How IPFS Works

A High-Level Overview of the InterPlanetary File System

Yiannis Psaras (@yiannisbot)
Protocol Labs - ResNetLab

original deck by @stebalien

Who am I:
Yiannis Psaras

- I work at Protocol Labs...

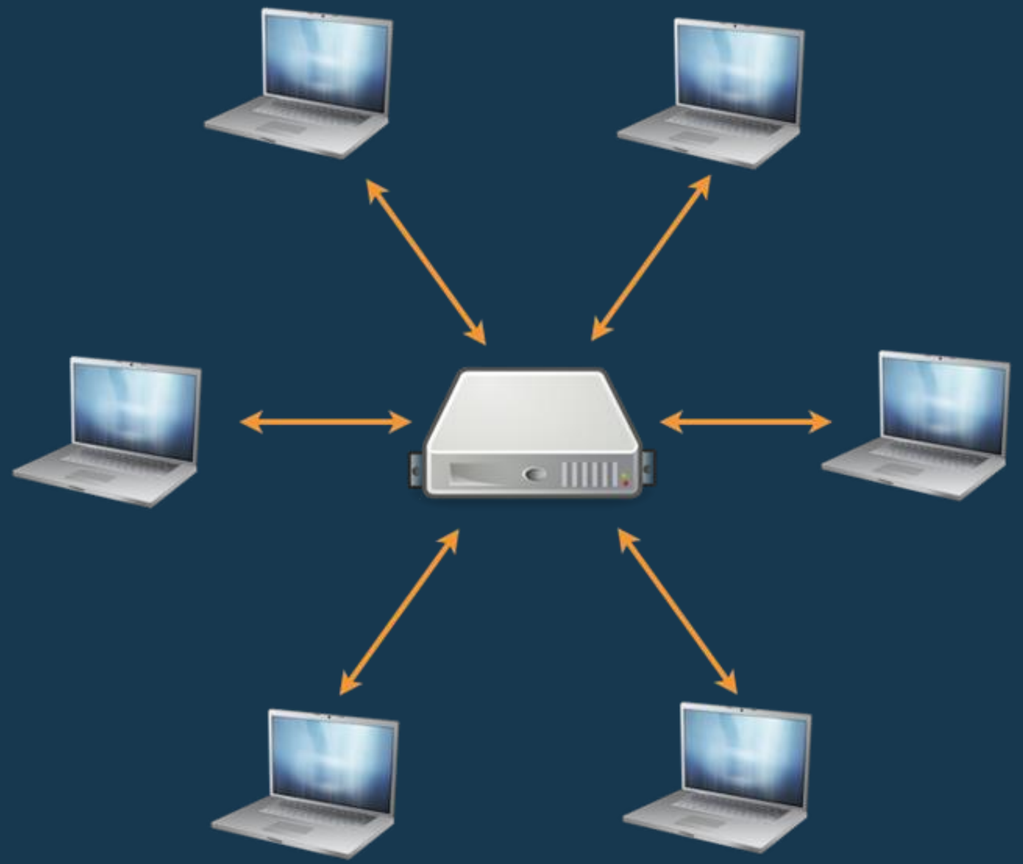
- ... on just a few of the IPFS Ecosystem Projects

IPFS libp2p IPLD Multiformats Cluster

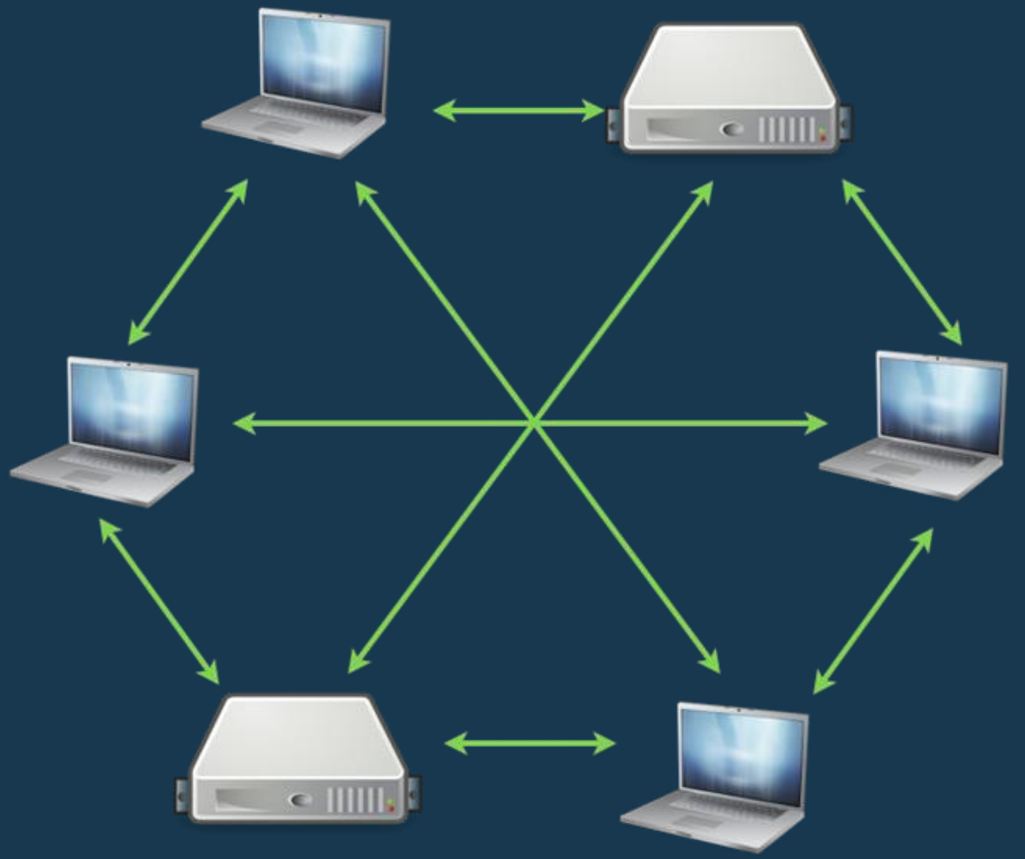
IPFS is a ***decentralized storage and delivery network*** which builds on fundamental principles of *P2P networking* and *content-based addressing*

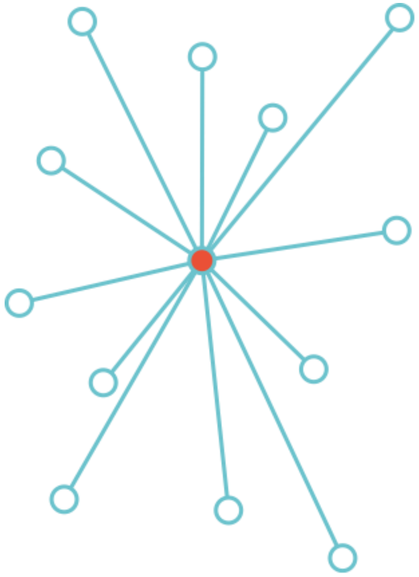
IPFS makes the web work peer-to-peer

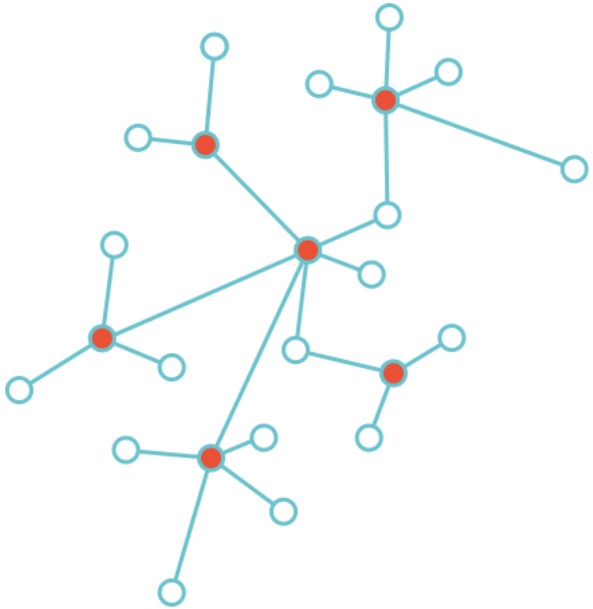
HTTP

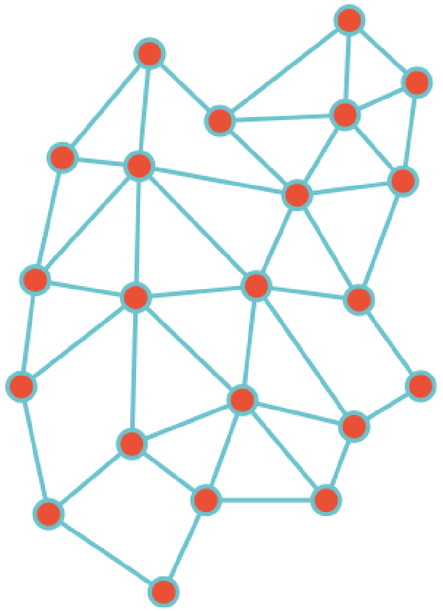


IPFS





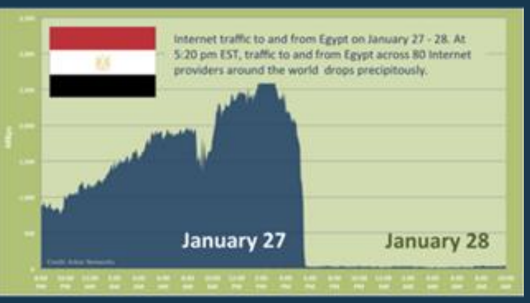




CENTRALIZED **DECENTRALIZED** **DISTRIBUTED**

Problems

emerging networks



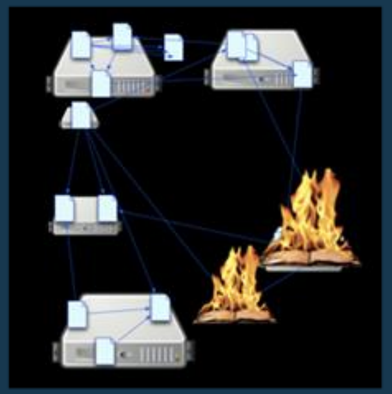
censorship

IPFS

Addresses

huge inefficiency

bad security model

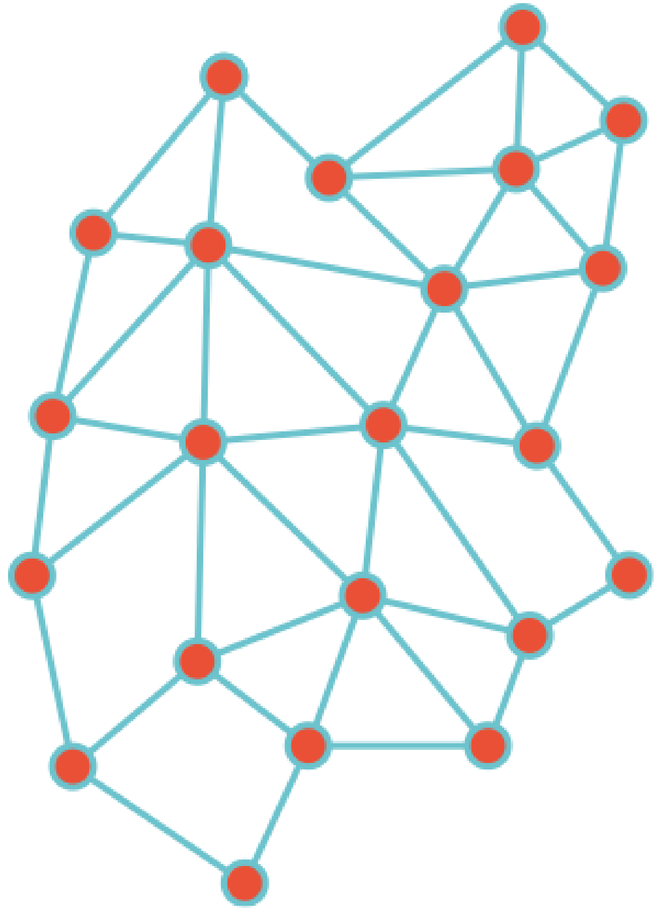


links break

no offline use

WHY DISTRIBUTED?

