How IPFS Works

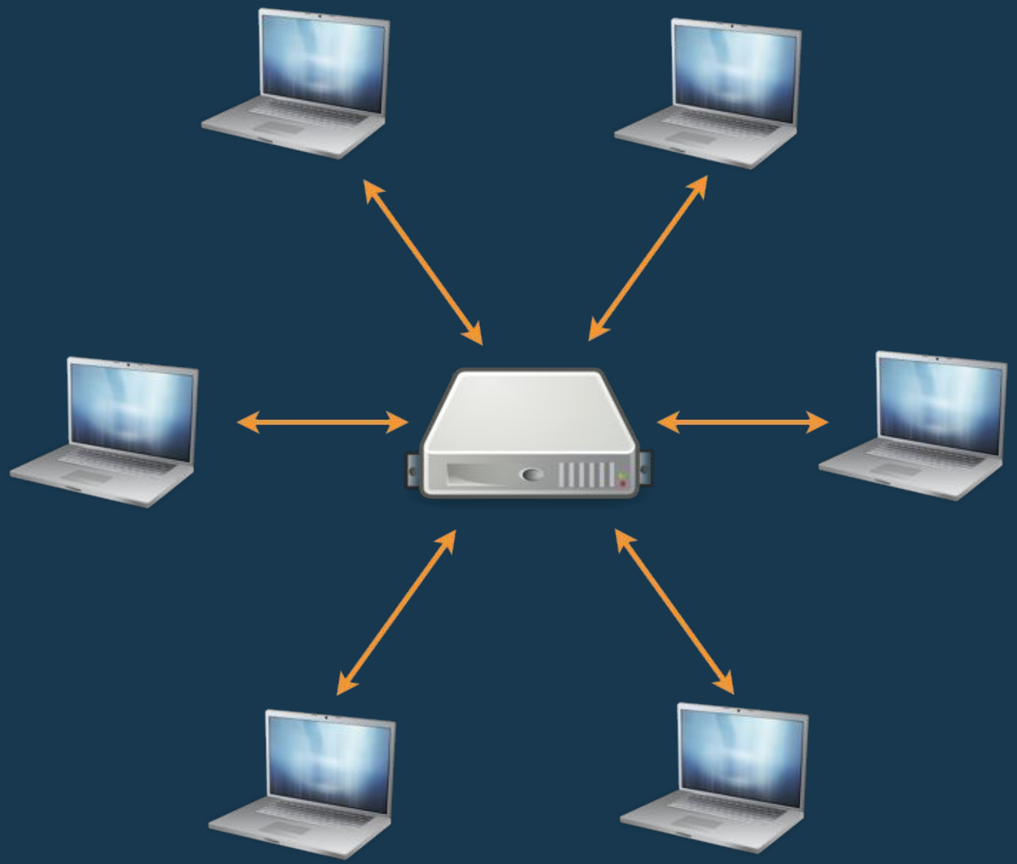
(approximately)

Sawood Alam (@*ibnesayeed*)
Old Dominion University

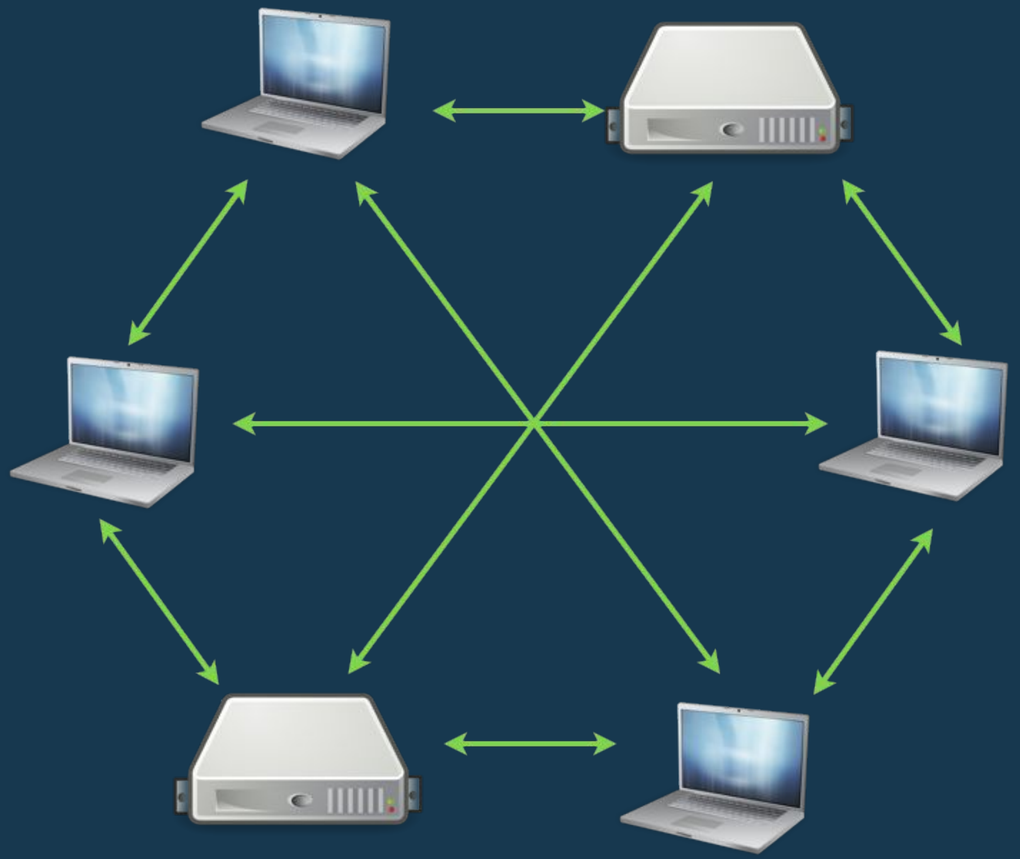
Original slides by Steven Allen (@*Stebalien*)

