Decentralized Storage with IPFS

How does it work under the hood?

FOSDEM - February 2023

Dennis Trautwein

Research Engineer

Protocol Labs

Who am I?

Dennis Trautwein

- Research Engineer **@Protocol Labs**
- Industrial Ph.D. candidate **@University of Göttingen**

@dennis-tra on **GitHub**
@dtrautwein_eu on **Twitter**
https://dtrautwein.eu on the **Web**
dennis@protocol.ai via **Email**

Today's

Agenda

- What is IPFS?
- Importing Content
- Connecting to the Network
- Content Routing
- Call Outs

WHAT IS IPFS?

What is IPFS?

In Words

stands for the InterPlanetary File System

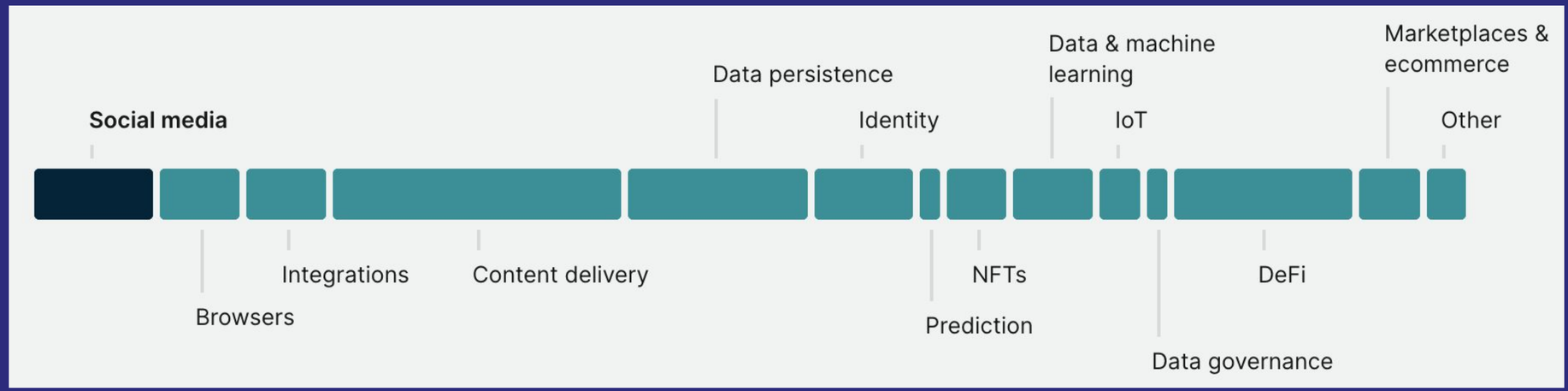
IPFS is a **decentralized storage and delivery network** which builds on *P2P networking* and *content-based addressing*.

IPFS is **not** a blockchain.

What is IPFS?

In Numbers

- Operational since 2015
- # of Requests: >> 1B requests (weekly)
- Volume of Traffic: hundreds of TBs
- Unique weekly users: tens of Ms



Disclaimer: These are estimates from our vantage points. IPFS is a decentralized network. Noone has a full view of the network. Real numbers are likely to be much higher than those.

What is IPFS?

Value Proposition

- Decouples content from hosts
- Permanent, verifiable links
- Censorship resistance
- Alleviate backbone addiction
  - Efficient bandwidth use
  - Offline friendly
  - Emerging networks

What is IPFS?

Value Proposition

Location Addressing fails on us

- URL points to a single copy
- No way to know where other copies are
- Not possible to validate integrity
  - e.g., DNS poisoning, change copy
- No Request Aggregation

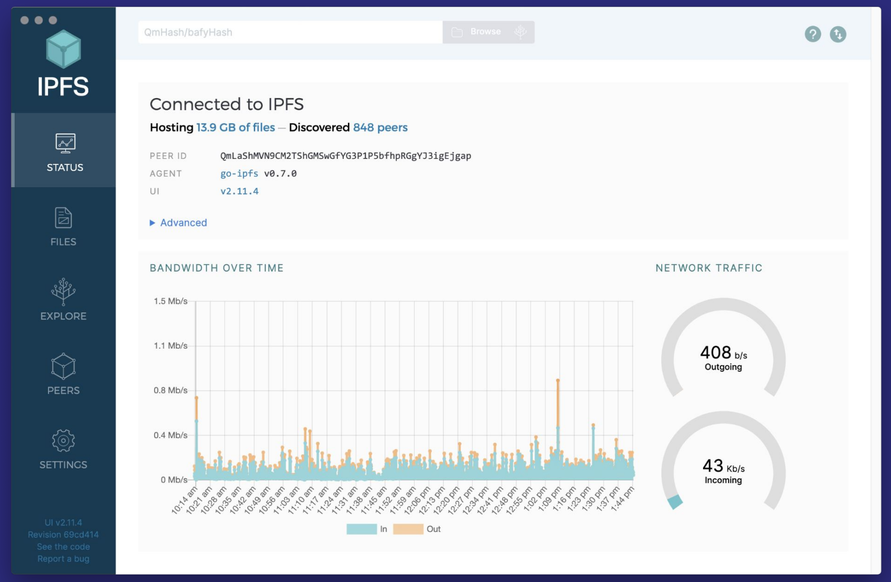
- Emerging Networks
- Offline Use
- Censorship
- Breaking Links

What is IPFS?

