1'19	Q2 '19	Q3 '19	Q4 '19	Q1 '20	Q2 '20
ANNOUNCEMENT	bus as usual	bus as usual	bus as usual	bus as usual	bus as usual	bus as usual
LOCK EXPIRY DATE, END OF	Qty (M)	Qty (M)	Qty (M)	Qty (M)	Qty (M)	Qty (M)
Q1'19	30.625					
Q2 '19	30.625	30.625				
Q3 '19	30.625	30.625	30.625			
Q4 '19	30.625	30.625	30.625	30.625		
Q1 '20	30.625	30.625	30.625	30.625	30.625	
Q2 '20	30.625	30.625	30.625	30.625	30.625	30.625
Q3 '20	30.625	30.625	30.625	30.625	30.625	30.625
Q4 '20	30.625	30.625	30.625	30.625	30.625	30.625
Q1 '21		30.625	30.625	30.625	30.625	30.625
Q2 '21			30.625	30.625	30.625	30.625
Q3 '21				30.625	30.625	30.625
Q4 '21					30.625	30.625
Q1 '22						30.625
Total Locked (M)	245.000	245.000	245.000	245.000	245.000	245.000
Total Unlocked (M)	-	-	-	-	-	-
TOTAL (M)	**245.000**	**245.000**	**245.000**	**245.000**	**245.000**	**245.000**

Figure 29: Tranche relocking schedule. Source: https://www.storj.io/blog/using-timelocked-tokens-to-support-long-term-sustainability

Finally, Swarm uses the BZZ token as a utility token to pay for storage on the network. The tokenomics model deployed by Swarm is a bonding curve, which determines the price of the token based on the supply thereof. Users can sell their tokens back to the bonding curve at any time at current market price. In Swarm, long-term storage orders require collateral to be pledged in the form of "promises". Similar to the previous networks, more storage usage means fewer tokens available on the market, which would have deflationary pressures on the token price, as users who want to buy the token must purchase from the bonding curve, which will increase the price with every additional token sold.



BZZ Supply	Price	Market cap
62,500,000.00	$0.32	$20,000,000
65,625,000.00	$1.38	$90,562,500
68,750,000.00	$5.79	$398,062,500
71,875,000.00	$22.91	$1,646,656,250
75,000,000.00	$85.65	$6,423,750,000
78,125,000.00	$303.55	$23,714,843,750
81,250,000.00	$1,023.88	$83,190,250,000
84,375,000.00	$3,298.71	$278,328,656,250
87,500,000.00	$10,185.09	$891,195,375,000
90,625,000.00	$30,227.56	$2,739,372,625,000
93,750,000.00	$86,461.67	$8,105,781,562,500
96,875,000.00	$238,933.43	$23,146,676,031,250
100,000,000.00	$639,312.94	$63,931,294,000,000
103,125,000.00	$1,659,571.97	$171,143,359,406,250
106,250,000.00	$4,187,069.68	$444,876,153,500,000
109,375,000.00	$10,284,257.75	$1,124,840,691,406,250
112,500,000.00	$24,628,668.34	$2,770,725,188,250,000
115,625,000.00	$57,585,970.53	$6,658,377,842,531,250

Figure 30: Shape of BZZ bonding curve. Source: https://medium.com/ethereum-swarm/swarm-and-its-bzzaar-bonding-curve-ac2fa9889914

Discussion

It is impossible to say that one network is objectively better than another. When designing a decentralized storage network, there are countless trade-offs that must be considered. While Arweave is great for permanent storage of data, Arweave is not necessarily suitable to move Web2.0 industry players to Web3.0 – not all data needs to be permanent. However, there is a strong sub-sector of data that does require permanence: NFTs and dApps.

If we look at other networks, we witness similar trade-offs: Filecoin is incentivizing Web2.0 storage providers to move their Storage to Web3.0 and is thus a driving a force in the adoption of decentralization. Filecoin's Proof of Spacetime algorithm is computationally heavy with slow write speeds, which means that it is more suitable for higher-value data (like their slogan "storing humanity's most important data") that does not change often. However, many applications require constant changes to their data. Crust Network fills this gap by providing storage that is computationally less intensive to prove.

Looking at how these projects store data, we can see that Crust Network and Arweave are the only ones that do not use erasure encoding. One may think that erasure encoding is the better option, due to the majority of projects using it, but that is not necessarily the case. Arweave does not need erasure encoding, as the Proof of Access consensus mechanism paired with the Wildfire mechanic ensures data is replicated aggressively across the network. On Crust Network data is replicated to at least 20 nodes, and in many cases to over 100 nodes. While this does have greater up-front bandwidth, being able to retrieve data from a large number of nodes simultaneously makes file retrieval fast and adds strong redundancy in case of failures or nodes leaving the network. Crust Network needs this high level of redundancy, because it does not yet have a data replenishment or repair mechanism like the other chains. Of the decentralized storage networks reviewed here, Crust Network is the youngest.

If we compare any project to Filecoin, we will see other chains support a higher degree of storage decentralization, but may be more centralized in other aspects, such as Storj in which a single satellite node can control a large cluster of storage nodes. If that satellite node goes offline, all access to files is lost. However, having satellites control repair processes autonomously is a huge upgrade compared with the manual repair processes required in Sia. Storj also gives Web2.0 users an easier first step into decentralized storage, by allowing any form of payment between users and satellites.

If we further compare Storj's approach to decentralization to that of the other projects, we will see that Storj's lack of system-wide consensus is indeed a purposeful design decision to increase network performance, as the network does not need to wait on consensus to proceed with fulfilling storage requests.

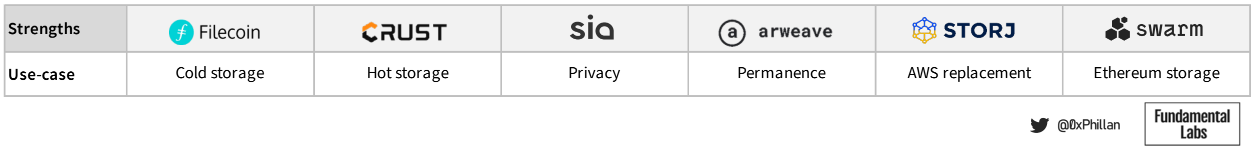
Swarm and Storj are the only protocols that do not have their own layer1 blockchain network, and instead rely on ERC20 tokens deployed on the Ethereum network. Swarm is directly integrated in the Ethereum network, with storage orders being directly controlled through Ethereum smart contracts.

This makes Swarm a strong choice for Ethereum native dApps and for storing the metadata of Ethereum-based NFTs, due to the convenience of proximity and same environment. Storj, while also based on Ethereum, is not that heavily integrated into the Ethereum ecosystem, however, can also benefit from smart contracts.

Sia and Filecoin use a storage market mechanism where storage providers can set their prices and are matched with storage users that are willing to pay those prices based on specific requirements, while in the other network storage pricing is protocol-dictated based on network-specific factors. Using a storage market means that users get greater choice regarding how their files are stored and secured but having the price set by the network reduces complexity and makes for an easier user experience.

Conclusion

There is no single best approach for the various challenges decentralized storage networks face. Depending on the purpose of the network and the problems it is attempting to solve, it must make trade-offs on both technical and tokenomics aspects of network design.



Strengths	Filecoin	CRUST	sia	arweave	STORJ	swarm
Use-case	Cold storage	Hot storage	Privacy	Permanence	AWS replacement	Ethereum storage

@0xPhillan Fundamental Labs

Figure 31: Summary of strong uses-cases for reviewed storage networks

In the end, the purpose of the network and the specific use-cases it attempts to optimize will determine the various design decisions.

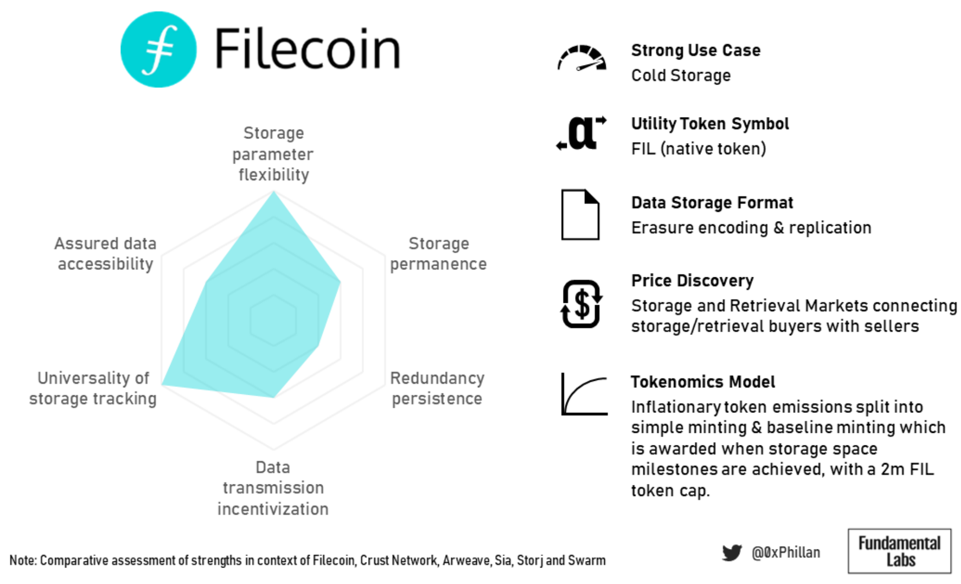
Comparative Network Profiling

What follows are summative profiles of the various storage networks comparing them amongst each other on a set of scales defined below. The scales used reflect comparative dimensions of these networks, however it should be noted that the approaches to overcome challenges of decentralized storage are in many cases not better or worse, but instead merely reflect design decisions.

- **Storage parameter flexibility**: the extent to which users have control over file storage parameters
- **Storage permanence**: the extent to which file storage can achieve theoretical permanence by the network (i.e., without intervention)
- **Redundancy persistence**: the networks ability to maintain data redundancy through replenishment or repair
- **Data transmission incentivization**: the extent to which the network ensures nodes generously transmit data
- **Universality of storage tracking**: the extent to which there is a consensus among nodes regarding the storage location of data
- **Assured data accessibility**: the ability of the network to ensure that a single actor in the storage process cannot remove access to files on the network

Higher scores indicate greater ability in each of the above.

Filecoin's tokenomics support growing the total network's storage space, which serves to store large amounts of data in an immutable fashion. Furthermore, their storage algorithm lends itself more to data that is unlikely to change much over time (cold storage).



Strong Use Case
Cold Storage

Utility Token Symbol
FIL (native token)

Data Storage Format
Erasure encoding & replication

Price Discovery
Storage and Retrieval Markets connecting storage/retrieval buyers with sellers

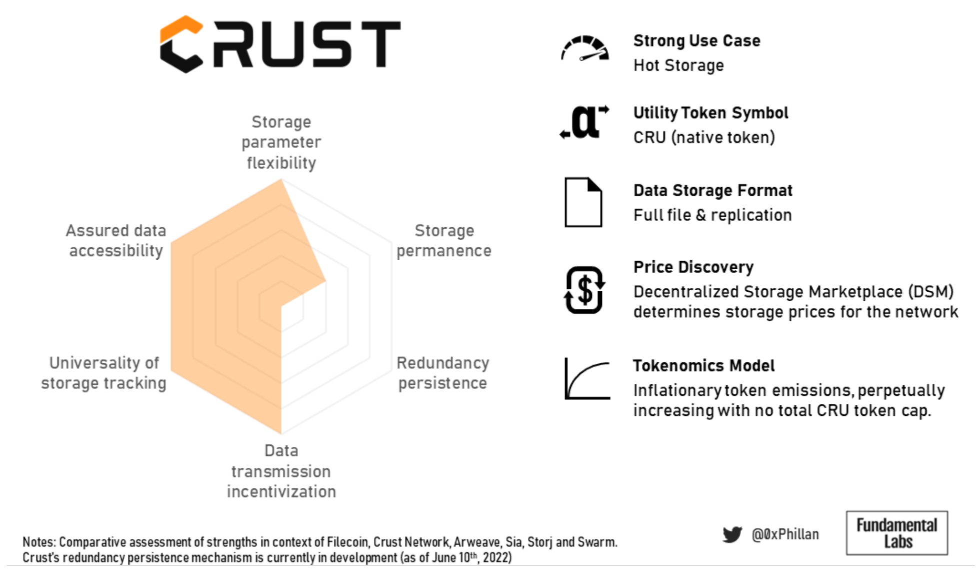
Tokenomics Model
Inflationary token emissions split into simple minting & baseline minting which is awarded when storage space milestones are achieved, with a 2m FIL token cap.

Note: Comparative assessment of strengths in context of Filecoin, Crust Network, Arweave, Sia, Storj and Swarm

Figure 32: Summative profile of Filecoin

Crust's tokenomics ensure hyper-redundancy with fast retrieval speeds which make it suitable for high traffic dApps and make it suitable for fast retrieval of data of popular NFTs.

Crust scores lower on storage permanence, as without persistent redundancy its ability to deliver permanent storage is heavily impacted. Nonetheless, permanence can still be achieved through manually setting an extremely high replication factor.



Strong Use Case
Hot Storage

Utility Token Symbol
CRU (native token)

Data Storage Format
Full file & replication

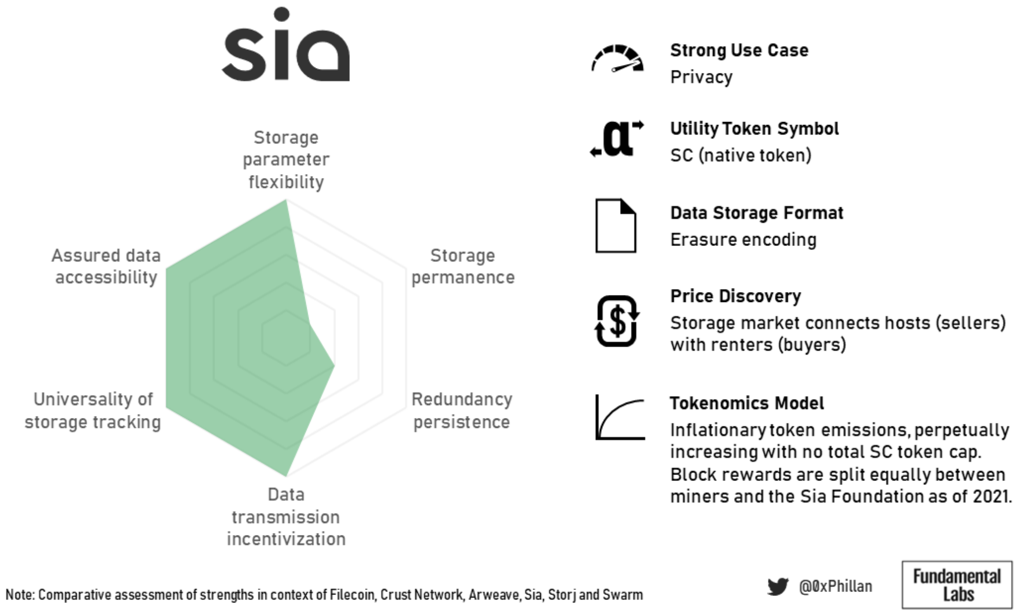
Price Discovery
Decentralized Storage Marketplace (DSM) determines storage prices for the network

Tokenomics Model
Inflationary token emissions, perpetually increasing with no total CRU token cap.

Notes: Comparative assessment of strengths in context of Filecoin, Crust Network, Arweave, Sia, Storj and Swarm.
Crust's redundancy persistence mechanism is currently in development (as of June 10th, 2022)

Figure 33: Summative profile of Crust

Sia is all about privacy. The reason manual health restoration by the user is required, is because nodes do not know what data fragments they are storing, and which data these fragments belong to. Only the data owner can reconstruct the original data from the fragments in the network.



Strong Use Case
Privacy

Utility Token Symbol
SC (native token)

Data Storage Format
Erasure encoding

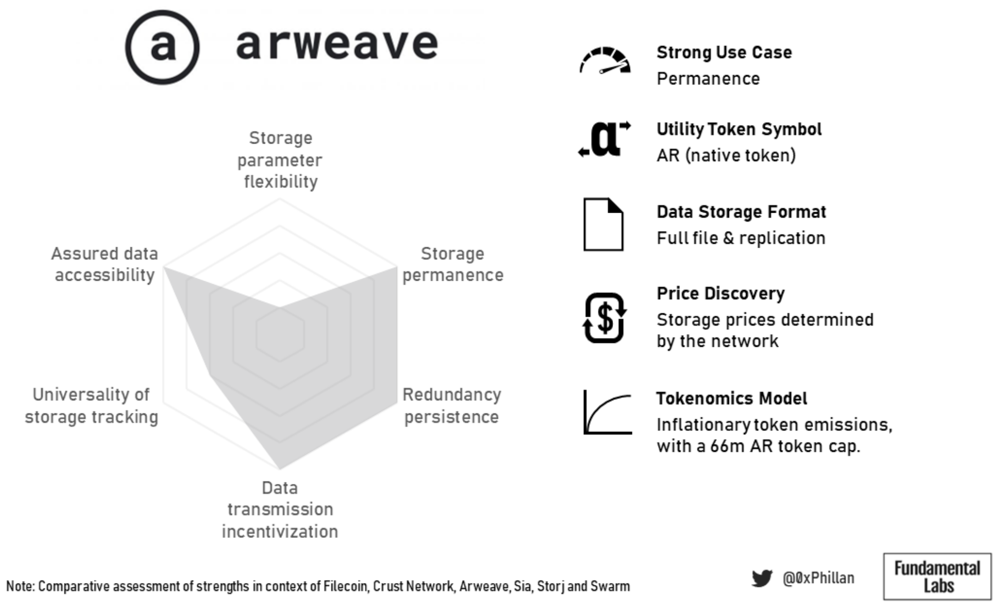
Price Discovery
Storage market connects hosts (sellers) with renters (buyers)

Tokenomics Model
Inflationary token emissions, perpetually increasing with no total SC token cap. Block rewards are split equally between miners and the Sia Foundation as of 2021.

Note: Comparative assessment of strengths in context of Filecoin, Crust Network, Arweave, Sia, Storj and Swarm

Figure 34: Summative profile of Sia

In contrast, Arweave is all about permanence. That is also reflected in their endowment design, which makes storage more costly but also makes them a highly attractive choice for NFT storage.