- **Resilience / Offline-first**
- **Speed**
- **Scalability**
- **Security**
- **Efficiency**
- **Trustless**



domain name /dns/example.com/foo/bar/baz.png

content address /ipfs/QmW98pJrc6FZ6/foo/bar/baz.png

THE IPFS STACK

IPFS is the result of combining multiple blocks commonly used to build distributed applications into a distributed-storage application.

IPFS uses libp2p, IPLD and Multiformats to provide content-addressed decentralized storage.

LIBP2P

libp2p is the peer-2-peer network-layer stack that supports IPFS. It takes care of host addressing, content and peer discovery through protocols and structures such as DHT and pubsub.

IPLD

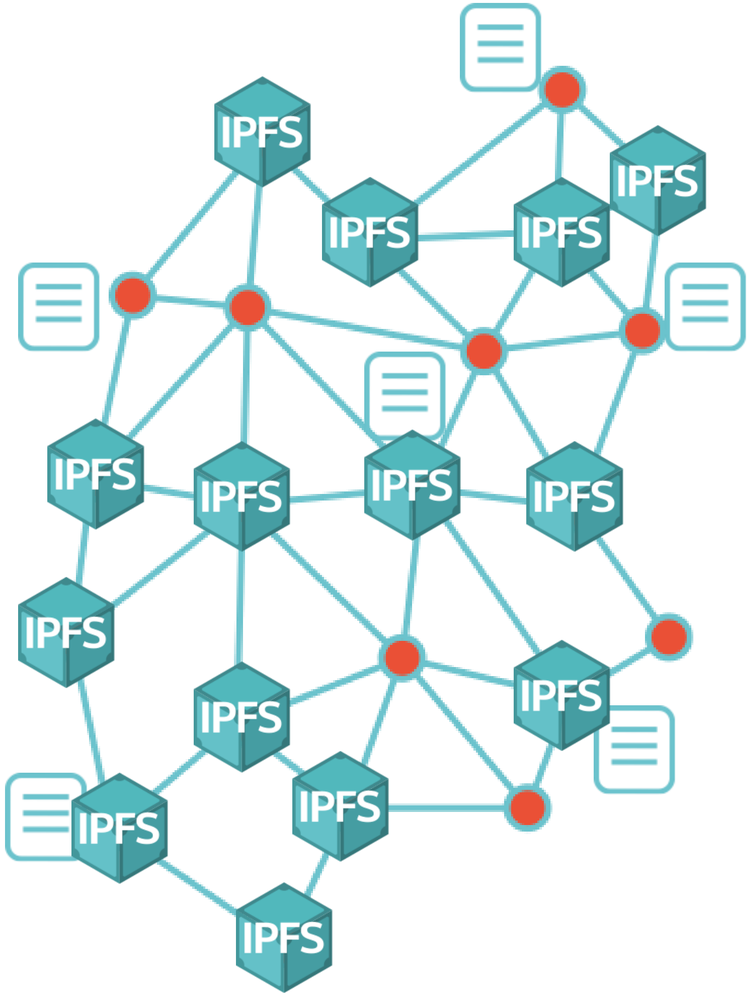
IPLD (InterPlanetary Linked Data) provides standards and formats to build Merkle-DAG data-structures, like those that represent a filesystem.

Multiformats

Multiformats provides formatting structures for self-describing values. These values are useful both to the data layer (IPLD) and to the network layer (libp2p)

KEY FACTS

- All content authenticated
- No central server - all peers are the same
- Content is never pushed to a different peer when adding it, only downloaded upon request.
- Content can be anything, from scientific datasets to blockchains.



IPFS

Data
orbit db
qri.io
arbore
Catena BLOCKCHAIN EXPLORER

Identity
civic
ION
NOMIOS
ZINC
handshake

Productivity
berty
PeerPad
PEERGOS

Social Media
3BOX
AKASHA
Indorse
Identifi
KARMA
Peepeth
busy

Integrations & Collabs
openSUSE
Guix
NETFLIX
CLOUDFLARE

Prediction and betting
AUGUR
CryptoBets
PLAY 2 WIN
VIRTUE POKER
Crypto Dice
MÖBIUS 2D

Marketplace
OpenBazaar
ORIGIN
BountyOx
Ethlance
NAME BAZAAR ENS ETH
PORTION

NFT
dlux DECENTRALIZED LIMITLESS USER EXPERIENCES
DIGITAL ART CHAIN
Decentraland
Glossy
mokens

Governance
GovBlocks
Democracy Earth
ARAGON ONE

Exchange
Dether
faa.st
Swap .online

Finance
REQUEST NETWORK
Bloom
colu.
SETTLE
MARKETPROTOCOL

Content
D.tube
EVERIPEDIA
BitTube
UNSTOPPABLE DOMAINS
ALEXANDRIA
dlive
Matters
Viewly
VIULY.COM
DSOUND
AUDIUS

Other
textile
MÓIBIT
#KarmaPay
TEMPORAL
SprtsHub
Simple As Water
ipwb
WINGS
adchain
MONITOR CHAIN
IPSE
Dappkit
IKU
点对点科技 DECENTRAL TECH
kauri.io
FILESTORM
INFURA

IPFS:

Lifecycle

Adding Files

Getting Files

IPFS:

Lifecycle

Adding Files

Getting Files

Content
Producer / Publisher

Content
Consumer

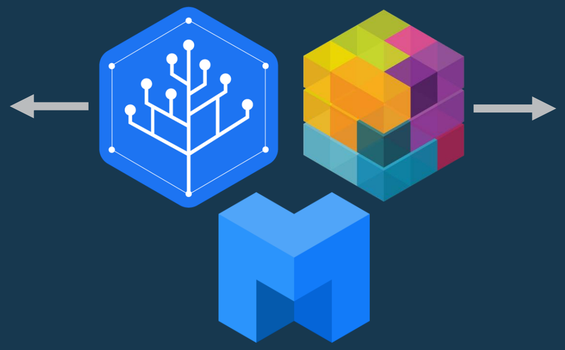
Import → **Name** → **Find** → **Fetch**

Chunking
UnixFS
IPLD

CID
Path
IPNS

Routing
DHT
Kademlia

Bitswap



Import | **Name** | **Find** | **Fetch**

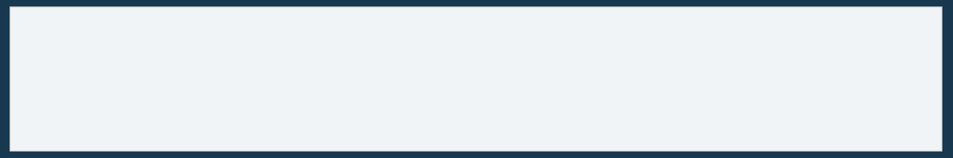
Chunking
UnixFS
IPLD

CID
Path
IPNS

Routing
DHT
Kademlia

Bitswap

Contiguous File:



Chunked File:

(each chunk is hashed)

- Deduplication
- Piecewise Transfer
- Seeking

Import | **Name** | **Find** | **Fetch**

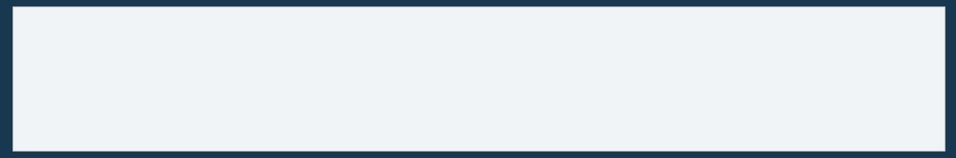
Chunking
UnixFS
IPLD

CID
Path
IPNS

Routing
DHT
Kademlia

Bitswap

Contiguous File:



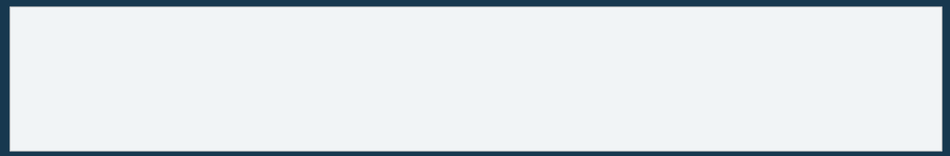
Chunked File:

- **Deduplication**
- Piecewise Transfer
- Seeking

Import | **Name** | **Find** | **Fetch**

Chunking	CID	Routing	Bitswap
UnixFS	Path	DHT	
IPLD	IPNS	Kademlia	

Contiguous File:



Chunked File:

- **Deduplication**
- Piecewise Transfer
- Seeking

Import | **Name** | **Find** | **Fetch**

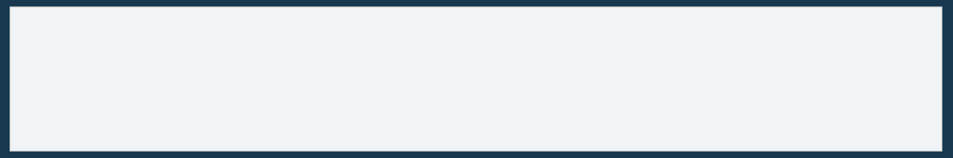
Chunking
UnixFS
IPLD

CID
Path
IPNS

Routing
DHT
Kademlia

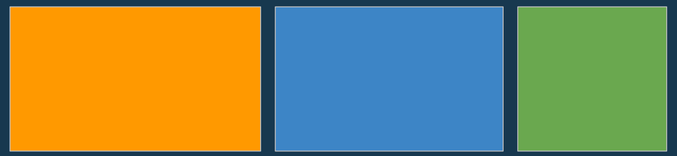
Bitswap

Contiguous File:



Chunked File:

Deduplicated:

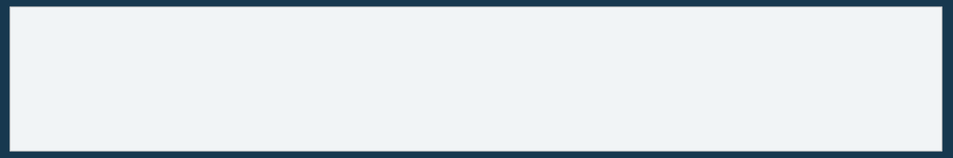


- **Deduplication**
- Piecewise Transfer
- Seeking

Import | **Name** | **Find** | **Fetch**

Chunking	CID	Routing	Bitswap
UnixFS	Path	DHT	
IPLD	IPNS	Kademlia	

Contiguous File:



Chunked File:

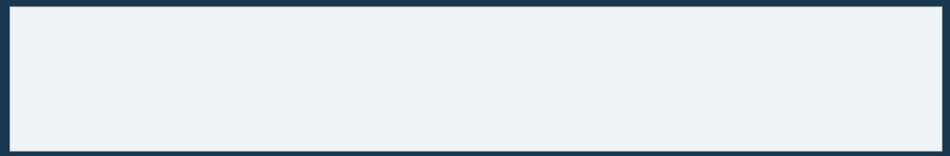
Fetched:

- Deduplication
- **Piecewise Transfer**
- Seeking

Import | **Name** | **Find** | **Fetch**

Chunking	CID	Routing	Bitswap
UnixFS	Path	DHT	
IPLD	IPNS	Kademlia	

Contiguous File:



Chunked File:

- Deduplication
- Piecewise Transfer
- **Seeking**

Import | **Name** | **Find** | **Fetch**

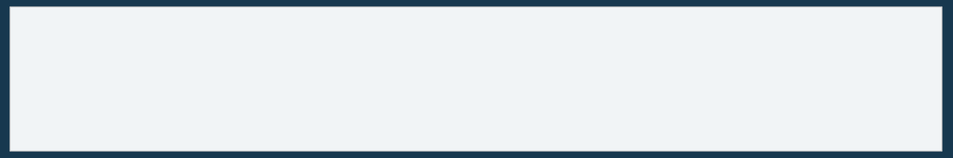
Chunking
UnixFS
IPLD

CID
Path
IPNS

Routing
DHT
Kademlia

Bitswap

Contiguous File:



- Deduplication
- Piecewise Transfer
- **Seeking**

Import | **Name** | **Find** | **Fetch**

Chunking	CID	Routing	Bitswap
UnixFS	Path	DHT	
IPLD	IPNS	Kademlia	

File Chunks:

Content addressing:

FOLDERS

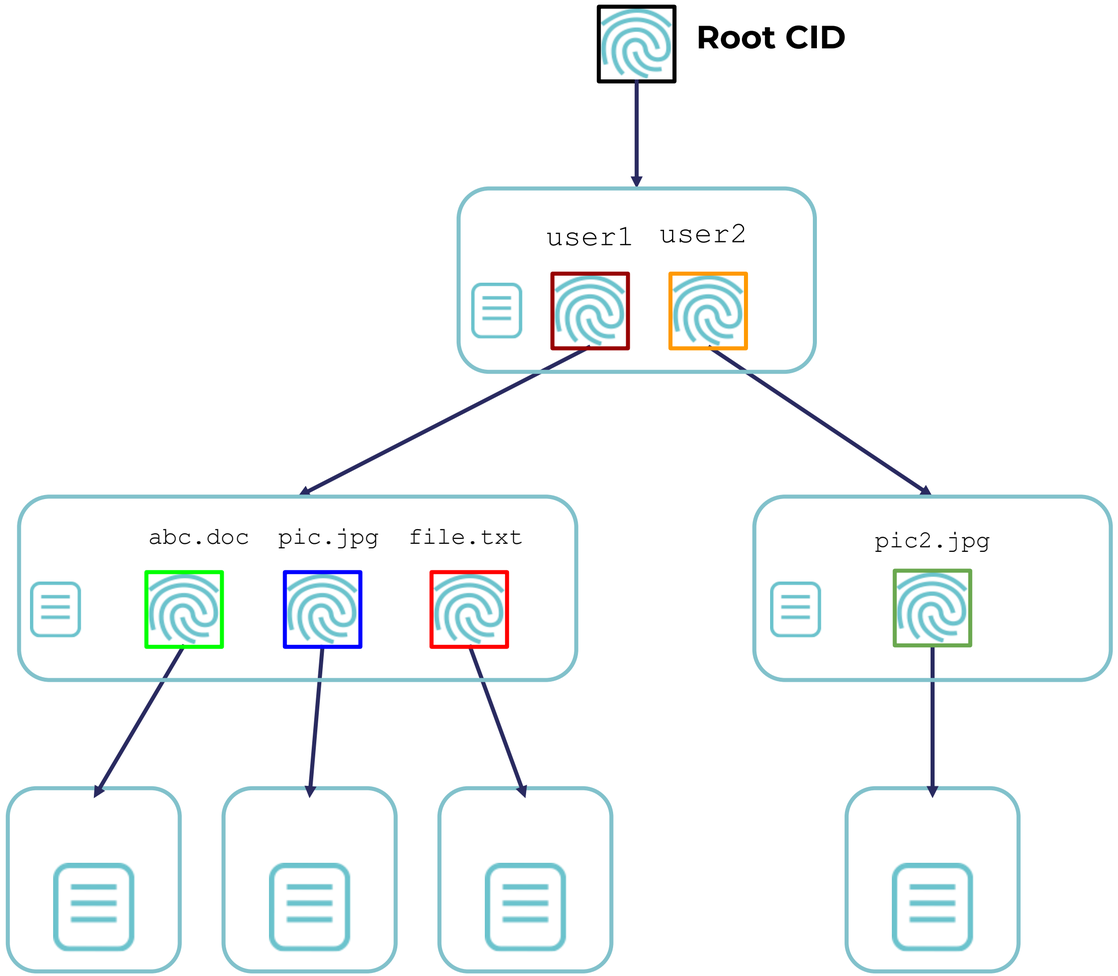
A folder is a special file which lists the files in it:

```
➔fileA -> <CID_A>
➔fileB -> <CID_B>
➔folderC -> <CID_C>
```

Content addressing:

FOLDERS

A folder is a special file which lists the files in it:

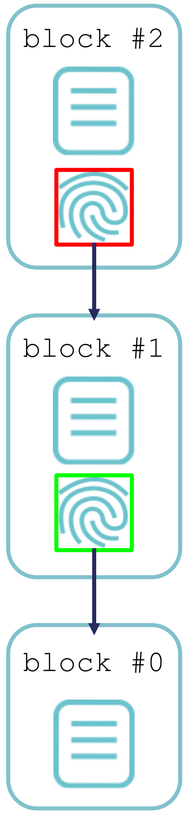


Content addressing:

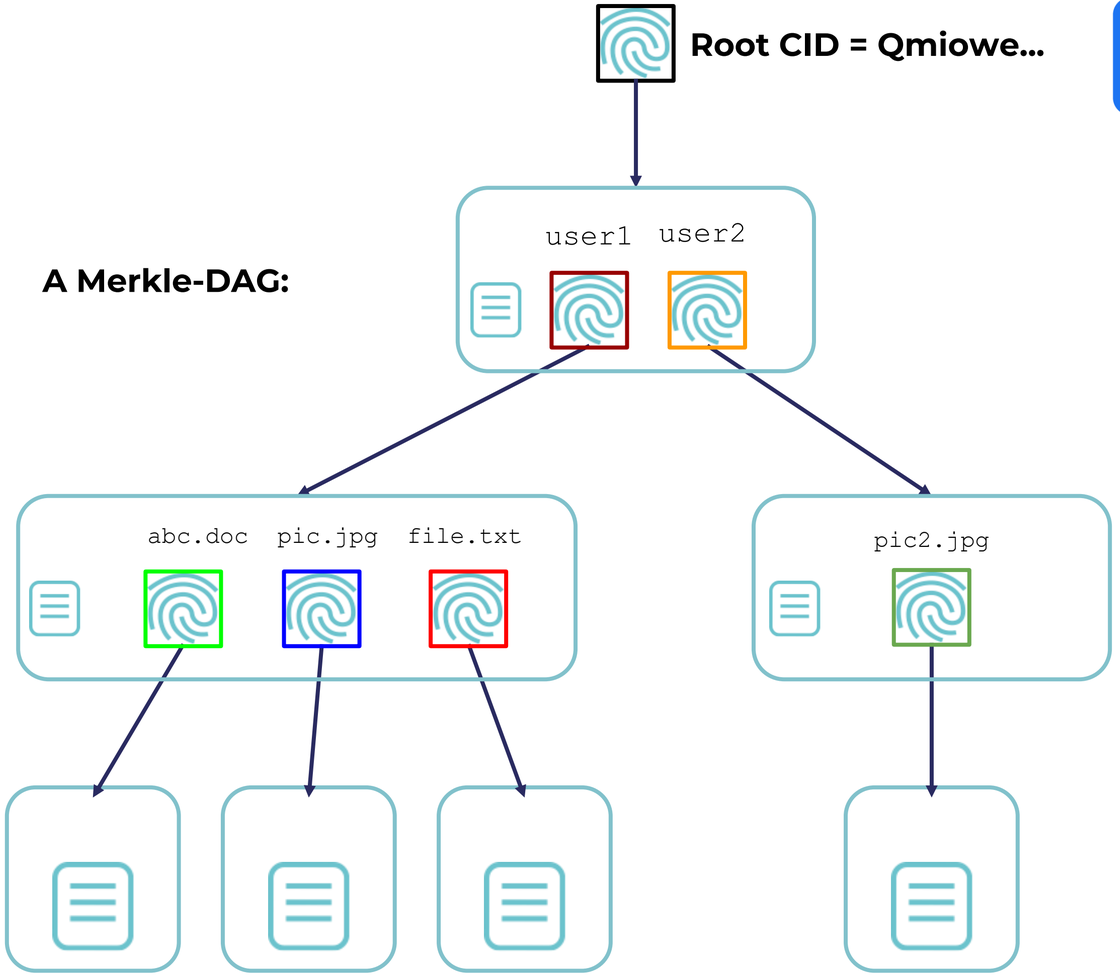
MERKLE-DAGs

Merkle-Direct-Acyclic-Graphs are graph data-structures where each node is content-addressed.