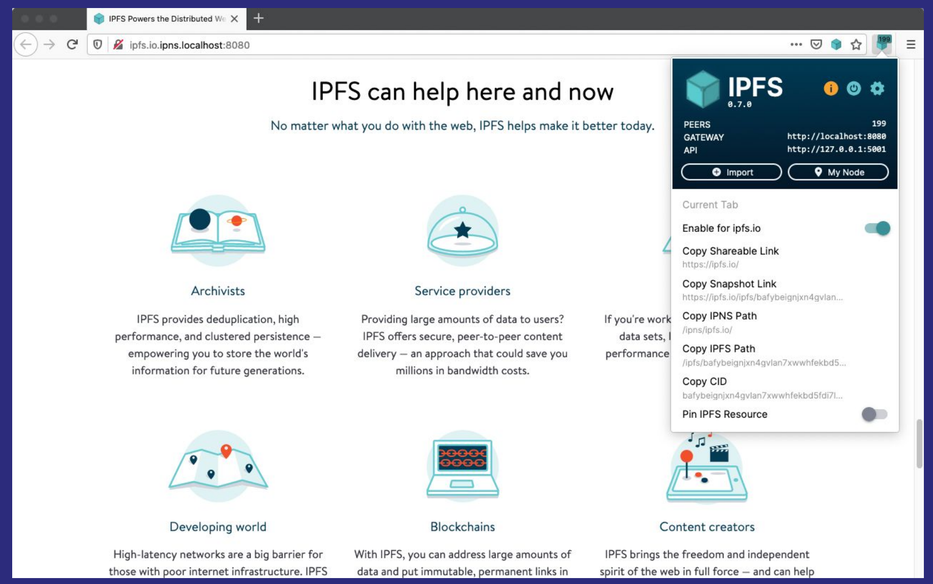
Installation

COMMAND LINE

IPFS Desktop



IPFS Companion



Brave/Opera

IMPORTING CONTENT

Importing Content

ipfs init

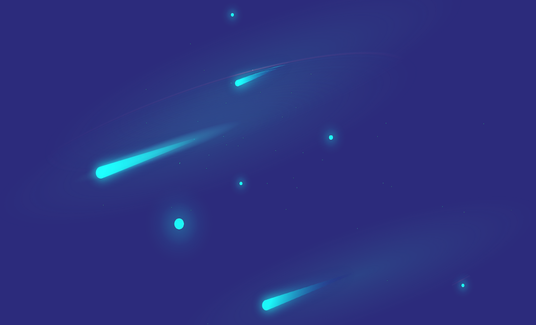
```
$ ipfs init
generating ED25519 keypair...done
peer identity: 12D3KooWB7JBgNBj3SaNujhY14zfGcxxWphFJPv8YDUkygtX9ETR
initializing IPFS node at /home/<user>/.ipfs
to get started, enter:

    ipfs cat /ipfs/QmQPeNsJPyVWPFDVHb77w8G42Fvo15z4bG2X8D2GhfbSXc/readme
```

- Generates PeerID
- Initializes IPFS Repository
- Just a Local Operation

Importing Content

ipfs add FILE

```
$ ipfs add kubo_v0.15.0_linux-amd64.tar.gz
added QmdVRTMRe9HiWBiFbyHveM4mKpKqvifo9CBEuTSGNuBgKZ kubo_v0.15.0_linux-amd64.tar.gz
 31.85 MiB / 31.85 MiB [============================================================] 100.00%
```



Importing Content

Content Identifier

```
$ ipfs add kubo_v0.15.0_linux-amd64.tar.gz
added QmdVRTMRe9HiWBiFbyHveM4mKpKqvifo9CBEuTSGNuBgKZ kubo_v0.15.0_linux-amd64.tar.gz
 31.85 MiB / 31.85 MiB [============================================================] 100.00%
```

Content Identifier (CID)

- Most Fundamental Ingredient
- Hash with Metadata
- Self-Describing
- Self-Certifying
- Immutable