Strong Use Case
Permanence

Utility Token Symbol
AR (native token)

Data Storage Format
Full file & replication

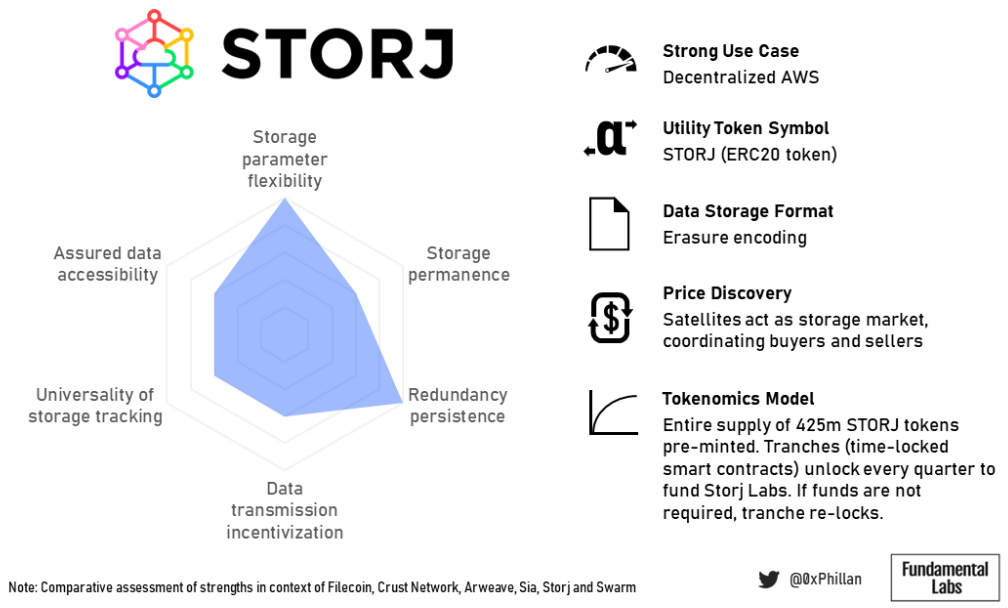
Price Discovery
Storage prices determined by the network

Tokenomics Model
Inflationary token emissions, with a 66m AR token cap.

Note: Comparative assessment of strengths in context of Filecoin, Crust Network, Arweave, Sia, Storj and Swarm

Figure 35: Summative profile of Arweave

Storj's business model seems to heavily factor in their billing and payment approach: Amazon AWS S3 users are more familiar with monthly billing. By removing complex payment and incentive systems often found in blockchain-based systems, Storj Labs sacrifices some decentralization but significantly reduces the barrier to entry for their key target group of AWS users.



Strong Use Case
Decentralized AWS

Utility Token Symbol
STORJ (ERC20 token)

Data Storage Format
Erasure encoding

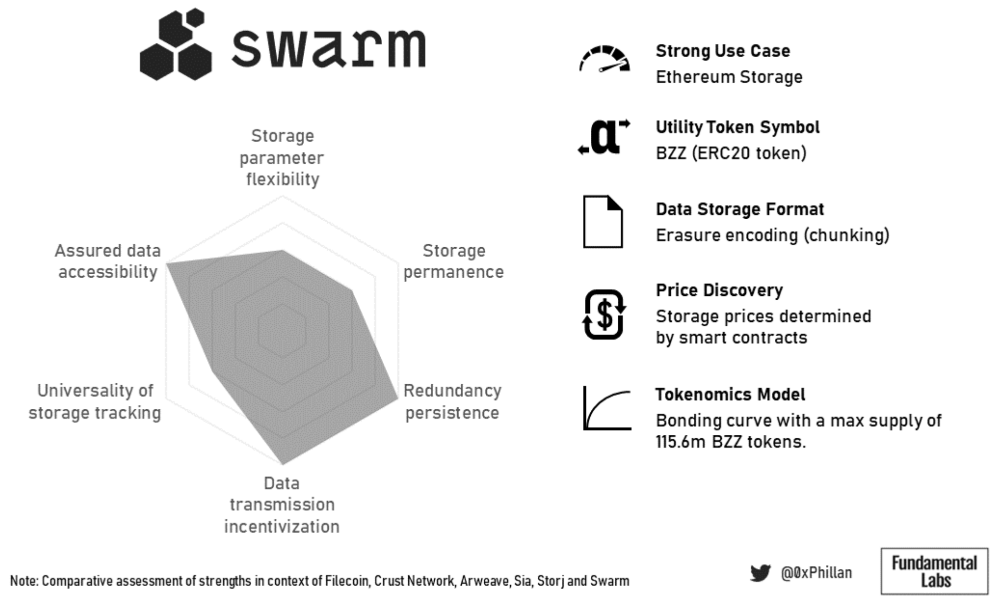
Price Discovery
Satellites act as storage market, coordinating buyers and sellers

Tokenomics Model
Entire supply of 425m STORJ tokens pre-minted. Tranches (time-locked smart contracts) unlock every quarter to fund Storj Labs. If funds are not required, tranche re-locks.

Note: Comparative assessment of strengths in context of Filecoin, Crust Network, Arweave, Sia, Storj and Swarm

Figure 36: Summative profile of Storj

Swarm's bonding curve model ensures storage costs remain relatively low overtime as more data is stored on the network, and its proximity to the Ethereum blockchain make it a strong contender to become the primary storage for more complex Ethereum-based dApps.



Strong Use Case
Ethereum Storage

Utility Token Symbol
BZZ (ERC20 token)

Data Storage Format
Erasure encoding (chunking)

Price Discovery
Storage prices determined by smart contracts

Tokenomics Model
Bonding curve with a max supply of 115.6m BZZ tokens.

Note: Comparative assessment of strengths in context of Filecoin, Crust Network, Arweave, Sia, Storj and Swarm

Figure 37: Summative profile of Swarm

The next frontier

Returning to the Web3 infrastructural pillars (consensus, storage, computation), we see that the decentralized storage space has a handful of strong players that have positioned themselves within the market for their specific use cases. This does not exclude new networks from optimizing existing solutions or occupying new niches, but this does raise the question: **what's next?**

The answer is: **computation**. The next frontier in achieving a truly decentralized internet is decentralized computation. Currently only few solutions exist that bring to market solutions for trustless, decentralized computation that can power complex dApps, which are capable of more complex computation at far lower cost than executing smart contracts on a blockchain.

Internet computer (ICP) and Holochain (HOLO) are networks that occupy a strong position in the decentralized computation market at time of writing. Nonetheless, the computational space is not nearly as crowded as the consensus and storage spaces. Hence, strong competitors are bound to enter the market sooner or later and position themselves accordingly. One such competitor is **Stratos (STOS)**. Stratos offers a unique network design through its decentralized data mesh technology, combining blockchain technology with decentralized storage, decentralized computation and decentralized databases.

We see decentralized computation, and specifically the network design of the Stratos network, as areas for future research.

Closing

Thank you for reading this research piece on decentralized storage. If you enjoy research that seeks to uncover the fundamental building blocks of our shared Web3 future, consider following @FundamentalLabs on Twitter.

Did I miss any interesting concepts, or other valuable information? Please reach out to me on Twitter @0xPhillan so we can enhance this research together.

Section 4:

Decentralized Storage Networks Deep-Dive

This section covers a technical deep-dive of the various storage networks covered in this research. This section is meant to give readers a high-level technical overview of each protocol and certain protocol mechanics.

Note: In this section there is much overlap with the previous section, and it is intended to be a deeper dive into various aspects of these decentralized storage solutions.

IPFS-based storage solutions

Filecoin, Crust and Pinata are all based on the Interplanetary File System (IPFS). IPFS, developed by Protocol Labs, is a popular decentralized file storage protocol that takes care of content addressing. This means IPFS can help users store files, as well as provide a way to find and retrieve those files, in the decentralized IPFS network. In simple terms, IPFS can be thought of as a communication protocol between nodes in a decentralized network that facilitates storage-related operations.

While the above is an oversimplification, it helps to illustrate why IPFS-based storage solutions have emerged to begin with: IPFS does not have any incentive mechanisms built in for people to operate nodes or store your files. Node operators need to pay for storage space, computational power and bandwidth, and without an incentive system it becomes difficult for node operators to justify operating a node.

The Interplanetary File System

The IPFS protocol hashes data with metadata creating a unique identifier called a content ID (CID). Once that CID is created, the content in that CID is unique and becomes immutable: it cannot be changed. If a user would like to make changes to a file, they need to re-upload the file which will be hashed again and a new CID is created. So when you look for content hosted on IPFS, you are actually telling the IPFS network to retrieve a specific piece of content instead of retrieving the content at a specific location.

IPFS nodes thus need a way to know where the content that a CID references is located. For this IPFS uses distributed hash tables (DHTs) and the Kademlia algorithm to map the addresses of peer nodes that hold specific content. When a user asks an IPFS node to retrieve the content of a CID, the node will query the DHT to retrieve addresses of all peers that have that particular CID.

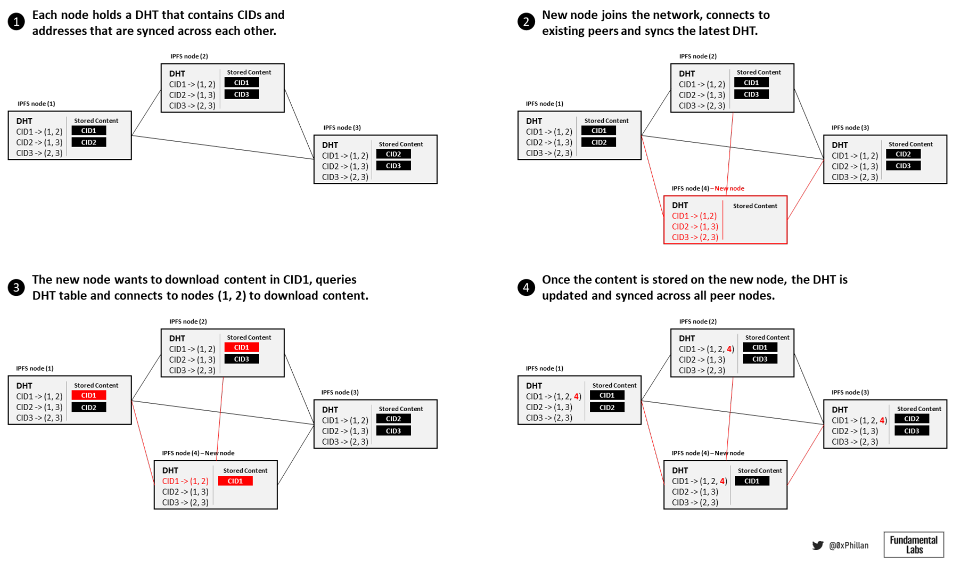

1 Each node holds a DHT that contains CIDs and addresses that are synced across each other.
2 New node joins the network, connects to existing peers and syncs the latest DHT.
3 The new node wants to download content in CID1, queries DHT table and connects to nodes (1, 2) to download content.
4 Once the content is stored on the new node, the DHT is updated and synced across all peer nodes.
IPFS node (1)
IPFS node (2)
IPFS node (3)
IPFS node (4) – New node
DHT
Stored Content
CID1 -> (1, 2)
CID2 -> (1, 3)
CID3 -> (2, 3)
CID1 -> (1, 2, 4)
CID1
CID2
CID3
@0xPhillan
Fundamental Labs

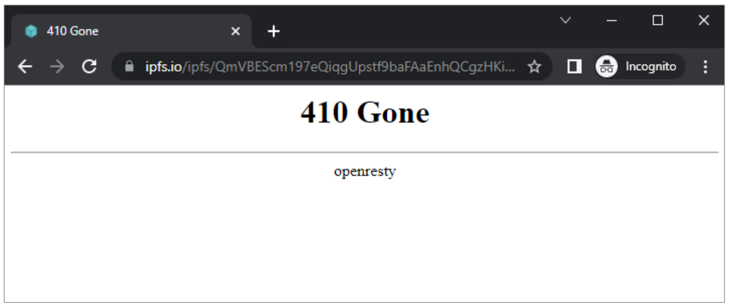
Figure 38: Brief overview of IPFS DHT and communication mechanism. Note: new nodes are bootstrapped with a list of common peers.

If content needs to be regularly refreshed (for example new versions of a dApp are released), instead of re-sharing a new CID, developers can use the Interplanetary Name System (IPNS) instead. In IPNS, a name is the hash of a public key. It is associated with a record containing information about the hash it links to that is signed by the corresponding private key. This way new records can be signed and published under one IPNS hash instead of sharing updated CIDs every time.

The key weakness of IPFS is that content is only available as a node is willing to store the data. During the duration that the data is stored it is indeed immutable, i.e., the content referenced by a CID will never change. However, without any incentive, a node can remove the data or go offline entirely, making the content unavailable. Furthermore, to access IPFS content you need to either run your own node (you don't need to broadcast content, but you will have DHTs that are regularly updated) or you need to communicate with an IPFS gateway, which is a node hosted by a third party that helps transmit the data you want to retrieve.

Furthermore, IPFS automatically deletes files from a node after a specific period of time. To ensure nodes maintain copies of data stored on a node, those files need to be "pinned". It is this pinning action that decentralized file storage providers use to store files for extended periods of time when using IPFS.

The Brave browser makes this process seamless, so much so that retrieving data from a CID feels no different than surfing the web. While interacting with a gateway is inherently considered no riskier than surfing the web, certain risks that you face when browsing the internet are also present when interacting with an IPFS gateway. For example, a malicious gateway could lie about the content it is serving you or track specific content retrievals.



410 Gone

openresty

Figure 39: Unavailable content through IPFS. https://ipfs.io/ipfs/QmVBEScm197eQiqgUpstf9baFAaEnhQCgzHKiXnkCoED2c

The main strengths of IPFS are that the protocol is open source, enabling any project to use decentralized storage , and that it is easy to integrate into existing projects with a strong community that backs the technology.

We will take a closer look at Filecoin, Crust Network and Pinata below, who use IPFS to enable decentralized storage.

Filecoin

Filecoin aims to be marketplace for data storage and retrieval, positioning itself to compete with data giants such as Amazon, Microsoft and Google and content delivery networks like Cloudflare. The Filecoin blockchain builds on the content addressing of IPFS to add long-term data persistence using cryptoeconomic incentives. Filecoin and IPFS are both developed by Protocol Labs.

Token

The FIL token is a utility token used to purchase storage and retrieval services, to incentivize nodes to provide and expand storage resources, and to maintain network consensus.

Storage Technology & Consensus Mechanisms

While Filecoin borrows technologies from IPFS such as content addressing, CIDs and Merkle DAGs, Filecoin nodes do not join or participate in the public IPFS DHT. On Filecoin, each CID refers to 32GB storage sector, and public IPFS nodes cannot view the contents of a CID without the right decryption provided by the Filecoin protocol.

On Filecoin, below two algorithms do most of the heavy lifting:

- Proof-of-Replication (PoRep): miners are rewarded in Filecoin if they can prove they've received the cryptographically stored information from the client (step 4 above).
- Proof-of-Spacetime (PoSt): miners are rewarded and not slashed if they can prove that data in the client's contract has been kept unchanged over a period of time.

The PoRep algorithm allows for miners to proof that data has been replicated to their own unique dedicated storage. This allows for miners to prove that they are indeed storing the number of unique physical copies requested by clients. This is achieved by "sealing" every replica of data independently. This is achieved by requiring storage miners to store pseudo-random permutation of the stored data unique the miners public key. If a miner commits to storing a certain number of replicas, each replica will have its own unique pseudo-random permutation that makes it unique.

Once files are properly stored and sealed, the miner generates the Merkle root of the replica and a proof of sealing. When the miner is challenged regarding whether or not they are indeed storing the content, the miner receives a random challenge which determines a specific leaf of the Merkle tree of the unique replica, based on the Merkle tree root. The miner then generates a proof of knowledge using the Merkle path that leads through the Merkle root.

PoSt builds on this primitive and proves that the uniquely stored replica has been stored over a certain period of time. When a random challenge is received, the miner generated proofs of replication in sequence, using the output of a proof as an input of the next over a specific amount of iterations corresponding to the storage duration. The PoSt mechanism is used to audit miners to ensure they are storing the right data for the right amount of time.

To become a storage miner, miners must first pledge both a storage pledge and a consensus pledge as collateral. If miners misbehave either maliciously or negligently by not executing storage contract properly, their pledge is slashed and burned (removed from the total circulation of Filecoin) by the protocol. This acts both as a disincentive for malice, as well as providing deflationary pressure on the token. 75% of block rewards earned by miners vest linearly over 180 days while 25% are made immediately available to improve miner cash flow and profitability.

In Filecoin, data storage is managed through storage contracts that are settled on the Storage Market – *for more details please see the Storage Process & Pricing Mechanism section.*

Apart from storage miners, Filecoin has another class of miner: the retrieval miner. While storage miners are responsible for storing data, retrieval miners are responsible for retrieving data and serving it to clients. Retrieval miners are not required to pledge collateral to become a miner and thus many storage miners tend to also take on the role of retrieval miners. Retrieval miners can obtain data directly from clients, thus enabling third party storage services directly linked to certain retrieval and storage nodes, or through the Retrieval Market. On the retrieval market, retrieval miners are rewarded in FIL tokens for retrieving and serving data to clients.

The Filecoin network uses a public blockchain ledger to track all storage allocations of all network participants. Whenever a new block is mined, the network checks if the required proofs for all stored data is available and if it is valid. Under the circumstance that proofs are missing or invalid, the miner is penalized by having some of their collateral removed. From here, if a specific fault threshold is exceeded, the network settles the storage order as failed and attempts to retrieve the faulty data from other nodes in order place a new storage order to the Storage Market. If the network is unable to retrieve the faulty data (i.e., ever storage miner storing this piece is faulty), that means the data is lost. In this case, the client gets refunded.

Storage Process & Pricing Mechanism

When new data is to be stored on the Filecoin network, a storage user must connect to a storage provider through the Filecoin storage market and negotiate storage terms, before placing a storage order this is handled through Filecoin's Storage Market. In the Storage Market, users must initiate a deal negotiation to start the storage process:

1. **Discovery** - The client identifies miners and determines their current asks.
2. **Negotiation** - Both parties come to an agreement about the terms of the deal, each party commits funds to the deal and data is transferred from the client to the provider.
3. **Publishing** - The deal is published on chain, making the storage provider publicly accountable for the deal.
4. **Handoff** - Once the deal is published, it is handed off and handled by the Storage Mining Subsystem. The Storage Mining Subsystem will add the data corresponding to the deal to a

sector, seal the sector, and tell the Storage Market Actor that the deal is in a sector, thereby marking the deal as active.

During this process, the user decides which type of erasure encoding (EC) is to be used and the replication factor thereof. With EC, data is broken down into constant-size fragments, which are each expanded and encoded with redundant data, so that only a subset of the fragments are required to reconstruct the original file. The replication factor refers to how often the data should be replicated to more storage sectors of the storage miner. Once the storage miner and the user agree on the terms, the data is transmitted to a storage miner and is stored in a storage miner's storage sector.

If users want to further increase redundancy, they need to engage in additional storage deals with additional storage providers, as there still exists the risk that one storage miner goes offline and with it all of their pledged storage sectors. Applications such as NFT.Storage and Web3.Storage built by Filecoin on the Filecoin protocol solve this through storing files with multiple storage miners, however at a protocol level users must manually engage with multiple storage miners.