IPFS makes the web work peer-to-peer

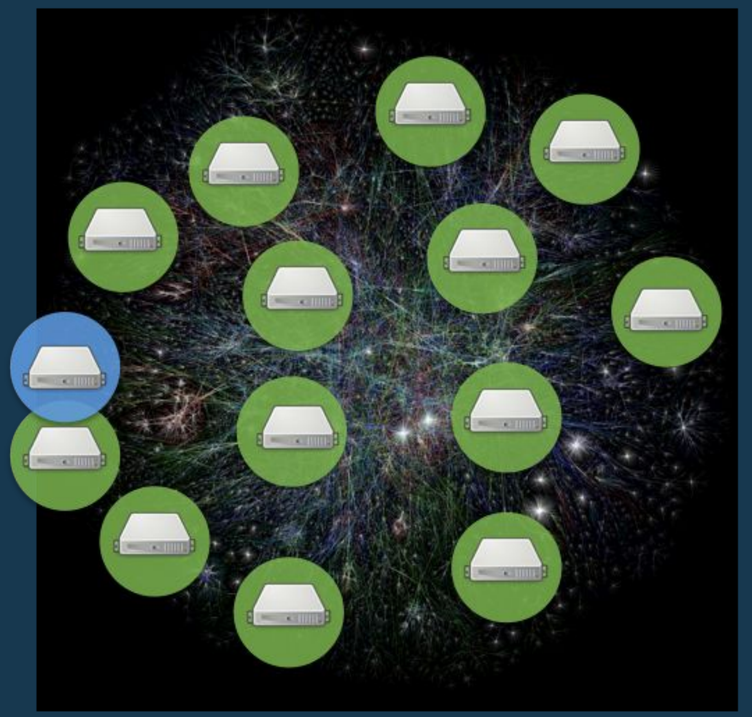
HTTP



IPFS



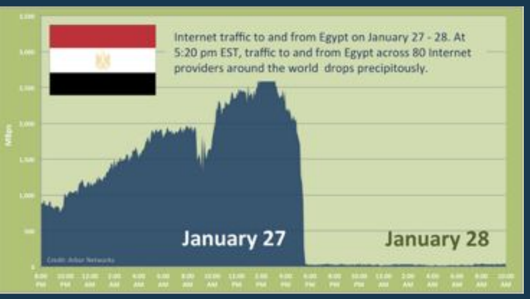
domain name /dns/example.com/foo/bar/baz.png



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

Problems

emerging networks



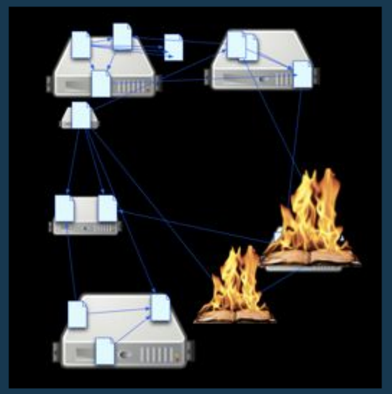
censorship

IPFS

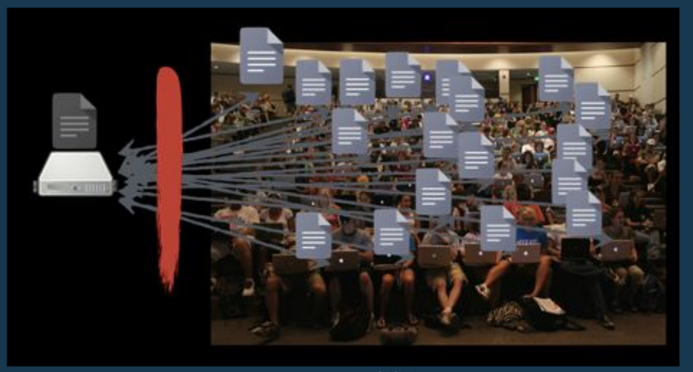
Addresses

huge inefficiency

bad security model



links break



no offline use

IPFS: Distributed Web Protocol

IPLD: authenticated data model & formats

libp2p: modular p2p networking library

Multiformats: future-proofing & upgradability

IPFS:

Lifecycle

IPFS:

Adding Files

```
λ: ipfs add -r docs
added QmZTR5bcpQD7cFgTorqxZDYaew1Wqgfbd2ud9QqGPAkK2V docs/about
added QmYCvbfNbCwFR45HiNP45rwJgvatpiW38D961L5qAhUM5Y docs/contact
added QmY5heUM5qgRubMDD1og9fhCPA6QdkMp3QCwd4s7gJsyE7 docs/help
added QmejvEPop4D7YUadeGqYWmZxHhLc4JBUCzJJHWMzdcMe2y docs/ping
added QmXgqKTbzdh83pQtKFb19SpMCpDDcKR2ujqk3pKph9aCNF docs/quick-start
added QmPZ9gcCEpqKTo6aq61g2nXGUhM4iCL3ewB6LDXZCtioEB docs/readme
added QmQ5vhrL7uv6tuoN9KeVBwd4PwfQkXdVVmDLUZuTNxqgvm docs/security-notes
added QmS4ustL54uo8FzR9455qaxZwuMiUhyvMcX9Ba8nUH4uVv docs
 5.97 KiB / 5.97 KiB [=============================================] 100.00%
```

IPFS:
Adding Files

```
λ: ipfs add -r docs
added QmZTR5bcpQD7cFgTorqxZDYaew1Wqgfbd2ud9QqGPAkK2V docs/about
added QmYCvbfNbCwFR45HiNP45rwJgvatpiW38D961L5qAhUM5Y docs/contact
added QmY5heUM5qgRubMDD1og9fhCPA6QdkMp3QCwd4s7gJsyE7 docs/help
added QmejvEPop4D7YUadeGqYWmZxHhLc4JBUCzJJHWMzdcMe2y docs/ping
added QmXgqKTbzdh83pQtKFb19SpMCpDDcKR2ujqk3pKph9aCNF docs/quick-start
added QmPZ9gcCEpqKTo6aq61g2nXGUhM4iCL3ewB6LDXZCtioEB docs/readme
added QmQ5vhrL7uv6tuoN9KeVBwd4PwfQkXdVVmDLUZuTNxqgvm docs/security-notes
added QmS4ustL54uo8FzR9455qaxZwuMiUhyvMcX9Ba8nUH4uVv docs
 5.97 KiB / 5.97 KiB [=============================================] 100.00%
```