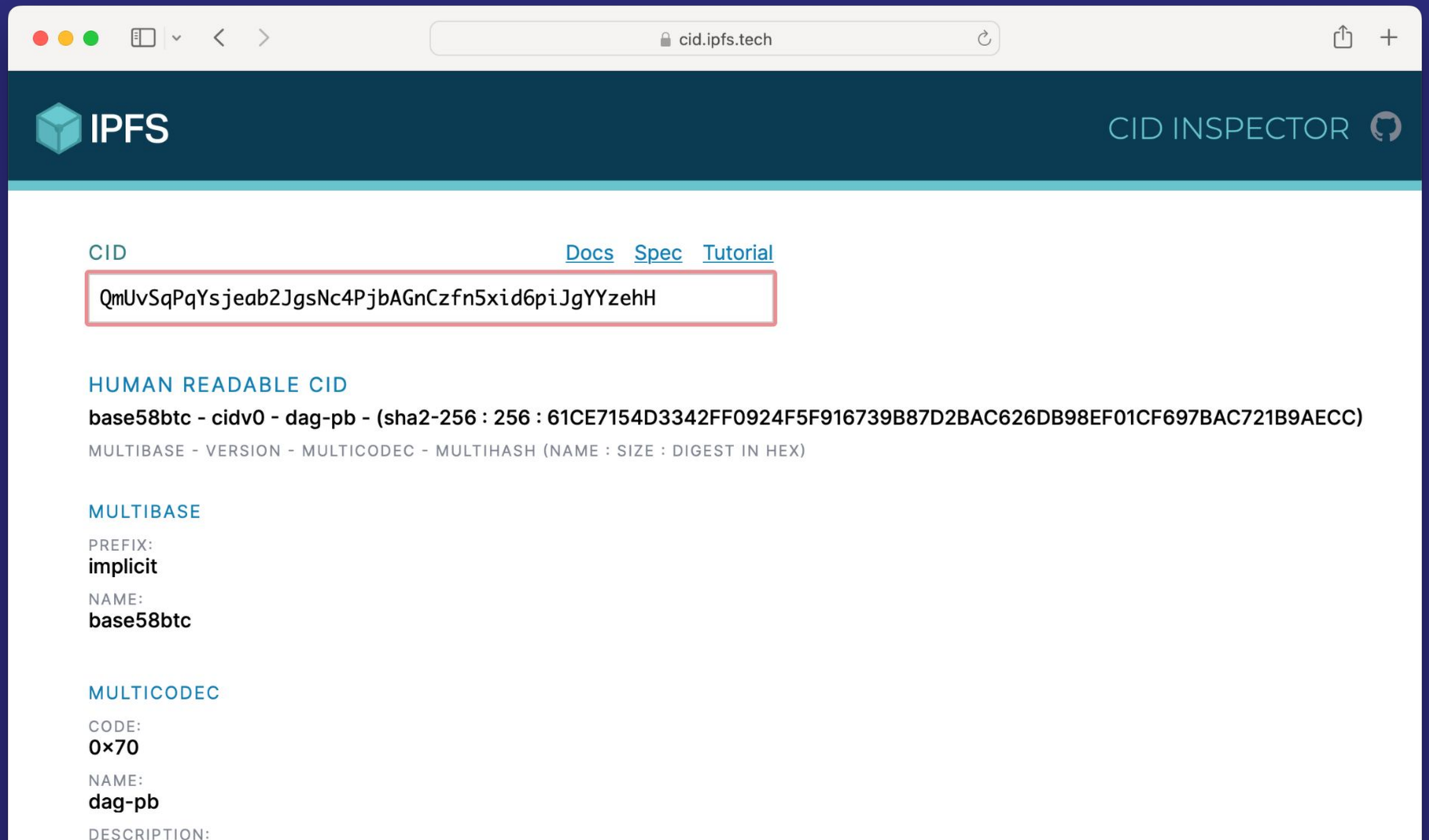
Multiformats

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

Importing Content

CID Inspector

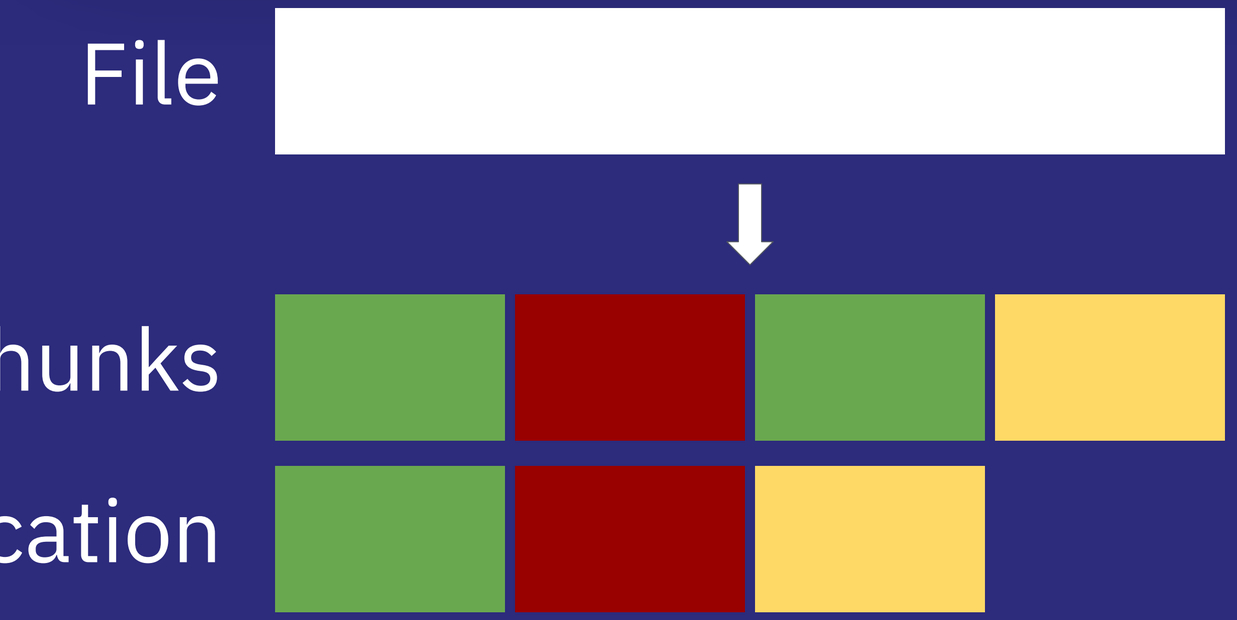
https://cid.ipfs.tech



Importing Content

Chunking

```
$ ipfs add kubo_v0.15.0_linux-amd64.tar.gz
added QmdVRTMRe9HiWBiFbyHveM4mKpKqvifo9CBEuTSGNuBgKZ kubo_v0.15.0_linux-amd64.tar.gz
 31.85 MiB / 31.85 MiB [============================================================] 100.00%
```



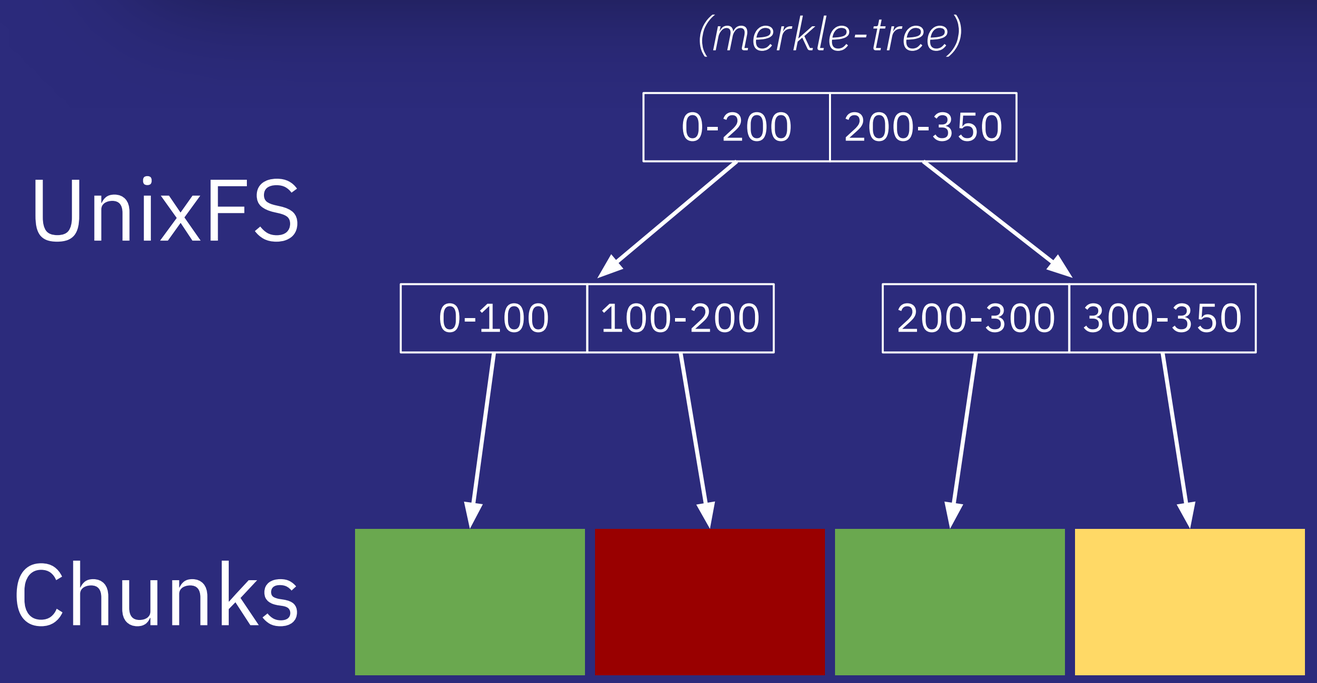
- Piecewise Transfer
- Deduplication
- Random Access