Tokenomics

- Baseline minting (770 million FIL): minted based on network performance relating to reaching storage capacity targets, incentivizing growing the network's storage capacity
- Simple minting (330 million FIL): minted independent on a 6-year half life based on time (97% minted in approximately 30 years)
- Mining reserve (300 million FIL): community controlled reserve tokens, with the purpose of incentivizing future types of mining
- Initial Coin Offering (200 million FIL): vesting over 3 years from protocol launch on October 15th, 2020
- Protocol Labs (300 million FIL) & Filecoin Foundation (100 million FIL): vesting over 6 years from protocol launch on October 15th, 2020
- Total slashed (30.5 million FIL) as of time of writing (*see https://filfox.info/en/address/f099*)

This gives us a (current) fully diluted circulation of FIL of roughly 1.97 billion tokens. The tokenomics are designed to incentivize storage and contract completion, while further incentivizing the expansion of total storage on the network.

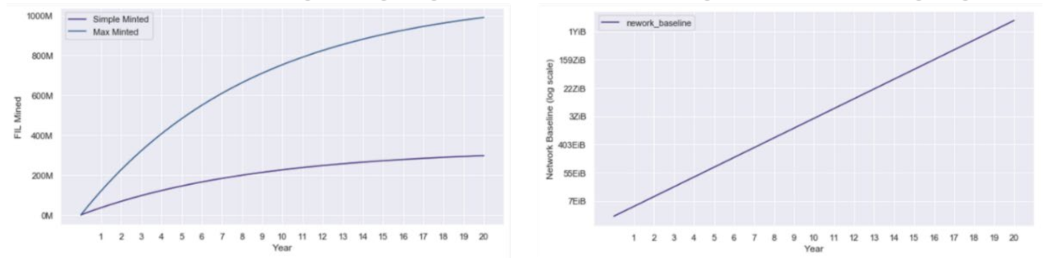


Figure 40: Max and Min Minting from Storage Mining and Max Baseline Minting. Source: https://filecoin.io/blog/filecoin-circulating-supply/

Crust Network

Crust Network aims to be the preferred storage network for Web3, acting as an interconnectivity and compatibility layer between blockchains and its own decentralized storage blockchain that interfaces with IPFS for file storage. Crust Network is built on the Substrate framework, primarily known from the Polkadot ecosystem.

Token

The CRU token is a utility token with the main functions of purchasing storage services and maintaining blockchain consensus through staking.

Storage Technology & Consensus Mechanisms

Crust Network is built on IPFS and participates in the public IPFS DHT. This means Crust Network IPFS nodes can be found by, and store CIDs and data of, non-Crust IPFS nodes. Crust further operates a network of IPFS public gateways to transmit and receive files.

IPFS itself, however, does not incentivize long-term storage of data nor does it incentivize the transmission or replication of data. Simply speaking, IPFS can be considered a data transfer and retrieval framework. This is where the Crust Network protocol comes in. The Crust Network protocol essentially manages and incentivizes two networks simultaneously that nodes can mine;

- the Crust Network blockchain (aka "*verification network*"),
- the *storage network*, which uses IPFS

Furthermore, Crust Network has two consensus mechanisms that run in parallel and require different proofs form the network:

- Meaningful Proof of Work (MPoW): provides technical assurance for the trusted execution of code in the Trusted Execution Environment (TEE)
- Guaranteed Proof of Stake (GPoS): manages storage orders and maintains blockchain consensus

MPoW is responsible for the generation of two proof types that require network consensus, namely environment proof and workload proof, and runs within a node's TEE. Simply put, the TEE is a hardware computing module on physical devices that is separated from the core processing units of the device. This module allows for verifiably untampered computing and can be used to retrieve information about the system or be used for complex calculations. The only limitation of joining the Crust Network therefore becomes whether the mining hardware supports TEE.

The environment proof ensures that new nodes that join the network have a valid TEE instance, by using a public key generated within the TEE and other nodes on the network executing a remote attestation procedure. This uses the public key and data produced by the new node and compares this against an expected result. If the results match, the new node passes the environment proof and the node identity and the TEE public key will be recorded on the Crust Network blockchain.

The workload proof deals with both the storage of new files, and also handles random data checks executed by the network. When a user stores a file on Crust, an operation within the TEE called the inspector will split the file into multiple pieces, which are each encrypted and hashed into a Merkle tree. As a result, in a node's external storage a node can only see encrypted file fragments and is not able to reconstruct the files stored on their own. When random data checks are initiated, the inspector algorithm within the TEE retrieves a random Merkle tree hash and relevant file fragment which is decrypted and re-hashed. Finally, the new hash is compared against the expected hash. The inspector not only checks file fragments stored on the nodes, but also the storage capacity of the node. Together these actions form a work report which is signed, verified by other nodes and recorded on the blockchain. If the file fragment cannot be retrieved or otherwise misbehaves, staked CRU is confiscated with a portion of the confiscated tokens being burned and the remainder being allocated as block rewards – *more details regarding staking in Guaranteed Proof of Stake explanation further below.*

While in this model Proof of Work calculations are still executed, only *meaningful* calculations based on changes to the node's storage state and validated through TEE are executed – hence the name Meaningful Proof of Work.

The workload proof also has a Proof of Running Tracking (PoRT) algorithm which enables the tracking of computational workloads. Although not activated on the Crust Mainnet at this point in time, this algorithm can enable decentralized computing.

The Guaranteed Proof of Stake (GPoS) consensus algorithm, which is essentially a proof of stake algorithm where nodes can only stake CRU tokens if they have provided storage capacity to the network, whereas staking capacity increases based on the total storage capacity provided to the network and proof that the nodes actively process storage orders from the network. In the network, MPoW generates environment and workload reports which are submitted to verifier nodes in the network as transactions, and using the GPoS consensus mechanism these verifier nodes then generate blocks and write them to the Crust Network blockchain. It should be noted here that by definition, verifier nodes must also provide storage resources (hence *Guaranteed* Proof of Stake).

Within the GPoS consensus mechanism, a further node type is distinguished: candidate nodes. These are verifier nodes that are not qualified for block generation due to insufficient staked CRU tokens. At the end of each block generation cycle, candidate nodes may become verifier nodes if the meet the required staking quota.

Finally, the last two participants in GPoS are guarantors and storage users. Guarantors can provide their CRU tokens to nodes on the network to help them increase their stake in order to reach the required staking quota to become a verifier node. By doing this, guarantors can earn guarantee income. Storage users are users on the network that pay in CRU to store files.

Below is a summary of the network actors and token flows:

- Verifier nodes:
    - Block packaging rewards (network token emissions, storage fees reward pool), enabled through sufficient token staking
    - Block cycle rewards (network token emissions)
    - Storage income (miner storage rewards)
- Candidate nodes:
    - Block cycle rewards (network token emissions)
    - Storage income (network token emissions)
- Guarantors:
    - Guarantee income (from verifier node block packaging rewards)
- Users:
    - Storage fees spending (users' storage fees spending split into miner storage rewards (20%) and staking reward pool (80%))

In summary, the MPoW algorithm ensures storage operations are validly and verifiably executed at a node level, and the GPoS algorithm is responsible for accepting and bundling storage work reports from nodes in the form of transactions and writing them to blocks. GPoS is hence the mechanism that brings together the various actors in the network.

On top of the MPoW and GPoS consensus mechanisms lives the Decentralized Storage Market (DSM), which is an algorithm that lives is executed on the nodes and is responsible for storage pricing, processing storage orders and file retrieval operations.

When a user wants to store a file on a node, the user is given a dynamically calculated storage price based on storage requirements and details of the data to be stored. The user then submits the storage request to the network with storage fees attached. The network accepts the files and transmits them to a node through IPFS, which saves the files locally using the TEE and MPoW algorithm. The storage proof is then generated (including proof of files stored and remaining storage capacity), verified by other nodes on the network and submitted to the GPoS algorithm. The GPoS algorithm locks the valid proofs to the blockchain and pays out relevant rewards to nodes that hold the files.

To ensure nodes do not only store files and are unwilling to transmit files when requested by nodes or users, a nodes ability to receive storage orders is linked to their willingness to transmit files. Crust implements the IPFS BitSwap credit system between nodes, in which nodes track the number of storage requests that were fulfilled by other nodes to them. Over time, nodes will either build credits (data shared) or build debt (data received). Selfish nodes that build debt with other nodes will eventually not receive any further files until they have replenished their credit. This essentially incentivizes nodes to always try to maintain a high credit, because only nodes with sufficient credit are awarded new storage orders. Also, before a retrieval request is actioned the node will check the Crust Network blockchain to see whether the file is indeed a valid file on the network. This ensures nodes' bandwidth isn't wasted. It should be noted here that only the four fastest merchants that can provide valid storage proofs are given storage rewards, further incentivizing fast node bandwidth.

When a file is retrieved on IPFS the file fragments are retrieved from a number of nodes to speed up the download process and are reconstructed on the node which issued the retrieval request. If a user requests a file through a gateway, the gateway reconstructs the file before passing it on to the user using the HTTP protocol (as described in the IFPS section).

Saving the real-time status of all files onto the Crust blockchain allows for the DSM to count all saved instances of files across the entire network, as well as retrieve their storage locations. As a result, the network is able to calculate the replication status of all files. Currently a mechanism to improve the replication status and hence the resilience of data on the network is being designed that will allow for dynamic replenishment of files on the network.

Summarizing the above; MPoW ensures file storage operations are valid, GPoS ensures at a network level actors are rewarded for their various roles in the network, and DSM acts as user/application interface and the coordinating mechanism between GPoS and MPoW.

Storage Process & Pricing Mechanism

In Crust Network, a storage user connects directly with Crust Network's Decentralized Storage Market (DSM), which dynamically calculates the network rate for storage based on file size, quantity and certain network metrics. On the DSM, the users directly settle storage fees based on their storage requirements and store files directly. The fees involved for storing on Crust include:

- Byte fee: fee levied on storage size and quantity of files
- On-chain state fee: fee for every storage order placed
- Basic fee: fee charged for the duration of the contract in 6-month increments

Formula:

$$Storage\ cost = (file\ size * byte\ fee) + (basic\ fee * storage\ duration) + state\ fee$$

Whereas storage duration is calculated in 6-month increments.

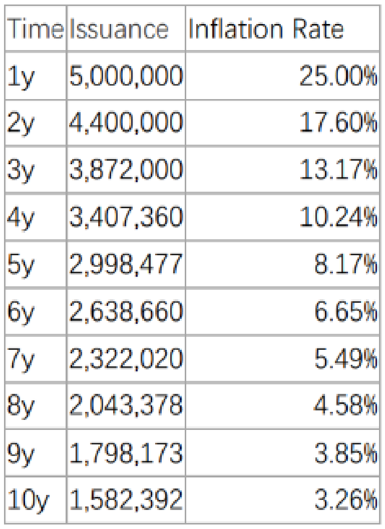
On Crust Network, the storage costs are based on network activity and denominated in CRU. If more files are stored, the price increases and if fewer files are stored the price decreases – similar to the relationship between Ethereum's base fee price for gas units. That means if the price of the CRU token changes over time, the USD cost for storage will increase, while generally it can be expected that storage fees remain more stable.

Tokenomics

Initial pool of 20 million CRU, split as follows:

- 5 million CRU for community development (25%)
- 2 million CRU for ecological growth (10%)
- 5 million CRU reserved for investors (25%)
- 4 million CRU for the technical team (20%)
- 4 million CRU foundation reservation (20%)

New tokens are issued on an annual basis with an annually reducing inflation coefficient. All newly emitted tokens are mining rewards for active nodes.



Time	Issuance	Inflation Rate
1y	5,000,000	25.00%
2y	4,400,000	17.60%
3y	3,872,000	13.17%
4y	3,407,360	10.24%
5y	2,998,477	8.17%
6y	2,638,660	6.65%
7y	2,322,020	5.49%
8y	2,043,378	4.58%
9y	1,798,173	3.85%
10y	1,582,392	3.26%

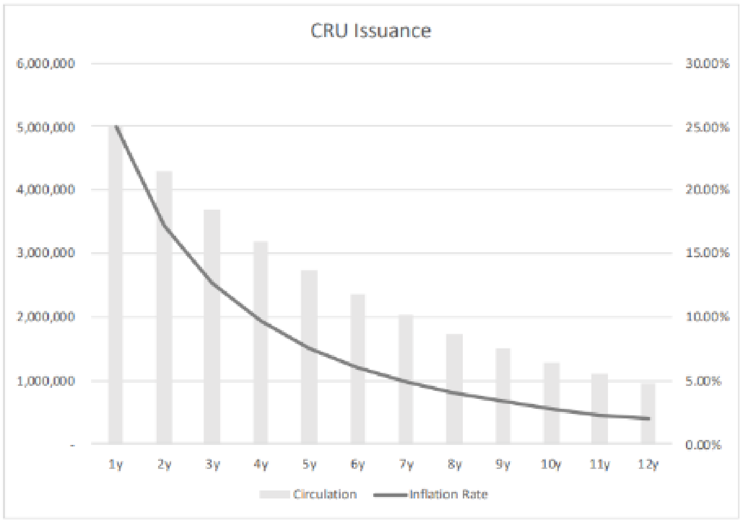


Figure 41: Crust Network token emissions. Source: Crust Economy Whitepaper (https://gw.crustapps.net/ipfs/QmRYJN6V5BzwnXp7A2Avcp5WXkgzyunQwqP3Es2Q789phF)

Pinata

Pinata is an IPFS pinning service and gateway provider. Through Pinatas API, users can directly store files on Pinata's own IPFS nodes. Users can also create dedicated gateways for storage and file retrieval. While Pinata operates on a the public IPFS network, they centrally control user accounts and billing. Pinata can be considered a centralized service built on a decentralized network.

Storage Technology

Pinata participates in the public IPFS DHT, meaning that anybody with the CID can access the content. Users can store files using an API key provided by Pinata upon sign-up, or directly using their online file manager.

Pricing Mechanism

Pinata utilizes a monthly pay-as-you-go model with three tiers of membership. The highest tier is a monthly flat-rate with bandwidth >100GB per month charged as you go.

Non-IPFS-based storage solutions

Apart from the three IPFS-based storage solutions mentioned above, many unique proprietary storage solutions have emerged over the years. In this section we will be taking a closer look at Arweave, Siacoin, Storj and Swarm.

Arweave

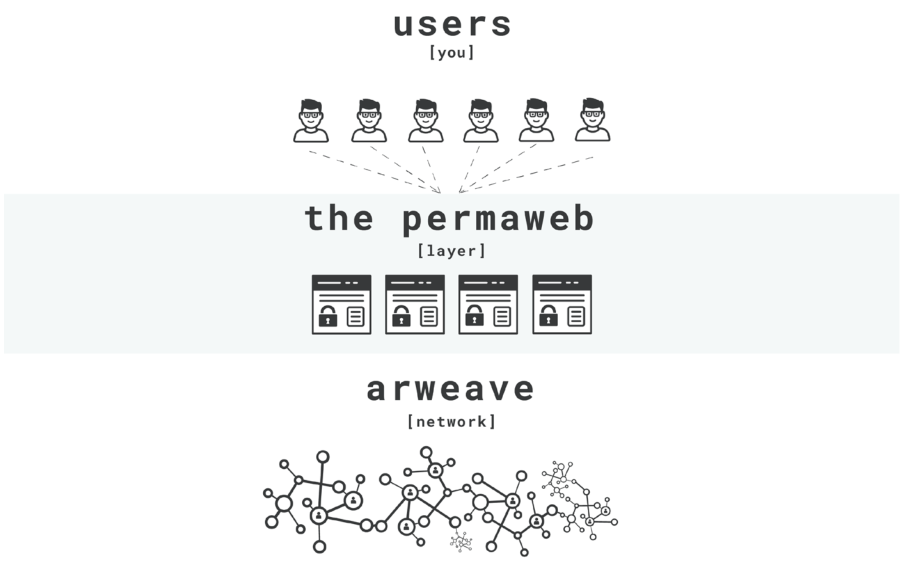


Figure 42: Relationship between Arweave and Permaweb. Source: https://www.arweave.org/technology#permaweb

Arweave is an open-source decentralized storage protocol that allows users to store data permanently with a single upfront fee. Arweave consists of two distinct parts: the blockweave and the permaweb.

- The blockweave is the blockchain-like storage layer of the Arweave network, where storage orders are processed and data is replicated.
- The permaweb is a human-readable layer built on top of the blockweave that is meant to mimic the world wide web as we know it – with the difference, that all websites and dApps once uploaded cannot be changed or altered. The permaweb, just like the world wide web, is accessible with a normal web browser using HTTP.

The Arweave protocol also supports smart contracts through its SmartWeave smart contract platform. Unlike smart contract-enabled blockchains such as Ethereum where smart contracts are computed on every node on the network, the state of smart contracts on Arweave are computed only on the local machines that request to run a smart contract.

Token

AR is the native utility token of the Arweave network, and is used to pay into storage endowments, which are paid out to miners to ensure that costs for storage and network bandwidth are covered indefinitely.

Storage Technology & Consensus Mechanism

The blockweave uses a consensus mechanism called Succint Random Proofs of Access (SPoRA), often simply referred to as "Proof-of-Access" (PoA). Whenever a node wants to accept new data, the miner must not only produce information about the previous block but must also provide cryptographic proof that they can access the "recall block" or "recall chunk", a randomly selected block of previously uploaded data, thus the name "Proof of Access". New data can only be uploaded to the system (and thus a new block mined) once the recall block has been verified. As a result of PoA, the blockweave is strictly speaking not a block *chain*, but instead resembles more of a graph structure, hence block *weave*.

