InterPlanetary File System (IPFS)

COMS4507 - Seminar Presentation
Course Coordinator - Marius Portmann

Rohith Kotia Palakirti (46544146)

Mitchell Clarke (45839153)

Akshay Varma (46452661)

How We Share Information: HTTP

Server shares data with multiple clients

HTTP works well enough because web pages are lightweight

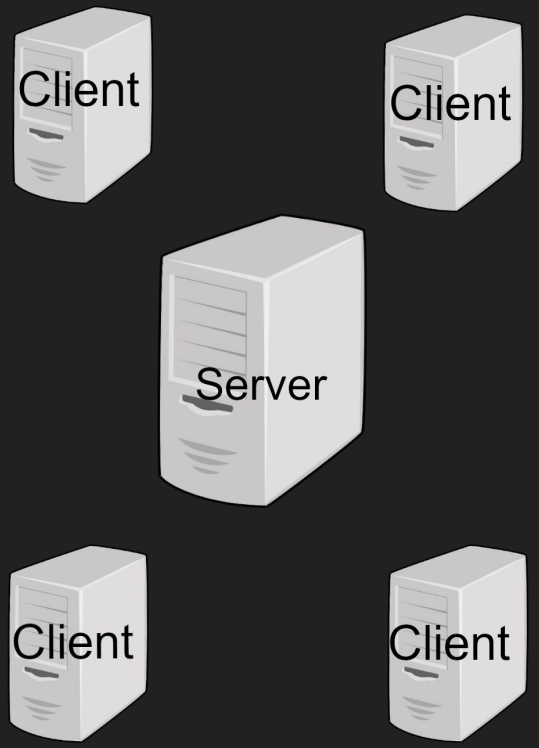
- What about larger files? HTTP is not the best solution.

What if the server is unavailable?

- No route
- Down for some reason
- Maximum connection limit reached

Multiple files need multiple HTTP requests

IPFS offers a solution



FastTrack P2P: the Origins

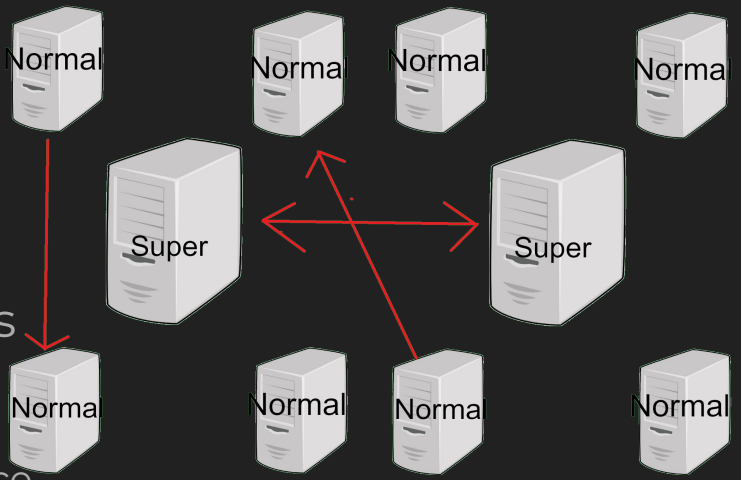
Introduced by Dutch company circa 2001

Computers are either “normal” or “super” nodes [2]

- Super nodes are fast computers (significant performance inequality in 2001)
- Super nodes handle searching on behalf of clients and advertising their files

UUHash algorithm: very fast, slow computers capable of effective use

- Its use allows nodes to get parts of file from separate nodes simultaneously
- Its use also can lead to massive undetected corruption of files [7]



UUHash
First 300KB -> hash
From 1MB hash for 300KB
From 2MB hash for 300KB
From 4MB hash for 300KB
From 2^nMB hash for 300KB...
Last 300KB -> hash
If end 300KB and previous 300KB overlap, ignore previous

P2P in Current Times: BitTorrent

Aimed at sharing large files (ISOs, video, audio, etc)

Similar to FastTrack: Super node becomes Tracker [3]

- (without the massive undetected file corruption)

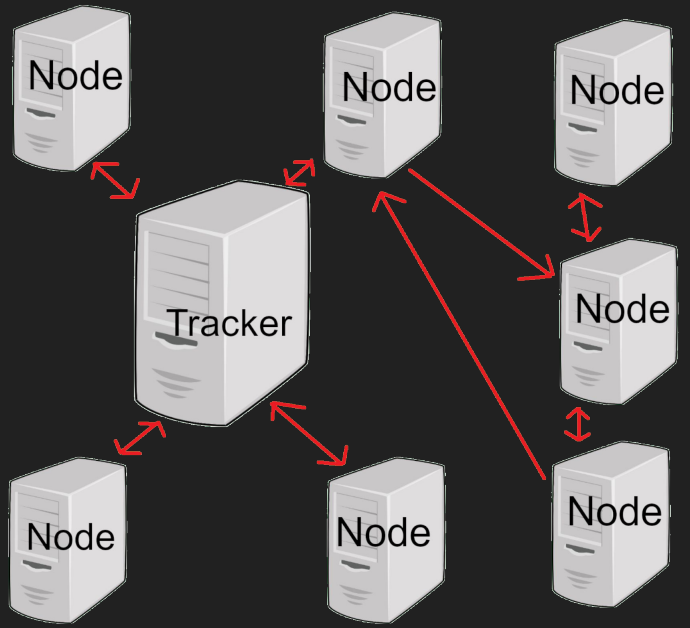
Distributed Hash Table technology [4]

- A tracker isn't required, nodes cooperate searching and advertising files

But, a node isn't required to positively contribute to the sharing

- "Leechers" only download files, then don't share once they have it

Limited set of nodes who can give blocks of wanted file



File Storage Structure and Addressing: Git

Tree data structure stores all committed versions of projects [6]

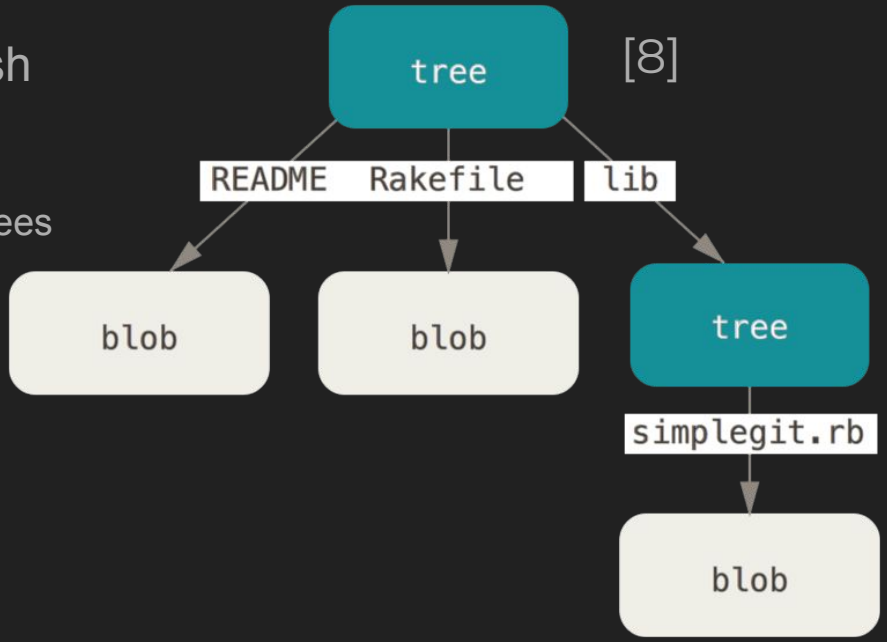
Each file version stored in its entirety

A branch of a tree references the file by its hash [5]

- Identical files are stored once, different branches or trees reference the same object

Removes redundancy and easily addresses content

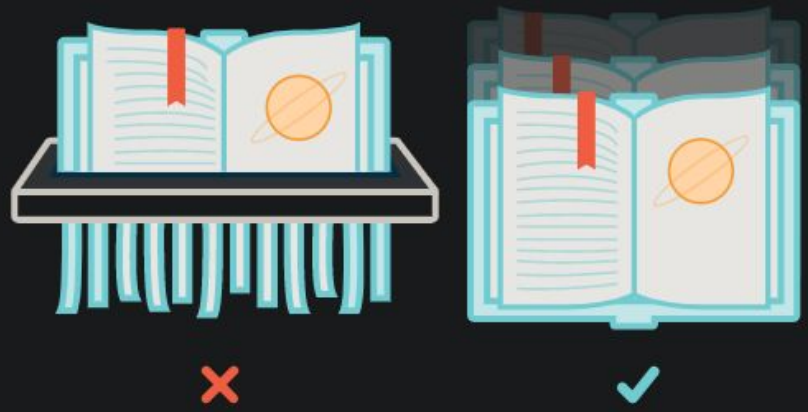
Object reference is universally unique



Today's web is inefficient and expensive

HTTP downloads files from one server at a time — but peer-to-peer IPFS retrieves pieces from multiple nodes at once, enabling substantial bandwidth savings. With **up to 60% savings for video**, IPFS makes it possible to efficiently distribute high volumes of data without duplication.

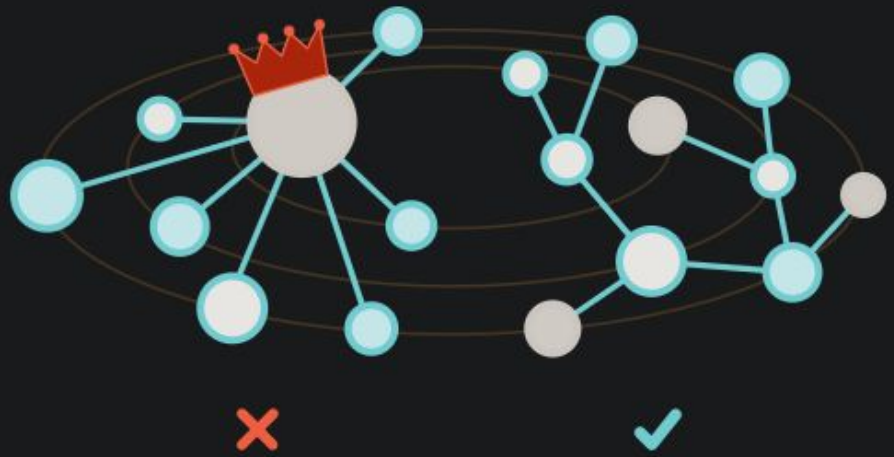
[9]



Today's web can't preserve humanity's history

The average lifespan of a web page is 100 days before it's gone forever. The medium of our era shouldn't be this fragile. IPFS makes it simple to set up resilient networks for mirroring data, and thanks to content addressing, files stored using IPFS are automatically versioned.