CID

-> CID: Content Identifier

-> IPFS Path: /ipfs/QmS4ustL54uo8FzR9455qaxZwuMiUhyvMcX9Ba8nUH4uVv

-> Gateway URL: https://ipfs.io/ipfs/QmS4ustL54uo8FzR9455qaxZwuMiUhyvMcX9Ba8nUH4uVv

IPFS:

Getting Files

```
λ: ipfs get -o docs /ipfs/QmS4ustL54uo8FzR9455qaxZwuMiUhyvMcX9Ba8nUH4uVv
Saving file(s) to docs
 6.39 KiB / 6.39 KiB [=============================================] 100.00% 0s
```

CID

IPFS:

Lifecycle

Adding Files

Getting Files

Import

Chunking
UnixFS
IPLD

Name

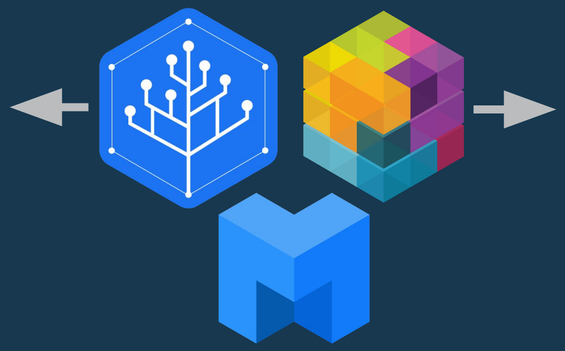
CID
Path
IPNS

Find

Routing
DHT
Kademlia

Fetch

Bitswap



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

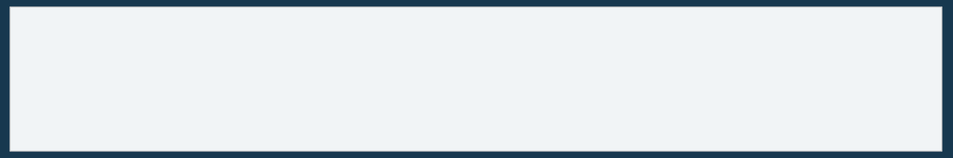
Chunking
UnixFS
IPLD

CID
Path
IPNS

Routing
DHT
Kademlia

Bitswap

Contiguous File:



Chunked File:

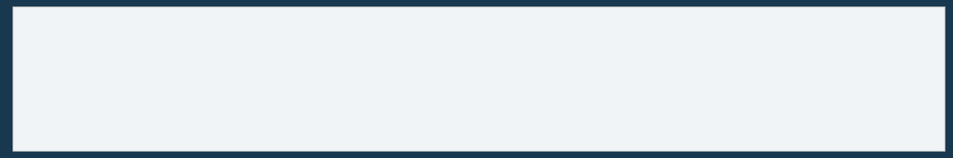
(each chunk is hashed)

- Deduplication
- Piecewise Transfer
- Seeking

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

Chunking	CID	Routing	Bitswap
UnixFS	Path	DHT	
IPLD	IPNS	Kademlia	

Contiguous File:



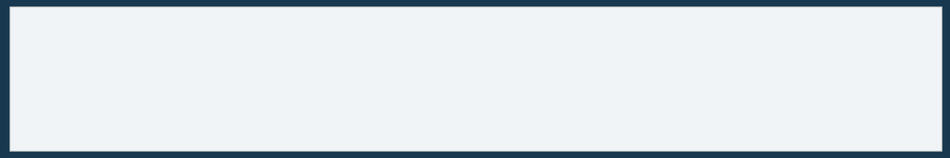
Chunked File:

- **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**

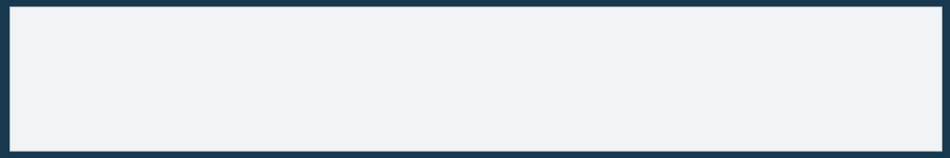
Chunking
UnixFS
IPLD

CID
Path
IPNS

Routing
DHT
Kademlia

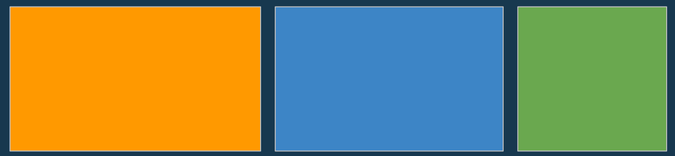
Bitswap

Contiguous File:



Chunked File:

Deduplicated:

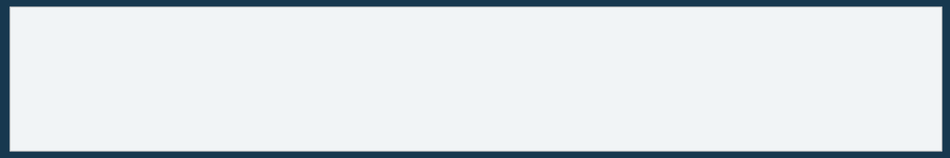


- **Deduplication**
- Piecewise Transfer
- Seeking

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

Chunking	CID	Routing	Bitswap
UnixFS	Path	DHT	
IPLD	IPNS	Kademlia	

Contiguous File:



Chunked File:

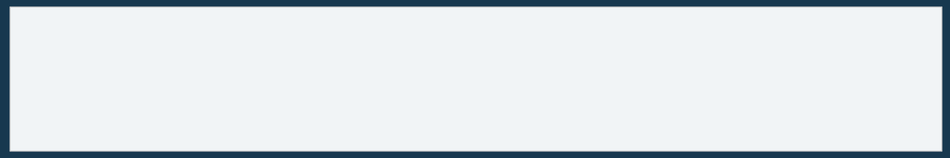
Fetched:

- 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**

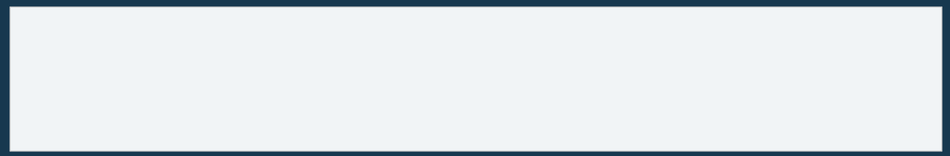
Chunking
UnixFS
IPLD

CID
Path
IPNS

Routing
DHT
Kademlia

Bitswap

Contiguous File:



Chunked File:

- Deduplication
- Piecewise Transfer
- **Seeking**

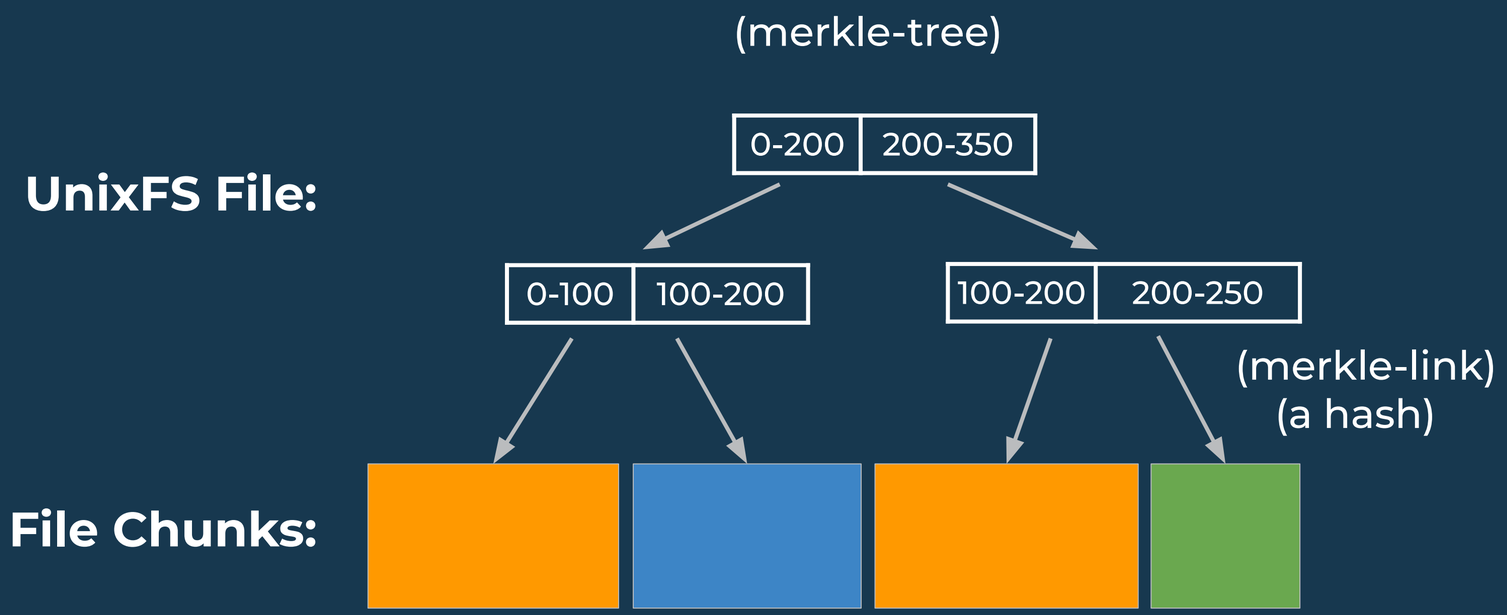
Import | **Name** | **Find** | **Fetch**

Chunking	CID	Routing	Bitswap
UnixFS	Path	DHT	
IPLD	IPNS	Kademlia	

File Chunks:

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

Chunking	CID	Routing	Bitswap
UnixFS	Path	DHT	
IPLD	IPNS	Kademlia	

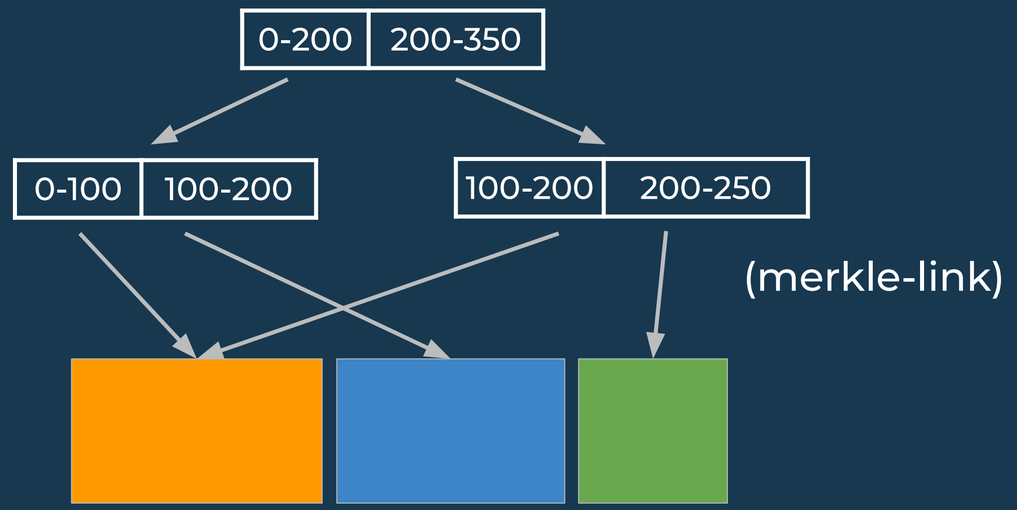


Import | **Name** | **Find** | **Fetch**

Chunking	CID	Routing	Bitswap
UnixFS	Path	DHT	
IPLD	IPNS	Kademlia	

(merkle-~~tree~~-***dag***) - directed acyclic graph

UnixFS File:

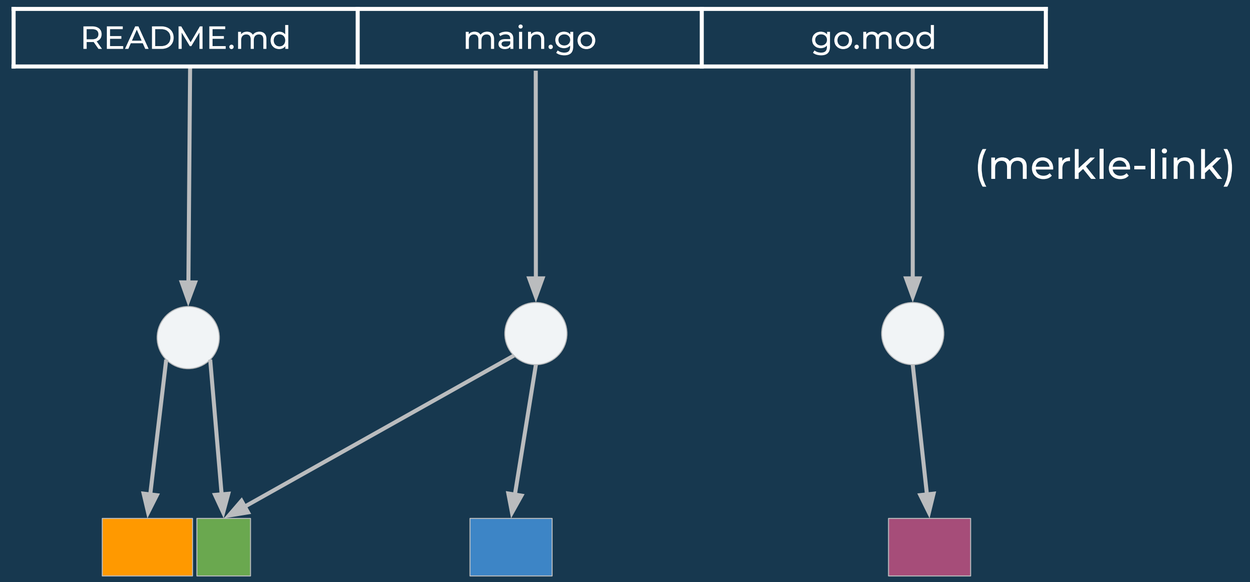


File Chunks:

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

Chunking	CID	Routing	Bitswap
UnixFS	Path	DHT	
IPLD	IPNS	Kademlia	

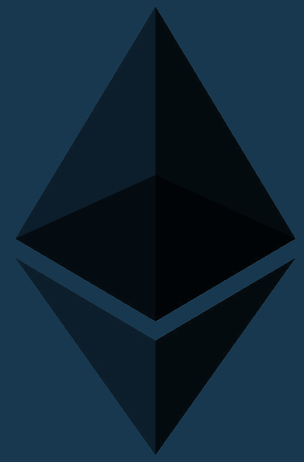
UnixFS Directory:



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

Chunking	CID	Routing	Bitswap
UnixFS	Path	DHT	
IPLD	IPNS	Kademlia	

Meta-format for understanding, encoding, and decoding merkle-linked data.



Chunking
UnixFS
IPLD

CID
Path
IPNS

Routing
DHT
Kademlia

Bitswap

Linked Data

```
http://b.com/Bar.json -> {
    "foo": http://a.com/Foo.json
}
http://a.com/Foo.json -> {
    "content": "I am foo"
}
```

Chunking
UnixFS
IPLD

CID
Path
IPNS

Routing
DHT
Kademlia

Bitswap

Linked Data

```
http://b.com/Bar.json -> {
    "foo": http://a.com/Foo.json
}
http://a.com/Foo.json -> {
    "content": "I am foo"
}
```

Authority

Chunking
UnixFS
IPLD

CID
Path
IPNS

Routing
DHT
Kademlia

Bitswap

Merkle-Linked Data

```
QmBar -> {
    "foo": QmFoo
}
QmFoo -> {
    "content": "I am foo"
}
```

- Immutable
- Authority Less

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

Content Identifier

QmS4ustL54uo8FzR9455qaxZwuMiUhyvMcX9Ba8nUH4uVv

bafybeibxm2nsadl3fnxv2sxcxmxaco2jl53wpeorjdzidjwf5aqdg7wa6u

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

Chunking
UnixFS
IPLD

CID
Path
IPNS

Routing
DHT
Kademlia

Bitswap

Digression:

Content Addressing / Location Addressing

Chunking
UnixFS
IPLD

CID
Path
IPNS

Routing
DHT
Kademlia

Bitswap

Digression: Content Addressing

Location Addressing

"My cat, Ozzy, is here."

Import | Name | Find | Fetch

Chunking
UnixFS
IPLD

CID
Path
IPNS

Routing
DHT
Kademlia

Bitswap

Content Addressing

"This is my cat, Ozzy."

Chunking
UnixFS
IPLD

CID
Path
IPNS

Routing
DHT
Kademlia

Bitswap

Digression: Content Addressing

Location Addressing

Chunking
UnixFS
IPLD

CID
Path
IPNS

Routing
DHT
Kademlia

Bitswap

Digression: Content Addressing

Location Addressing

Chunking
UnixFS
IPLD

CID
Path
IPNS

Routing
DHT
Kademlia

Bitswap

Digression: Content Addressing

Location Addressing

Chunking
UnixFS
IPLD

CID
Path
IPNS

Routing
DHT
Kademlia

Bitswap

Digression: Content Addressing

Location Addressing

"That's the wrong cat!"