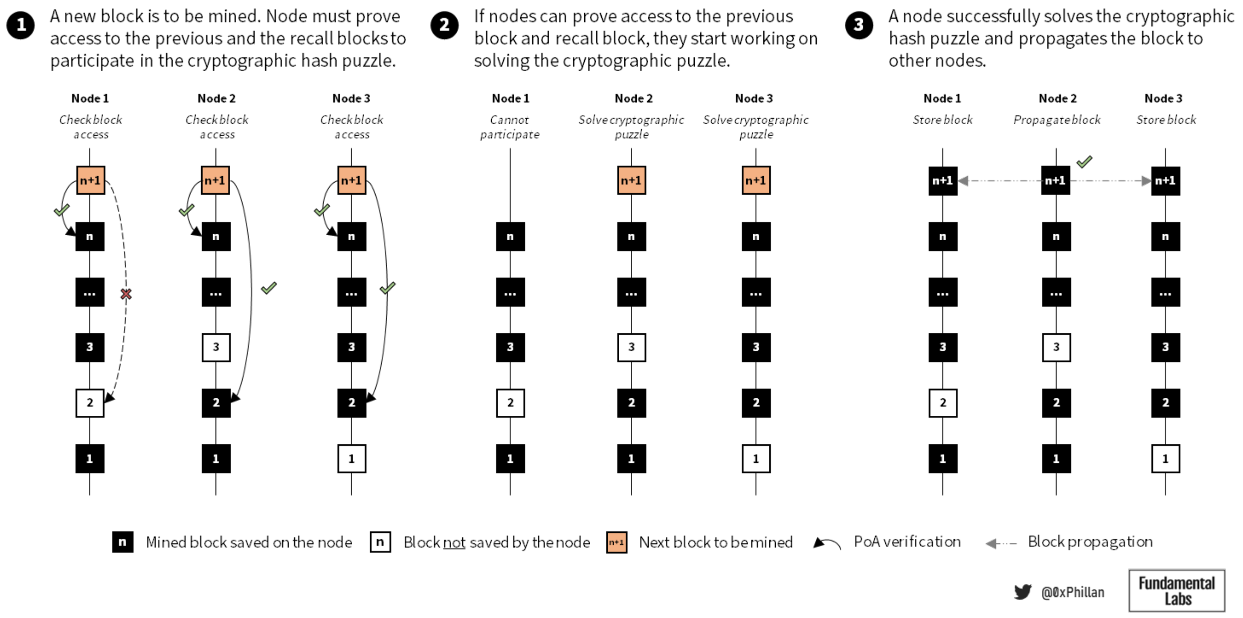

1 A new block is to be mined. Node must prove access to the previous and the recall blocks to participate in the cryptographic hash puzzle.
2 If nodes can prove access to the previous block and recall block, they start working on solving the cryptographic puzzle.
3 A node successfully solves the cryptographic hash puzzle and propagates the block to other nodes.
Node 1
Check block access
Node 2
Check block access
Node 3
Check block access
Node 1
Cannot participate
Node 2
Solve cryptographic puzzle
Node 3
Solve cryptographic puzzle
Node 1
Store block
Node 2
Propagate block
Node 3
Store block
n+1
n
...
3
2
1
Mined block saved on the node
Block not saved by the node
Next block to be mined
PoA verification
Block propagation
@0xPhillan
Fundamental Labs

Figure 43: Simplified illustration of PoA algorithm

In Arweave, the block data structure ensures that all data required to process new blocks and new transactions included into each individual block. As a result, it is not necessary for new miners who join the network to download the entire history of all blocks. When a new miner joins, it requests a Block Hash List and a Wallet list from trusted peers, which allow old blocks to be requested and verified and for transactions to be verified, even without possessing the block in which the last wallet was verified.

This leads to another challenge: apart from being able to request blocks from other nodes, old miners must be willing to spend their network and computational resources to deliver that block to new miners. Arweave solves this with the "wildfire" mechanic. In this mechanic nodes rank their peers based on the peer's generosity (sending new transactions) and the peer's responsiveness (responding to requests for information). Nodes then gossip (i.e. communicate) preferentially to higher ranked peers. This incentivizes nodes to actively communicate with the network, as active communication is rewarded with a higher likelihood of being able to mine the next block.

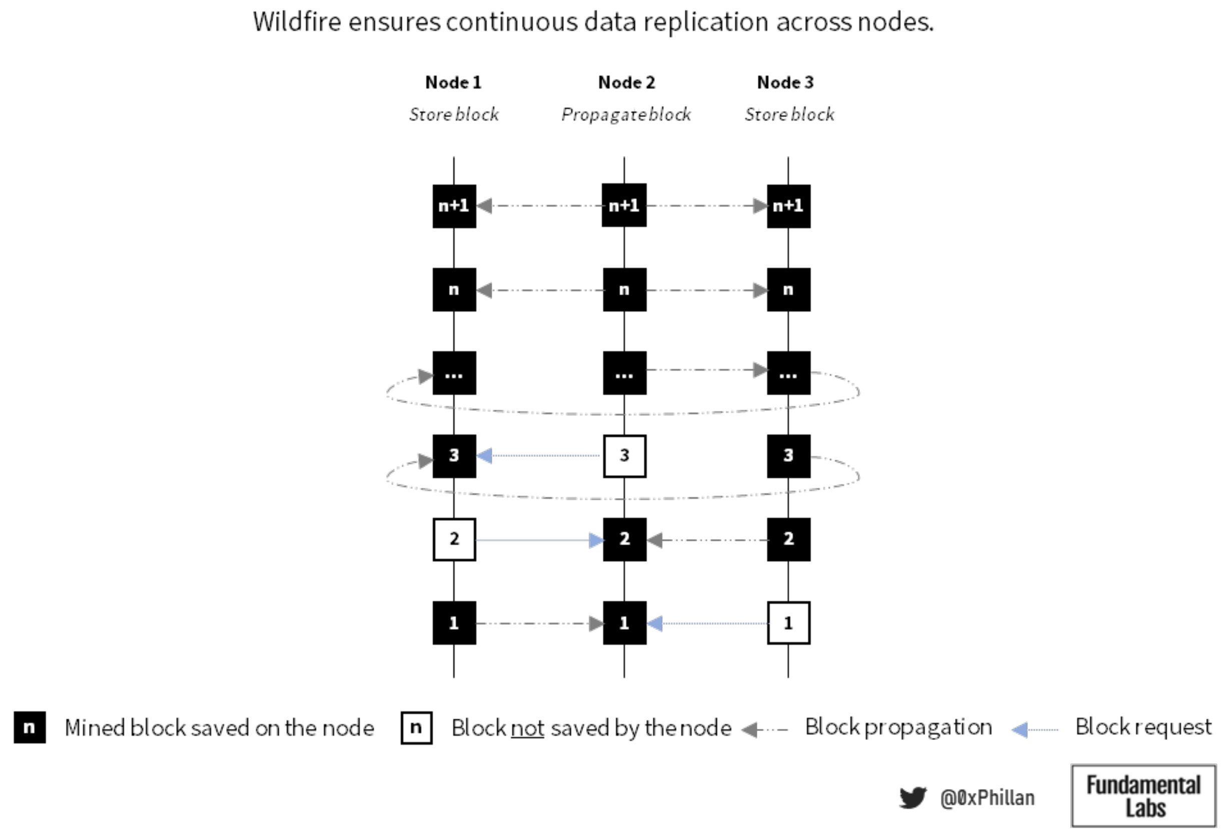


Figure 44: Simplified illustration of Wildfire mechanic creating the weave-like structure of the blockweave

Unlike Filecoin, where the consensus mechanism incentivizes expansion of storage resources, the PoA consensus mechanism incentivizes long-term data storage and redundancy maximization through the storing of 'rare' blocks, as miners would compete with fewer other miners in the Proof-of-Work puzzle for the same level of block reward.

The PoA consensus algorithm is essentially an extension of the PoW consensus algorithm, as when a node has the cryptographic proof required to mine recall block, they must still compete the smaller group of miners to solve a cryptographic puzzle. Once the cryptographic puzzle is solved and the data is added to the network, miners are rewarded with AR tokens.

The Arweave protocol monitors and moderates content that is published to Arweave, to ensure that certain materials that node operators do not agree with can be filtered out. While there are some automatic checks in place, every node can decide for itself which data to download and which to ignore. Hence when a user uploads files to the network, they can pick a gateway that adheres to a content policy that fits their needs. *For more information about Arweave's content policy, please consult the Arweave yellow paper.*

Data Permanence & Pricing Mechanism

When users wish to store files on Arweave, they most pay a one-time upfront fee. Part of this fee is paid to a miner to add the data to the network, but a majority of the fee is contributed towards a storage endowment, which using Arweave's assumptions should covers the cost of storing the data indefinitely. In their calculations, over the last 50 years the average annual rate of decline of the cost of storing 1GB per hour has been -30.57%.

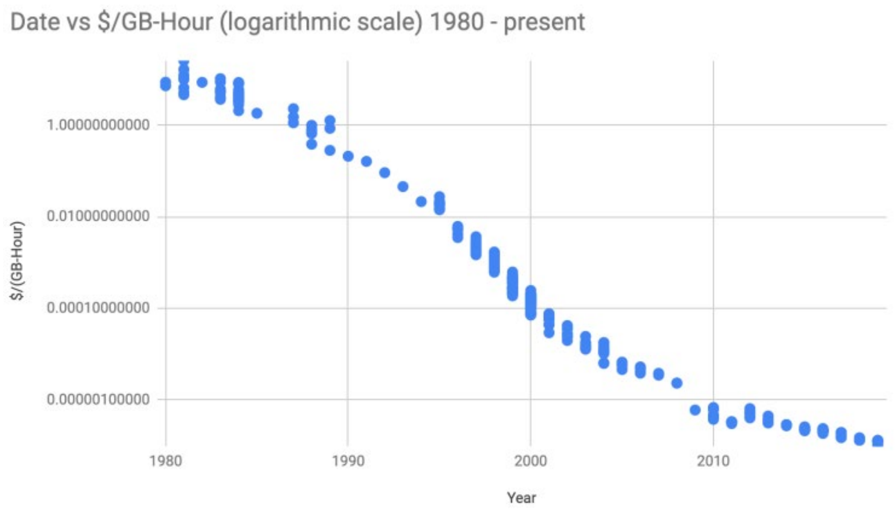


Figure 45: decline of storage costs for 1GBh. Source: Arweave Yellow Paper

Arweave applies three key assumptions to enable data to be stored permanently (or more precisely, as long as the Arweave network exists):

1. The cost a user pays to upload and store data to the Arweave network covers the first 200 years of storage, assuming a -0.5% per year decline in storage costs, which appears very conservative against the -30.57% average annual decline of the past 50 years. If the decline in cost per GBh is greater than -0.5% per year, the total number of years that the data will be stored is increased accordingly.
2. Tokens in the endowment are expected to grow in value in fiat terms over time, acting as an additional long-term stabilizing factor.
3. The Arweave protocol actively avoids releasing tokens from the endowment: if miners have already surpassed profitability through tokens emitted from inflationary block rewards and the transaction fees received when new data is added to the blockchain, payouts from the endowment do not take effect. This would lead to a 'float' of tokens in the endowment, further extending the length of time that data is stored on the network.

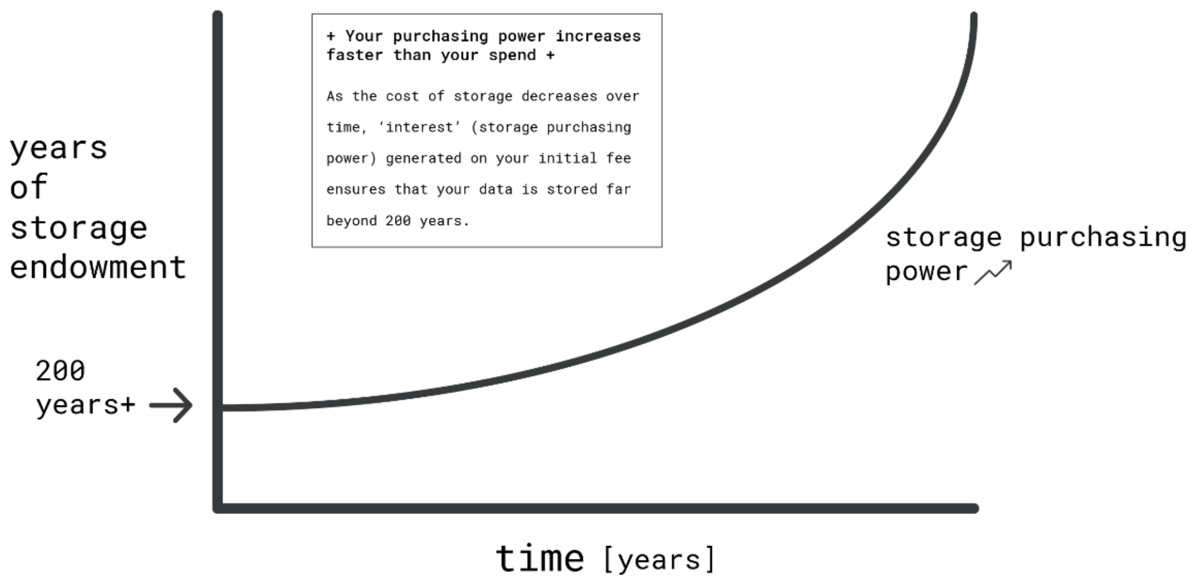


Figure 46: visualization of assumptions driving data permanence Source: https://www.arweave.org/technology#permaweb

This means that even if nodes leave the network over time, new nodes will join to continue storing the data for the economic incentives the system offers – as long as storage revenue is greater than the cost of storage.

There are 3 components that determine the cost of storing data on Arweave:

$$StorageCost(D) = Size(D) * FiatCost_{PerpetualStorage} * Price_{USD}$$

Where,

• Size(D)	= Size of the data that is being stored
• $\text{FiatCost}_{\text{PerpetualStorage}}$	= Fiat cost of perpetually storing 1 GB of data today
• $\text{Price}_{\text{USD}}$	= Price of AR (Arweave's native token) in fiat terms

Mining Rewards

When a miner successfully solves the PoA hashing puzzle, assuming they have access to the recall block, mining rewards are split into three parts:

- Transaction fees
- Inflationary token emissions (pre-defined for every block, gradually decreasing at a rate dependent on the block height)
- Endowment payments

The mining reward per block is outside of the control of the miner. The equation for a miner to maximize their mining revenue consists of:

- The probability of being able to participate in the mining of a new block is a function of the probability that the miner holds the data requested to be validated in the recall block. To maximize this likelihood, nodes are incentivized to store as many blocks as possible.
- The probability of a node being able to be the first to solve the cryptographic puzzle, which is a function of the nodes hashing power in comparison to the network's average hashing power.
- The historical active participation in the wildfire sub-mechanic, which increases how soon the node can get information on the next block.

Tokenomics

- 55 million AR generated at genesis (8th June, 2018)
- 11 million AR as inflationary emissions introduced as block rewards, which are halved continuously until all tokens are in circulation

Fully diluted token circulation will be 66 million AR tokens. Arweave does not deploy token burning mechanisms.

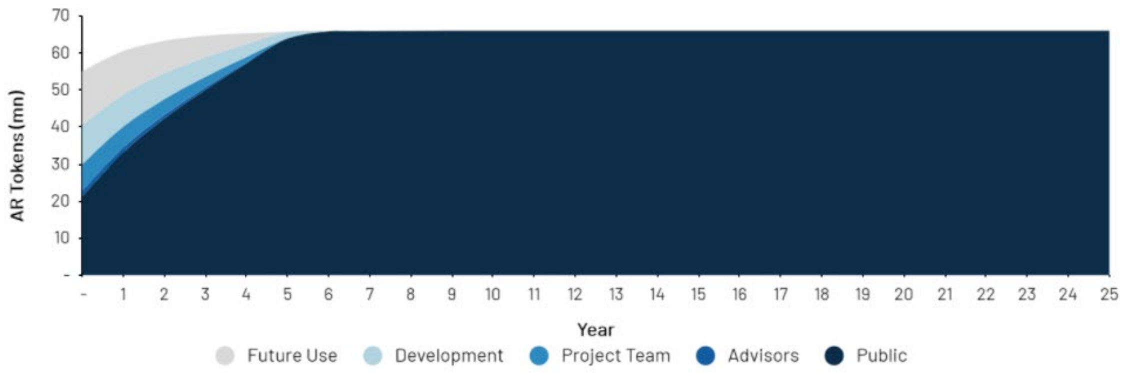


Figure 47: AR token inflation and team allocation. Source: https://medium.com/amber-group/arweave-enabling-the-permaweb-870ade28998b

Sia

Sia is a storage blockchain with the primary aim to bring back data ownership to the individual and prevent data censorship by any actors. Unlike IPFS-based filed storage and Arweave where entire files are replicated between nodes, Sia splits up files that are uploaded, encrypts the data fragments, and replicates those fragments across its network.

In the Sia network, file storage users are referred to as “renters”, because they essentially rent storage space on the network. The “node” is your specific installation of Sia, and a “host” is a storage space provider on the Sia network that earns Siacoin for providing their storage.

Sia, similar to Filecoin, works with contracts between renters and hosts that determine how much data is stored for how long and at which price. These storage contracts that define the terms of arrangement between renters and hosts are stored on Sia’s native blockchain, the Sia. The data to be saved (defined in the contract) is then stored on hosts’ machines. When a contract is formed, the host commits to submitting proofs that the files agreed upon in the contract are indeed stored on their machine.

Sia uses both incentivization and slashing to ensure files are stored for the duration of a contract: regularly submitting file storage proofs is rewarded with Siacoins and missing to submit a proof is penalized.

Architecturally speaking, Sia can be considered a hard fork of the Bitcoin network protocol, with adjustments made to transactions specifically to enable storage: the Bitcoin Script transaction language was entirely removed and replaced only with timelock-enabled M-of-N multi-signature transactions. The functionality of this transaction type is also extended in Sia with contracts, proofs, and contract updates. All this together enables the previously mentioned storage contract functionality in what is essentially a Proof-of-Work storage blockchain.

Token

The Sia Blockchain has two tokens:

- Siacoin: the native utility token of the Sia blockchain used for storage contracts
- Siafunds: a revenue-sharing token on the network that gives holders Siacoins for every completed contract on the network

Storage Technology & Consensus Mechanism

The primary driver behind Sia’s storage mechanism is the storage contract. The storage contract specifies all aspects of the agreement between the renter and the host. Furthermore, the storage contract includes a Merkle tree hash that is created by generating 30 constant sized segments from the file which are then hashed into a Merkle tree using the Threefish algorithm and each file segment is hashed using a different encryption key. This enables the verification of storage proofs and ensures the files are not tampered with. Each piece is then replicated on several different hosts, enabling privacy and increased censorship-resistance: no single host can reconstruct the file with their piece alone, and taking down a single piece does not corrupt the file. In fact, Sia uses Reed-Solomon erasure coding to generate the 30-piece segments, which ensures redundancy: only 10 of the 30 segments are required to fully recover a file, meaning even if a majority of the segments are lost, the files can still be recovered. Furthermore, if a host does go offline, the file is automatically replicated to a new host.

Apart from what files should be saved for how long at which price and the file's Merkle tree hash, the storage contract also specifies:

- Challenge frequency: specifies frequency of periodic valid proof submissions by host
- Payout parameters: reward rules for valid proofs, invalid or missing proof, and the maximum number of proofs that can be missed

When a challenge is to be submitted, there is a timeframe during which the host can submit the valid proof to the network called a challenge window. If a valid proof is submitted, the host is rewarded a certain amount of the funds pre-paid into the contract. This is Sia's **"Proof of Storage"** algorithm. If a host does not submit the proof, those funds are instead sent to a "missed proof" address, which is essentially a burn address. If too many proofs are missed, the contract "unsuccessfully terminates" and the remaining coins in the storage contract are sent to the missed proof address. The contract also unsuccessfully terminates if the funds in the storage contract are exhausted before the end of the contract duration – *more on this in the storage process & pricing mechanism section.*

If a storage contract is successfully concluded, meaning the contract was sufficiently funded and the host successfully submitted periodical proofs over the duration of the contract, then the contract is successfully terminated, and remaining coins sent to the host.

No matter if a proof is validly submitted or missed, or the contract successfully or unsuccessfully concludes, every action or lack thereof creates an output ID similar to a UTXO in Bitcoin. And just like Bitcoin, every transaction input must be a prior transaction output, meaning Sia can be considered a UTXO-based blockchain.