[9]



Today's web is centralized, limiting opportunity

The Internet has turbocharged innovation by being one of the great equalizers in human history — but increasing consolidation of control threatens that progress. IPFS stays true to the original vision of an open, flat web by delivering technology to make that vision a reality. [9]

Today's web is addicted to the backbone

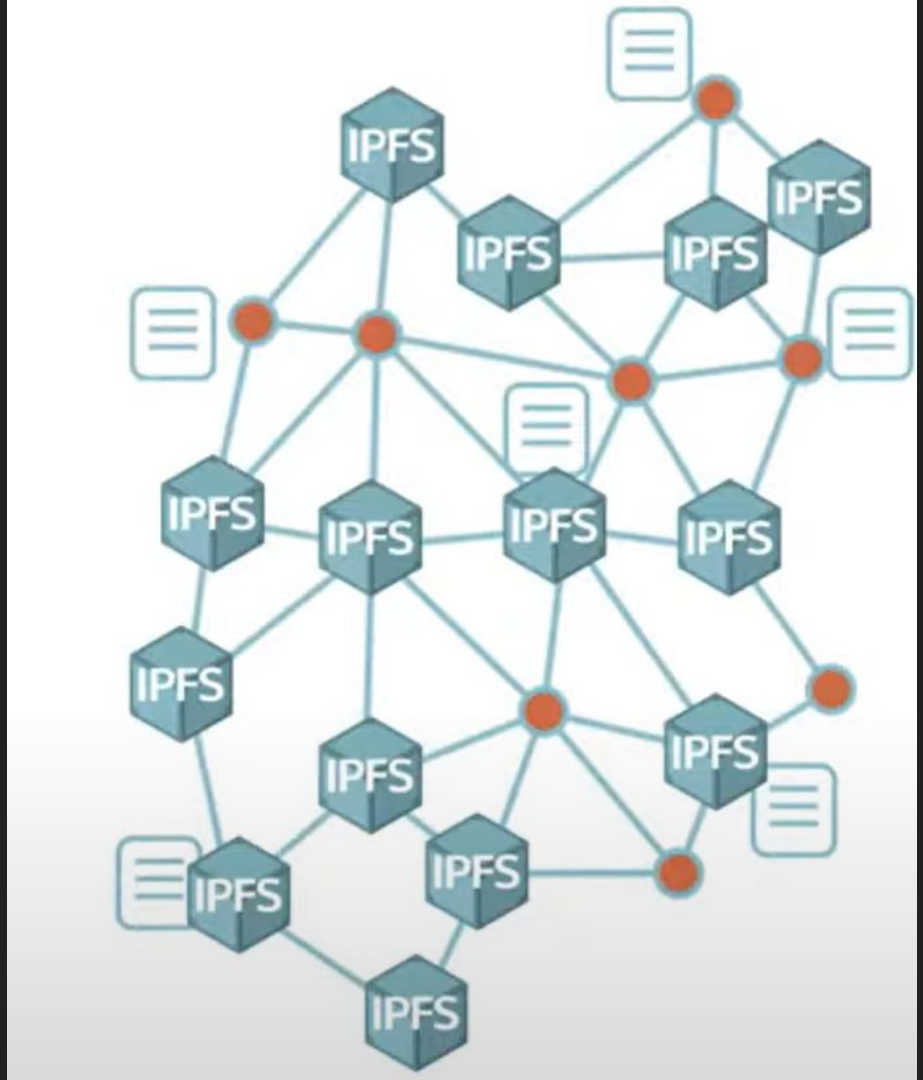
IPFS powers the creation of diversely resilient networks that enable persistent availability — with or without internet backbone connectivity. This means better connectivity for the developing world, during natural disasters, or just when you're on flaky coffee shop wi-fi.

[9]

What is IPFS?

Interplanetary:

- Distributed: no central server
- Resilient/Offline first
- To upgrade the web



IPFS is a protocol:

Defines a content-addressed file system

Coordinates content delivery

Combines Kademlia + BitTorrent + Git

IPFS is a file system:

Has directories and files

Is a mountable filesystem (via FUSE)

IPFS is a web:

Can be used to view documents like the conventional web

Files are accessible via HTTP at https://ipfs.io/<path>

Browsers and extensions can learn to use the ipfs:// URL or dweb:/ipfs/ URI schemes directly

Hash-addressed content guarantees authenticity

IPFS is modular:

Connection layer over any network protocol

Routing layer

Uses a routing layer DHT (Kademlia/Coral)

Uses a path-based naming service

Uses a BitTorrent-inspired block exchange

IPFS is a CDN:

Add a file to the file system locally, and it's now available to the world

Caching-friendly (content-hash naming)

BitTorrent-based bandwidth distribution

IPFS uses crypto:

Cryptographic-hash content addressing

Block-level deduplication

File integrity plus versioning

File-system-level encryption plus signing support

IPFS is p2p:

Worldwide peer-to-peer file transfers

Completely decentralized architecture

No central point of failure

IPFS has a name service:

IPNS, an SFS-inspired name system

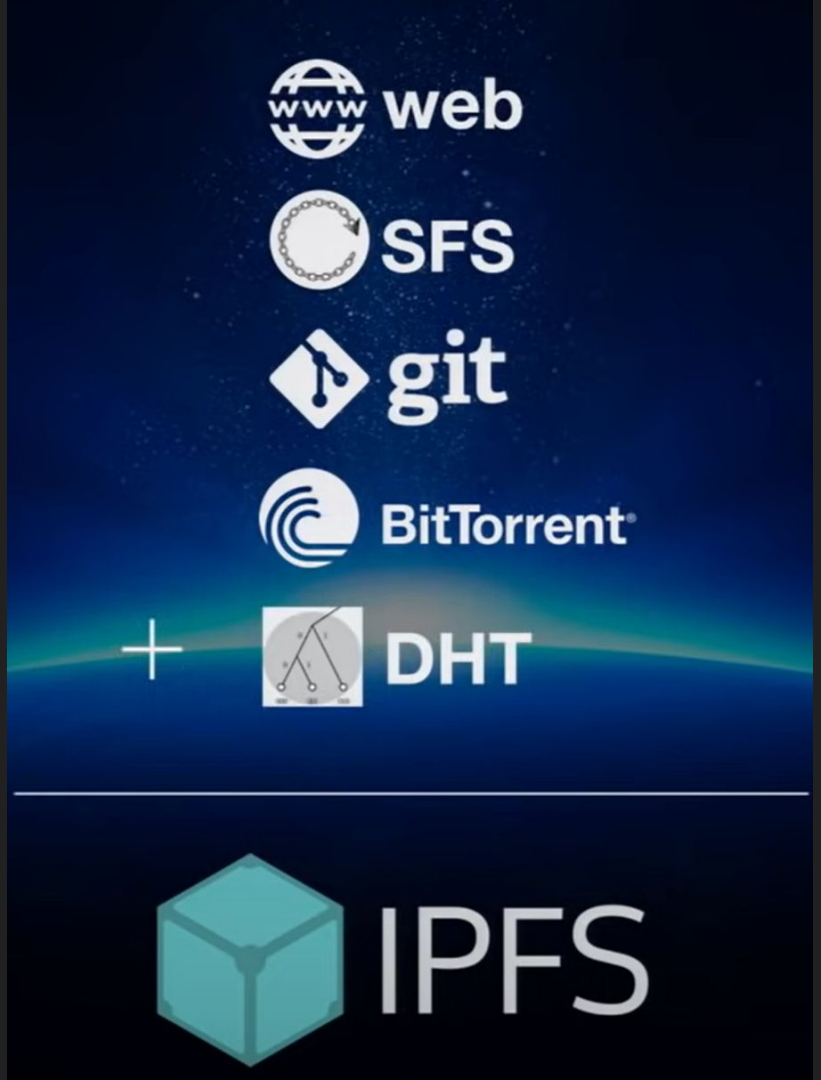
Global namespace based on PKI

It serves to build trust chains

It's compatible with other NSes

Can map DNS, .onion, .bit, etc to IPNS

How does it work?



[9]

How does it work?

There are three fundamental principles to understanding IPFS:

1. Unique identification via content addressing
2. Content linking via directed acyclic graphs (DAGs)
3. Content discovery via distributed hash tables (DHTs)

These three principles build upon each other to enable the IPFS ecosystem. Let's start with content addressing and the unique identification of content.

How does it work?

There are three fundamental principles to understanding IPFS:

1. **Unique identification via content addressing**
2. Content linking via directed acyclic graphs (DAGs)
3. Content discovery via distributed hash tables (DHTs)

These three principles build upon each other to enable the IPFS ecosystem. Let's start with content addressing and the unique identification of content.

Content Addressing

IPFS uses content addressing to identify content by what's in it rather than by where it's located.

Right now, content on the internet is found by location, such as:

https://en.wikipedia.org/wiki/Bitcoin

Or even on your local computer, like:

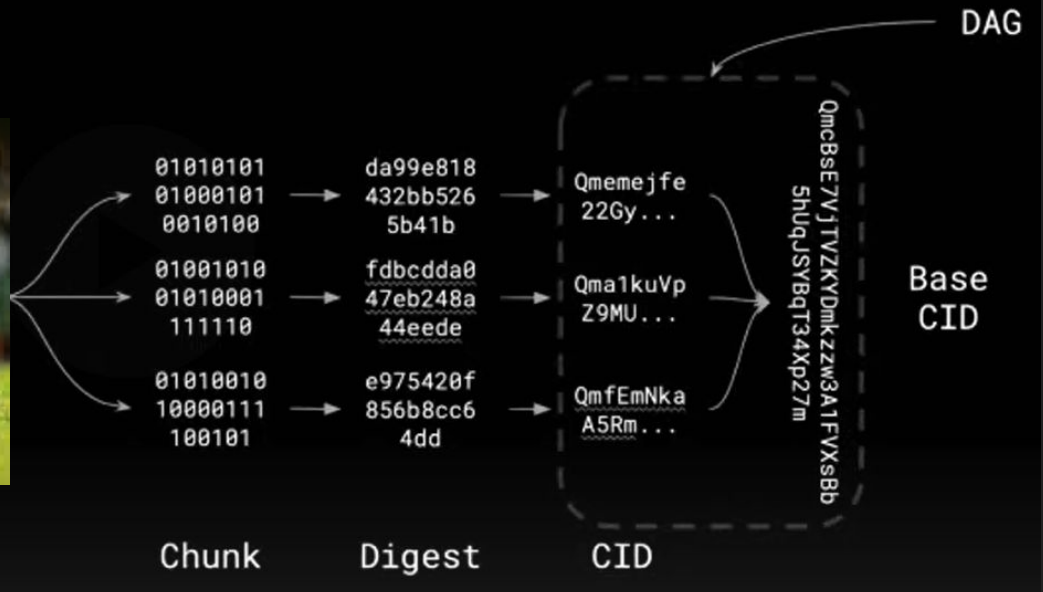
/home/rohith/docs/coms4507/ipfs.ppt

https://www.puppies.com/beagle.jpg

https://www.puppies.com/beagle.jpg

https://www.puppies.com/beagle.jpg

Content Addressing

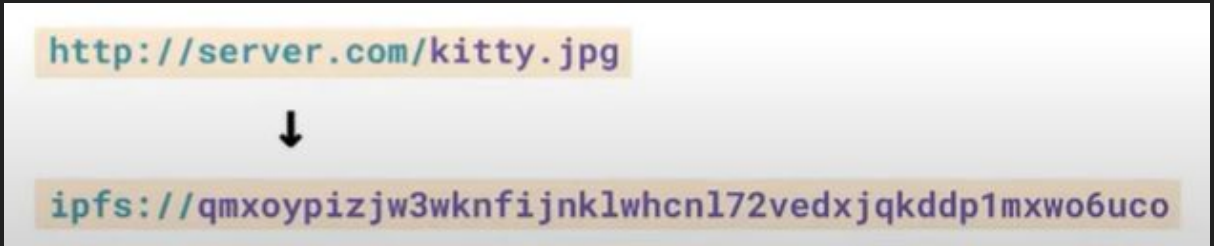


```
http://10.20.30.40/foo/bar/baz.png

/ipfs/QmW98pJrc6FZ6/foo/bar/baz.png
```

What happens you add a file to IPFS?

When you add a file to IPFS, your file is split into smaller chunks, cryptographically hashed, and given a unique fingerprint called a content identifier (CID). This CID acts as a permanent record of your file as it exists at that point in time.



[9]

Immutability

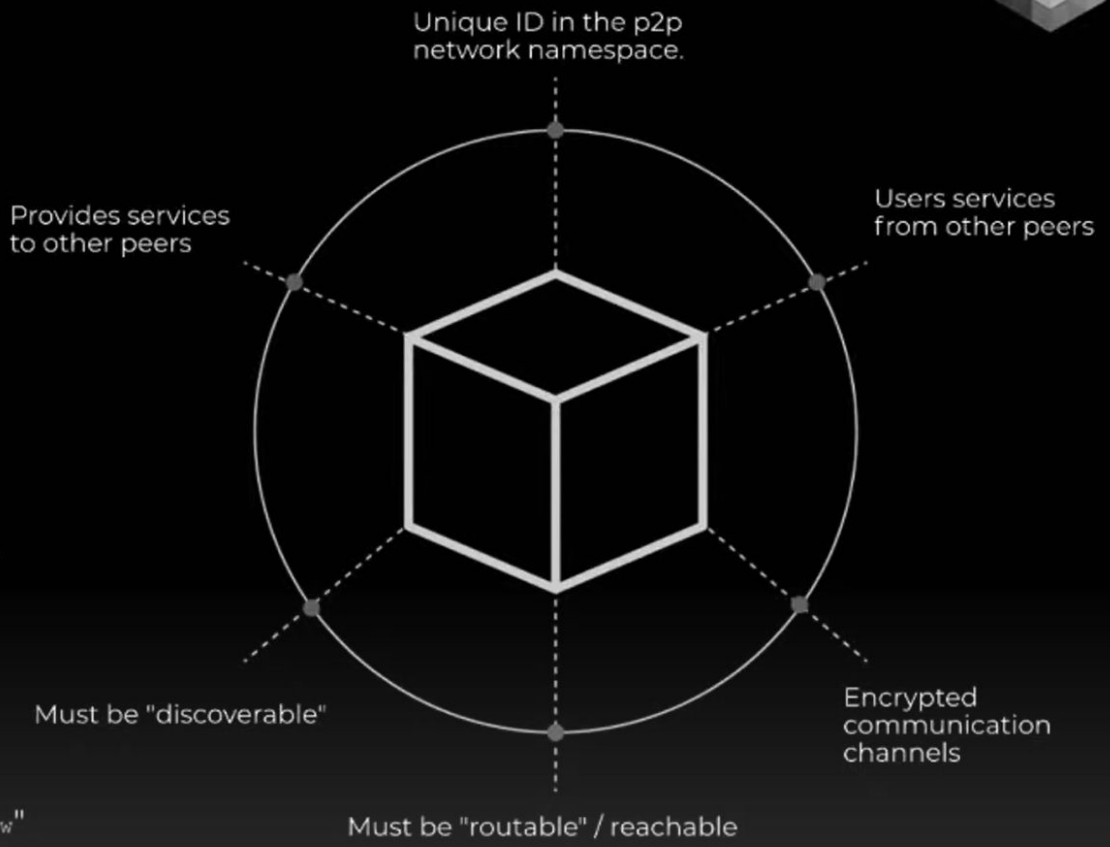
If you add a new version of your file to IPFS, its cryptographic hash is different, and so it gets a new CID. This means files stored on IPFS are resistant to tampering and censorship — any changes to a file don't overwrite the original, and common chunks across files can be reused in order to minimize storage costs.

Peer to Peer Networks

When other nodes look up your file, they ask their peer nodes who's storing the content referenced by the file's CID. When they view or download your file, they cache a copy — and become another provider of your content until their cache is cleared.

Content routing:

The Peer



The Peer is the unit of peer-2-peer systems.

Every peer uses a a cryptographic key pair (similar to HTTPs) for the purposes of:

- Identity: a unique name in the network: "`QmTuAM7RMnMqKnTq6qH1u9JiK5LqQvUxFdnrcM4aRHxeew`"
- Channel security (encryption)

[9]

What happens you add a file to IPFS?

A node can pin content in order to keep (and provide) it forever, or discard content it hasn't used in a while to save space. This means each node in the network stores only content it is interested in, plus some indexing information that helps figure out which node is storing what.

But what if you expect data to change?

IPNS uses public-key cryptography to produce a hash that links to an IPFS hash. This creates a system of updateable pointers to your immutable CIDs.

[9]

Do I need to remember a long string of CIDs?

No.

IPNS

DNSLink

How does it work?

There are three fundamental principles to understanding IPFS:

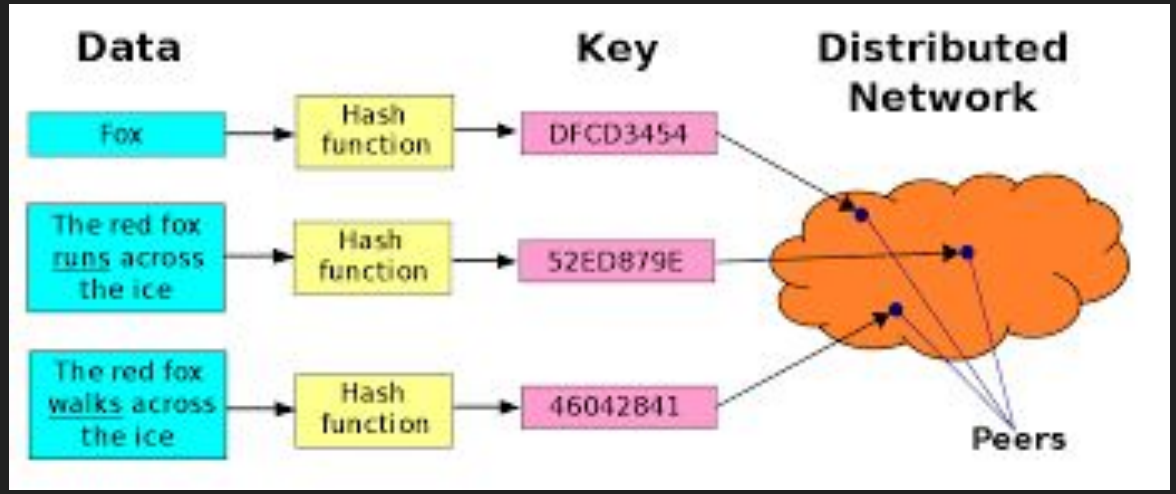
1. Unique identification via content addressing
2. **Content discovery via distributed hash tables (DHTs)**
3. Content linking via directed acyclic graphs (DAGs)

These three principles build upon each other to enable the IPFS ecosystem.

Distributed Hash Tables

A DHT is a dictionary like interface to data that is stored on nodes that are distributed across the network

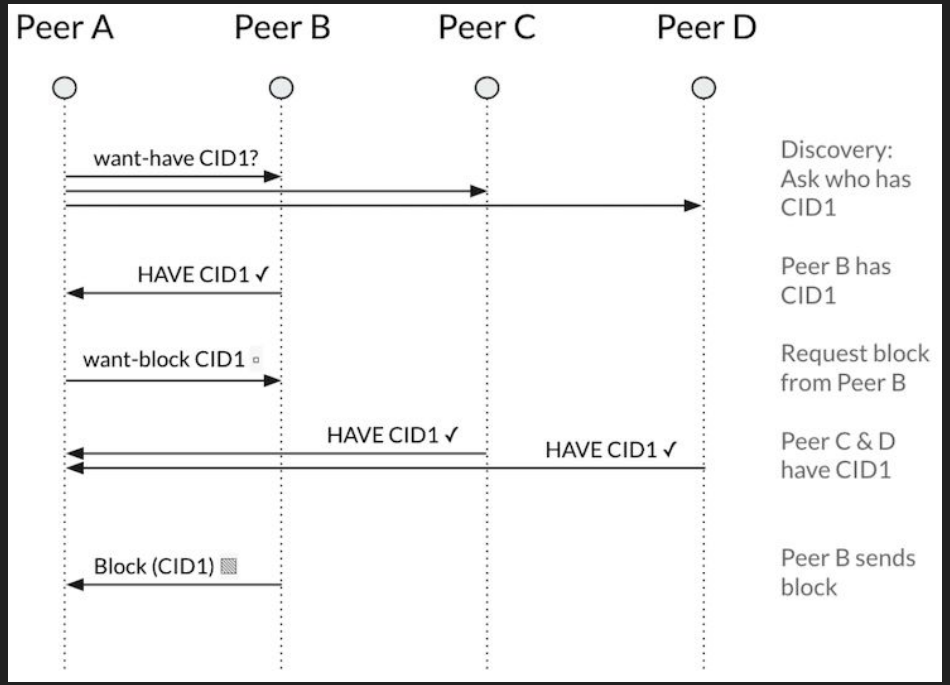
The node that gets to store a particular key is found by hashing that key.