(Each Chunk Hashed)

Importing Content

UnixFS

```
$ ipfs add kubo_v0.15.0_linux-amd64.tar.gz
added QmdVRTMRe9HiWBiFbyHveM4mKpKqvifo9CBEuTSGNuBgKZ kubo_v0.15.0_linux-amd64.tar.gz
 31.85 MiB / 31.85 MiB [============================================================] 100.00%
```



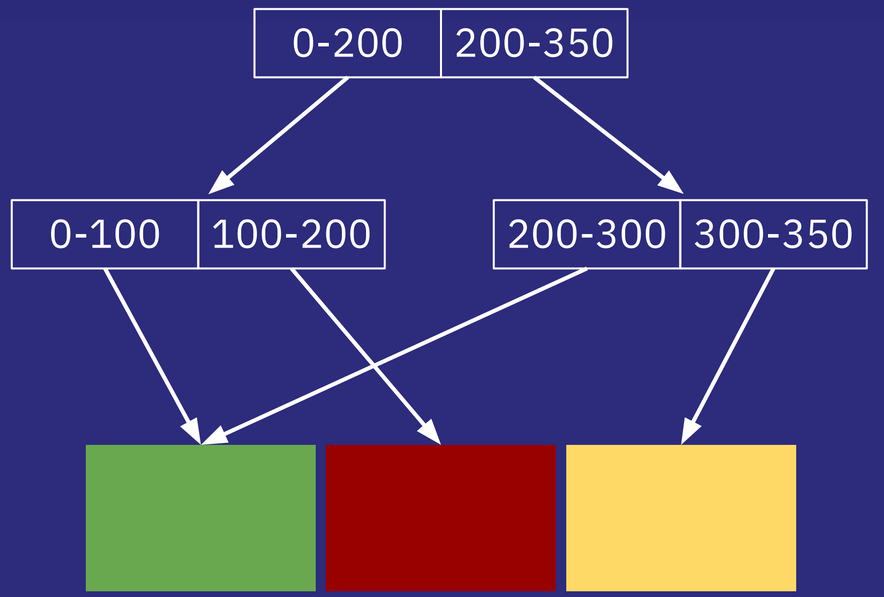
Importing Content

UnixFS

```
$ ipfs add kubo_v0.15.0_linux-amd64.tar.gz
added QmdVRTMRe9HiWBiFbyHveM4mKpKqvifo9CBEuTSGNuBgKZ kubo_v0.15.0_linux-amd64.tar.gz
 31.85 MiB / 31.85 MiB [============================================================] 100.00%
```