The storage proofs, which are to be submitted periodically by the host to the network, are created based on a randomly selected portion of the originally stored file and a list of hashes from the file's Merkle tree. These are then validated against the Merkle tree hash previously submitted when the storage contract was locked to prove that the host is indeed storing the file, and no alterations were made to the file.

Sia ensures hosts are incentivized to stay online and to complete contracts instead of intentionally dropping ongoing contracts for more lucrative contracts through putting up collateral up-front and through a host scoring mechanism. If a host goes offline or doesn't keep persistently keep renter data, they lose their collateral, and their host score is impacted. The host score is comprised of:

- Host uptime: longer uptime scores higher (below 95% induces collateral slashing)
- Competitive pricing: lower prices for storage and bandwidth score higher
- Collateral: equilibrium scores higher (impacts storage pricing: low collateral means host has nothing to lose, high collateral leads to storage price increases for renters)
- Available storage space: more remaining storage scores higher
- Host age: older scores higher
- Interaction weight: being open to storage orders scores higher
- Version adjustment: newer version of the Sia client scores higher

Furthermore, later in the lifecycle of the Sia project a Proof of Burn mechanic was introduced to prevent Sybil attacks. A Sybil attack is when an attacker creates a large number of identities in an identity system to become the majority in order to dictate what the system deems is true. In Sia this could mean creating a majority of new hosts to dictate which transactions should be included in a block. Hosts are expected to burn around 0.5-2.5% of their revenue to prove they are legitimate nodes on the network. This makes creating new identities extremely expensive and further instructs the aforementioned reputation system.

Finally, Sia relies on a Proof of Work consensus algorithm (specifically the Blake2b algorithm) to mine new blocks on the blockchain. While Proof of Storage creates transactions that validate storage proofs and are broadcast to the network, the Proof of Work algorithm bundles these transactions into blocks and adds them to the blockchain. As a result, the hashed Merkle tree of each transaction becomes publicly visible on the blockchain, meaning any host that holds a file segment can validate it against the public ledger storage contract entry.

Sia is also capable of allowing edits to uploaded files, however the precise mechanism thereof is beyond the scope of this research.

Storage Process & Pricing Mechanism

On the Sia network, an automatic storage marketplace is used to find a host to store data. The cost for storage, quoted and settled in Sia's native Siacoin token, is determined by supply and demand by hosts and renters: "If hosts find they can lower prices and win more data to store as a result, they'll do it. If renters are willing to pay more to store on high-quality hosts, those hosts might raise their prices." (https://docs.sia.tech/)

On Sia, you do not specify the duration of which you want to store your files. Instead, storage contracts are always three months long, and are automatically extended when one month remains, assuming the contract isn't cancelled by the renter (e.g., due to insufficient funds) or the files' health drops below minimum redundancy (hosts stop hosting the file).

The mechanism used to store files longer than three months is by setting an allowance that automatically refills over time from your wallet balance. This allowance is used to pay for storage contract extensions. Any funds that were filled into the allowance, but not allocated to storage contracts through transactions, are returned to the renter. The allowance refill mechanism is separate to the contract renewal mechanism and starts about halfway into a contract period.

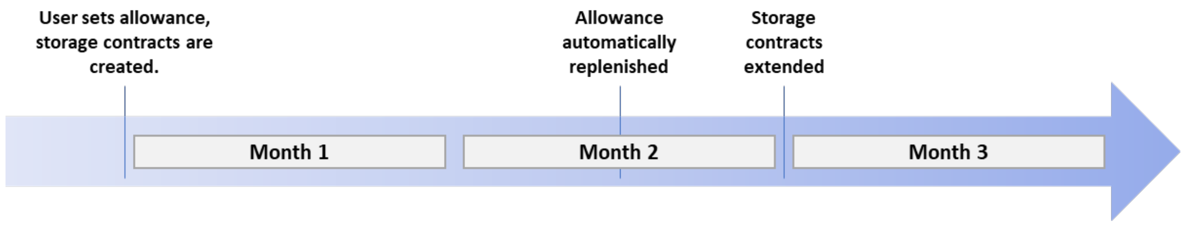

User sets allowance, storage contracts are created.
Allowance automatically replenished
Storage contracts extended
Month 1
Month 2
Month 3

Figure 48: Visualization of allowance and contract extensions mechanism timing

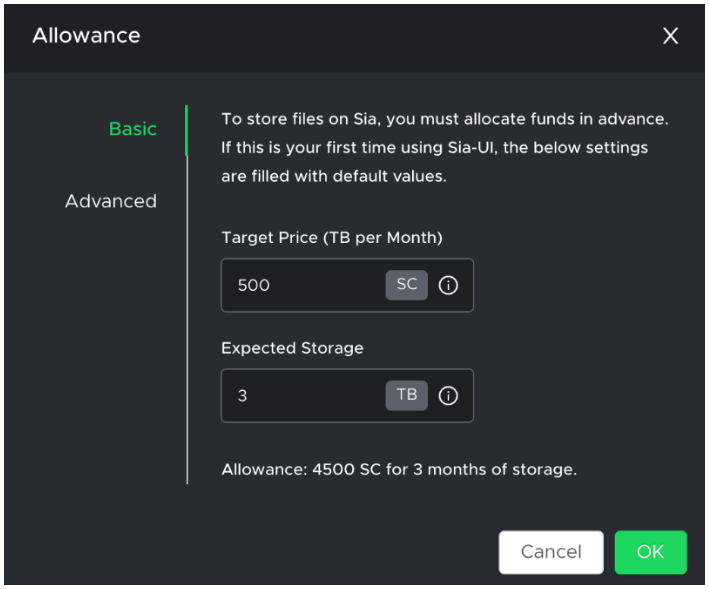
However, users must regularly open the Sia application to ensure the allowance is replenished – this is not executed autonomously through smart contracts but is a process that the Sia software executes when it is manually launched on a renter's machine. As a result of these mechanics, the renter only needs to set the target price and the expected total storage size to be used. The renter is then automatically connected to suitable hosts, which have all storage and bandwidth related costs set as part of their storage contract pricing setup.

The formula for storage allowance calculation is:

$$Allowance = TargetPrice_{TB/month} * StorageSize_{TB} * ContractDuration$$

Whereas;

- TargetPrice is the storage price per TB per month denoted in Siacoins
- StorageSize refers to the expected total used storage size in TB
- ContractDuration refers to the 3-month contract cycles in Sia



To store files on Sia, you must allocate funds in advance. If this is your first time using Sia-UI, the below settings are filled with default values.

Target Price (TB per Month): 500 SC

Expected Storage: 3 TB

Allowance: 4500 SC for 3 months of storage.

Figure 49: Renter storage contract settings; Source: https://docs.sia.tech/renting/how-to-rent-storage-on-sia

As users transmit, store, and retrieve data, they pay three different fees to the hosts from their allowance:

- Contract formation fees: transaction fee for creating a storage contract on the blockchain
- Storage fees: cost of renting storage space
- Bandwidth fees: bandwidth used for uploading or downloading data and for repair-related bandwidth

The replication of a split file across multiple hosts is called "health" and renters must regularly open the Sia client to refresh the health of their files. If a file's health is 100% it means that all pieces of the file are available on the Sia network. This manual action is a result of Sia's encryption approach: because nobody but the renter can construct the full file with each segment's encryption key, only the renter can verify with the network whether the file's replication status is healthy.

Sia recommends opening the client at least once a month and have the client run over night to ensure housekeeping items such as allowance refill, storage contract extensions and file replication can run in the background.

Skynet & Skylink

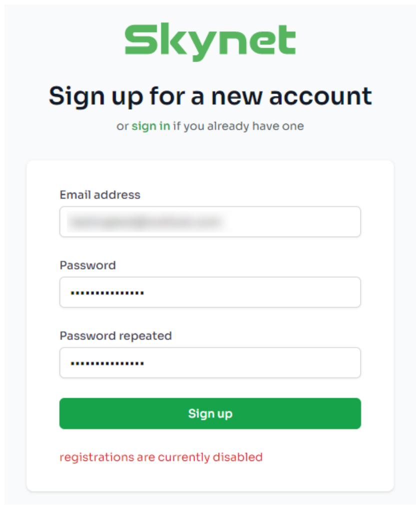
Skynet (https://siasky.net/) is Sia's implementation of a content delivery network, and closely resembles CIDs in IPFS or Arweave permaweb links: uploading data to Skynet generates a unique identifier, known as a Skylink. Just like with IPFS CIDs, a Skylink is a cryptographic hash generated from the file contents, and thus closely resemble CIDs in look:

```
https://siasky.net/CADIvje1Fdy2FP2TeBsYAbHfUsNug98wE7SYArdyczDaDg
```

Skynet uses "Portals" to let users access files through Skylinks. These closely resemble gateways in IPFS in that they provide an access point to the data on the Sia network without having to download the Sia client. Similar to IPNS, Skynet also has their own decentralized domain name service called Handshake Names, which all Portals support.

In the previous section we mention that on the Sia network a client must open their client at least once a month to ensure that housekeeping operations are completed. On Skynet, these housekeeping actions are automatically completed by Portals, and they also pay all host-related fees a renter would cover if they were to directly rent space on Sia. Finally, on Skynet Portals can decide their own pinning policies, which determine how long a file will be available on the network.

Since the network and underlying data is secured through the Sia blockchain, and pricing mechanics of storing on the Sia blockchain controlled through the automatic storage marketplace, Skynet adds another financial layer on top. According to Skynet docs, Portals operate a freemium model where basic access to the network and saving small files is subsidized by Portals, however, to unlock faster network speeds more storage space, users must pay a premium. Unfortunately, we were not able to test this feature as sign-ups for Skynet were closed as of time of writing.


Skynet

Sign up for a new account

or sign in if you already have one

Email address

Password

Password repeated

Sign up

registrations are currently disabled

Figure 50: Skynet registrations disabled.

Tokenomics

There is no maximum supply of Siacoins and the supply of Siacoins is perpetually inflating at a rate of 30,000 Siacoins per block. Initially the blocks emitted were (300,000 – blockheight) blocks, however after 270,000 blocks the block reward is locked at a minimum issuance of 30,000 per block. As of time of writing, the Sia blockchain has reached over 366,000 blocks with a total circulation of 47.4 billion Siacoins. The Sia blockchain has no pre-minted blocks, meaning that when the chain was launched circulation of Sia was 0.

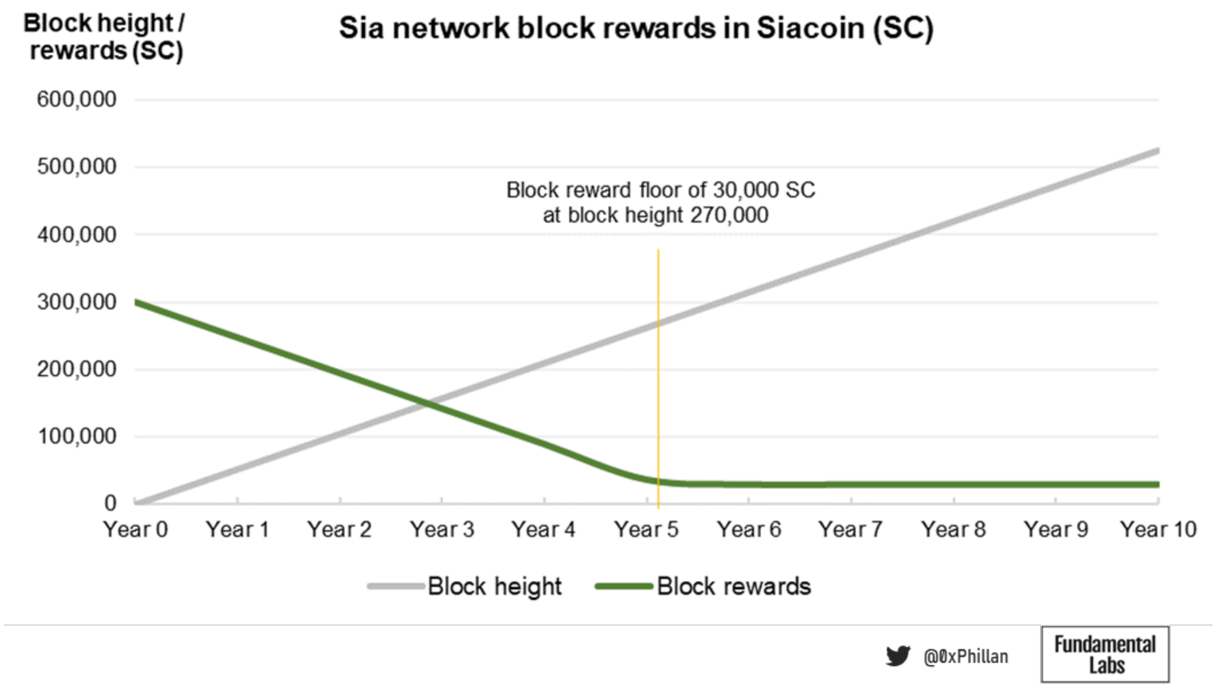


Figure 51: Sia network block rewards vs block height over time.

Apart from the block reward schedule above, Sia introduced a hardfork in its 6th year of operation to include a block subsidy and minted additional Siacoins, to fund the Sia Foundation, a non-profit entity meant to support, develop and promote the Sia network. The block subsidy is 30,000 SC per block paid out every 4,380 blocks (roughly a month at an average block time of 10 minutes) which equates to an annual subsidy of around 1.57 billion SC (roughly $8 million USD at token value at time of writing). The initial Sia Foundation subsidy was also approximately 1.57 billion SC, but was created all at once instead of being created overtime with block rewards.

https://docs.sia.tech/get-started-with-sia/siacoin-total-supply

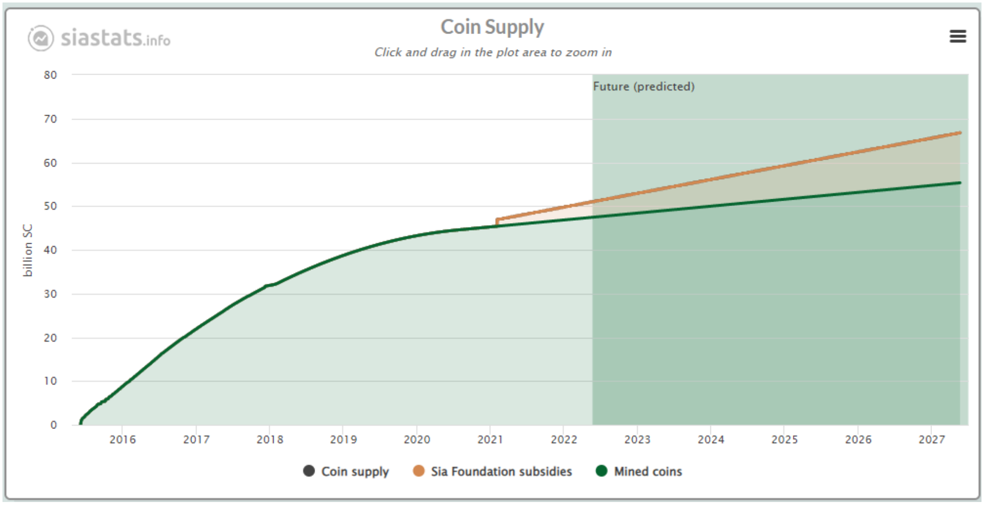


Figure 52: Annual growth of Siacoin supply and Foundation coin minting. Source: https://siastats.info/macroeconomics

The amount of coins burned in Sia, even after the Proof of Burn mechanism was activated (which appears to have been activated in early 2020 given the sudden increases in burn; no official sources were found on the activation date), is far below the coin inflation at only roughly 500k SC burned per year vs roughly 3.14 billion SC minted annually (1.57 billion block rewards and 1.57 billion Sia Foundation subsidy).

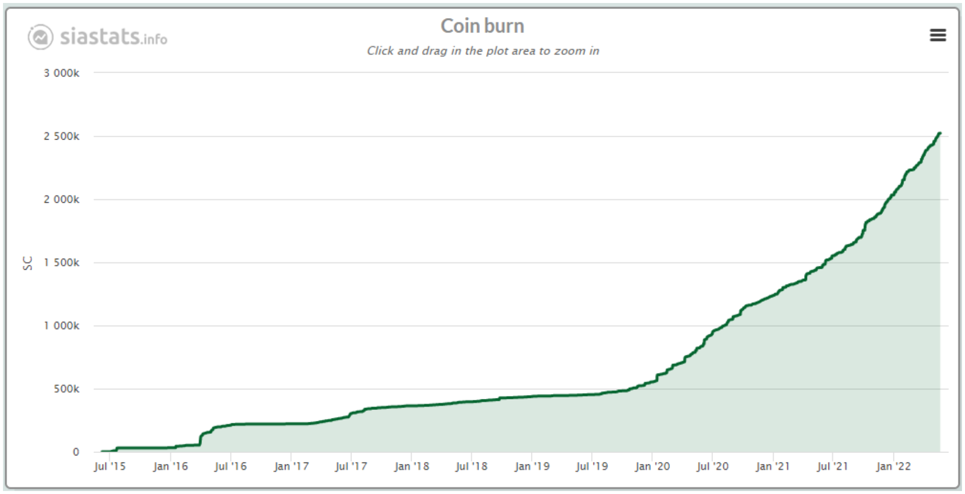


Figure 53: Siacoin burn mid-2015 to early 2022. Source: https://siastats.info/macroeconomics

Sia has a second cryptocurrency which are called Siafunds (SF). In total 10,000 SF were created and were entirely pre-mined. Siafunds allow for holders to share 3.9% of the revenue from successful storage contracts and were created for the purpose to ensure long-term funding for the development of the project. According to the Sia wiki, Nebulous holds 8576 SF, with the remainder being in public circulation as a result of a crowdfunding campaign to fund early development of the project.

Storj V3

Stroj is a decentralized file storage and content delivery network that aims to replace Amazon Web Services (AWS) S3. The Storj protocol implements a peer-to-peer storage system that encrypts, shards and distributes data to nodes around the world, bearing some similarities to Sia, all while avoiding the use of a blockchain. This design decision is meant to foster greater scalability: in a system where actions are meant to have only milliseconds of latency, the time overhead of waiting for a blockchain to reach consensus makes a blockchain an unsuitable mechanism for a decentralized storage provider with the ambition to replace AWS both in scale and performance.

Storj is currently on its third major iteration, hence “V3”.

Token

The Storj network is uses the Storj token on the Ethereum network as the default payment mechanism for storage and bandwidth payments. While the Storj token is the default, the network is designed in a way to allow for other payment mechanisms to be adopted in the future.