BitSwap

```
Want-list {
  QmZtmD2qt6fJot32nabSP3CUjicnypEBz7bHVDhPQt9aAy, WANT,
  QmTudJSaoKxtbEnTddJ9vh8hbN84ZLVvD5pNpUaSbxwGoa, WANT,
  ...
}
```

Discovery


Peer A
Peer B
Peer C
Peer D
want-have CID1?
HAVE CID1 ✓
want-block CID1
HAVE CID1 ✓
HAVE CID1 ✓
Block (CID1)
Discovery: Ask who has CID1
Peer B has CID1
Request block from Peer B
Peer C & D have CID1
Peer B sends block

How does it work?

There are three fundamental principles to understanding IPFS:

1. Unique identification via content addressing
2. Content discovery via distributed hash tables (DHTs)
3. **Content linking via directed acyclic graphs (DAGs)**

These three principles build upon each other to enable the IPFS ecosystem.

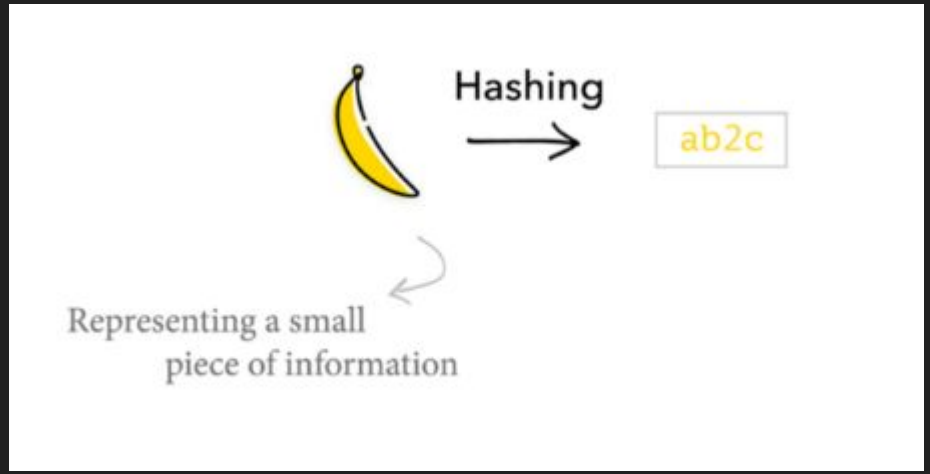
Merkle DAG

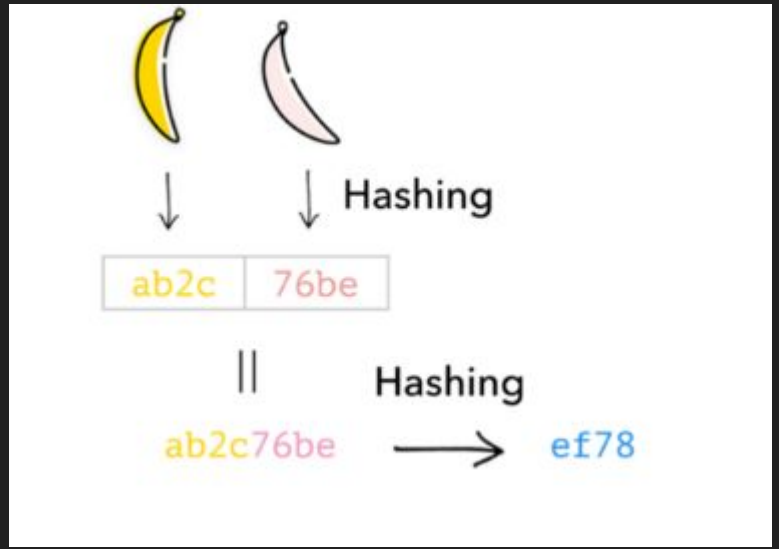
To give structure to the DHT and let users find the data they need when they need it IPFS uses a data structure called a Merkle DAG inspired by the git protocol.

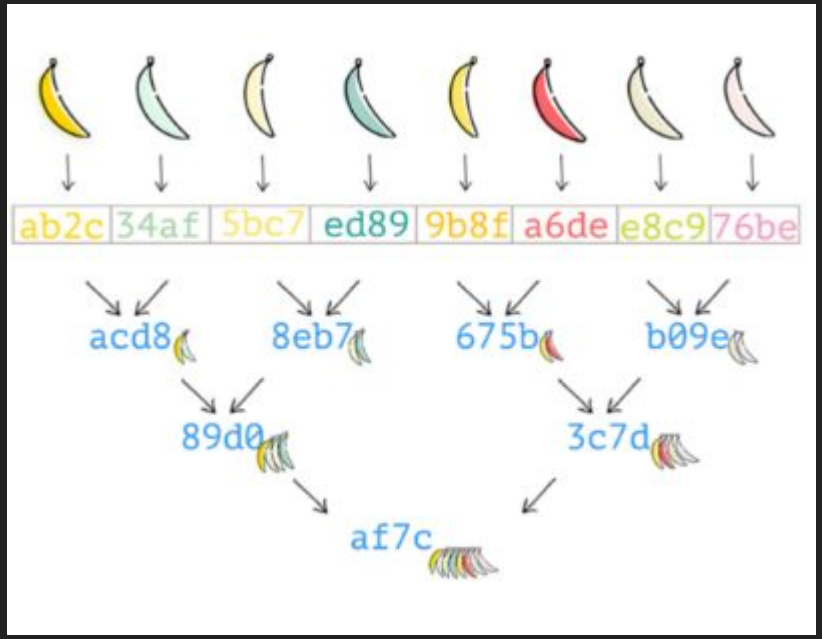
Git uses a Merkle DAG to enable distributed version control for source code and IPFS uses it to give structure to the entire network. It's simple and flexible.

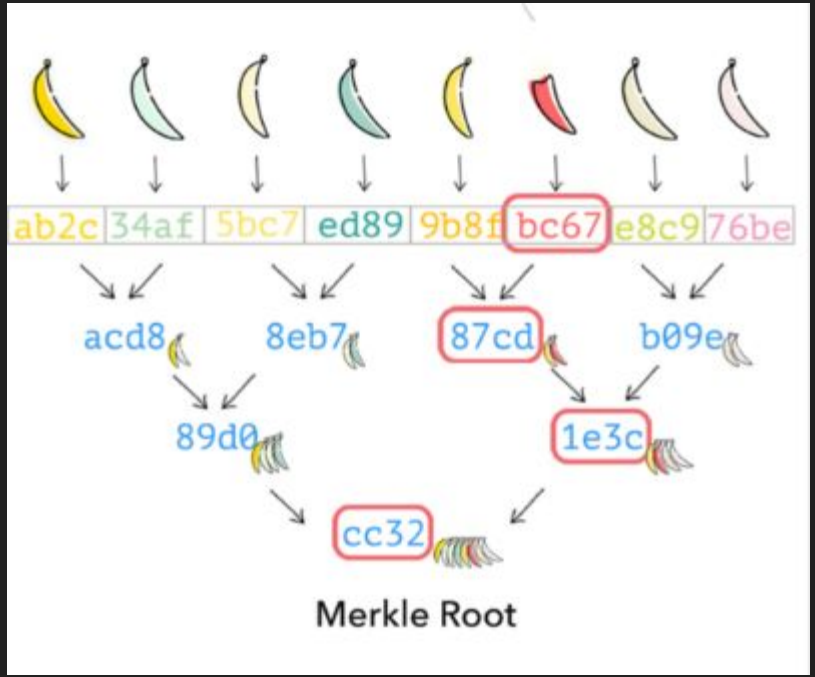
We can conceptualize it as a series of nodes connected together in the form of a directed acyclic graph.

It can look like a singly linked list or a tree so whenever data is added to the DHT the system generates a public/private key pair and the user gets both.

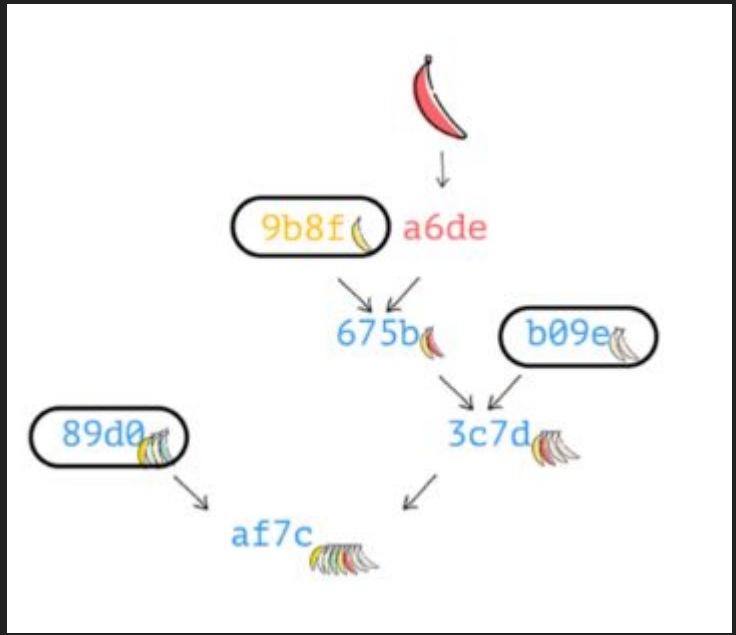

Hashing
ab2c
Representing a small
piece of information


Hashing
ab2c
76be
Hashing
ab2c76be
ef78


ab2c
34af
5bc7
ed89
9b8f
a6de
e8c9
76be
acd8
8eb7
675b
b09e
89d0
3c7d
af7c


ab2c
34af
5bc7
ed89
9b8f
bc67
e8c9
76be
acd8
8eb7
87cd
b09e
89d0
1e3c
cc32
Merkle Root

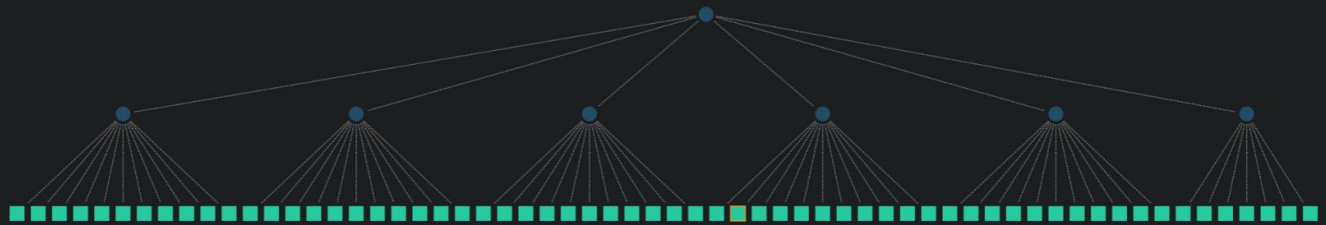

bc67
9b8f
b09e
89d0


9b8f
a6de
675b
b09e
89d0
3c7d
af7c

IPFS

DAG builder

CID v0 | Fixed 512 byte chunks | UnixFS leaves | Balanced DAG | 11 children max | Reset