UnixFS

*(merkle-~~tree~~-**dag**)*



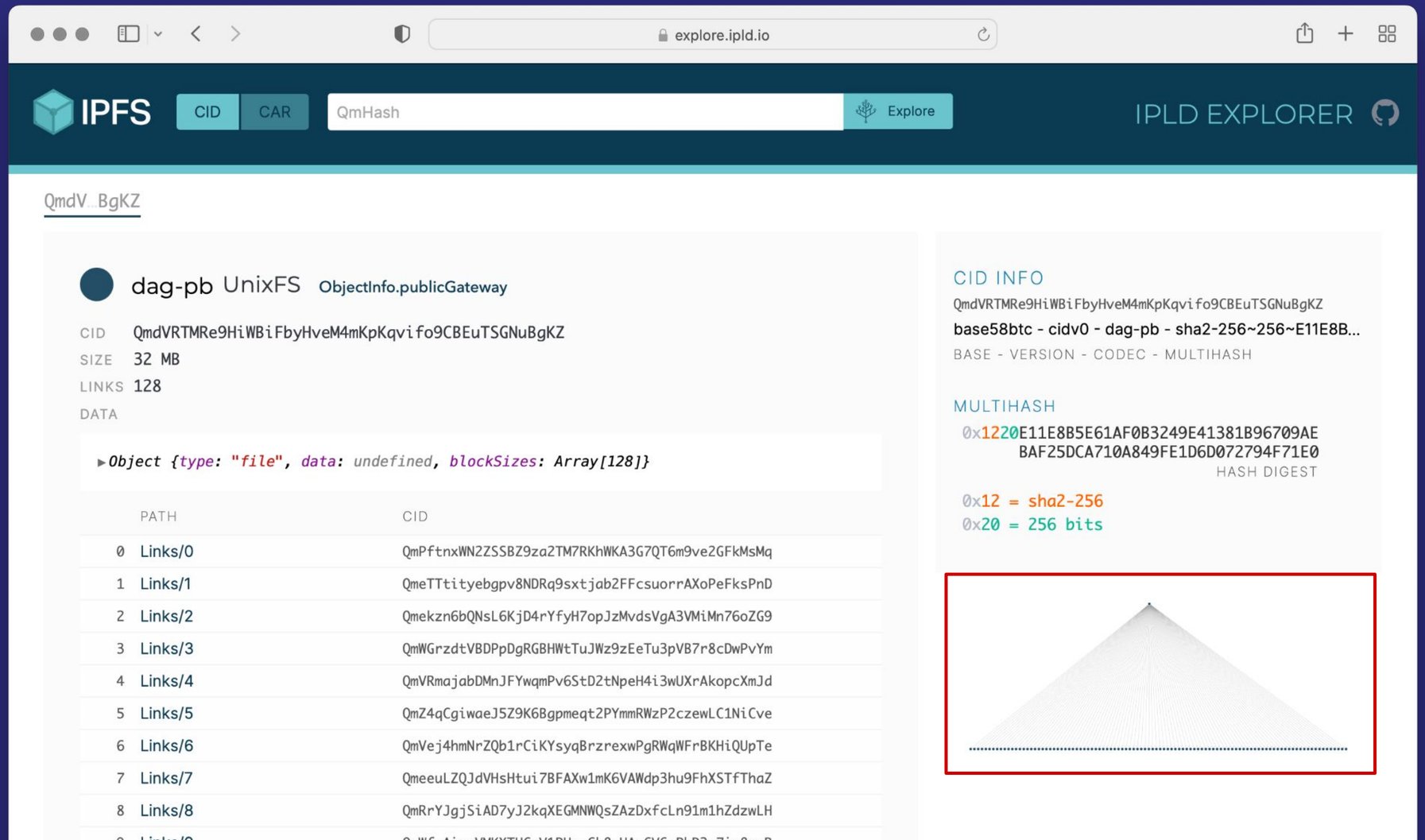
Chunks

IPLD

Importing Content

IPLD Explorer

https://explore.ipld.io



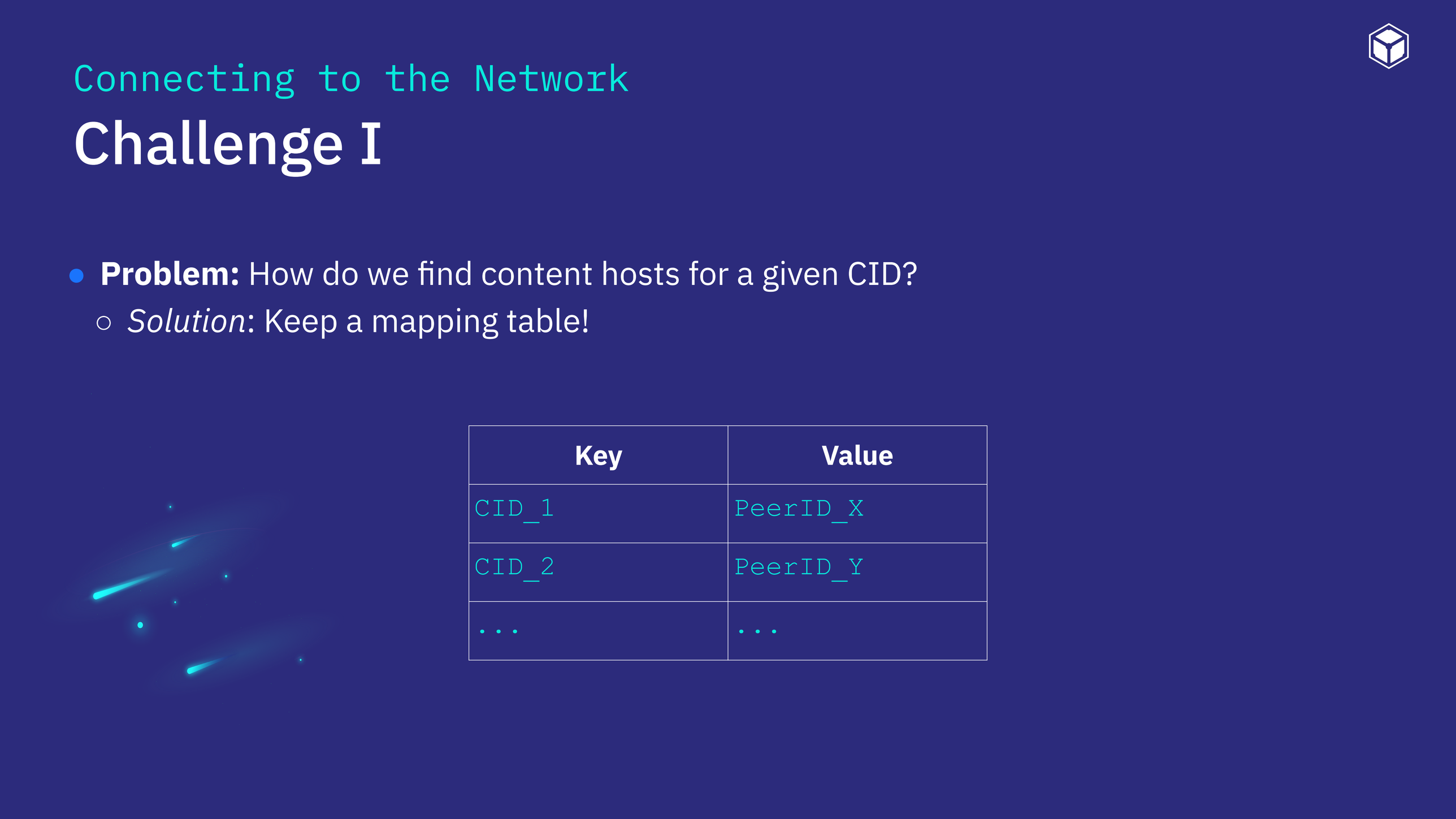
CONNECTING TO THE NETWORK

Connecting to the Network

Daemon

- Long running network-connected IPFS node
- Connects to Bootstrap Peers
- Learns about other Peers
- Announces itself to the Network

```
$ ipfs daemon
Initializing daemon...
Kubo version: 0.15.0
Repo version: 12
System version: amd64/linux
Golang version: go1.18.5
Swarm listening on /ip4/xxx.xxx.xxx.xxx/tcp/4001
Swarm listening on /ip4/xxx.xxx.xxx.xxx/udp/4001/quic
Swarm listening on /ip4/127.0.0.1/tcp/4001
Swarm listening on /ip4/127.0.0.1/udp/4001/quic
Swarm listening on /ip6/xxxx:xxxx:xxxx:xxxx::1/tcp/4001
Swarm listening on /ip6/xxxx:xxxx:xxxx:xxxx::1/udp/4001/quic
Swarm listening on /ip6/::1/tcp/4001
Swarm listening on /ip6/::1/udp/4001/quic
Swarm listening on /p2p-circuit
Swarm announcing /ip4/xxx.xxx.xxx.xxx/tcp/4001
Swarm announcing /ip4/xxx.xxx.xxx.xxx/udp/4001/quic
Swarm announcing /ip4/127.0.0.1/tcp/4001