A blockchain:



A Merkle-DAG:



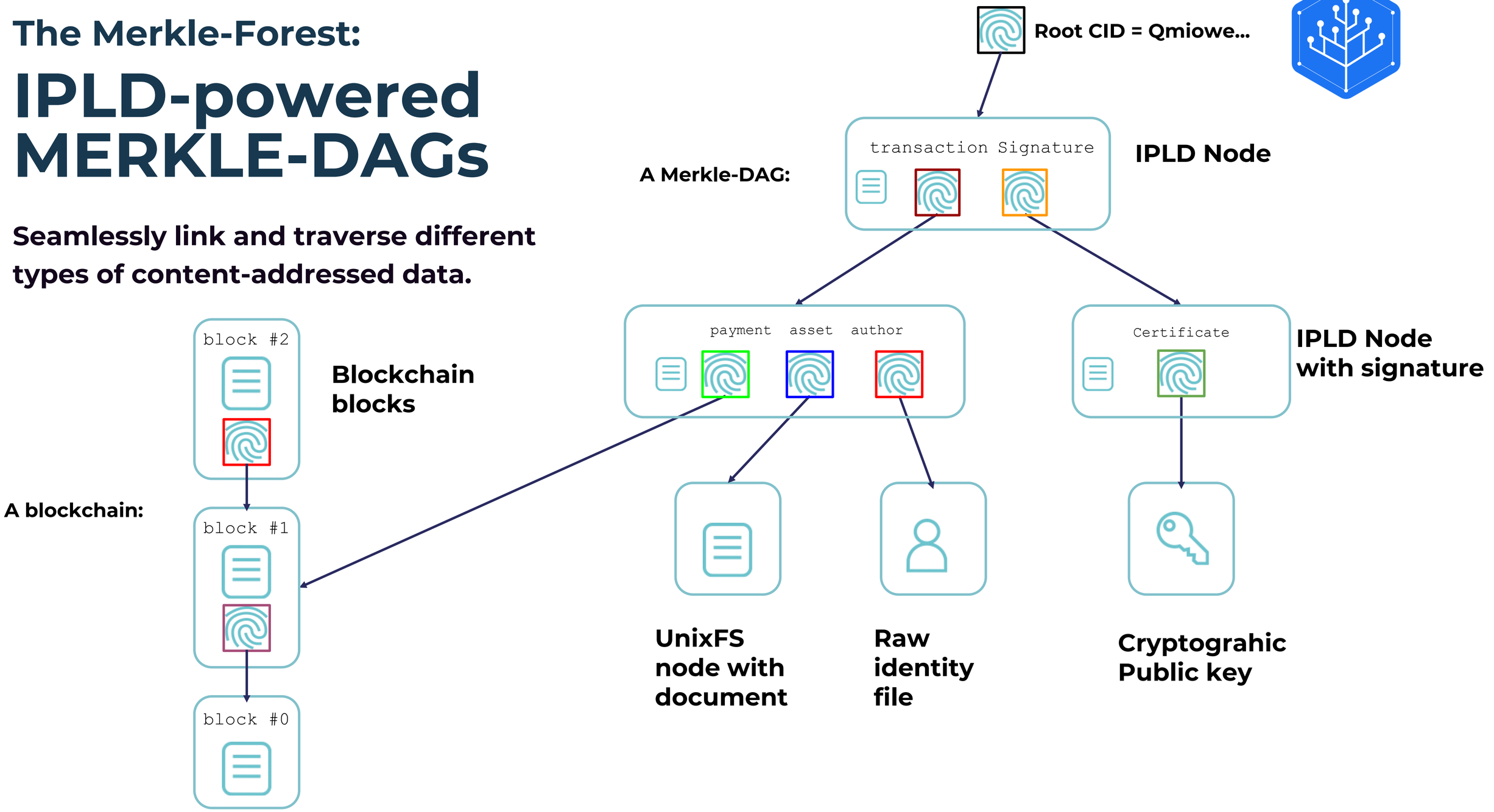
Location-based identifier -> IPFS Content-based Identifier:

`http://something.com/news/index.html` **->** `ipfs://Qmiowe.../news/index.html`

The Merkle-Forest:

IPLD-powered MERKLE-DAGs

Seamlessly link and traverse different types of content-addressed data.



Location Addressing

abc.com/poodle.jpg

VS

Content Addressing

Chunking
UnixFS
IPLD

CID
Path
IPNS

Routing
DHT
Kademlia

Bitswap

Content Identifier

QmS4ustL54uo8FzR9455qaxZwuMiUhyvMcX9Ba8nUH4uVv

bafybeibxm2nsadl3fnxv2sxcxmxaco2jl53wpeorjdzidjwf5aqdg7wa6u

CIDs are:

- used for **content addressing**
- **self describing**
- used to name every piece of data in IPFS/IPLD
- basically a **hash** with some **metadata**

Chunking
UnixFS
IPLD

CID
Path
IPNS

Routing
DHT
Kademlia

Bitswap

Immutable

Verifiable

Trustless

Permanent

Import	Name	Find	Fetch
Chunking	**CID**	Routing	Bitswap
UnixFS	Path	DHT	
IPLD	IPNS	Kademlia	

CIDs: What do they look like?

```
<base>base(<cid-version><multicodec><multihash>)
```

Import	Name	Find	Fetch
Chunking	**CID**	Routing	Bitswap
UnixFS	Path	DHT	
IPLD	IPNS	Kademlia	

Multiformats: Self-describing data

`<base>base(<cid-version><multicodec><multihash>)`

- Multicodec: a non-magic number to uniquely identify a format, protocol, etc.
- Multihash: a self describing hash digest.
- Multibase: a self describing base-encoded string.

Chunking	CID	Routing	Bitswap
UnixFS	Path	DHT	
IPLD	IPNS	Kademlia	

Multiformats: Self-describing data

Multicodec: a non-magic number.

name,	tag,	code,	description
identity,	multihash,	0x00,	raw binary
ip4,	multiaddr,	0x04,	
dccp,	multiaddr,	0x21,	
dnsaddr,	multiaddr,	0x38,	
protobuf,	serialization,	0x50,	Protocol Buffers
cbor,	serialization,	0x51,	CBOR
raw,	ipld,	0x55,	raw binary
...			

github.com/multiformats/multicodec

Chunking
UnixFS
IPLD

CID
Path
IPNS

Routing
DHT
Kademlia

Bitswap

Multiformats: Self-describing data

Multihash: a self-describing hash digest:

- Hash Function (*multicodec*)
- Hash Digest Length
- Hash Digest

github.com/multiformats/multihash

Import	Name	Find	Fetch
Chunking	**CID**	Routing	Bitswap
UnixFS	Path	DHT	
IPLD	IPNS	Kademlia	

Multiformats: Self-describing data

Multibase: a self-describing base encoding.

- A multibase prefix.
  - b - base32
  - z - base58
  - f - base16
- Followed by the base encoded data.

*b*afybeibxm2...

Chunking	**CID**	Routing	Bitswap
UnixFS	Path	DHT	
IPLD	IPNS	Kademlia	

Self Describing

58	bitcoin-block	ipld	0xb0	Bitcoin Block
59	bitcoin-tx	ipld	0xb1	Bitcoin Tx

- CIDv0: **QmS4u...**
  - Base58 encoded sha256 multihash
- CIDv1: **bafybei...**
  - Multibase encoded (ipld format multicodec, multihash) tuple.
- Why CIDv1?
  - Can be encoded in arbitrary bases (base32, base58, etc.).
  - Can link *between* merkle-dag formats using the *ipld format* multicodec.

Import	Name	Find	Fetch
Chunking	CID	Routing	Bitswap
UnixFS	Path	DHT	
IPLD	**IPNS**	Kademlia	

IPNS maps **Public Keys** to *paths*

/ipns/QmMyKey -> /ipfs/QmFoo (signed)

IPNS is *mutable*

/ipns/QmMyKey -> /ipfs/**QmSomethingNew**

IPNS can point to arbitrary paths

/ipns/QmMyKey -> /ipns/QmYourKey

Import → Name → Find → Fetch

Import	Name	Find	Fetch
Chunking UnixFS IPLD	CID Path IPNS	**Routing** **DHT** **Kademlia**	Bitswap

Enter libp2p

A Modular P2P Networking Stack

Content Address (CID) → Location Address (Peer)

Chunking
UnixFS
IPLD

CID
Path
IPNS

Routing
DHT
Kademlia

Bitswap

DECENTRALIZED PROCESS ADDRESSING

libp2p's raison d'être is the ability to locate, connect, authenticate, negotiate and interact efficiently with any process in the world, no matter the runtime (server, browser, IoT, embedded, etc.) so long as its identity is cryptographically derived from its public key; and have all of that happen in a seamless manner (e.g. NAT, relay, packet switching), even as those processes relocate, roam, evolve and mutate over time. It is juxtaposed to endpoint addressing (e.g. IP networks).

Import → Name → Find → Fetch

Chunking
UnixFS
IPLD

CID
Path
IPNS

Routing
DHT
Kademlia

Bitswap

Transports

Multiplexers

Secure Channels

Peer Discovery

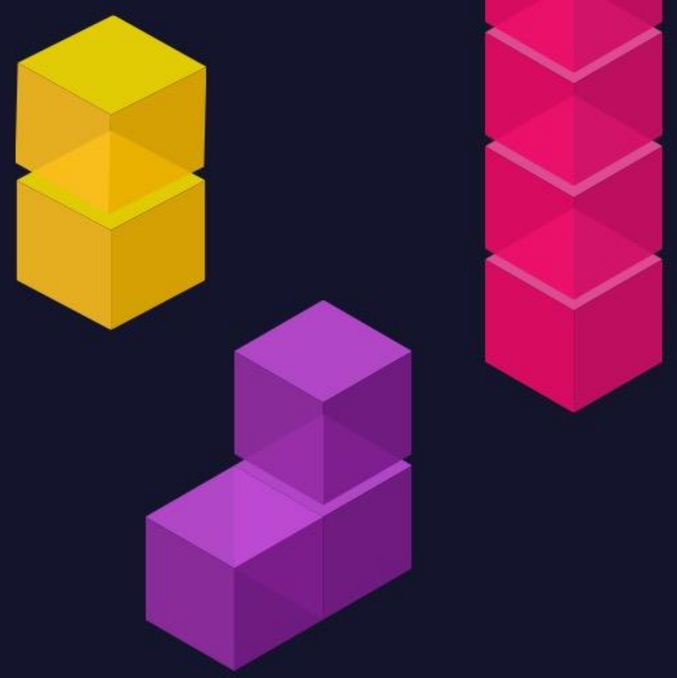
libp2p

Pubsub

NAT Traversal

Peer Routing

Content Routing



transport-agnostic addresses

MULTIADDRS

/ip4/104.236.179.241/tcp/4001/p2p/QmPeer…

0x0468ecb3f1060fa1 (omitting the id)

- Composable, future-proof, upgradeable transport-agnostic addresses.
- binary byte-encoded packed format, with a canonical string representation.
- encodes addressing, transport, routing, identity, encryption (future) of a peer.
- flexible varint-based byte[] format vs. hardcoding assumptions.