Chunking
UnixFS
IPLD

CID
Path
IPNS

Routing
DHT
Kademlia

Bitswap

Digression: Content Addressing

Location Addressing

"That's the wrong cat!"

(But you can't know that!)

Import | **Name** | Find | Fetch

Chunking
UnixFS
IPLD

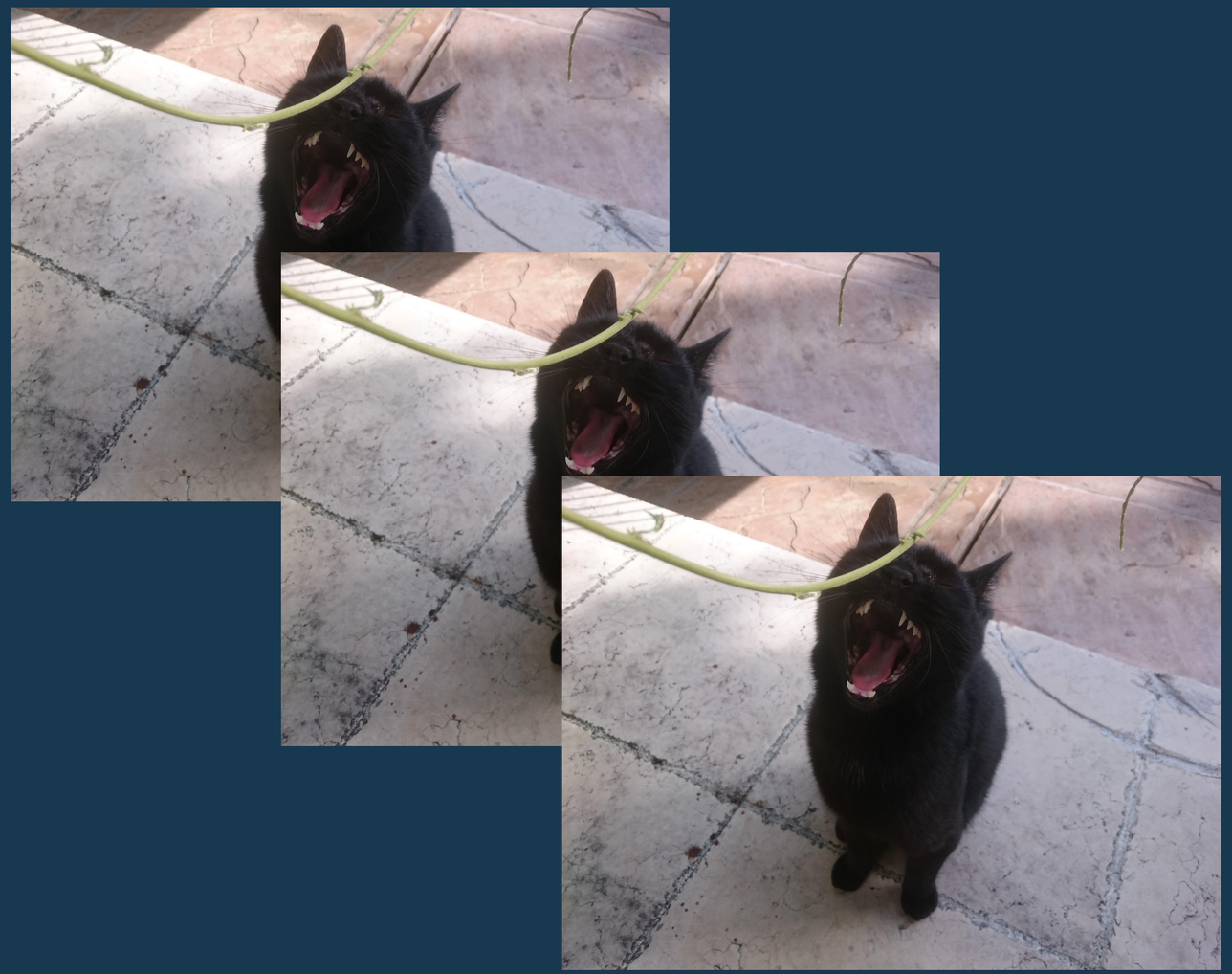
CID
Path
IPNS

Routing
DHT

Bitswap

Verifiable, Immutable, **Trustless**

Permanent



Chunking
UnixFS
IPLD

CID
Path
IPNS

Routing
DHT
Kademlia

Bitswap

Digression:

Multiformats: Self Describing Data

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

Digression: Multiformats

- 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	

Digression: Multiformats

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

Digression: Multiformats

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	

Digression: A bit of metadata

Multibase: a self-describing base encoding.

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

*b*afybeibxm2...

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

Self Describing

- CIDv0: **Qm**S4u...
  - Base58 encoded sha256 multihash
- CIDv1: **bafy**bei...
  - 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.

Chunking
UnixFS
IPLD

CID
Path
IPNS

Routing
DHT
Kademlia

Bitswap

IPFS uses **paths**, not **URIs/URLs**:

Like URLs, paths are **namespaced**:

/ipfs/QmFoo/welcome.txt
/ipns/QmBar/index.html

Unlike URLs, paths are ***recursive***:

/dns/github.com/tcp/22/ssh/git

Versus:

git+ssh://github.com:22

Not Composable!