Swarm announcing /ip4/127.0.0.1/udp/4001/quic
Swarm announcing /ip6/xxxx:xxxx:xxxx:xxxx::1/tcp/4001
Swarm announcing /ip6/xxxx:xxxx:xxxx:xxxx::1/udp/4001/quic
Swarm announcing /ip6/::1/tcp/4001
Swarm announcing /ip6/::1/udp/4001/quic
API server listening on /ip4/127.0.0.1/tcp/5001
WebUI: http://127.0.0.1:5001/webui
Gateway (readonly) server listening on /ip4/127.0.0.1/tcp/8080
Daemon is ready
```



Connecting to the Network

Challenge I

- **Problem:** How do we find content hosts for a given CID?
  - *Solution*: Keep a mapping table!

Key	Value
CID_1	PeerID_X
CID_2	PeerID_Y
...	...

Connecting to the Network

Challenge II

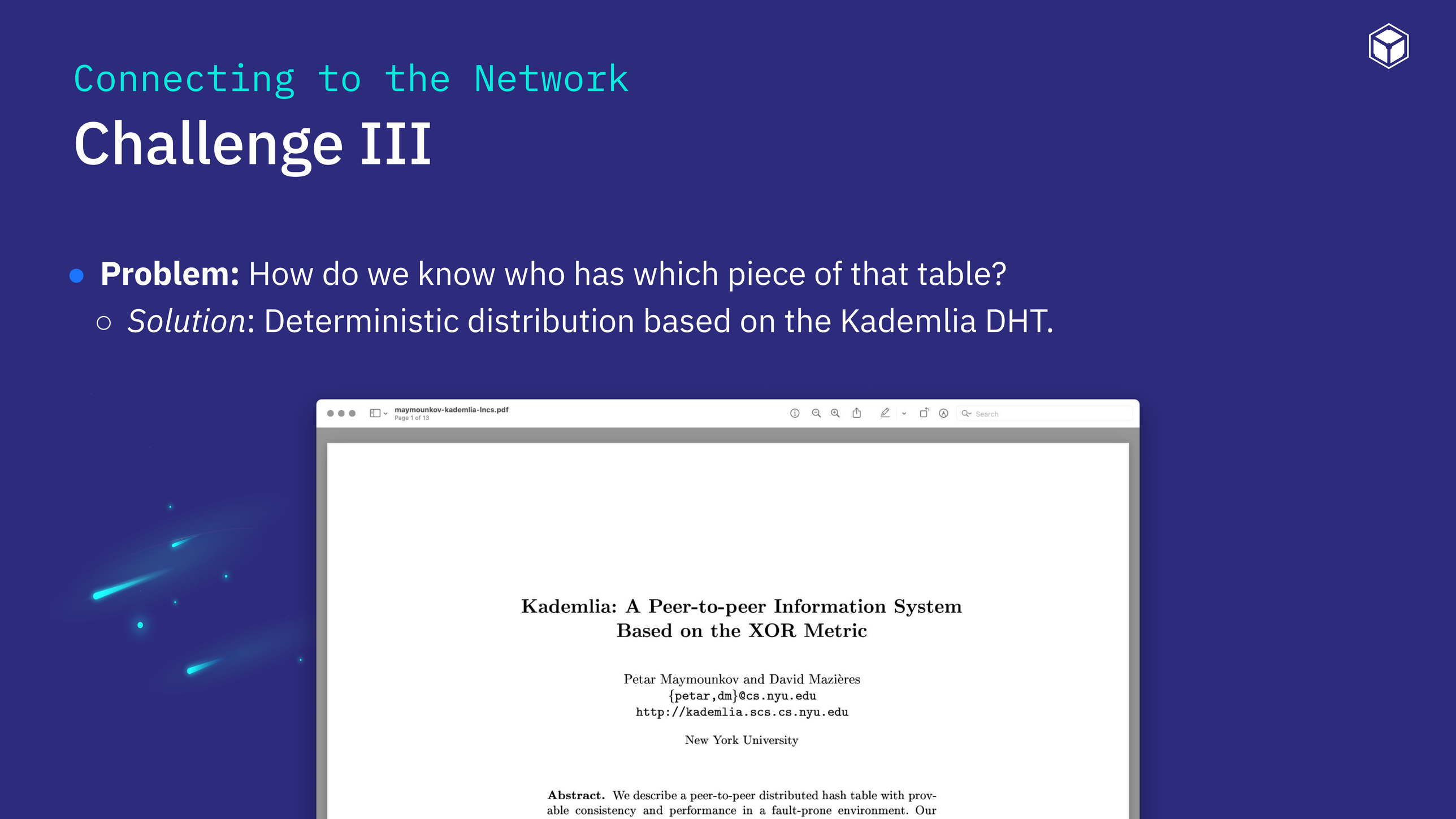
- **Problem:** The mapping table gets too big!
  - *Solution*: Split and distribute the table to each participating peer

PeerID_X

Key	Value
CID_1	PeerID_Y
...	...

PeerID_Y

Key	Value
CID_2	PeerID_X
...	...



Connecting to the Network

Challenge III

- **Problem:** How do we know who has which piece of that table?
  - *Solution*: Deterministic distribution based on the Kademlia DHT.