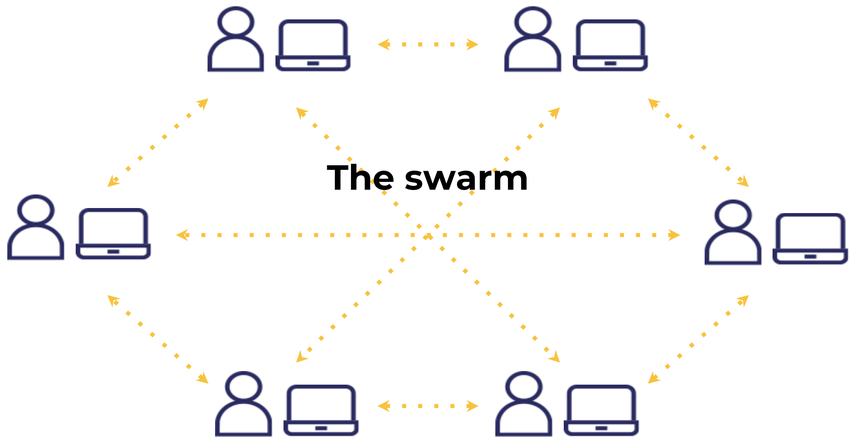
Chunking
UnixFS
IPLD

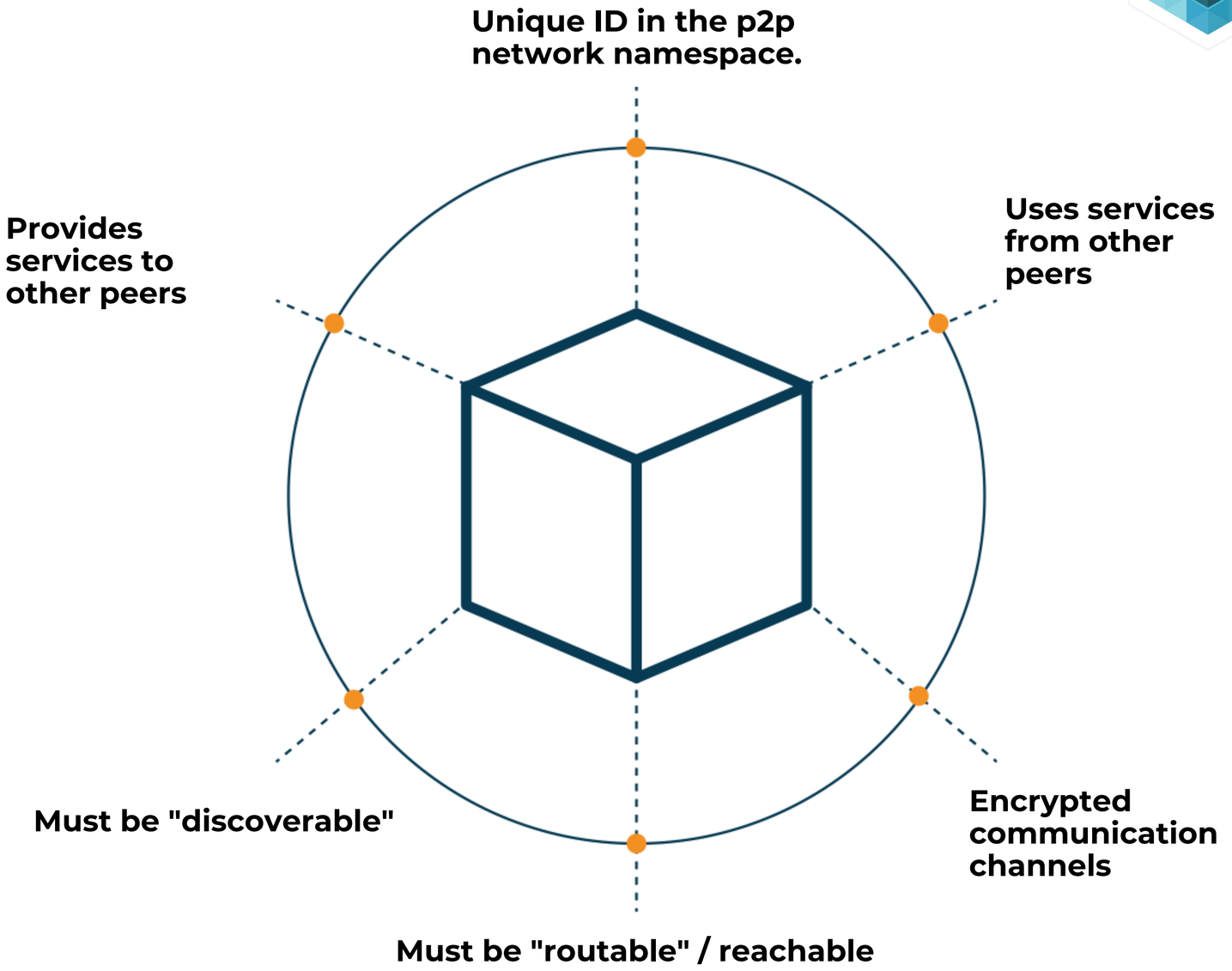
CID
Path
IPNS

Routing
DHT
Kademlia

Bitswap

Content routing:
The Peer





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)**

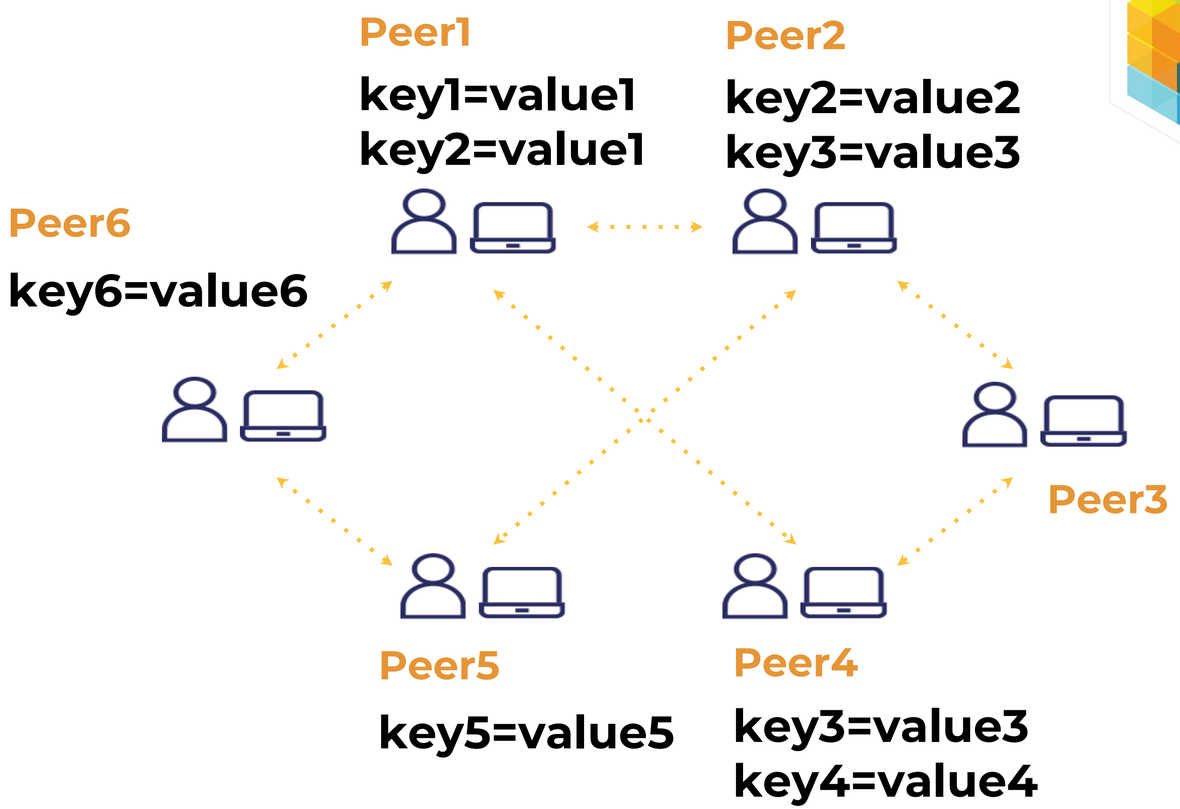
Import | Name | Find | Fetch

Import	Name	Find	Fetch
Chunking	CID	Routing	Bitswap
UnixFS	Path	**DHT**	
IPLD	IPNS	Kademlia	

Content routing:

THE DHT

A Distributed Hash Table (DHT) provides a 2-column table (key-value store) maintained by multiple peers.



Each row is stored by peers based on similarity between the key and the peer ID. We call this "distance":

- A peer ID can be "closer" to some keys than others
- A peer ID can be "closer" to other peers

The DHT in IPFS is used to provide:

- Content discovery (ContentID=PeerID)
- Peer routing (PeerID=/ip4/1.2.3.4/tcp/1234)

Example	
keys	values
key1	value1
key2	value2
...	...

Actual IPFS DHT contents	
keys (Content IDs or Peer IDs)	values
/ipfs/Qmabc	Qmpid1
/ipns/Qmzxy	/ipfs/Qmabc
Qmpid1	192.1.2.3, 42.53.1.23
Qmpid2	/relay/Qmpid1
...	...