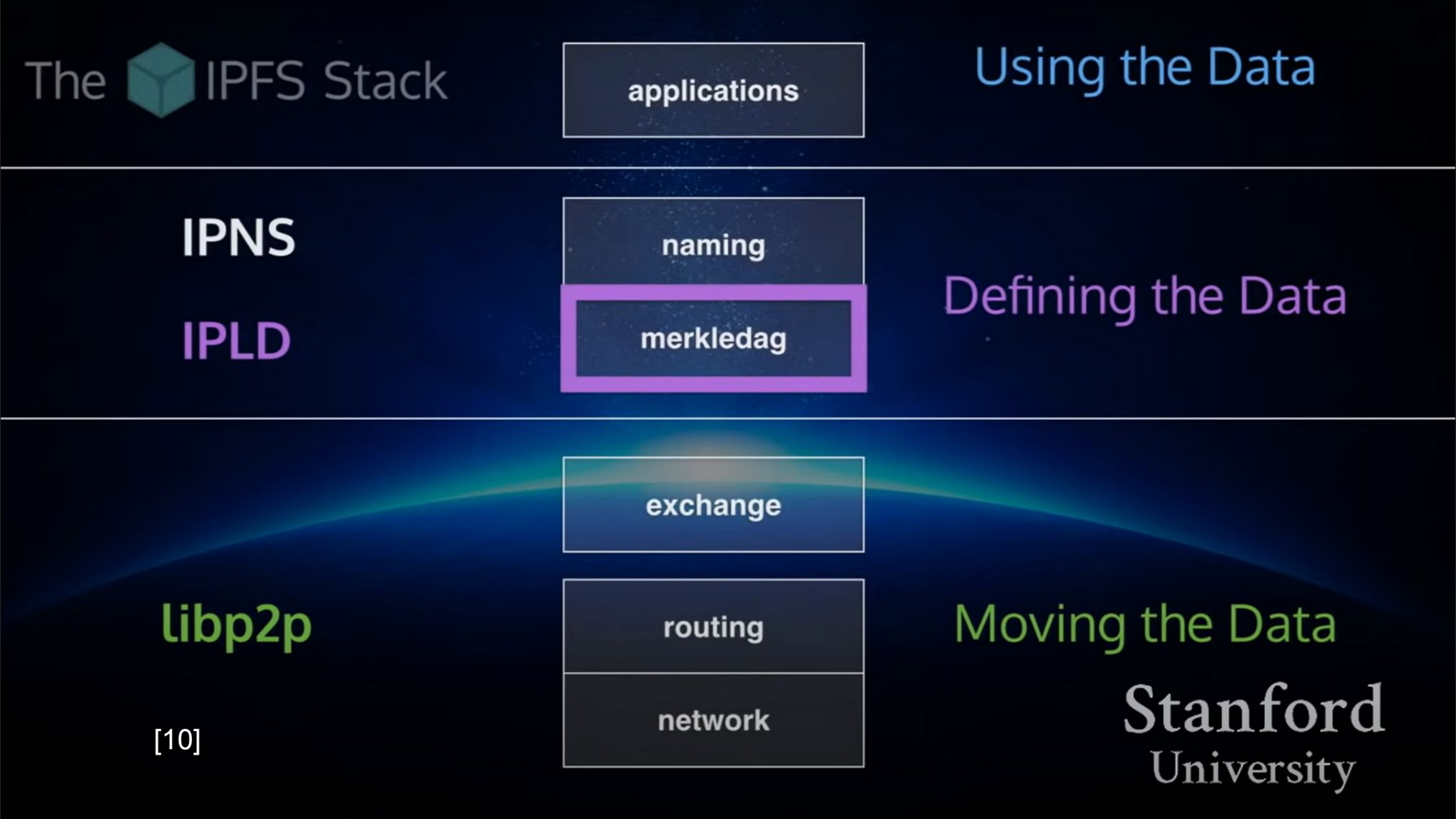
UnixFS File

CID
QmVdco2Nyo3BcHS1rNU58zn3GnDtFP9dBJLHaup87cZijR

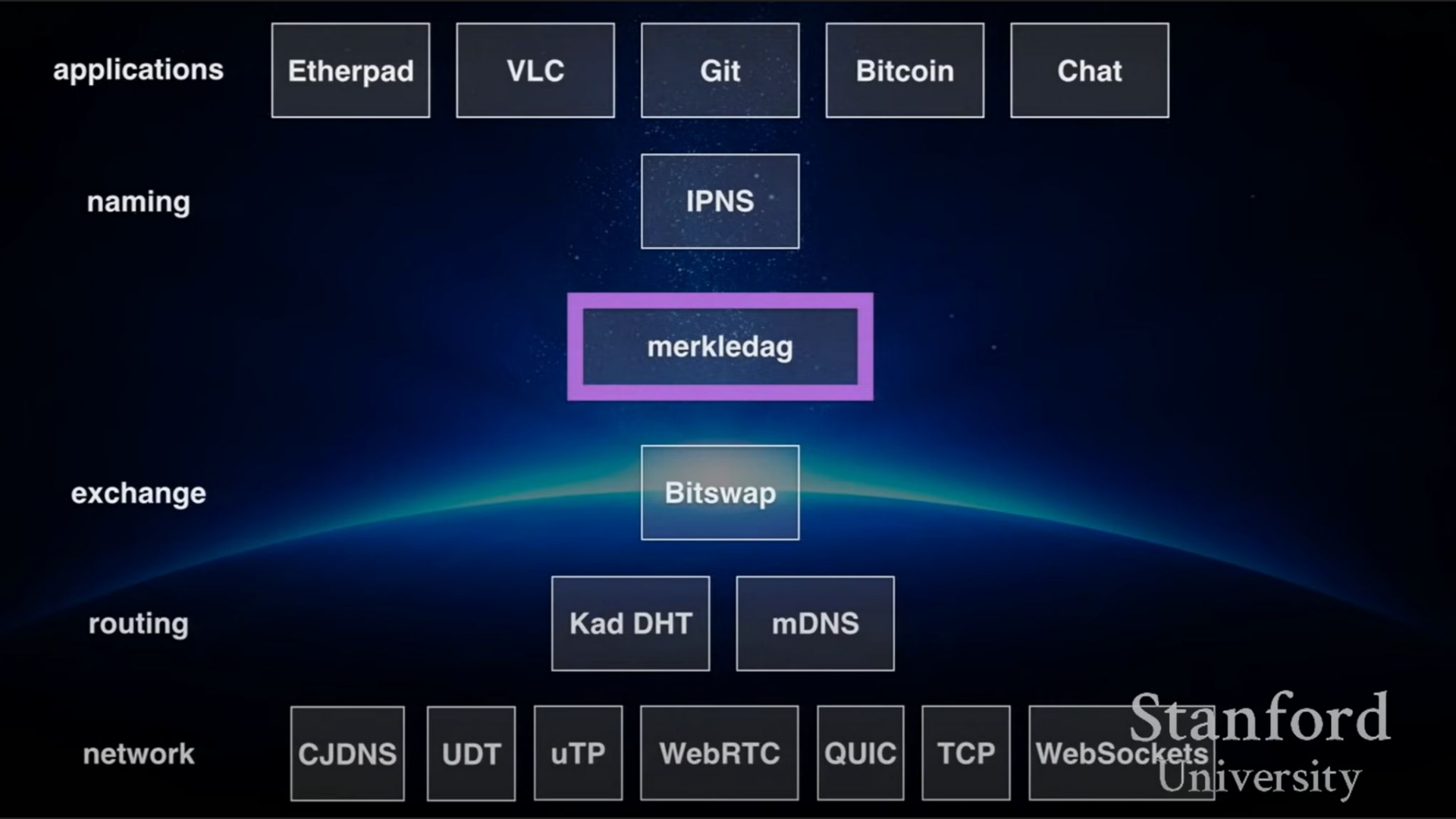
523 bytes total

512 bytes data

0 links

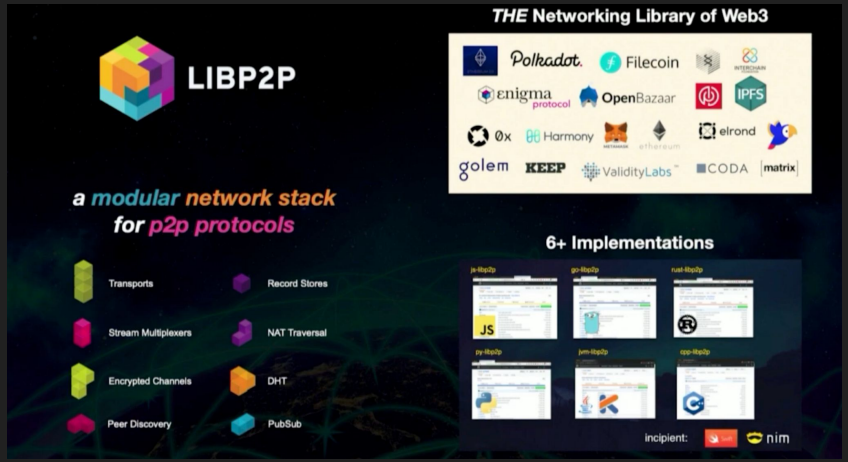

The IPFS Stack
applications
Using the Data
IPNS
IPLD
naming
merkledag
Defining the Data
exchange
libp2p
routing
network
Moving the Data
Stanford University

[10]


applications
Etherpad
VLC
Git
Bitcoin
Chat
naming
IPNS
merkledag
exchange
Bitswap
routing
Kad DHT
mDNS
network
CJDNS
UDT
uTP
WebRTC
QUIC
TCP
WebSockets
Stanford University

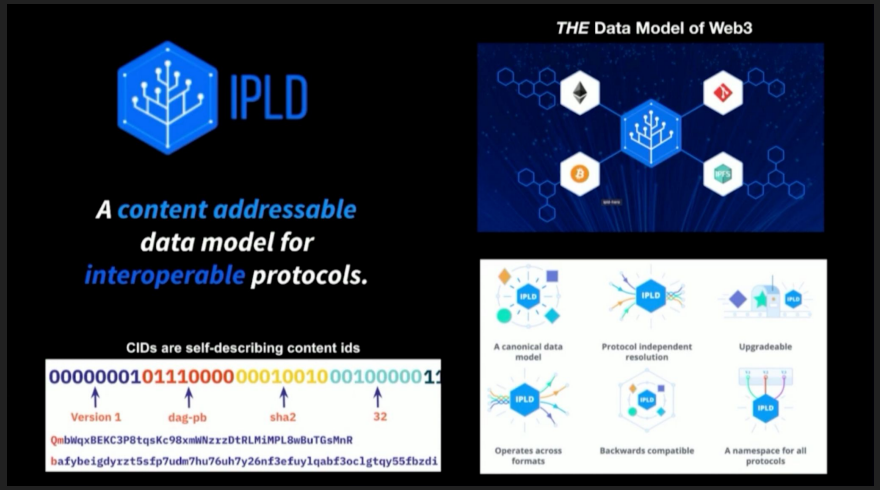
IPFS - Web 3.0

- IPFS surrounds other technologies around web3.0 - libP2P, IPLD
- libP2P - filecoin, Ethereum 2, polkadot
- IPLD - content addressable data model , CIDs
- Layers - work with any layer without using the layer above it.