Kademlia: A Peer-to-peer Information System Based on the XOR Metric

Petar Maymounkov and David Mazières
{petar,dm}@cs.nyu.edu
http://kademlia.scs.cs.nyu.edu

New York University

Abstract. We describe a peer-to-peer distributed hash table with provable consistency and performance in a fault-prone environment. Our

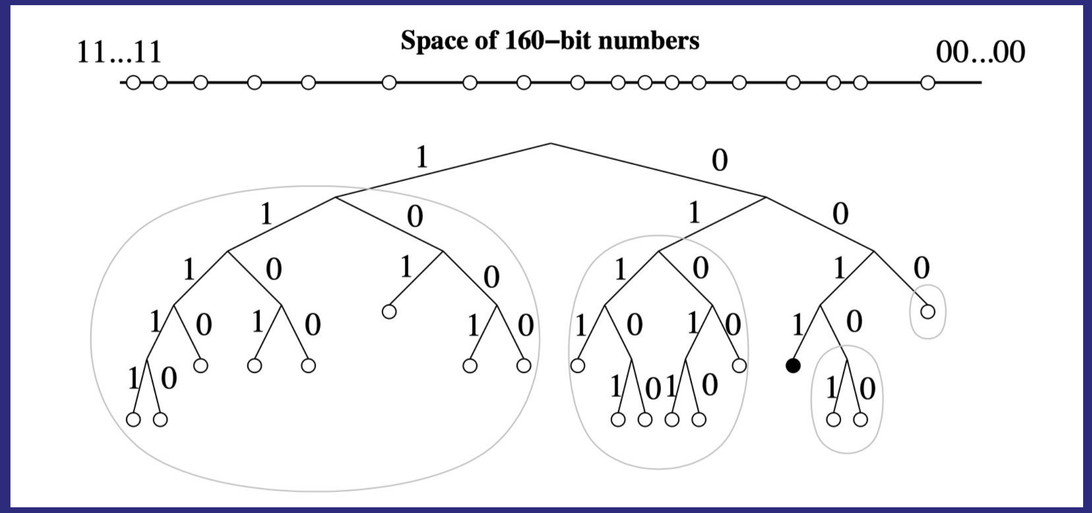
CONTENT ROUTING

Connecting to the Network

Distributed Hash Table

- IPFS uses adaptation of the Kademlia DHT
  - 256 bit key space (SHA256)
- Distributed system that maps keys to values
  - 2-column table
  - **Provider Records:** CID -> PeerID
  - **Peer Records:** PeerID -> Network Addresses
- Two key features:
  - XOR Distance Metric: XOR
    - notion of closeness (not geographically!)
  - Tree-based routing
- O(log N) lookups

Key	Value
CID_1	PeerID_X
CID_2	PeerID_Y
PeerID_X	Network Addr.
...	...



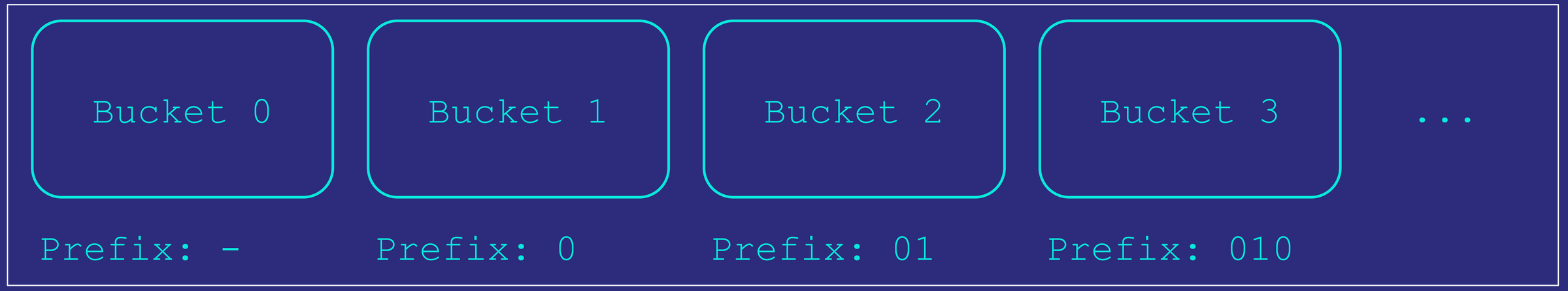
Connecting to the Network

Bootstrapping

- Calculate SHA256 of own PeerID
- Ask bootstrap nodes if they know peers whose SHA256(PeerID) start with:
  - `1...` (no common prefix)
  - `00...` (one common prefix)
  - `011...` (two common prefixes)

01001110010000010111...

Routing Table



Connecting to the Network

Retrieving Content

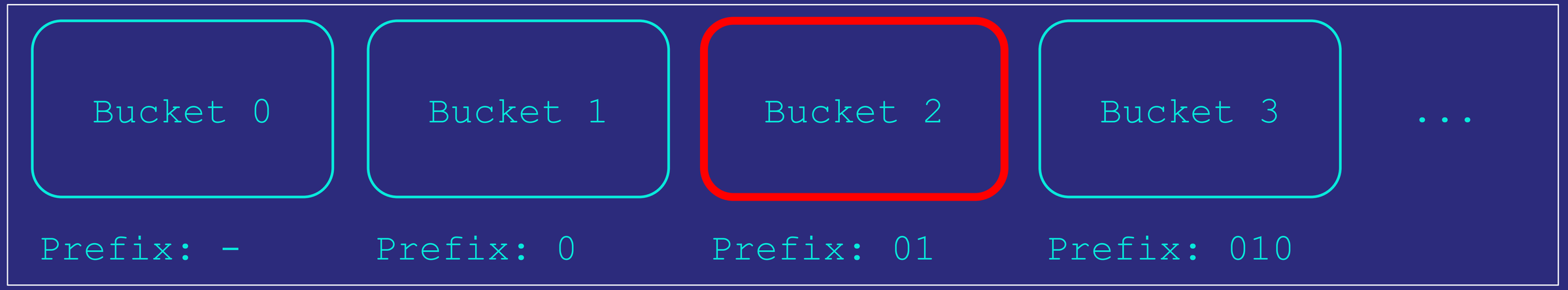
```
$ ipfs get QmUvSqPqYsjeab2JgsNc4PjbAGnCzfn5xid6piJgYYzehH
Saving file(s) to QmUvSqPqYsjeab2JgsNc4PjbAGnCzfn5xid6piJgYYzehH
 2.10 MiB / 2.10 MiB [================================] 100.00% 0s
```

- Calculate SHA256 of CID

0111101111100010101111...