Chunking
UnixFS
IPLD

CID
Path
IPNS

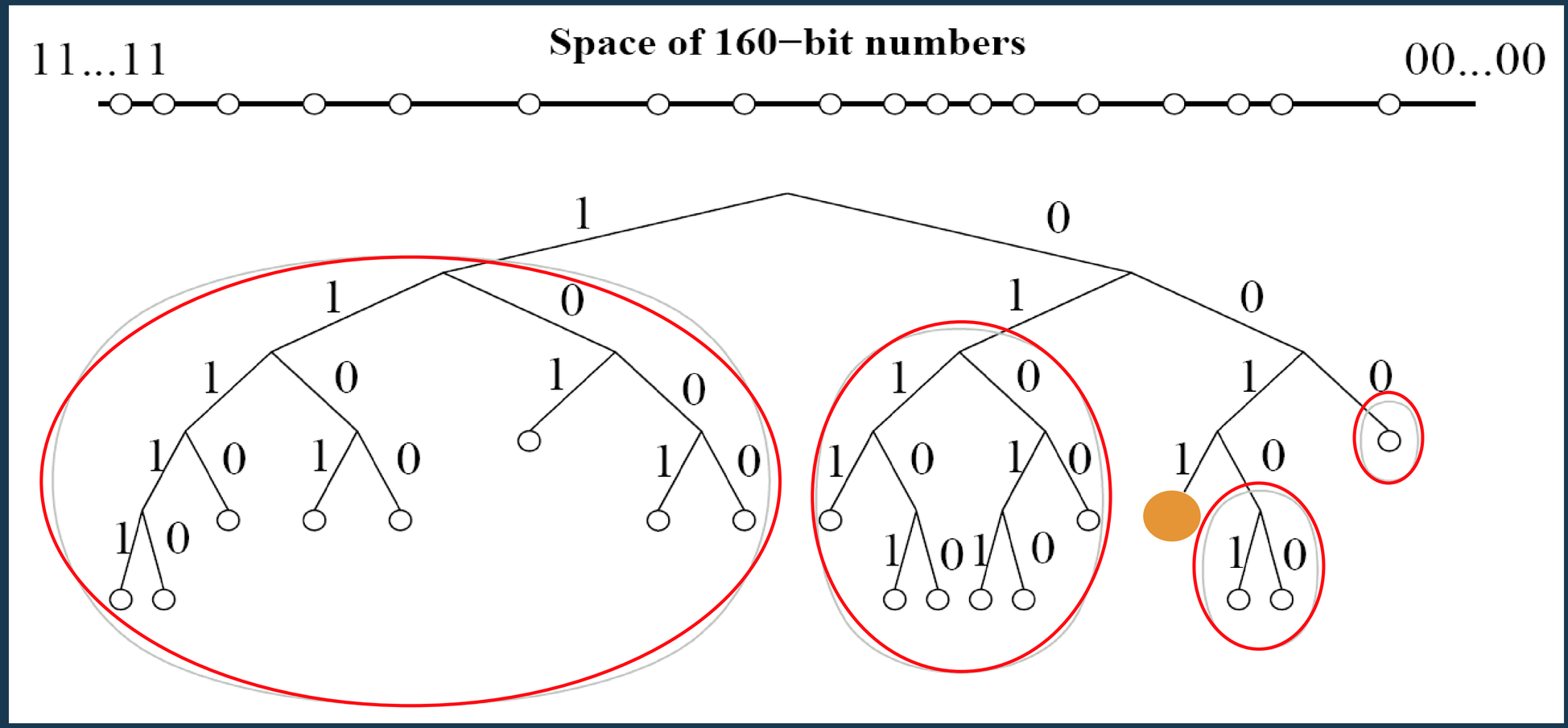
Routing
DHT
Kademlia

Bitswap

The Kademlia Distributed Hash Table

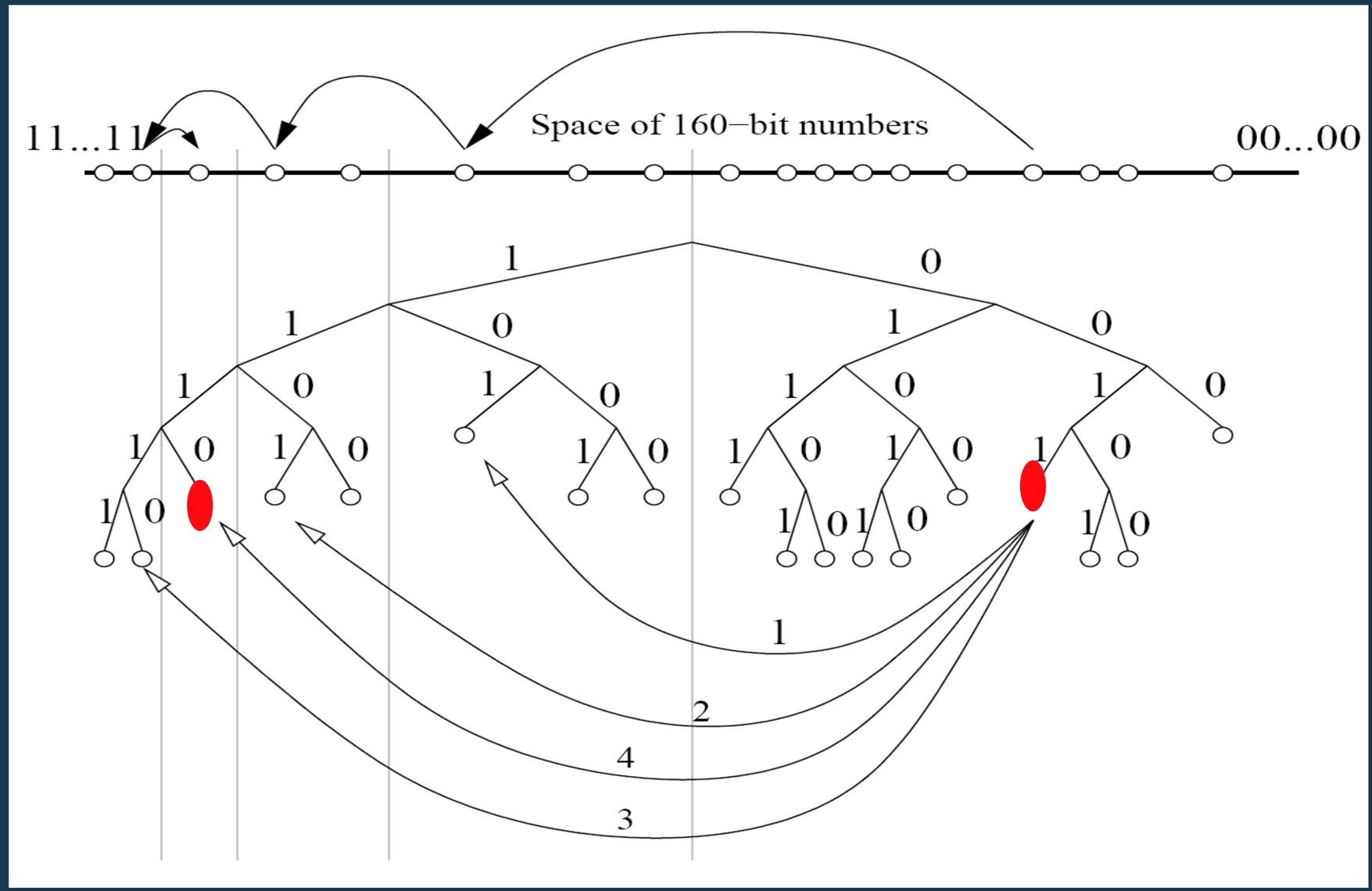
Kademlia Binary Tree

Subtrees for a node 0011.....