IPFS - Web 3.0

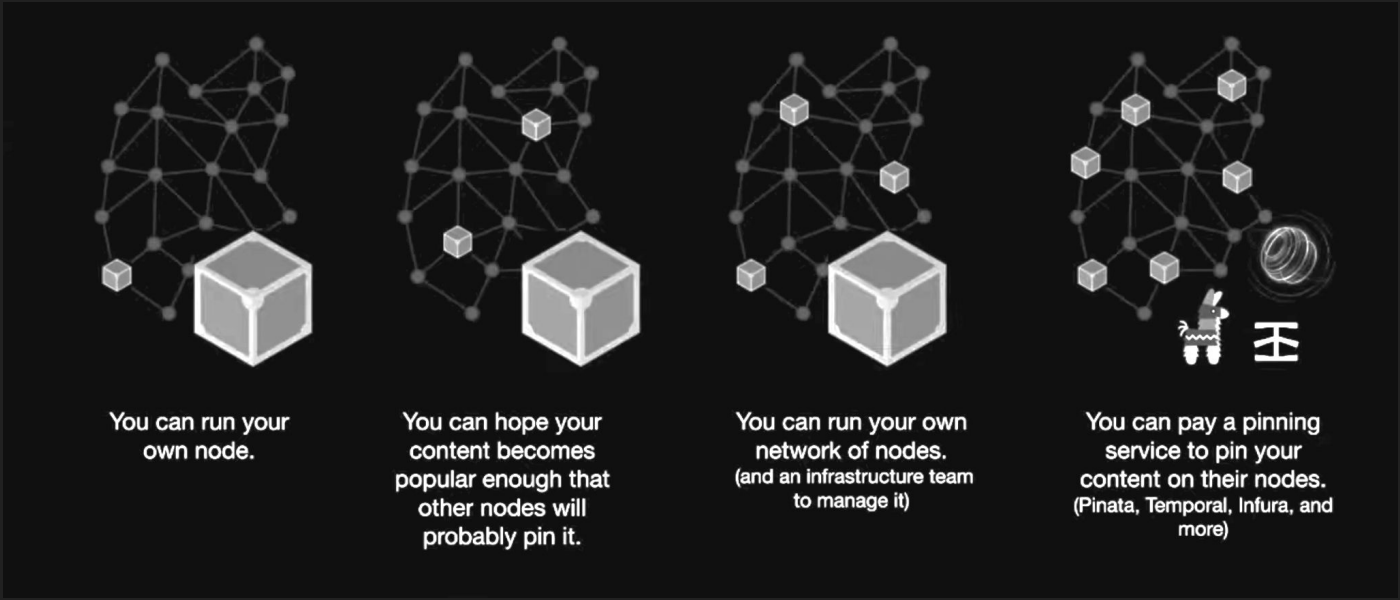
- IPFS surrounds other technologies around web3.0 - libP2P, IPLD
- libP2P - filecoin, Ethereum 2, polkadot
- IPLD - content addressable data model , CIDs
- Layers - work with any layer without using the layer above it.



IPFS - Web 3.0

- IPFS surrounds other technologies around web3.0 - libP2P, IPLD
- libP2P - filecoin, Ethereum 2, polkadot
- IPLD - content addressable data model , CIDs
- Layers - work with any layer without using the layer above it.

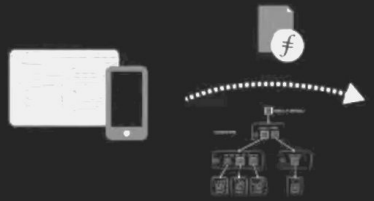
Using IPFS in the wild



Filecoin

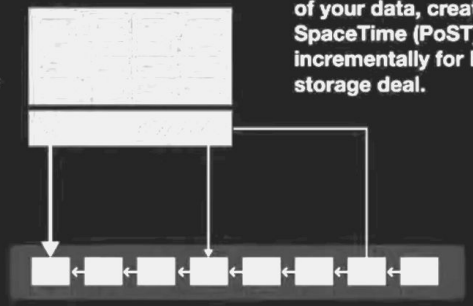
1 You (or your app) make a deal with a provider to store your data.

Filecoin preps your data by transforming it into a Directed Acyclic Graph (DAG). It uses the same process as IPFS, so the resulting CIDs are compatible.



2 Providers generate an initial Proof of Replication (PoRep). That's part of the storage deal that they publish to the Filecoin blockchain.

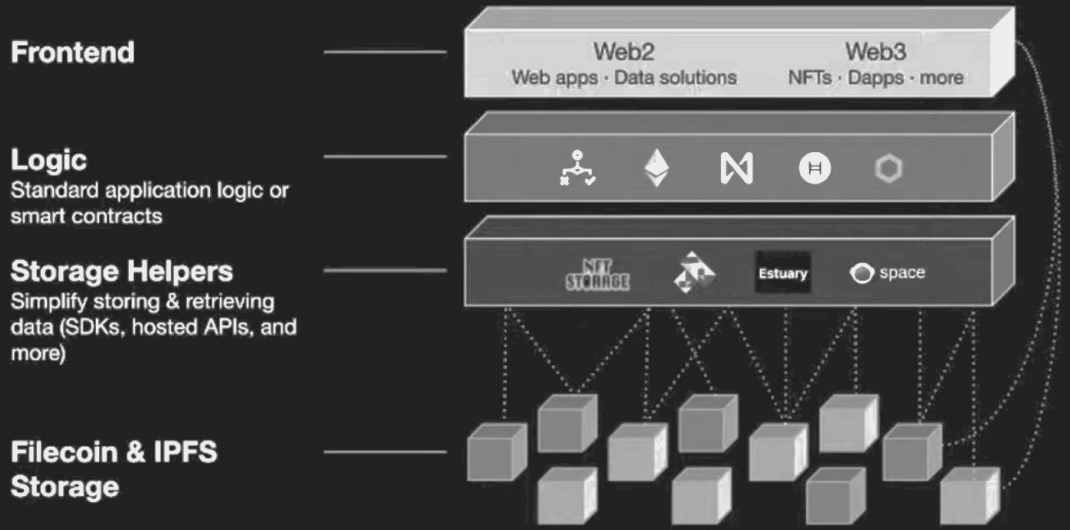
3 Over time, providers have to keep proving that they still have subsets of your data, creating Proofs of SpaceTime (PoST). They are paid incrementally for keeping up the storage deal.



4 Get your data by making retrieval deals with providers.

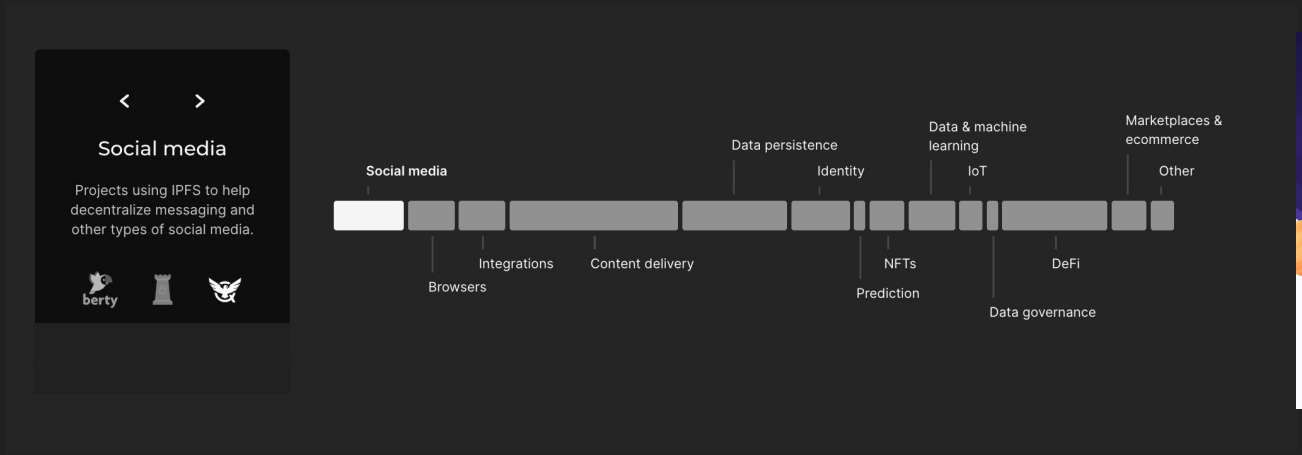
You can also check the chain for proof that it's being stored (and stored correctly) at any time.