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/Qm**SomethingNew**

IPNS can point to arbitrary paths

/ipns/QmMyKey -> /ipns/Qm**YourKey**

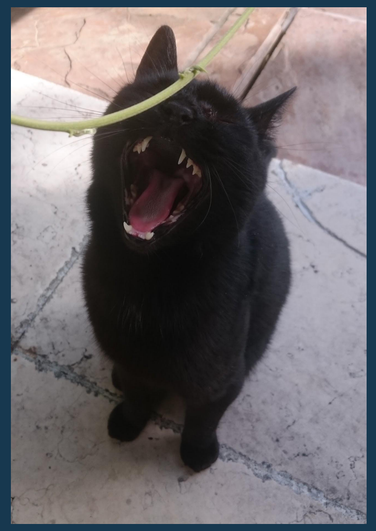
Import | **Name** | **Find** | Fetch

Chunking
UnixFS
IPLD

CID
Path
IPNS

Routing
DHT
Kademlia

Bitswap



Content Address (CID)

Location Address (Peer)

Chunking
UnixFS
IPLD

CID
Path
IPNS

Routing
DHT
Kademlia

Bitswap

Solution: Keep a "routing table"

What	Who
QmFoo	Ozzy
QmBar	Izzy

Chunking
UnixFS
IPLD

CID
Path
IPNS

Routing
DHT
Kademlia

Bitswap

But the table is too *BIG!*

What	Who
QmFoo	Ozzy
QmBar	Izzy
... millions of lines later ...	
QmXXXXXXXXXXXXXXX	Seth (not a cat!)

Chunking
UnixFS
IPLD

CID
Path
IPNS

Routing
DHT
Kademlia

Bitswap

Solution: *Distribute* the routing table and give a little bit to each peer.

Ozzy Knows

What	Who
QmBar	Izzy
...	

Izzy Knows

What	Who
QmFoo	Ozzy
...	

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

How do we know who has what piece of the routing table?

Chunking
UnixFS
IPLD

CID
Path
IPNS

Routing
DHT
Kademlia

Bitswap

How do we know who has what piece of the routing table?

Solution: *Deterministically* distribute the routing table.

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

Distance Metric: Is peer X closer to content C than peer Y?

Query Algorithm: Given the distance metric, how do we find the peers *closest* to C.

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

Distance Metric: `XOR(HASH(C), HASH(Peer))`

Query Algorithm:

1. Ask the closest peers you know for closer peers.
2. Remember the closest peers.

Chunking
UnixFS
IPLD

CID
Path
IPNS

Routing
DHT
Kademlia

Bitswap

Distance Metric: "Is this closer?"

Query Algorithm: "How do I get closer?"

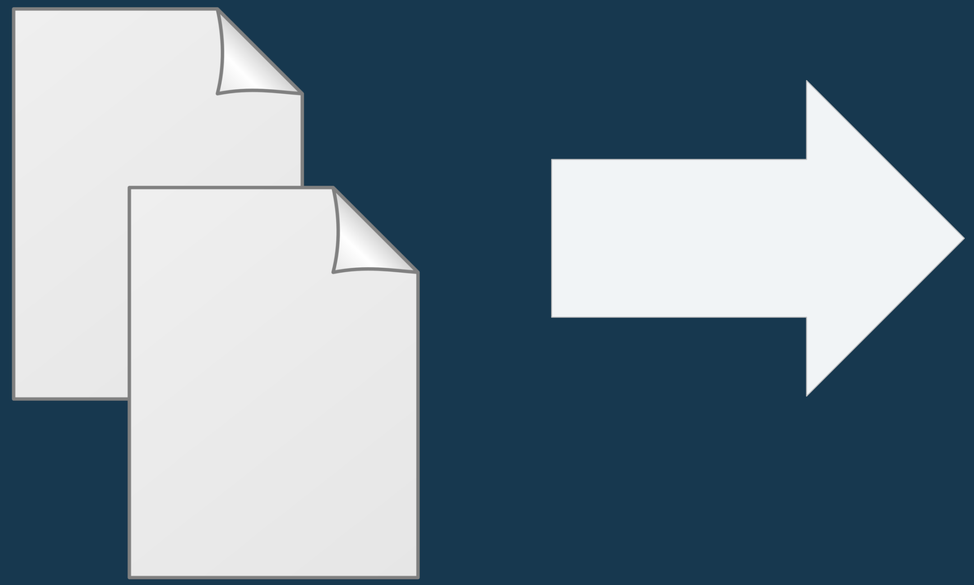
Import | **Name** | **Find** | **Fetch**