Storage Technology & Storage Mechanism

The Storj network differentiates actors in the network based on three peer classes:

- Storage nodes (object storage servers): provide storage space and bandwidth, expected to remain online at all times
- Uplink (clients): application or service that wants to store or retrieve data, not expected to remain online
- Satellite (metadata servers): caches node address information, stores per-object metadata, maintains storage node reputation, manages billing and payments, verifies file integrity, and reconstructs files, and manages authorization

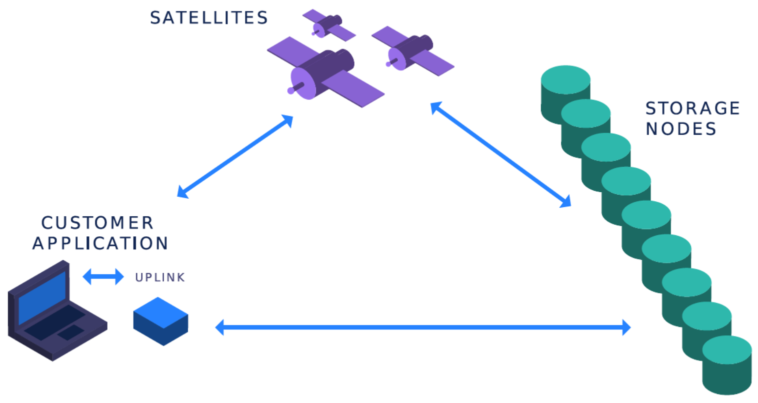


Figure 54: The three peer classes. Source: Storj V3 Whitepaper

These three peer classes form a symbiotic relationship in that they all rely on each other for this system to work in a trustless manner. Unlike other systems where consensus must be achieved to mine a block to a blockchain that runs validations, the peer classes operate independently and can form clusters within which file storage and transfer operations are executed in what is essentially a decentralized reputation-based file storage system.

The main actions executed by actors on the Storj network include:

- Node identity creation
- Data storage
- Data transfer (inbound & outbound)
- Audits (validation of integrity of stored data)
- Data repair (reconstructing files with poor integrity)
- Authorization management
- Reputation database management
- Payment and billing

At a high level, storage users pay satellites to coordinate storage with storage nodes. To connect with a satellite, users use a customer application or uplink, that facilitates communication with a satellite. Once nodes have been selected by the satellite, the uplink connects directly to the storage node to store files. The files are split into equally sized segments using the

In this protocol satellites are the coordinators of the system and manage various administrative overheads, including maintaining an up-to-date database of where file segments have been stored, launching audits and coordination data repair efforts. Satellites are also the billing and payment centers of Storj, as they track how much data has been retrieved from storage nodes, how much data storage nodes have saved at any given moment in time, and which repair efforts a storage node was involved in. Satellites then pay storage nodes on a regular basis using the Storj ERC20 token that lives on the Ethereum blockchain. The Storj token is in this sense purely a utility token meant to pay for storage transactions in the network.

Storage nodes rent their hard drive space and provide bandwidth to allow for uplinks to send data to store or retrieve stored data from storage nodes, the connections of which are coordinated by satellites. When an uplink receives data to upload, it splits the data using Reed-Solomon erasure codes, similarly to Sia, which creates 80 constant-size file segments each encrypted using a different encryption key. The satellite tells the uplink which storage nodes to connect to and signs that message. Finally, the uplink then connects to storage nodes providing the signed message from the satellite (called a bandwidth allocation) to transfer the individual file segments to the storage node.

When a file is transmitted, it is not transmitted at once, instead the segment and bandwidth allocation is broken down into smaller pieces, which are transmitted and validated one by one. Bandwidth allocations are stored by storage nodes to claim bandwidth payments from satellites. Splitting up a transmission like this ensures that a storage node cannot go offline to avoid receiving and storing a full file, and still claim payment for the full bandwidth allocation.

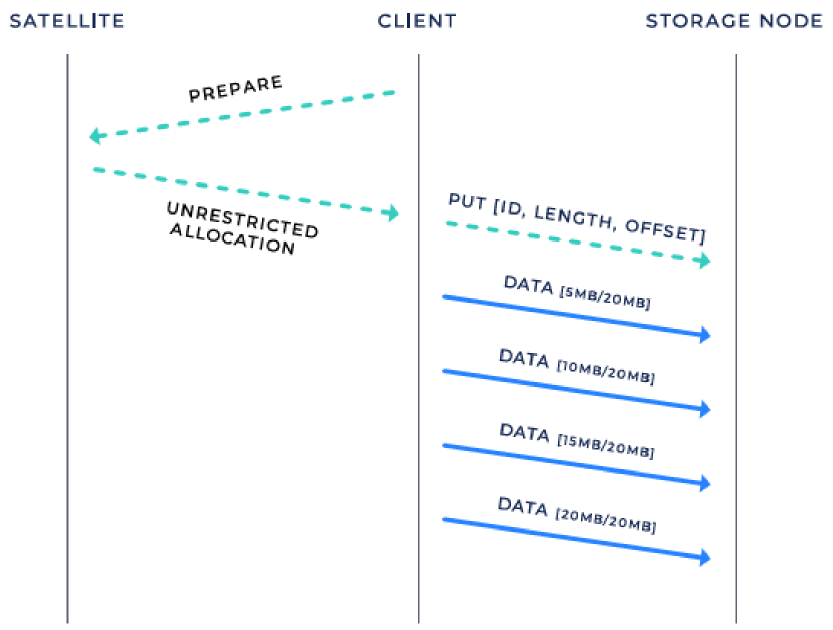


Figure 55: Diagram of a put operation. Source: Storj V3 Whitepaper

It's important to note that initial inbound transmissions of data for storage are not paid for on the Storj network. Instead, storage nodes are paid for all other uses of their bandwidth (e.g., data retrieval, repair work) and for the storage space used.

In this system there is no single source of truth, such as a blockchain, which stores and manages all storage orders and token transactions. Instead, every satellite and storage node maintain databases that hold metadata about network participants they've interacted with. While satellites also hold metadata on file locations (i.e., server addresses), both satellites and storage nodes hold data relating to the performance of their counterparts. These databases form Storj's audit, data repair and reputation systems.

Audits are executed regularly by satellites and ensure data availability on storage nodes. A satellite sends a challenge to a storage node requesting proof that the node has indeed stored the data it is expected to have stored. Audits first choose a "stripe" – a subset of a segment – and then run an algorithm across all erasure shares stored across storage nodes to identify faulty data. When sufficient storage nodes return correct information (which they are incentivized to do), any incorrect or missing responses can be identified. Audit results feed into Storj's reputation system – *more on that later*. Storage nodes that fail these audits are eventually removed from a satellite's storage node database and can lose funds held in escrow. Furthermore, these storage nodes may also receive limited to no future payments, further incentivizing to adhere to the system.

When a node goes offline, taking with it pieces all the segments stored on it's node. If the segments fall below a certain safety threshold (set by uplinks as desired durability), the satellite marks the pieces as missing and starts the data repair process. Since the satellite holds the data locations in a local database, it can reverse-lookup other storage locations of these segments. These are downloaded by the satellite, reconstructed, and the missing pieces will be regenerated and uploaded to new nodes. To assess whether the repair was successful, a validation hash is stored in the satellite's database, and compared against a piece hash retrieved from the storage node after storage is complete.

Storj's storage node reputation system consists of four parts:

- Proof of Work (PoW) identity system: to enter the network and communicate with satellites, storage nodes must prove they are invested by solving a PoW puzzle. The difficulty of the puzzle is set arbitrarily by satellites and the system is expected to self-balance over time.
- Vetting process: unvetted storage nodes are slowly added as additional storage targets for storage requests. They are selected as additional nodes on top of a satellites existing preferred storage nodes so as not to affect network integrity, but also to allow for data collection about the node.
- Filtering system: nodes that fail audits, fail to return data, are too slow or do not have enough uptime are disqualified by the satellite for future storage operations. Once disqualified, a storage node must restart the vetting process to re-enter the network.
- Preference system: based on storage node latency, history of reliability and uptime, geographic location and other collected data, they are given a greater selection likelihood for new data uploads.

The preference system only determines where *new* data is stored, and does not affect already stored data or repair data.

Since Storj is a trustless system, storage nodes have a reputation system of their own to determine trustworthy satellites. Storage nodes collect data on payment, demand generation and performance history. If satellites score poorly, storage nodes will avoid accepting their data. Furthermore, when new satellites join the system, storage nodes will start a vetting process of their own, restricting interactions with new satellites and collecting data to gauge their trustworthiness.

The role of satellites is extremely important in this system, and requires satellites to be constantly running – as should these go offline, repair processes would stop and eventually all stored copies of the file segments will disappear. It should be noted here that a satellite *instance* does not necessarily constitute one physical machine. Instead, a satellite can run as several servers, and can be backed by a horizontally scalable trusted database to ensure greater uptime. Furthermore, uplinks can connect to multiple satellites to increase data availability and permanence. Nonetheless, if a satellite goes offline, all data coordinated across storage nodes by that satellite will be inaccessible: the data will remain online on storage nodes, however will become inaccessible as the retrieval mechanism of the satellite would be unavailable.

As can be seen from the above, Storj implements various techniques to ensure data availability, tamper-protection and privacy. Storj also allows the editing of data – part of their mission to become an AWS S3 competitor – through the use of authorizations. Users will communicate with a satellite to request adding, removing and editing of files, and if these users have the right authorizations, they will be authenticated which will allow them make changes to their uploaded data according to their authorization configurations.

Pricing Mechanics & Data Permanence

In Storj uplinks, satellites and storage nodes are three distinct actors with distinct functions, all facing different target groups. Uplinks are end-user facing and build a user-friendly way for people with data storage requirements to store data, without interacting with the backend architecture, namely the satellites and storage nodes. Users – through uplinks – end up paying satellites to coordinate communication with storage nodes, the latter of which are often operated by anybody who has additional bandwidth and storage capacity to spare.

In this structure, satellites act as a sort of escrow that collect user payments and hold these, while storage nodes deliver on the storage and retrieval requirements and are paid in regular cycles by satellites based on files stored and bandwidth used in STORJ tokens, and uplinks handle user payments to satellites in multiple currency formats (STORJ, fiat money or other means).

Public storage node payments appear to be set centrally by Storj, and are as follows:

- Storage - STORJ tokens at $1.50 per TB per month (including the increased data size resulting from erasure encoding). If 2TB of storage is used in a month, storage revenue would be $3 worth of STORJ tokens.
- Egress bandwidth - $20 per TB for egress bandwidth related to file retrieval
- Audit & repair bandwidth - $10 per TB for egress bandwidth related to file audit and repair bandwidth

Through the Storj website, users can get a free plan to start testing Storj decentralized cloud services with 150gb storage limit and 150gb bandwidth per month, or get a pro account for which storage costs $4 per TB per month and bandwidth $7 per TB (an additional per-segment fee of $0.0000088 applies as well).

Since users pay only for what they use, there is a clear difference in what users pay and what storage nodes earn. This has two reasons: first, Storj incentivized higher bandwidth with higher revenue, and second, Storj withholds a certain amount of revenue, which storage operators lose if they leave the network. Since this is a public network and users regularly join and leave the network, the revenue from collected witholdings is used to further fund storage node bandwidth:

- Months 1-3: 75% of revenue is withheld, 25% is paid to the Node Operator
- Months 4-6: 50% of revenue is withheld, 50% is paid to the Node Operator
- Months 7-9: 25% of revenue is withheld, 75% is paid to the Node Operator
- Months 10-15: 100% of Storage Node revenue is paid to the Node Operator

- After Month 15: 50% of total withholdings are returned, with the remaining 50% held until the Node gracefully exits the network

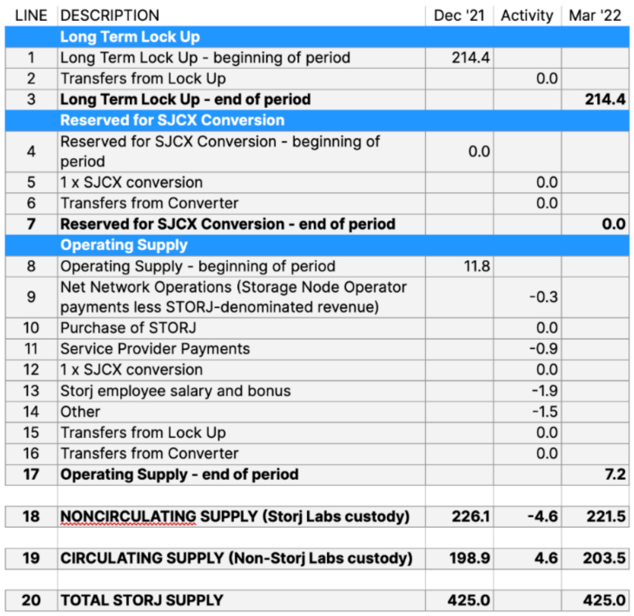
It is important to note here that as time passes, previous withholdings are not returned to the storage node, but instead the withholding amount for new revenues is reduced. If a storage node stays operational for 15 months, then half of historical withholdings are returned, and the other half becomes claimable if they do a graceful exit. A graceful exit refers to storage nodes permanently leaving the network, for which the node triggers a special command with the satellite that coordinates the moving of all data stores on the node to other storage nodes.

Storj assumes that all data is meant to be stored permanently, unless the file is given a specific time-to-live (TTL) value during upload, which is essentially an expiration date. If no TTL value is set, the files will stay online for as long as the user pays the satellite through the uplink and the uplink remains online.

Tokenomics

The STORJ token is the utility token used for paying for storage and bandwidth on the Storj decentralized storage network. Previously Storj operated using a Bitcoin-based token with the ticker SJCX, however throughout 2017 Storj began allowing users to convert their Bitcoin-based tokens to their Ethereum ERC20 equivalent, the STORJ token. The total supply of STORJ ERC20 tokens is fixed and pre-minted at 425 million STORJ.

Of the 425 million STORJ in circulation, as of March 2022, 203.5m are in public circulation and 221.5m are in custody of Storj Labs, *see items 18 and 19 in below figure*.

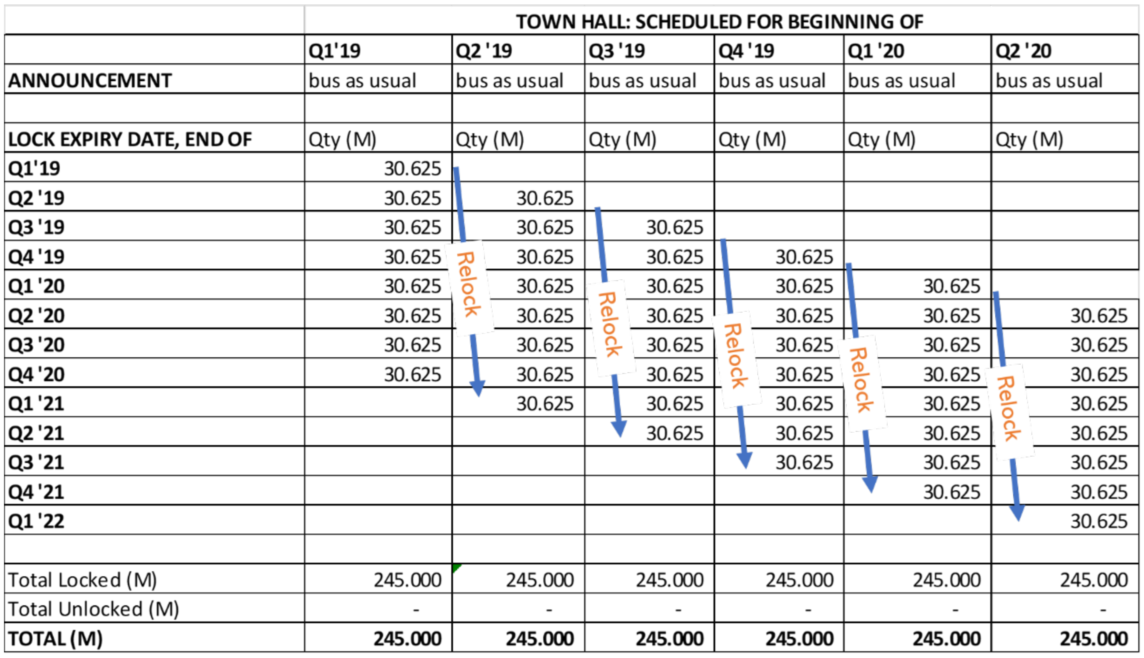


LINE	DESCRIPTION	Dec '21	Activity	Mar '22
	Long Term Lock Up			
1	Long Term Lock Up - beginning of period	214.4		
2	Transfers from Lock Up		0.0	
3	**Long Term Lock Up - end of period**			**214.4**
	Reserved for SJCX Conversion			
4	Reserved for SJCX Conversion - beginning of period	0.0		
5	1 x SJCX conversion		0.0	
6	Transfers from Converter		0.0	
7	**Reserved for SJCX Conversion - end of period**			**0.0**
	Operating Supply			
8	Operating Supply - beginning of period	11.8		
9	Net Network Operations (Storage Node Operator payments less STORJ-denominated revenue)		-0.3	
10	Purchase of STORJ		0.0	
11	Service Provider Payments		-0.9	
12	1 x SJCX conversion		0.0	
13	Storj employee salary and bonus		-1.9	
14	Other		-1.5	
15	Transfers from Lock Up		0.0	
16	Transfers from Converter		0.0	
17	**Operating Supply - end of period**			**7.2**
18	**NONCIRCULATING SUPPLY (Storj Labs custody)**	**226.1**	**-4.6**	**221.5**
19	**CIRCULATING SUPPLY (Non-Storj Labs custody)**	**198.9**	**4.6**	**203.5**
20	**TOTAL STORJ SUPPLY**	**425.0**		**425.0**

Figure 56: STORJ Token Balances & Flows Report: Q1 '22. Source: https://www.storj.io/blog/storj-token-balances-and-flows-report-q1-2022

The tokens that are in Storj Labs custody are broken into eight different tranches, each containing 30.625 million STORJ that are unlocked every quarter for eight consecutive quarters. At time of writing (May 25th, 2025) this represents roughly $16.8 million USD. Each tranche is in a time-locked smart contract, so even if Storj Labs would want to redeem these tokens, they would have to need 8 consecutive quarters to withdraw the full amount.