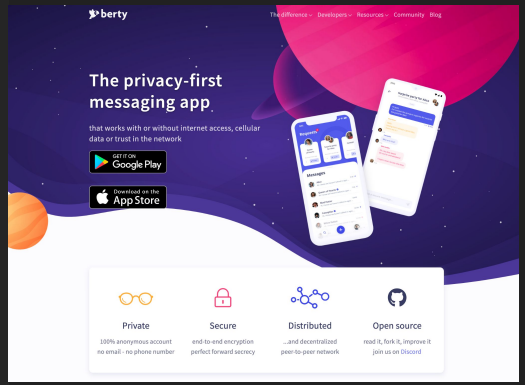
Web3 - Enabled Architectures



IPFS - Ecosystem

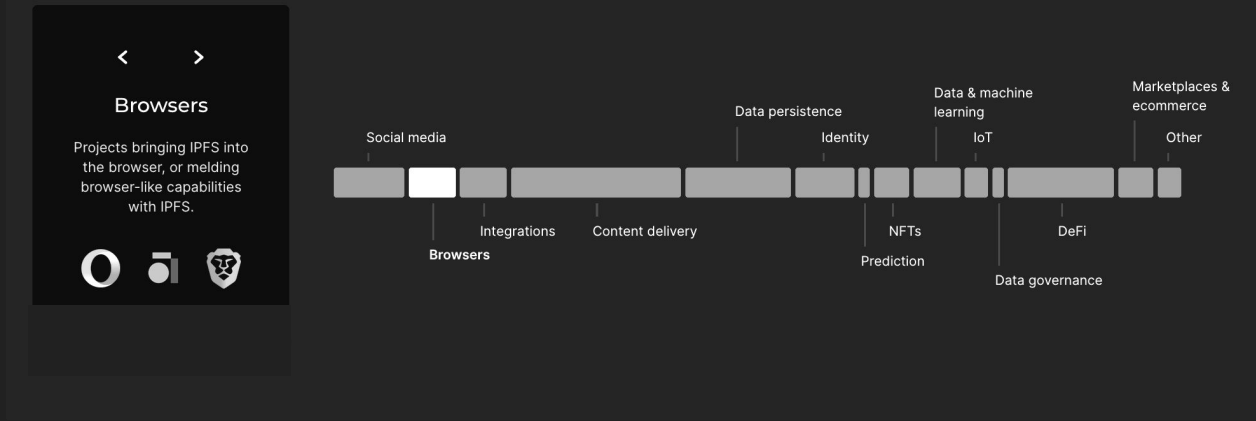
Social Media

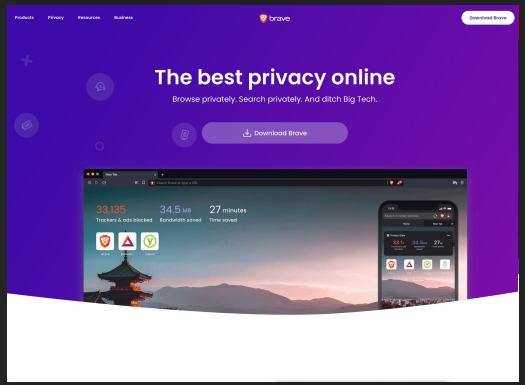




IPFS - Ecosystem

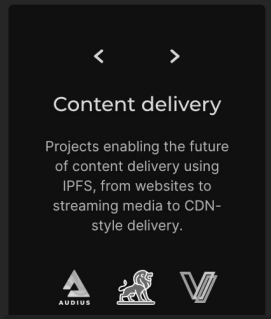
Browsers

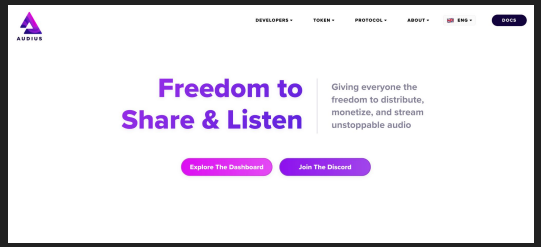


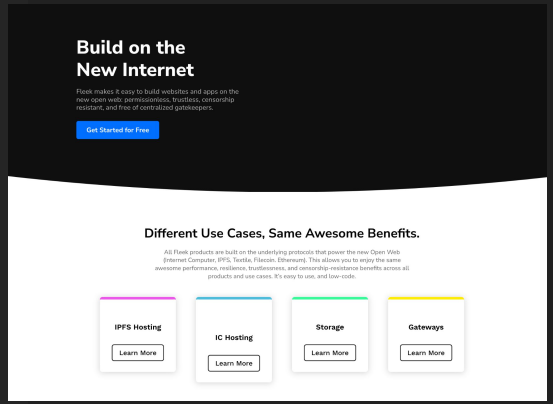


IPFS - Ecosystem

Content Delivery

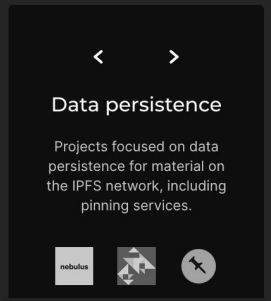


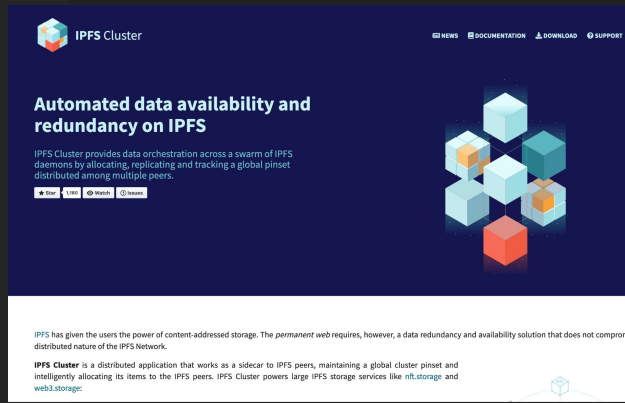




IPFS - Ecosystem

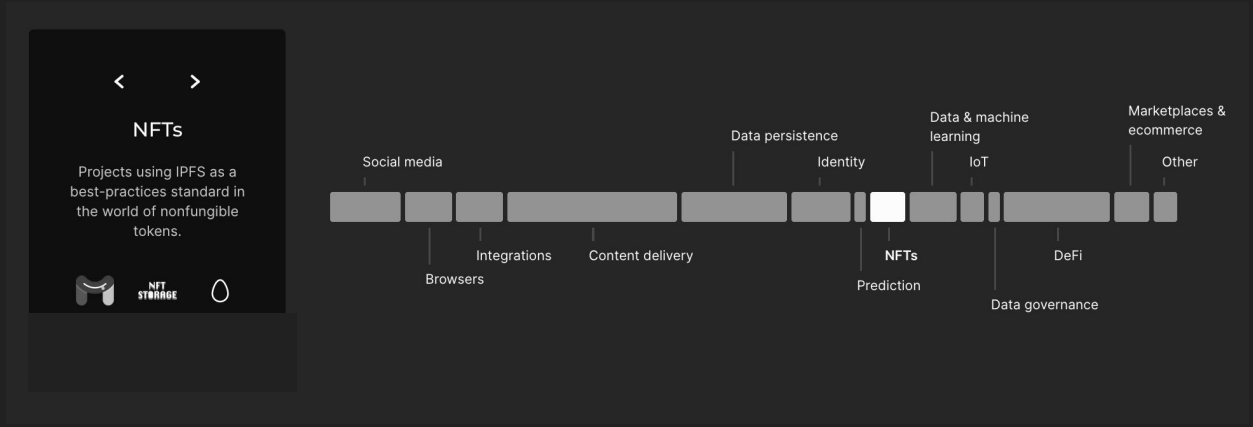
Data Persistence

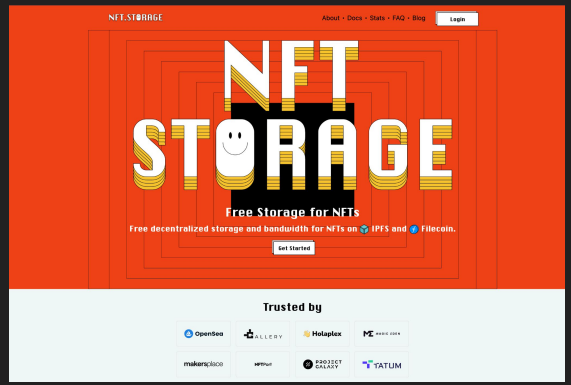




IPFS - Ecosystem

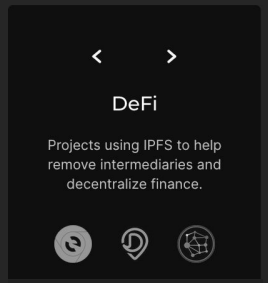
NFTs

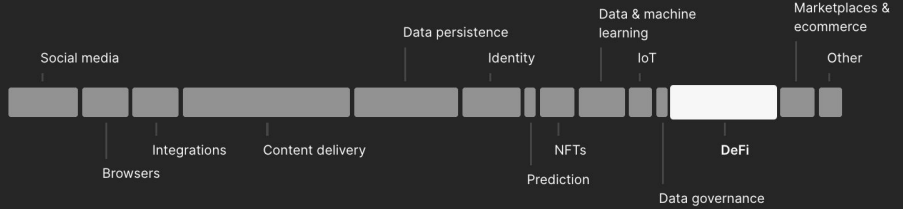


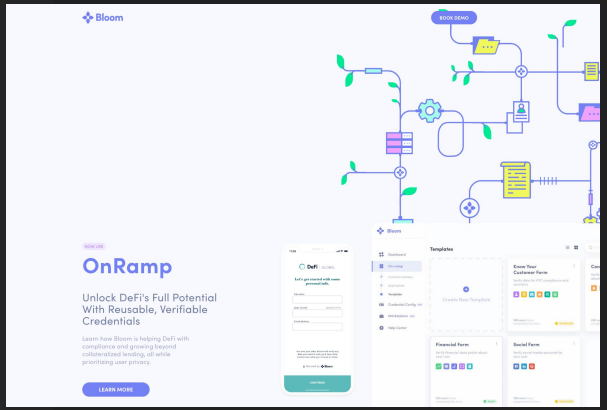


IPFS - Ecosystem

DeFi







IPFS - Today

Problems IPFS Addresses today -

- Censorship
- Offline use
- Content integrity
- Security

3.8 Using IPFS

IPFS is designed to be used in a number of different ways. Here are just some of the usecases I will be pursuing:

1. As a mounted global filesystem, under `/ipfs` and `/ipns`.
2. As a mounted personal sync folder that automatically versions, publishes, and backs up any writes.
3. As an encrypted file or data sharing system.
4. As a versioned package manager for *all* software.
5. As the root filesystem of a Virtual Machine.
6. As the boot filesystem of a VM (under a hypervisor).
7. As a database: applications can write directly to the Merkle DAG data model and get all the versioning, caching, and distribution IPFS provides.
8. As a linked (and encrypted) communications platform.
9. As an integrity checked CDN for large files (without SSL).
10. As an encrypted CDN.
11. On webpages, as a web CDN.
12. As a new Permanent Web where links do not die.

The IPFS implementations target:

(a) an IPFS library to import in your own applications.

(b) commandline tools to manipulate objects directly.

(c) mounted file systems, using FUSE [?] or as kernel modules.