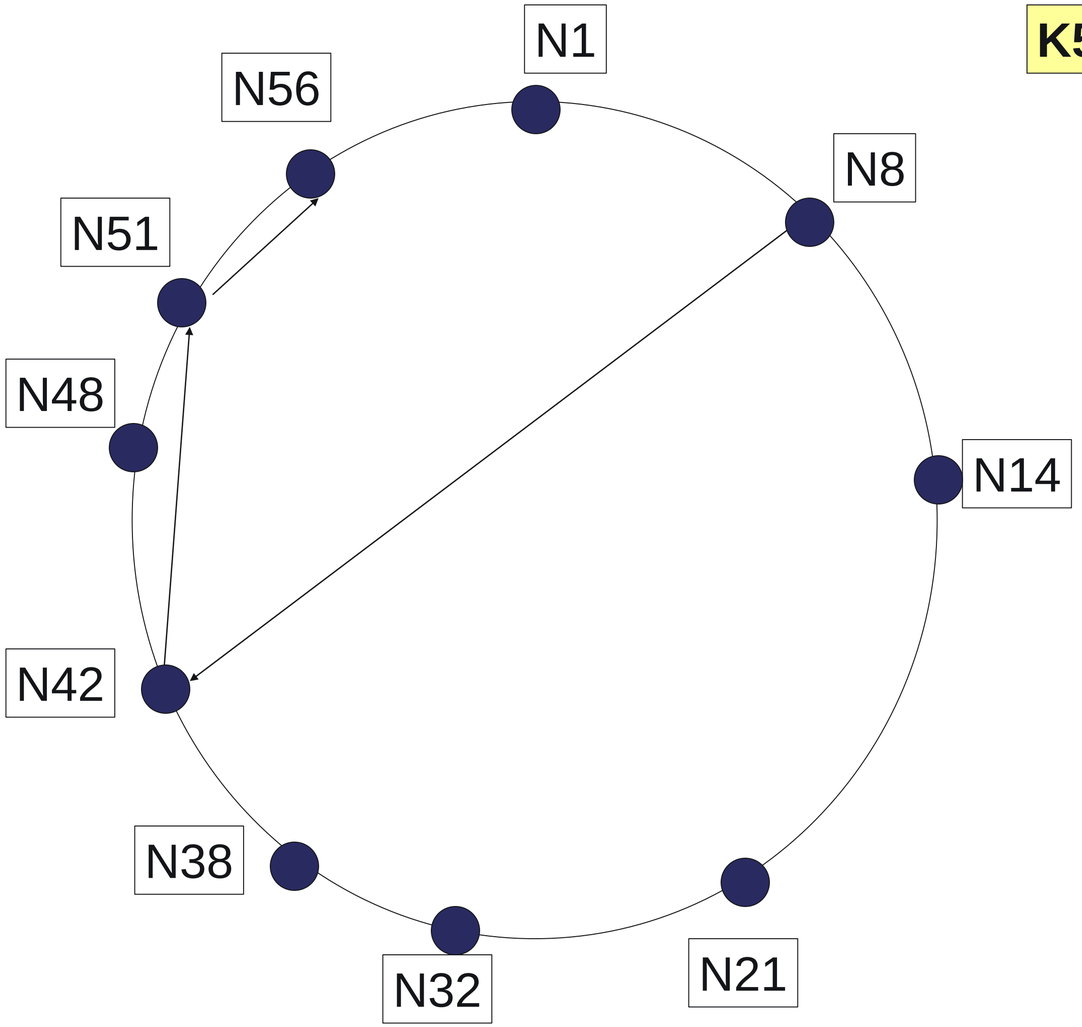


Kademlia Search

An example of lookup: node 0011 is searching for 1110.....in the network

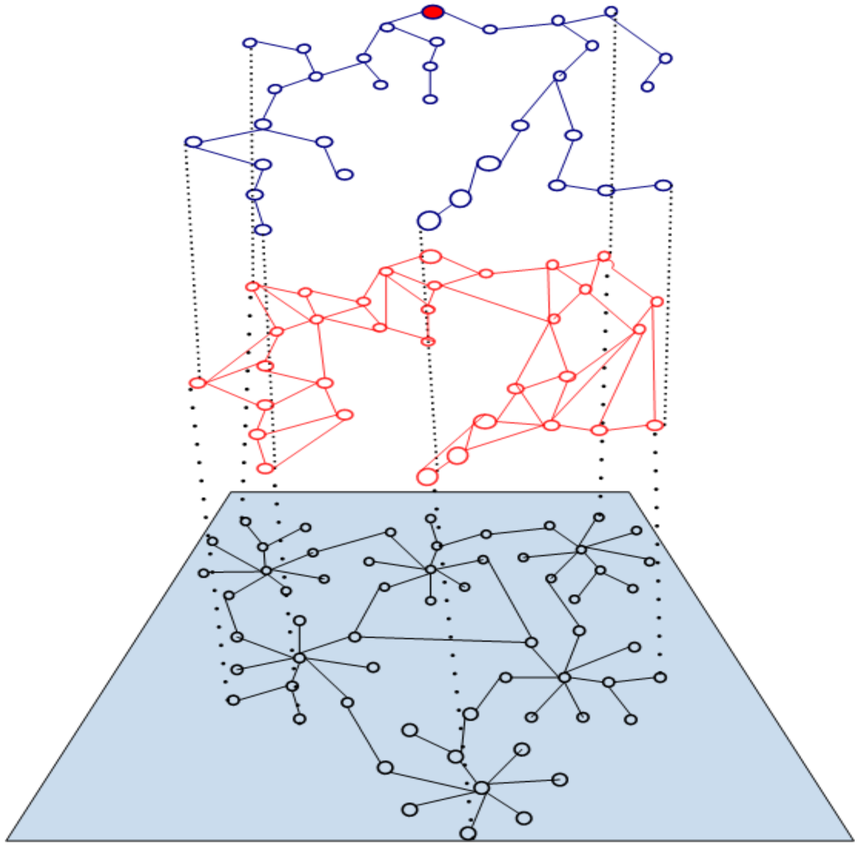


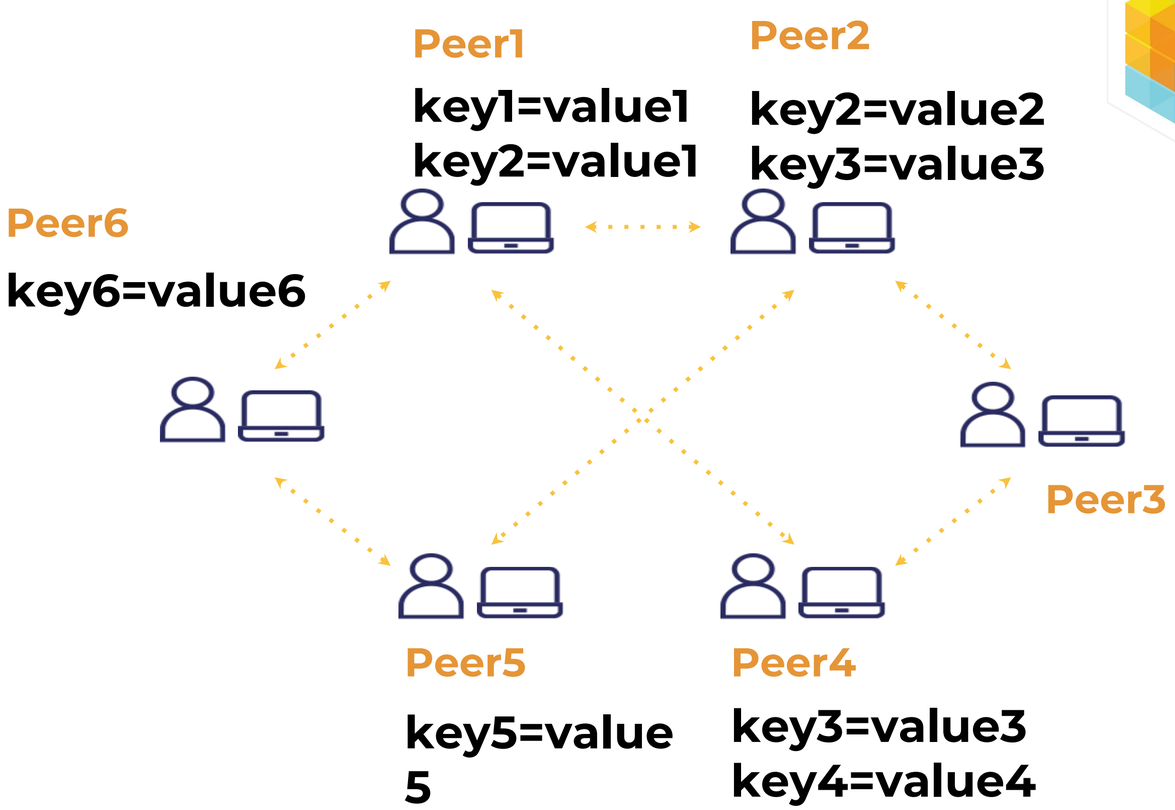
Lookup Using Finger Table



Content routing:

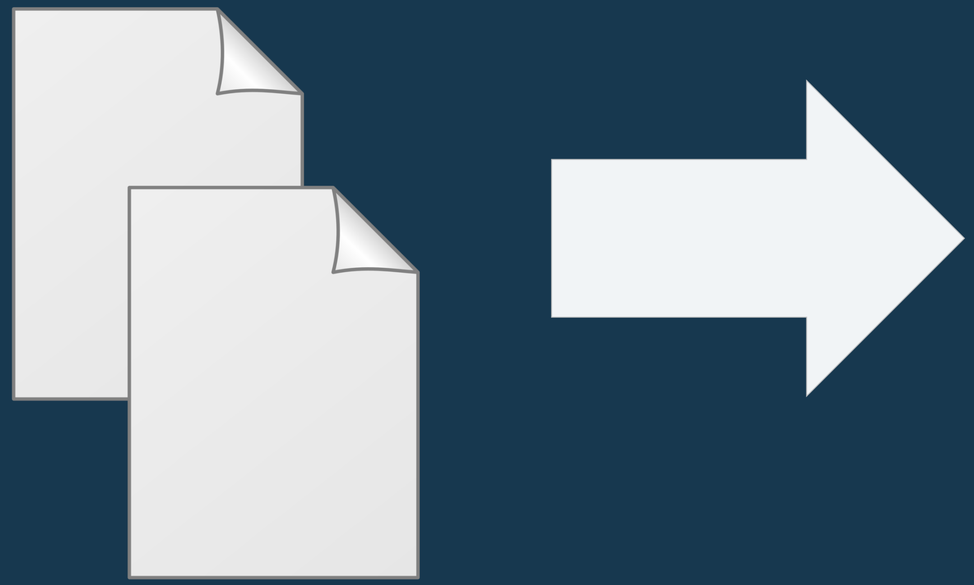
THE DHT Overlay vs Underlay





Import → Name → Find → Fetch

Import	Name	Find	Fetch
Chunking UnixFS IPLD	CID Path IPNS	Routing DHT Kademlia	**Bitswap**