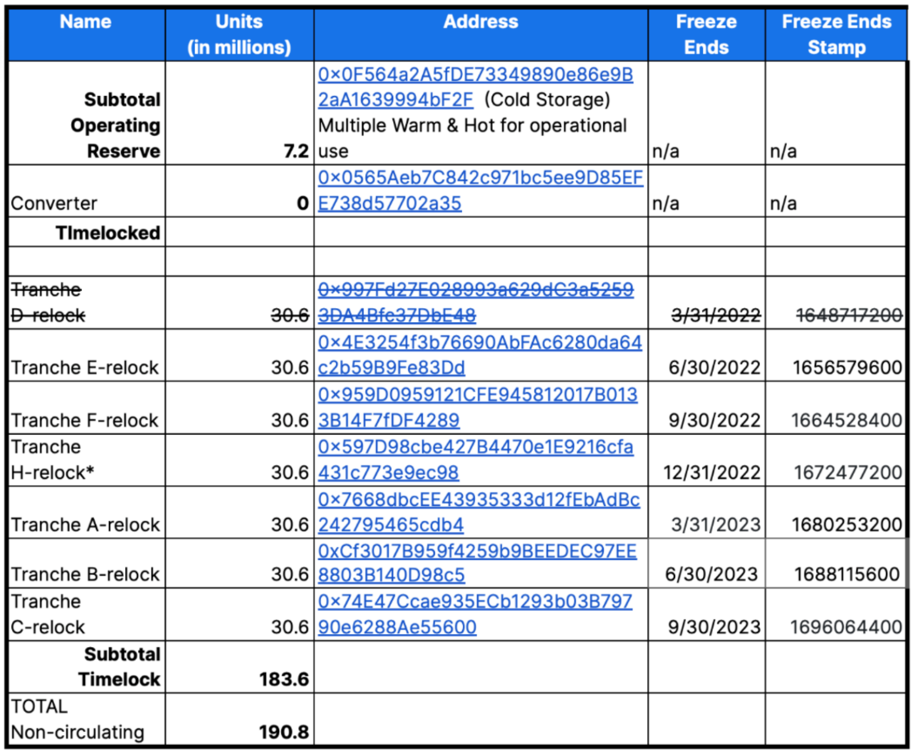
In December 2018 when these tranches were introduced, Storj Labs committed to relocking the first tranche and appending it to the unlocking of the last tranche to essentially delay any unlockings without affecting the one tranche per quarter arrangement. Whenever a tranche is not used, the tranche get relocked as Storj Labs attempts to finance its operations through operating the public Storj network and keeps these tranches as financial reserves.


Relock
Relock
Relock
Relock
Relock

	TOWN HALL: SCHEDULED FOR BEGINNING OF					
	Q1'19	**Q2 '19**	**Q3 '19**	**Q4 '19**	**Q1 '20**	**Q2 '20**
ANNOUNCEMENT	bus as usual	bus as usual	bus as usual	bus as usual	bus as usual	bus as usual
LOCK EXPIRY DATE, END OF	Qty (M)	Qty (M)	Qty (M)	Qty (M)	Qty (M)	Qty (M)
Q1'19	30.625					
Q2 '19	30.625	30.625				
Q3 '19	30.625	30.625	30.625			
Q4 '19	30.625	30.625	30.625	30.625		
Q1 '20	30.625	30.625	30.625	30.625	30.625	
Q2 '20	30.625	30.625	30.625	30.625	30.625	30.625
Q3 '20	30.625	30.625	30.625	30.625	30.625	30.625
Q4 '20	30.625	30.625	30.625	30.625	30.625	30.625
Q1 '21		30.625	30.625	30.625	30.625	30.625
Q2 '21			30.625	30.625	30.625	30.625
Q3 '21				30.625	30.625	30.625
Q4 '21					30.625	30.625
Q1 '22						30.625
Total Locked (M)	245.000	245.000	245.000	245.000	245.000	245.000
Total Unlocked (M)	-	-	-	-	-	-
TOTAL (M)	**245.000**	**245.000**	**245.000**	**245.000**	**245.000**	**245.000**

Figure 57: Tranche relocking schedule. Source: https://www.storj.io/blog/using-timelocked-tokens-to-support-long-term-sustainability

Storj committed to giving 60-days notice if there are to be any changes to the tranche unlocking schedule. Of the 8 tranches, currently two tranches have not been relocked. The first discontinuation of a relock was at the end of Q3 2020, and the second occured in the end of Q1 2022. The stated purpose for unlocking these tokens was to support the networks continuing growth, thus increasing tokens in circulation to 234.1 million STORJ and reducing tokens in Storj Labs custody to 190.8 million STORJ.



Name	Units (in millions)	Address	Freeze Ends	Freeze Ends Stamp
Subtotal Operating Reserve	**7.2**	0×0F564a2A5fDE73349890e86e9B2aA1639994bF2F (Cold Storage) Multiple Warm & Hot for operational use	n/a	n/a
Converter	**0**	0×0565Aeb7C842c971bc5ee9D85EFE738d57702a35	n/a	n/a
TImelocked				
~~Tranche D-relock~~	~~30.6~~	~~0×997Fd27E028993a629dC3a52593DA4Bfe37DbE48~~	~~3/31/2022~~	~~1648717200~~
Tranche E-relock	30.6	0×4E3254f3b76690AbFAc6280da64c2b59B9Fe83Dd	6/30/2022	1656579600
Tranche F-relock	30.6	0×959D0959121CFE945812017B0133B14F7fDF4289	9/30/2022	1664528400
Tranche H-relock*	30.6	0×597D98cbe427B4470e1E9216cfa431c773e9ec98	12/31/2022	1672477200
Tranche A-relock	30.6	0×7668dbcEE43935333d12fEbAdBc242795465cdb4	3/31/2023	1680253200
Tranche B-relock	30.6	0xCf3017B959f4259b9BEEDEC97EE8803B140D98c5	6/30/2023	1688115600
Tranche C-relock	30.6	0×74E47Ccae935ECb1293b03B79790e6288Ae55600	9/30/2023	1696064400
Subtotal Timelock	**183.6**			
TOTAL Non-circulating	**190.8**			

Figure 58: Storj Labs custody Ethereum addresses as of April 28th. Source: https://www.storj.io/blog/storj-token-balances-and-flows-report-q1-2022

If we ignore changes in circulating tokens and only consider the fixed supply, it can be assumed that in the long-term the value of the STORJ token will increase as both supply and demand side actors in Storj network need to use the Storj token for storage activities.

Swarm

Swarm is a decentralized storage network built on the Ethereum network and incentivized through the BZZ token, an Ethereum-based ERC20 token. Swarm's vision is to "extent the blockchain with peer-to-peer storage and communication to realize the world computer that can serve as an operating system and deployment environment for decentralized applications".

Swarm also aims to provide freedom of information through permissionless publishing and privacy through features such as anonymous browsing, deniable storage, untraceable messaging, and file representation formats that leak no metadata.

Anybody with additional hard drive space and bandwidth can join the Swam.

The network has been in development as early as 2015, and as of February 2022 the network could reliably upload and download roughly 5mb of data at speeds of 6.47MiB/s up and 12.47MiB/s down. Finally, data on Ethereum swarm is accessible through human readable formats and can be resolved through ENS domains.

Token

The utility token of the Swarm network is the BZZ token, which is an ERC20 token that lives on the Ethereum blockchain. The token supply is dynamic and changes with purchases/sales of the token from and to the bonding curve, which determines the price of the token. *More details in the tokenomics section.*

Storage Technology & Consensus Mechanism

Swarm consists of four interconnected yet clearly separable layers that together form the infrastructure of Swarm, of which the overlay network and an API to access that network form the core of the Swarm protocol.

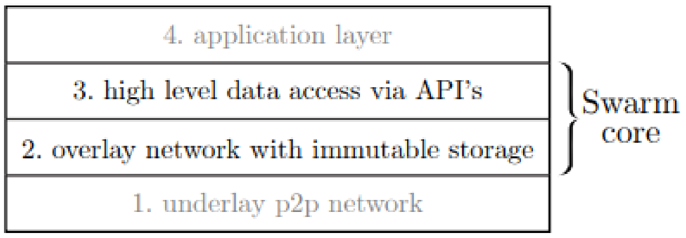

4. application layer
3. high level data access via API's
2. overlay network with immutable storage
1. underlay p2p network
Swarm core

Figure 59: Swarm's layered design. Source: The Book of Swarm

The overlay network determines how files are stored and represents the protocols underlying storage model. The storage model developed by Swarm is called Distributed Immutable Store of Chunks (DISC) and forms the basis of how to nodes communicate with each other. Swarm is essentially a slightly different interpretation of a distributed hash table (DHT), similar to how IPFS nodes manage and keep track of the various nodes they have connected to, but instead of storing where files are to be found DISC directly stores tiny pieces of files (i.e., chunks) – *more on that later*.

In Swarm, nodes are expected to make decisions in regards to which other nodes to connect to based on their proximity so that local connection decisions can reach globally optimal routing of messages (known as Kademlia connectivity). Every node tracks both the network address and Swarm address of other nodes, the latter of which tracks enables to define the proximity of two addresses to each other.

Using the degree of proximity, nodes that are closest to each other form a fully connected neighborhood requiring a minimum of 8 nodes, and also connect to 8 further nodes each in an increasingly lesser different degree of proximity to the neighborhood (i.e., they are farther away from each other).

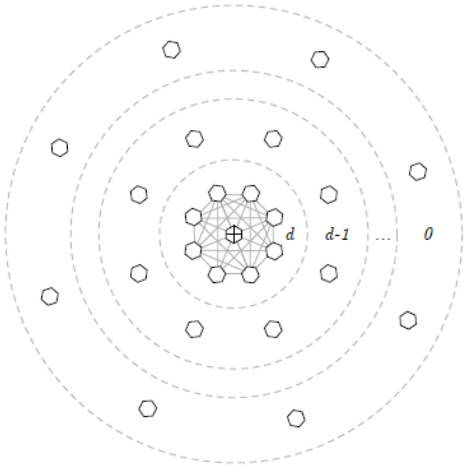


Figure 60: Kademlia connectivity used in Swarm. Source: Swarm Whitepaper

This network design ensures that messages intended for nodes that are very far away from each other can always reach their destination, even if the nodes are not directly connected.

Swarm stores data on these nodes as chunks, which represent 4 kilobytes of data with an address that exists in the same address space as node addresses, enabling the calculation of proximity of nodes and chunks. Swarm requires nodes that are in close proximity of a data chunk to store that same chunk locally, thus creating clusters of replicated data within a neighborhood. Since a chunk is essentially just a segment of a larger file, without the context of what the full file is meant to be, nodes are unable to rebuild the full file. Chunks can be encrypted for additional privacy. Furthermore, to ensure file redundancy and consistent availability when nodes leave or join the network, nodes continuously synchronize chunks with their neighbors.

To retrieve a chunk from a neighborhood, a client communicates with a node which is in close proximity to itself requesting the retrieval of the chunk. Using the chunk address and the Kademlia algorithm, the nodes recursively forward the message through various nodes in varying proximity layers until they reach the neighborhood hosting the file, which then returns the file along the same route. If any node along the way happens to have the chunk in their local storage, it is sent back as a response instead. Nodes are incentivized to cache chunks of data to reduce bandwidth usage of the network. This is achieved through *opportunistic caching*, which refers to the caching chunks of distant neighborhoods to receive payment for retrieval of those chunks. Cached data lives in the caching subsystem of a Swarm node.

In Swarm, each node has two local subsystems, namely the reserve and the cache. In simple terms, the reserve stores chunks that have postage stamps attached to it. Postage stamps are purchased through BZZ tokens and indicate the value a user places on storing these files on Swarm. When a file is stored on a node, the postage stamp acts as a sort of rent that decreases over time. Once the value of the stamp reaches a certain threshold, it is moved from the reserve (i.e., paid for storage) to the cache.

The cache stores chunks that are not protected by the reserve, either because the storage stamp value has reduced over time, or because the cached chunk is from a distant node. Chunks in cache are ranked by their latest retrieval as a means to indicate the popularity of the chunk and whether it is worth to continue storing the chunk. The cache is regularly cleared of unpopular chunks, ensuring that popular content is permeated across the network and easily retrievable, while also maximizing income for nodes: When nodes on the network return a chunk from a retrieval request, the nodes earn BZZ tokens, hence economically incentivizing the holding of as many chunks as possible.

The method of retrieving and transferring files described above increases anonymity in the network, because a node forward request and an initial request initiation are identical in terms of structure. This ambiguity obfuscates the identity of those retrieving files.

However, this approach leads to unpopular chunks of data to disappear from nodes overtime, thus impacting permanence of the system. To combat this, Swarm implements a postage lottery system called “RACE” (raffle, apply, claim and earn), that is executed through smart contracts on the Ethereum blockchain. While the detailed mechanism of race goes beyond the scope of this research, it suffices to know that these raffles:

- act as spot checks on nodes
- for nodes presents an opportunity to earn additional income
- encourage nodes to stay online as they would otherwise miss raffles
- require nodes to store the right data and properly maintain the stored chunks

Finally, in order to reconstruct files, nodes need to be able to understand which chunks belong to which files, and the downloader needs to be able to verify the correctness of the chunks. In Swarm, every chunk address is unique, implying a unique address-payload association. This uniqueness creates the immutability of the chunk, as only that chunk can contain the data embedded in it. The canonical content addressed chunk in Swarm is called a binary Merkle tree chunk (BMT chunk), and the address of BMT chunks is calculated using the binary tree hash algorithm (BMT hash).

Swarm has two kinds of chunks; a content addressed chunk and a single owner chunk. While these differ in terms of data structure, both use the BMT hash verify chunk integrity and to reconstruct the full file. Ultimately, users can use their Swarm hash (also known as bzzhash) to signal to the network to retrieve all chunks and recreate the file. For more details on the BMT hashing algorithm, please refer to The Book of Swarm.

For additional protection against data loss, caused by nodes going offline or being otherwise unable to access data, Swarm applies Cauchy-Reed-Solomon erasure coding to 4 kilobyte sized chunks of the file before they are hashed into the Merkle tree. This allows the network to retrieve data, even when a portion of the chunks are inaccessible.

Finally, Swarm includes a pinning function which allows nodes to save all chunks locally and prevent the chunks from being removed.

Data Permanence & Pricing Mechanics

Swarm applies the Swarm Account Protocol (SWAP) to incentivize nodes to collaborate with each other in routing messages, while decreasing frivolous bandwidth use. Nodes track the relative bandwidth consumption with peers they connect with, creating a debts and credits balancing mechanism between any two nodes at any given point in time. When node A requests data from node B, and node B responds, then node B has a credit surplus, while node A has debt liabilities. This can continue until a certain threshold is reached, after which node B will not accept further requests until node A has repaid liabilities. To pay back liabilities and have both nodes return to a balanced state of fewer debts and liabilities within the threshold, node A can either wait for reciprocal requests from node B which would reduce node A's debts, or they send a cheque that can be cashed out for BZZ tokens to node B pay back the debt. This creates a "service-for-service" relationship between nodes.

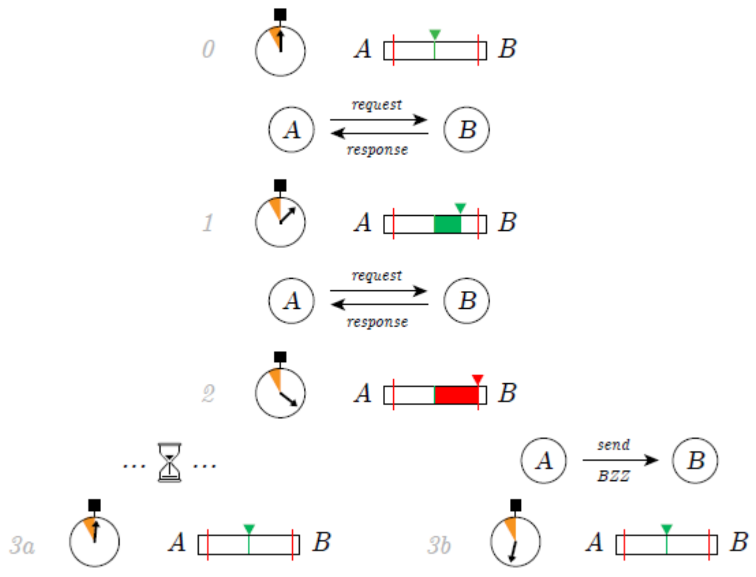


Figure 61: Swarm Accounting Protocol. Source: Swarm whitepaper.

Cheques are handled on-chain by a smart contract. Nodes must decide for themselves whether to cash a cheque upon receipt, or to wait to reduce transaction costs on the Ethereum network. If the node waits, however, they increase the risk of settlement failure, i.e., the check can bounce due to the cheque provider moving funds outside of their chequebook wallet. This is where Swarm employs a reputation system: because the smart contract records failed cheque withdrawals, nodes can see publicly which other nodes did not make good on their cheques and can refrain from communicating with that node in the future.

Apart from the SWAP protocol, nodes can also earn additional BZZ by holding unpopular data and participating in the RACE lottery system.

The SWAP and RACE systems are positive incentive mechanisms. Swarm also employs negative incentive systems called “competitive insurance”. Competitive insurance requires nodes to store every bit of promised, and failure to do so is not only unprofitable, but outright catastrophic to the insurer. While SWAP incentivizes short term data storage, and RACE incentivized long-term data storage of popular files, competitive insurance incentivizes long-term data storage of any files stored on the network no matter their popularity, as well as simultaneously prevent users from spinning up new nodes to sell empty long-term storage promises, only to cash-out and deactivate their node shortly after.

The competitive insurance system works with a deposit system. Nodes that want to sell long-term storage (aka promissory storage) must have a stake verified and locked-in with an Ethereum-based smart contract at the time of making their promise – essentially a security deposit. If the security deposit has been locked, the node is entitled to make storage promises up to the duration of the locked stakes. If, during the promise period, a node fails to prove ownership of the data they promised to store, they lose their entire security deposit. If a user or a node finds that content with a promise is inaccessible, they can submit a challenge to a smart contract that handles the verification process.

Nodes are compensated for their promises over time. When a user stores data on a node, they pay up-front for the entire storage duration. This amount is locked, and is released in installments to the node as long as they can provide proof of custody of the files.

Swarm let’s end users, through client software, determine the amount of data and duration that data is to be stored on the network. The price for storage is automatically calculated through the client software and smart contracts. Swarm allows for different levels of data retention:

- minimal – a few hours
- temporary – a week
- long term – a year
- forever – 10 years

The purchase of storage space over time in Swarm is called a postage subscriptions, and they are managed by the postage subscription API, which shows users how much data of a specific subscription has been uploaded and for how long it can be stored at its current price (e.g. 88/100 megabytes for 23 days).

To access one's content on Swarm, one either needs to run a Swarm node, or use a public gateway. This setup is similar to that of IPFS. Swarm recommends to only retrieve encrypted content through one's own node, as the if using a public gateway, as soon as the content leaves the gateway and is transmitted over http to the user it will not be encrypted anymore.

Tokenomics

While the project has been with the Ethereum Foundation since 2015 within their Geth team, only in June 2021 Swarm launched a public token sale. Details of the token distribution can be found below:

- 27.6 million BZZ (42%) – Early Token Sale (Early backers in Sept 2020, Private round in Dec 2020)
- 5.17 million BZZ (8%) – Public Token Sale (June 2021, unlocked in August 2021)
- 15.87 million BZZ (23%) – Ecosystem (Infrastructure for L1 solutions, development, airdrops, grants, donations)
- 12.50 million BZZ (19%) – Present and Future Team Members
- 4.9 million BZZ (7%) – Swarm Foundation (protocol, network and business development, marketing and community support)

The price of the BZZ token is determined by a bonding curve that is controlled by smart contracts instead of traditional market makers, where purchases and sales of tokens to the bonding curve will directly adjust the price that users pay for new tokens. This makes it prohibitively expensive to buy or dump large amounts of tokens at once, thus protecting the utility of the token against speculative actions.