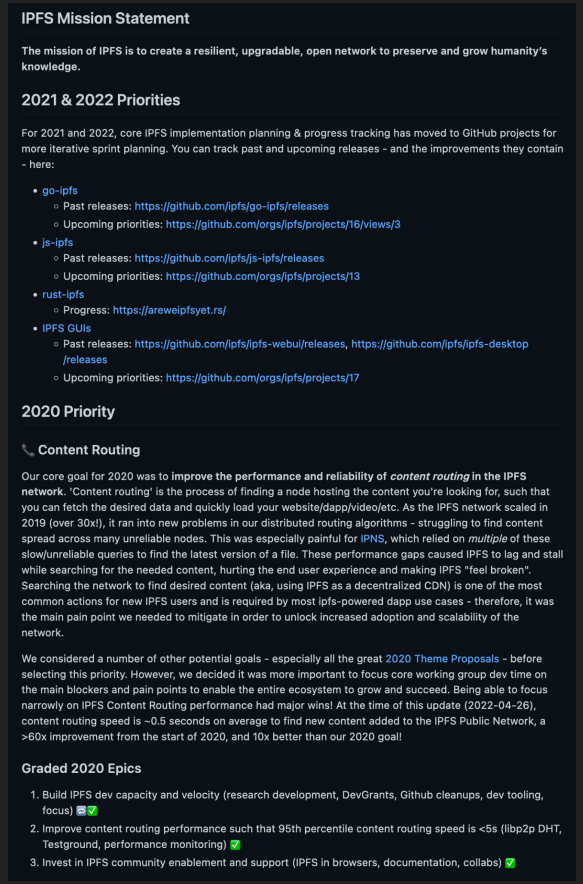
Initial IPFS draft use cases [1]

Challenges with IPFS

- Lack of strong economic incentives
- Unreliable with private data
- Lack of on-chain proofs

The Future of IPFS - Roadmap

- For 2020, made content routing a priority.
- Future goals - distributing large files easily, Focusing on go and javascript implementations, and creating a self archiving web.



IPFS Mission Statement

The mission of IPFS is to create a resilient, upgradable, open network to preserve and grow humanity's knowledge.

2021 & 2022 Priorities

For 2021 and 2022, core IPFS implementation planning & progress tracking has moved to GitHub projects for more iterative sprint planning. You can track past and upcoming releases - and the improvements they contain - here:

- go-ipfs
  - Past releases: https://github.com/ipfs/go-ipfs/releases
  - Upcoming priorities: https://github.com/orgs/ipfs/projects/16/views/3
- js-ipfs
  - Past releases: https://github.com/ipfs/js-ipfs/releases
  - Upcoming priorities: https://github.com/orgs/ipfs/projects/13
- rust-ipfs
  - Progress: https://areweipfsyet.rs/
- IPFS GUIs
  - Past releases: https://github.com/ipfs/ipfs-webui/releases, https://github.com/ipfs/ipfs-desktop/releases
  - Upcoming priorities: https://github.com/orgs/ipfs/projects/17

2020 Priority

📞 Content Routing

Our core goal for 2020 was to **improve the performance and reliability of *content routing* in the IPFS network**. 'Content routing' is the process of finding a node hosting the content you're looking for, such that you can fetch the desired data and quickly load your website/dapp/video/etc. As the IPFS network scaled in 2019 (over 30x!), it ran into new problems in our distributed routing algorithms - struggling to find content spread across many unreliable nodes. This was especially painful for IPNS, which relied on *multiple* of these slow/unreliable queries to find the latest version of a file. These performance gaps caused IPFS to lag and stall while searching for the needed content, hurting the end user experience and making IPFS "feel broken". Searching the network to find desired content (aka, using IPFS as a decentralized CDN) is one of the most common actions for new IPFS users and is required by most ipfs-powered dapp use cases - therefore, it was the main pain point we needed to mitigate in order to unlock increased adoption and scalability of the network.

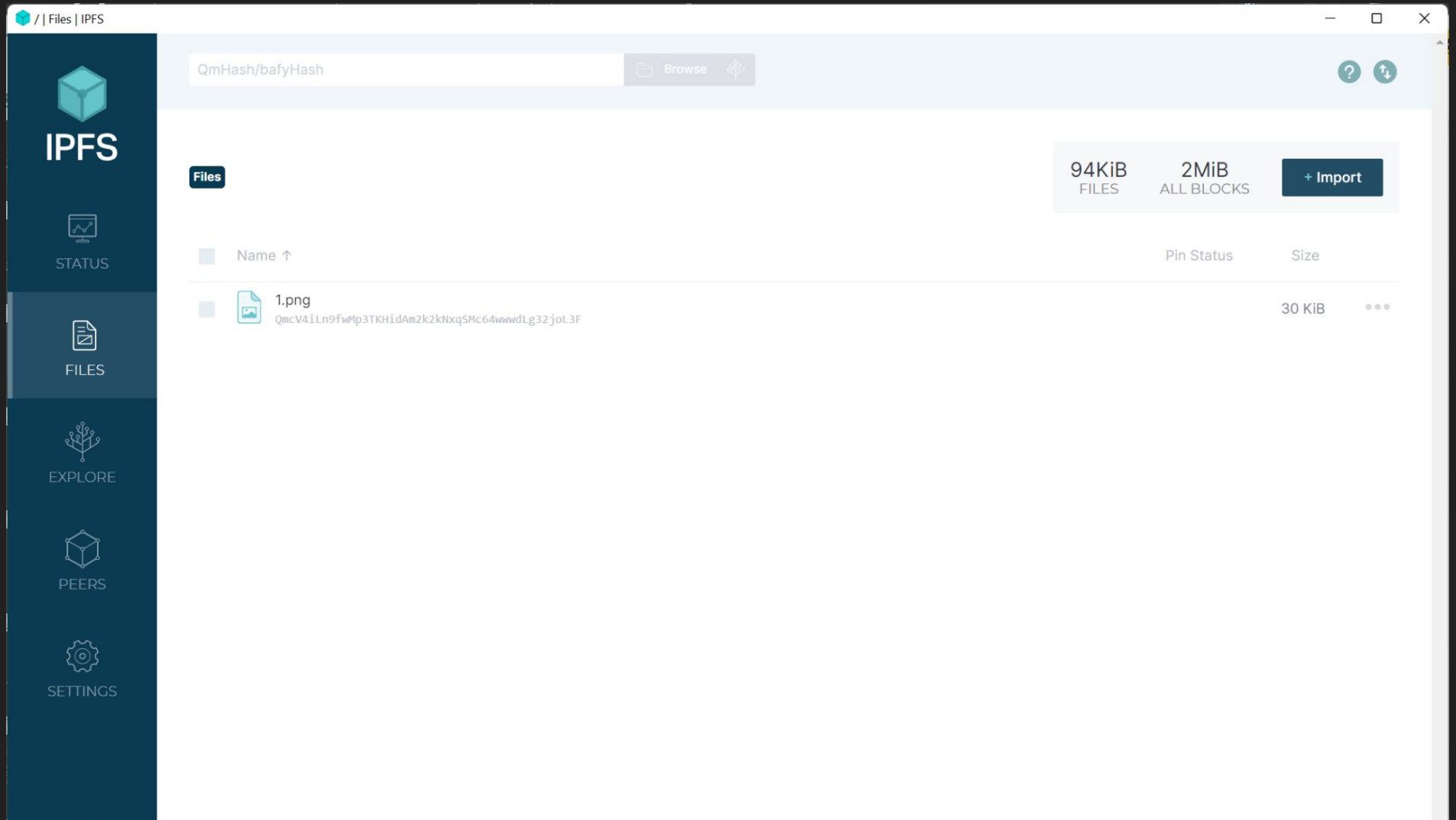
We considered a number of other potential goals - especially all the great 2020 Theme Proposals - before selecting this priority. However, we decided it was more important to focus core working group dev time on the main blockers and pain points to enable the entire ecosystem to grow and succeed. Being able to focus narrowly on IPFS Content Routing performance had major wins! At the time of this update (2022-04-26), content routing speed is ~0.5 seconds on average to find new content added to the IPFS Public Network, a >60x improvement from the start of 2020, and 10x better than our 2020 goal!

Graded 2020 Epics

1. Build IPFS dev capacity and velocity (research development, DevGrants, Github cleanups, dev tooling, focus) ✅
2. Improve content routing performance such that 95th percentile content routing speed is <5s (libp2p DHT, Testground, performance monitoring) ✅
3. Invest in IPFS community enablement and support (IPFS in browsers, documentation, collabs) ✅

Conclusion

- At its core, IPFS is still a global versioned filesystem.
- Push the web, towards a more democratic internet.
- Publishing information needs no publisher
- Users can trust the content without trusting the peers
- Content lives on forever
- Towards a more Permanent Web


/ | Files | IPFS
IPFS
QmHash/bafyHash
Browse
STATUS
FILES
EXPLORE
PEERS
SETTINGS
Files
94KiB
FILES
2MiB
ALL BLOCKS
+ Import
Name ↑
Pin Status
Size
1.png
QmcV4iLn9fwMp3TKHidAm2k2kNxqSMc64wwwdLg32joL3F
30 KiB

References

[1] J. Benet. “IPFS - Content Addressed, Versioned, P2P File System”, *(Draft 3).*
[Online]. Available: https://docs.ipfs.io/concepts/further-reading/academic-papers/ (accessed May 21, 2022).

[2] Academic Kids. “FastTrack” Academickids.com.
https://academickids.com/encyclopedia/index.php/FastTrack (accessed May 21, 2022).

[3] Kaspersky. “What Is BitTorent and Is It Safe?” Kaspersky.com.
https://www.kaspersky.com/resource-center/definitions/bittorrent (accessed May 21, 2022).

[4] M. Leiva-Gomez. “MTE Explains: How BitTorrent DHT Peer Discovery Works” Maketecheasier.com.
https://www.maketecheasier.com/how-bittorrent-dht-peer-discovery-works/ (accessed May 21, 2022).

[5] J. Loeliger and M. McCullough, “Chapter 4. Basic Git Concepts”, in *Version Control with Git, 2nd Edition*. US: O'Reilly Media, Inc. [Online]. Available:
https://www.oreilly.com/library/view/version-control-with/9781449345037/ch04.html (accessed May 21, 2022).

[6] Git. “About” Git-scm.com.
https://git-scm.com/about/branching-and-merging (accessed May 21, 2022).

[7] Academic Kids. “UUHash” Academickids.com.
https://academickids.com/encyclopedia/index.php/UUHash (accessed May 21, 2022).

[8] Git. “10.2 Git Internals - Git Objects” Git-scm.com.
https://git-scm.com/book/en/v2/Git-Internals-Git-Objects (accessed May 21, 2022).

[9] IPFS. “IPFS - Documentation” docs.ipfs.io.

https://docs.ipfs.io (accessed May 21, 2022).

[10] “Stanford Seminar - IPFS and the Permanent Web”

https://www.youtube.com/watch?v=HUVmypx9HGI (Accessed May 21, 2022).