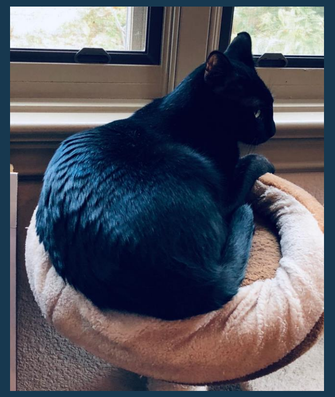
Import → Name → Find → Fetch

Chunking
UnixFS
IPLD

CID
Path
IPNS

Routing
DHT
Kademlia

Bitswap

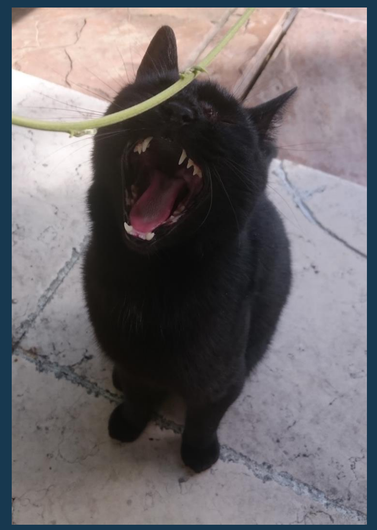


Izzy Wants

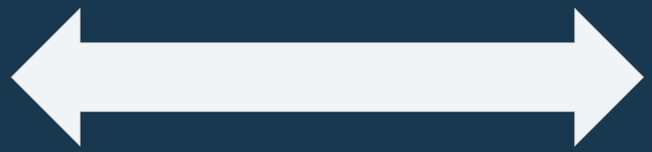
- QmTreats
- QmToy

Ozzy Wants

- QmCuddles
- QmFood
- QmAttention



Izzy



Ozzy

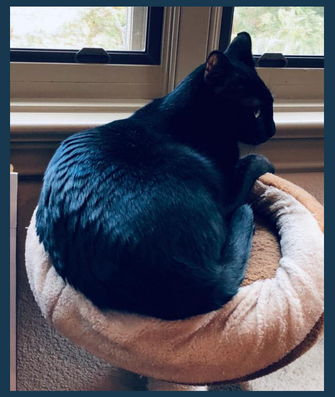
Import → Name → Find → Fetch

Chunking
UnixFS
IPLD

CID
Path
IPNS

Routing
DHT
Kademlia

Bitswap



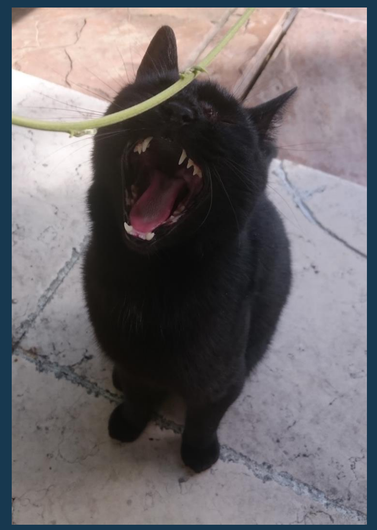
Izzy

Ozzy Wants

- QmCuddles
- QmFood
- QmAttention

Izzy Wants

- QmTreats
- QmToy



Ozzy

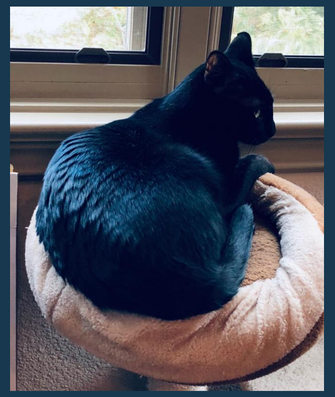
Import → Name → Find → Fetch

Chunking
UnixFS
IPLD

CID
Path
IPNS

Routing
DHT
Kademlia

Bitswap



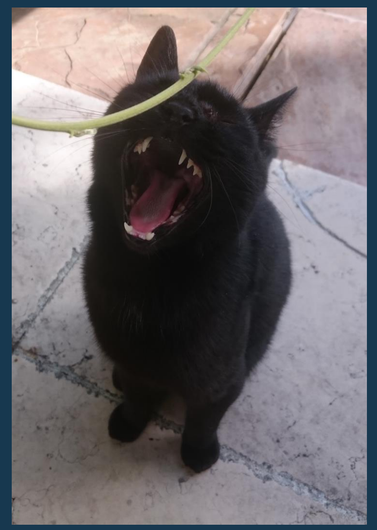
Izzy

Ozzy Wants

- QmCuddles
- *QmFood*
- *QmAttention*

Izzy Wants

- QmTreats
- *QmToy*



Ozzy

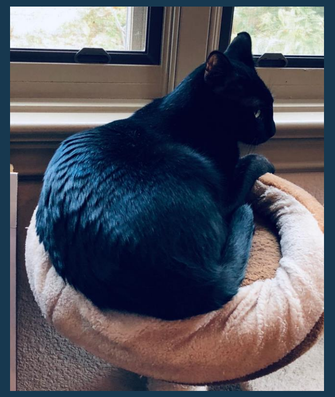
Import → Name → Find → Fetch

Chunking
UnixFS
IPLD

CID
Path
IPNS

Routing
DHT
Kademlia

Bitswap



Izzy

Ozzy Wants

- QmCuddles
- *QmFood*
- *QmAttention*

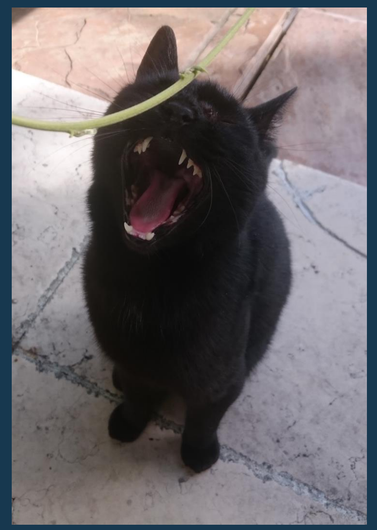
QmToy

Izzy Wants

- QmTreats
- *QmToy*

QmFood

QmAttention



Ozzy

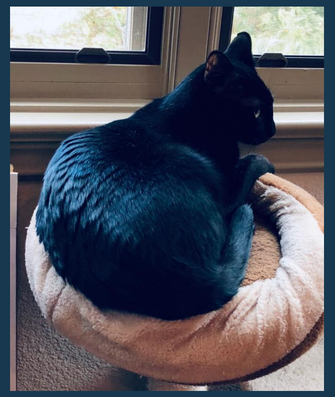
Import | Name | Find | Fetch

Chunking
UnixFS
IPLD

CID
Path
IPNS

Routing
DHT
Kademlia

Bitswap



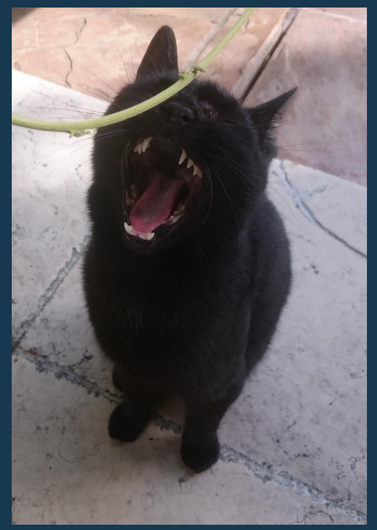
Izzy

Ozzy Wants

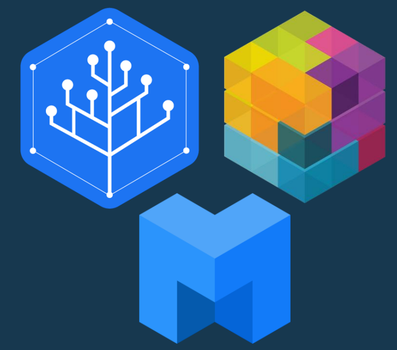
- QmCuddles

Izzy Wants

- QmTreats



Ozzy



Chunking	CID	Routing	Bitswap
UnixFS	Path	DHT	
IPLD	IPNS	Kademlia	

Publishing Content:

1. Chunking
2. Obtaining the CID
3. Adding Content to the network
   - Content is not replicated, only provider record is stored in the DHT

Consuming Content from the browser:

1. Local node acts as client
2. Connects to public IPFS Gateway
3. Public IPFS Gateway acts as a full IPFS Server node

Consuming Content as an IPFS Peer:

1. Get CID (out of band)
2. Walk the DHT to resolve CID to PeerID
3. Contact PeerID to ask for CID
4. Fetch content and cache a copy
5. Serve local copy upon subsequent request
6. **In parallel:** send your WANTLIST to all connected peers through BitSwap

Import	Name	Find	Fetch
Chunking UnixFS IPLD	CID Path IPNS	Routing DHT Kademlia	Bitswap

Try it out!

Find out more about the IPFS project:

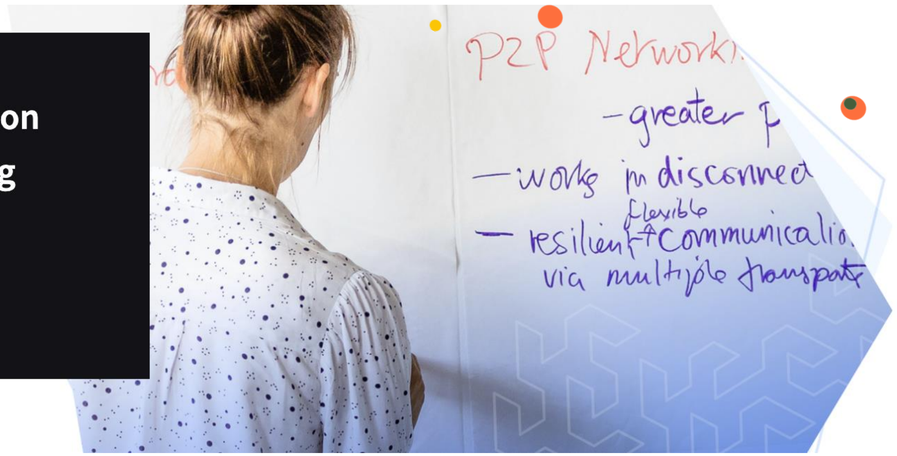
- IPFS: documentation and download https://docs.ipfs.io
- libp2p documentation: https://docs.libp2p.io
- ProtoSchool: Interactive tutorials on decentralized data structures https://proto.school
- IPFS Companion browser extension: Upgrade your browser with IPFS superpowers: https://github.com/ipfs-shipyard/ipfs-companion
- IPFS Desktop: https://github.com/ipfs-shipyard/ipfs-desktop
- DNSLink: https://dnslink.io

Lots more to look out for!

Protocol Labs
Research

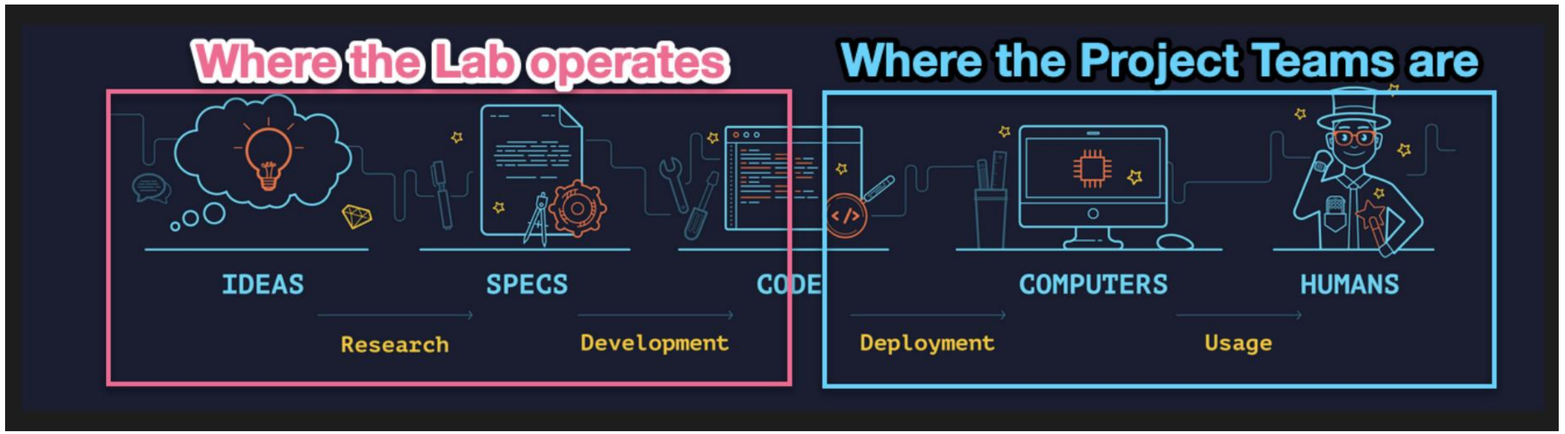
We explore the future of decentralization and examine the infrastructure limiting what you can do with technology.

Our research philosophy ›

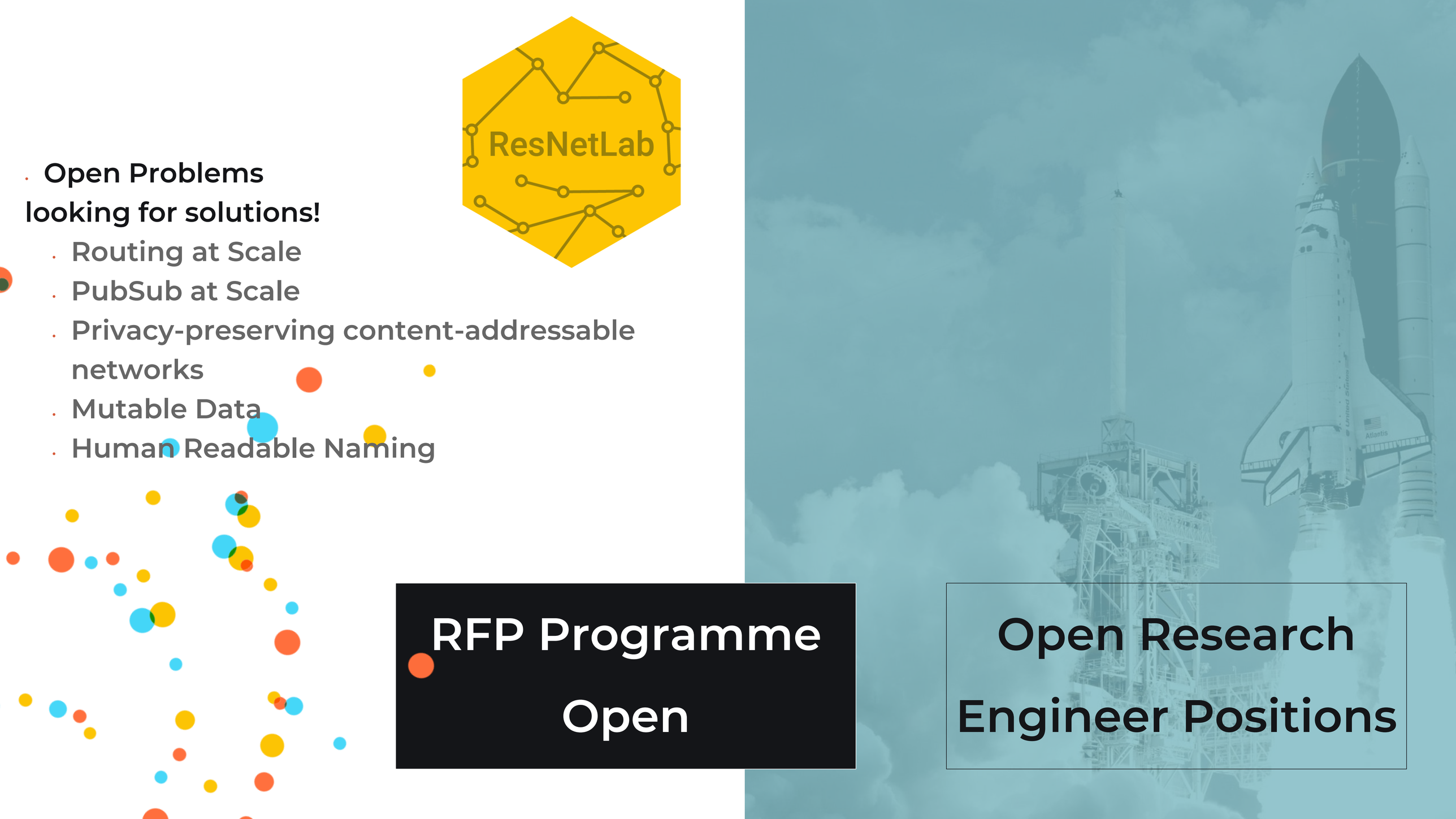


https://research.protocol.ai/

ResNetLab



https://research.protocol.ai/research/groups/resnetlab/



- **Open Problems looking for solutions!**
  - Routing at Scale
  - PubSub at Scale
  - Privacy-preserving content-addressable networks
  - Mutable Data
  - Human Readable Naming

RFP Programme Open

Open Research Engineer Positions