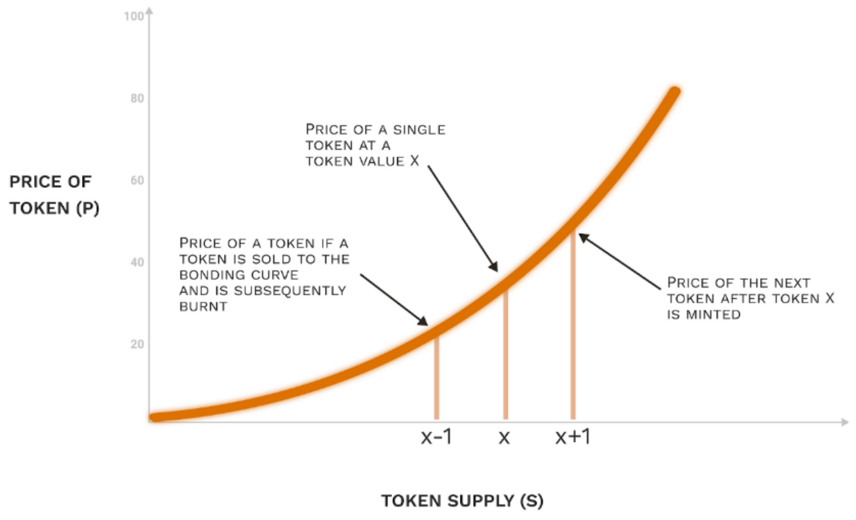
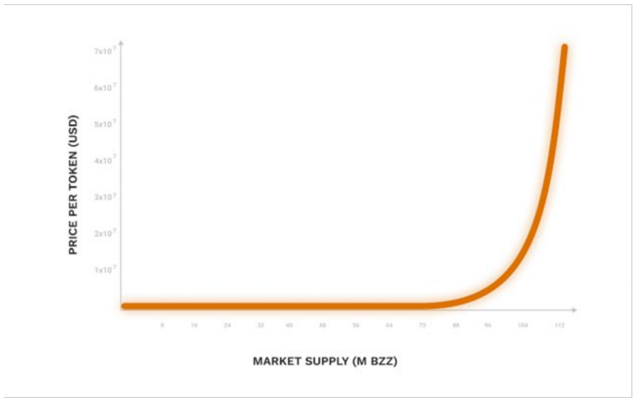
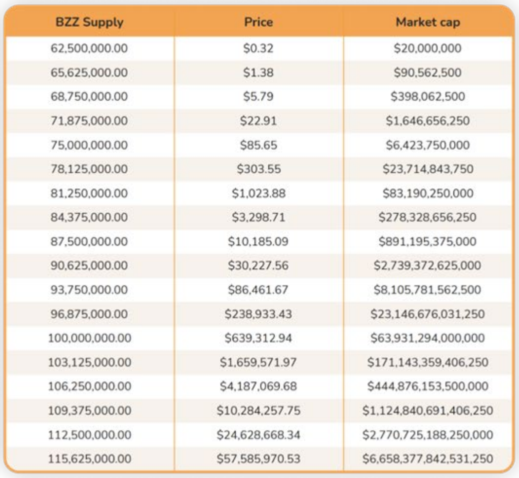

100
80
60
40
20
PRICE OF TOKEN (P)
PRICE OF A SINGLE TOKEN AT A TOKEN VALUE X
PRICE OF A TOKEN IF A TOKEN IS SOLD TO THE BONDING CURVE AND IS SUBSEQUENTLY BURNT
PRICE OF THE NEXT TOKEN AFTER TOKEN X IS MINTED
x-1
x
x+1
TOKEN SUPPLY (S)

Figure 62: Swarm bonding curve explanation on Bzzaar exchange (bzz.echange). Source: https://medium.com/ethereum-swarm/swarm-and-its-bzzaar-bonding-curve-ac2fa9889914

As a result of this bonding curve, the actual circulating supply of tokens is in constant fluctuation. Although there is a hardcoded maximum of 125 million BZZ tokens, it's extremely unlikely the higher end of token supply will ever circulate, due to the shape of the bonding curve which steepens heavily.





BZZ Supply	Price	Market cap
62,500,000.00	$0.32	$20,000,000
65,625,000.00	$1.38	$90,562,500
68,750,000.00	$5.79	$398,062,500
71,875,000.00	$22.91	$1,646,656,250
75,000,000.00	$85.65	$6,423,750,000
78,125,000.00	$303.55	$23,714,843,750
81,250,000.00	$1,023.88	$83,190,250,000
84,375,000.00	$3,298.71	$278,328,656,250
87,500,000.00	$10,185.09	$891,195,375,000
90,625,000.00	$30,227.56	$2,739,372,625,000
93,750,000.00	$86,461.67	$8,105,781,562,500
96,875,000.00	$238,933.43	$23,146,676,031,250
100,000,000.00	$639,312.94	$63,931,294,000,000
103,125,000.00	$1,659,571.97	$171,143,359,406,250
106,250,000.00	$4,187,069.68	$444,876,153,500,000
109,375,000.00	$10,284,257.75	$1,124,840,691,406,250
112,500,000.00	$24,628,668.34	$2,770,725,188,250,000
115,625,000.00	$57,585,970.53	$6,658,377,842,531,250

Figure 63: Shape of BZZ bonding curve. Source: https://medium.com/ethereum-swarm/swarm-and-its-bzzaar-bonding-curve-ac2fa9889914

The bonding curve smart contract address can be found here:
https://etherscan.io/address/0x4f32ab778e85c4ad0cead54f8f82f5ee74d46904

Although the bonding curve is fully automated, the Swarm Foundation maintains control to manually shut down the bonding curve in emergency situations, which include:

- A critical or exploitable bug in the bonding curve contract is discovered
- MakerDAO discovers a critical bug or is shut down for any reason; and
- DAI loses its peg to the USD

As of writing, there are roughly 63.5 million BZZ tokens in circulation, which is -2.7 million BZZ tokens below the 66.2 million tokens that were minted and distributed during token launch. Indicating that tokens have likely been sold back to the bonding curve since launch.

Real-time circulating supply: https://tokenservice.ethswarm.org/circulating_supply

Real-time bonding curve: https://tokenservice.ethswarm.org/token_price

References

References have been split by category for easier review.

All references were accessed between May 5th and 31st, 2022.

Arweave

Amber Group (2011) *Arweave: Enabling the Permaweb*. Available at: https://medium.com/amber-group/arweave-enabling-the-permaweb-870ade28998b

Arweave (2021) *Storage Endowment*. Available at: https://arwiki.wiki/#/en/storage-endowment (Accessed May 29th, 2022)

Arweave (2021) *What is the Permaweb*. Available at: https://arwiki.wiki/#/en/the-permaweb (Accessed May 29th, 2022)

Arweave (2022) *Storage Price Stabilization*. Available at: https://arwiki.wiki/#/en/Storage-Price-Stabilization (Accessed May 29th, 2022)

Arweave (n.d.) *Technology*. Available at: https://www.arweave.org/technology

Arweave Fees (2021) *Arweave Fees*. Available at: https://arweavefees.com/

Crypto Valley Journal (n.d.) *Permaweb*. Available at: https://cvj.ch/en/glossary/permaweb/

Gemini (2022) *Arweave: A Permanent, Decentralized Internet*. Available at: https://www.gemini.com/cryptopedia/arweave-token-ar-coin-permaweb

Gemini (n.d.) *Glossary: Permaweb*. Available at: https://www.gemini.com/cryptopedia/glossary#permaweb

Messari (n.d.) *Arweave – Data Storage Protocol*. Available at: https://messari.io/asset/arweave/profile/token-usage

Tom McWright (2019) *How much does Arweave cost?* https://observablehq.com/@tmcw/how-much-does-arweave-cost

Wayne Jones (n.d.) *Can data really be stored forever?* Available at: https://ardrive.io/can-data-really-be-stored-forever/

Crust

Crust Network (2020) *Crust Technical White Paper v1.9.9*. Available at: https://gw.crustapps.net/ipfs/QmP9WqDYhreSuv5KJWzWVKZXJ4hc7y9fUdwC4u23SmqL6t

Crust Network (2021) *Crust Token Metrics & Economies.* Available at: https://medium.com/crustnetwork/crust-token-metrics-economics-84592efc6d1f

Crust Network (2021) *Economy Whitepaper.* Available at: https://gw.crustapps.net/ipfs/QmRYJN6V5BzwnXp7A2Avcp5WXkgzyunQwqP3Es2Q789phF

Crust Network (n.d.) *DSM*. Available at: https://wiki.crust.network/docs/en/DSM

Crust Network Subscan (n.d.) *Storage Price Calculator*. Available at: https://crust.subscan.io/tools/storage_calculator

Filecoin

Coinlist (2017) *Filecoin Token Sale Economies*. Available at: https://coinlist.co/assets/index/filecoin_2017_index/Filecoin-Sale-Economics-e3f703f8cd5f644aecd7ae3860ce932064ce014dd60de115d67ff1e9047ffa8e.pdf

File.App (n.d.) Available at: https://file.app/

Filecoin (2020) *Date Transfer in Filecoin*. Available at: https://spec.filecoin.io/systems/filecoin_files/data_transfer/

Filecoin (2020) *Storage Market in Filecoin*. Available at: https://spec.Filecoin.io/systems/Filecoin_markets/storage_market/

Filecoin (2020) *Understanding Filecoin Circulating Supply.* Available at: https://filecoin.io/blog/filecoin-circulating-supply/

Filecoin (2020) *What sets it apart: Filecoin's proof system*. Available at: https://filecoin.io/blog/posts/what-sets-us-apart-filecoin-s-proof-system/

Filecoin (2021) *How storage and retrieval deals work on Filecoin*. https://Filecoin.io/blog/posts/how-storage-and-retrieval-deals-work-on-Filecoin/

Filecoin (n.d.) *IPFS and Filecoin.* Available at: https://docs.Filecoin.io/about-Filecoin/ipfs-and-Filecoin/

Filecoin (n.d.) *IPFS*. Available at: https://spec.filecoin.io/#section-libraries.ipfs

James Duade (2021) *FileCoin: Decentralized Cloud Storage Competitor To AWS, Microsoft Azure, And Google Cloud*. Available at: https://seekingalpha.com/article/4419553-Filecoin-decentralized-cloud-storage-competitor-aws-microsoft-google

IPFS

Brave (2021) *IPFS Support in Brave*. Available at: https://brave.com/ipfs-support/

Hussein Nasser (2021) *The IPFS Protocol Explained with Examples – Welcome to the Decentralized Web*. Available at: https://www.youtube.com/watch?v=PlvMGpQnqOM

IPFS (n.d.) *InterPlanetary Name System (IPNS).* Available at: https://docs.ipfs.io/concepts/ipns/

James (2018) *The technology behind IPFS.* Available at: https://medium.com/coinmonks/the-technology-behind-ipfs-and-what-can-ipfs-do-c7009fe42bab

Sia

Cryptopedia Staff (2021) *Sia: A Highly Decentralized Data Storage Solution*. Available at: https://www.gemini.com/cryptopedia/siacoin-mining-sia-crypto-sc-coin-decentralized-storage#section-siacoin-sc-and-siafunds

Sia Docs (2022) *Sia 101*. Available at: https://docs.sia.tech/get-started-with-sia/sia101

Sia Wiki (2017) *Hosting*. Available at: https://siawiki.tech/host/hosting

Sia Wiki (2017). *Using the UI for* renting. Available at: https://siawiki.tech/renter/using_the_ui_for_renting

SiaStats.info (2018) *"When coin burn?" Every day, and 400 000 SC so far.* Available at: https://www.reddit.com/r/siacoin/comments/8te2e4/when_coin_burn_every_day_and_400_000_sc_so_far/

Taek (2015) *[ANN] Sia - Decentralized Storage.* Available at: https://bitcointalk.org/index.php?topic=1060294.msg11372055#msg11372055

Taek (n.d.) *How Sia Works.* Available at: https://web.archive.org/web/20171102065557/https:/forum.sia.tech/topic/108/how-sia-works

Vorick, D. and Champine, L. (2014) *Sia: Simple Decentralized Storage*. Available at: https://sia.tech/sia.pdf

Storj

Gleeson, J. (2019) *Cloud Storage Prices Haven't Changed Much in 4 Years but They're About To*. Available at: https://www.storj.io/blog/cloud-storage-prices-havent-changed-much-in-4-years-but-theyre-about-to

Gleeson, J. and Ihnatiuk, V. (2021). *Sharing Space for Fun and Profit—Part 2*. Available at: https://www.storj.io/blog/sharing-space-for-fun-and-profit-part-2

Johnson, K. (2022) *STORJ Token Balances and Flows Report: Q1 2022*. Available at: https://www.storj.io/blog/storj-token-balances-and-flows-report-q1-2022

Storj DCS Docs (n.d.) *Getting Started on DCS*. Available at: https://storj-labs.gitbook.io/dcs/

Storj Docs (n.d.) *Billing, Payment and Accounts*. Available at: https://docs.storj.io/dcs/billing-payment-and-accounts-1/pricing/

Storj Forum (2020) *Single point of Failure if a Satellite is down?* Available at: https://forum.storj.io/t/single-point-of-failure-if-a-satellite-is-down/4577

Storj Labs (n.d.) *Transparent Pricing*. Available at: https://www.storj.io/pricing

Storj Labs, Inc (2018) *Storj: A Decentralized Cloud Storage Network Framework*. Available at: https://www.storj.io/storjv3.pdf

Storj Labs, Inc and Subsidiaries (2018) *V3 White Paper Executive Summary*. Available at: https://www.storj.io/Storj-White-Paper-Executive-Summary.pdf

Storj Nlog (2019) *What Storage Node Operators Need to Know About Satellites*. Available at: https://www.storj.io/blog/what-storage-node-operators-need-to-know-about-satellites

Swarm

Altcoin Disrupt (2021) *What is Swarm? ICO Upcoming? Will Swarm 100X? Decentralized? $10K - $100K*. Available at: https://www.youtube.com/watch?v=rxPlYf9Pe2A

Coding Bootcamps (n.d.) *How to Work with Ethereum Swarm Storage*. Available at: https://www.coding-bootcamps.com/blog/how-to-work-with-ethereum-swarm-storage.html

ETHDenver (2022) *The State Of Ethereum Swarm – Angela Vitzthum*. Available at: https://www.youtube.com/watch?v=22HfkeEmOK4

Ethereum Wiki (2020) *Swarm Hash*. Available at: https://eth.wiki/concepts/swarm-hash

Munair (2021) *A Case for Swarming Medical History*. Available at: https://munair.medium.com/a-case-for-swarming-medical-history-77baa5e40424

Swarm (n.d.) *Swarm Docs*. Available at: https://docs.ethswarm.org/docs/

Swarm Hive (2021) *BZZ Tokenomics*. Available at: https://medium.com/ethereum-swarm/swarm-tokenomics-91254cd5adf

Swarm Hive (2021) *Swarm and its "Bzzaar" Bonding Curve*. Available at: https://medium.com/ethereum-swarm/swarm-and-its-bzzaar-bonding-curve-ac2fa9889914

Swarm team (2021) *SWARM: Storage and communication infrastructure for a self-sovereign digital society*. Available at: https://www.ethswarm.org/swarm-whitepaper.pdf

Thebojda (2022) *A Brief Introduction to Ethereum Swarm*. Available at: https://hackernoon.com/a-brief-introduction-to-ethereum-swarm

Trón, V. (2021) *The book of Swarm: storage and communication infrastructure for self-sovereign digital society back-end stack for the decentralised web*. V1.0 pre-release 7. Available at: https://www.ethswarm.org/The-Book-of-Swarm.pdf

Miscellaneous

BitcoinFees.net (n.d.) *Bitcoin Fees*. Available at: https://bitcoinfees.net/ (Accessed May 15th, 2022)

BitcoinTalk Forum (2021) *Some questions about OP_Return*. Available at: https://bitcointalk.org/index.php?topic=5310671.0

Daniel, E. and Tschorsch, F. (2022) *IPFS and Friends: A Qualitative Comparison of Next Generation Peer-to-Peer Data Networks*. Available at: https://arxiv.org/pdf/2102.12737.pdf

Dfinity (n.d.) *Blockchain's future.* Available at: https://dfinity.org/

Dr. Wood, G. (2016) *Ethereum: A Secure Decentralized Generalised Transaction Ledger EIP-150 Revision*. Available at: http://gavwood.com/paper.pdf

Ethereum Foundation (2022) *Gas and Fees*. Available at: https://ethereum.org/en/developers/docs/gas/

Ethereum Foundation (2022) *Introduction to Dapps.* Available at: https://ethereum.org/en/developers/docs/dapps/

Etherscan (2021) *CryptopunksData Smart Contract*. Available at: https://etherscan.io/address/0x16f5a35647d6f03d5d3da7b35409d65ba03af3b2#code

Fenbushi Capital (2021) *Stratos: build "convenience store" network for decentralized storage*. Available at: https://fenbushi.medium.com/stratos-build-convenience-store-network-for-decentralized-storage-7cd7dd0d2b49

Georgios Konstantopoulos & Leo Zhang (2021) *Ethereum Blockspace - Who Gets What and Why*. Available at: https://research.paradigm.xyz/ethereum-blockspace

Internet Computer (2022) Computation and Storage Costs. Available at: https://internetcomputer.org/docs/current/developer-docs/deploy/computation-and-storage-costs

Kofi Kufuor (2021) *The State of NFT Data Storage*. Available at: https://thecontrol.co/the-state-of-nft-data-storage-c471c1af58d5

Larva Labs (2021) *On-chain Cryptopunks*. Available at: https://www.larvalabs.com/blog/2021-8-18-18-0/on-chain-cryptopunks

SnapFingersEditor (2021) *Decentral Storage for NFTs | SnapFingers Weekly #9.* Available at: https://medium.com/snapfingers/decentral-storage-for-nfts-snapfingers-weekly-9-2d9315320847

Stackexchange Forum Bitcoin (2015) *Explanation of what an OP_RETURN transaction looks like*. Available at: https://bitcoin.stackexchange.com/questions/29554/explanation-of-what-an-op-return-transaction-looks-like (Accessed May 15th, 2022)

Stackexchange Forum Ethereum (2016) *What is the cost to store 1KB, 10KB, 100KB worth of data into the ethereum blockchain?* Available at: https://ethereum.stackexchange.com/questions/872/what-is-the-cost-to-store-1kb-10kb-100kb-worth-of-data-into-the-ethereum-block

Uniswap (2020) *Uniswap Interface + IPFS*. Available at: https://uniswap.org/blog/ipfs-uniswap-interface

Uniswap (2022) *Uniswap Frontend Interface Release on Github.* Available at: https://github.com/Uniswap/interface/blob/main/.github/workflows/release.yaml

Uniswap (2022) *Uniswap Github Code Repository.* Available at: https://github.com/Uniswap/interface