Chunking
UnixFS
IPLD

CID
Path
IPNS

Routing
DHT
Kademlia

Bitswap



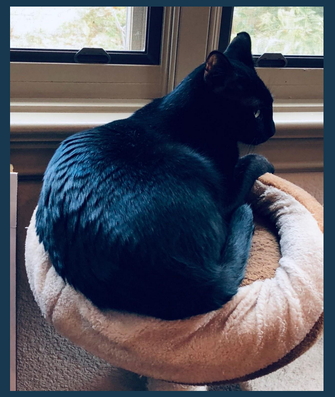
Import | Name | Find | Fetch

Chunking
UnixFS
IPLD

CID
Path
IPNS

Routing
DHT
Kademlia

Bitswap

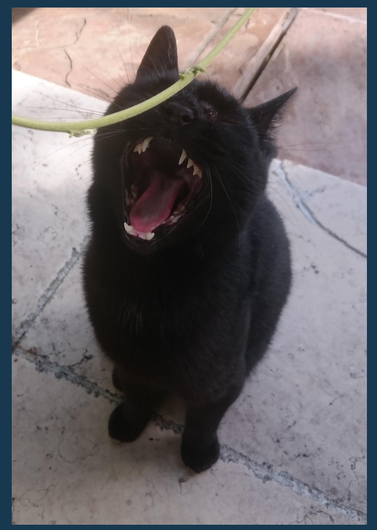


Izzy Wants

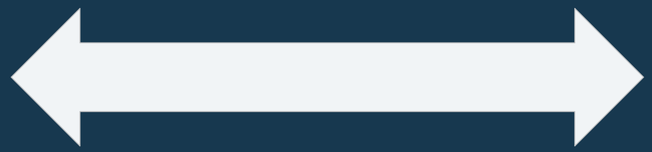
- **QmTreats**
- **QmToy**

Ozzy Wants

- **QmCuddles**
- **QmFood**
- **QmAttention**



Izzy



Ozzy

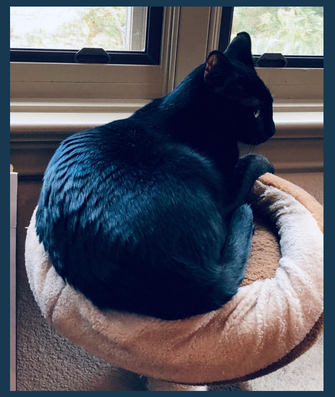
Import | Name | Find | Fetch

Chunking
UnixFS
IPLD

CID
Path
IPNS

Routing
DHT
Kademlia

Bitswap



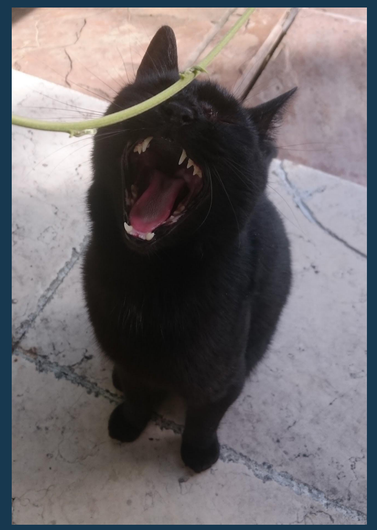
Izzy

Ozzy Wants

- QmCuddles
- QmFood
- QmAttention

Izzy Wants

- QmTreats
- QmToy



Ozzy

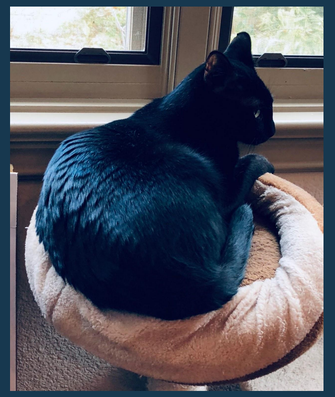
Import → Name → Find → Fetch

Chunking
UnixFS
IPLD

CID
Path
IPNS

Routing
DHT
Kademlia

Bitswap



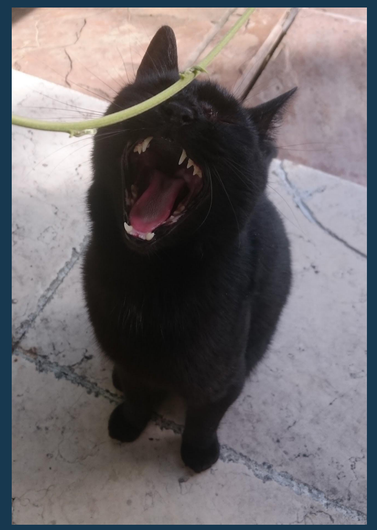
Izzy

Ozzy Wants

- QmCuddles
- *QmFood*
- *QmAttention*

Izzy Wants

- QmTreats
- *QmToy*



Ozzy

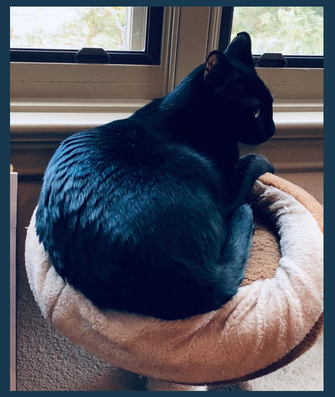
Import → Name → Find → Fetch

Chunking
UnixFS
IPLD

CID
Path
IPNS

outing
HT
demlia

Bitswap



Izzy

Izzy Wants

- QmTreats
- *QmToy*

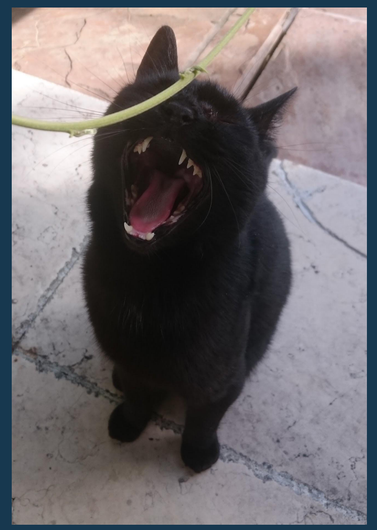
QmToy

Ozzy Wants

- QmCuddles
- *QmFood*
- *QmAttention*

QmFood

QmAttention



Ozzy

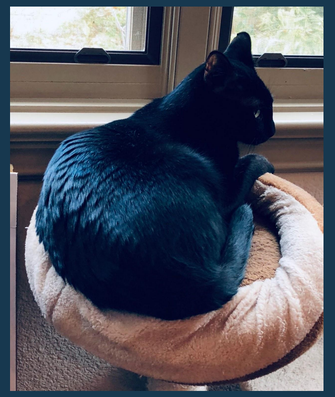
Import → Name → Find → Fetch

Chunking
UnixFS
IPLD

CID
Path
IPNS

Routing
DHT
Kademlia

Bitswap



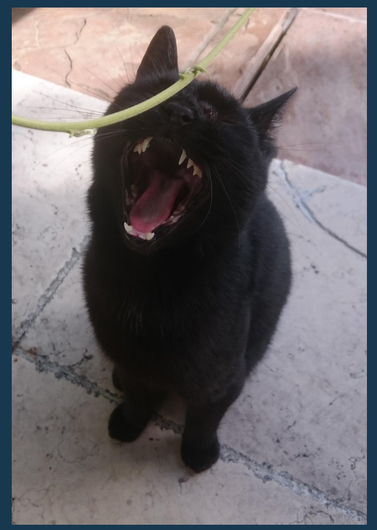
Izzy

Ozzy Wants

- QmCuddles

Izzy Wants

- QmTreats



Ozzy

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

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

How IPFS Works

(approximately)

Sawood Alam (@*ibnesayeed*)
Old Dominion University

Original slides by Steven Allen (@*Stebalien*)