- Locate appropriate bucket
- Get list of peers in that bucket
- Start parallel request for that CID
  - If peer know that CID: Returns Provider Record (CID -> PeerID mapping)
  - If peer doesn't know that CID: Returns list of closer peers

Routing Table



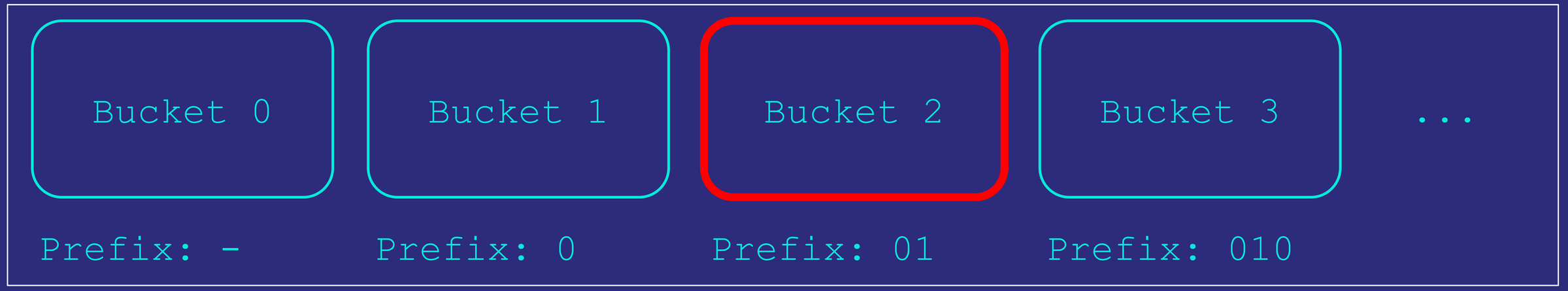
Connecting to the Network

Publishing Content

- Calculate SHA256 of CID
- Locate appropriate bucket
- Get list of peers in that bucket
- Start parallel request for closer peer to that CID
- Terminate when the closest known three peers have been successfully queried
- Store Provider Record with the 20 closest peers

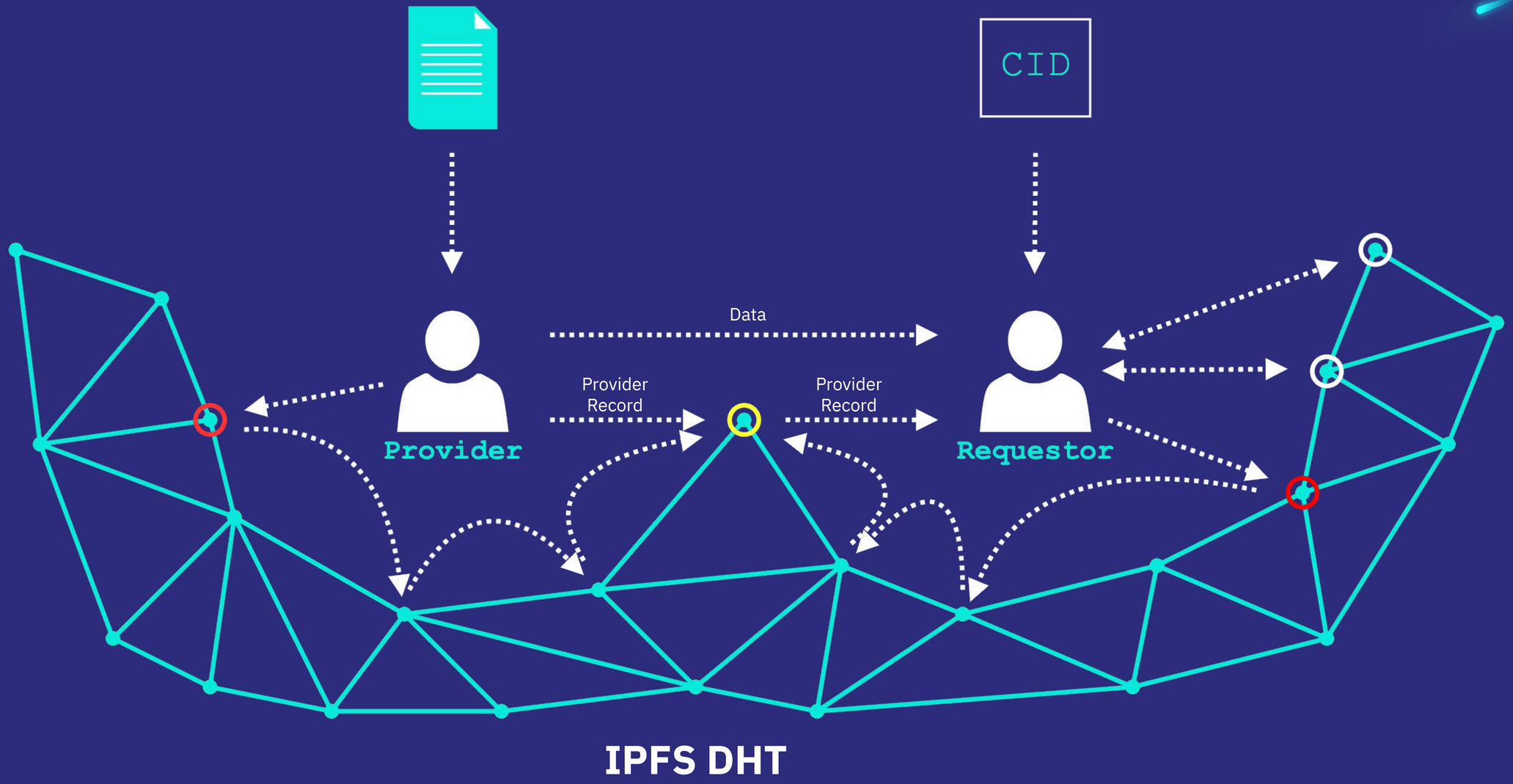
0111101111100010101111...

Routing Table



Summary

Content Lifecycle



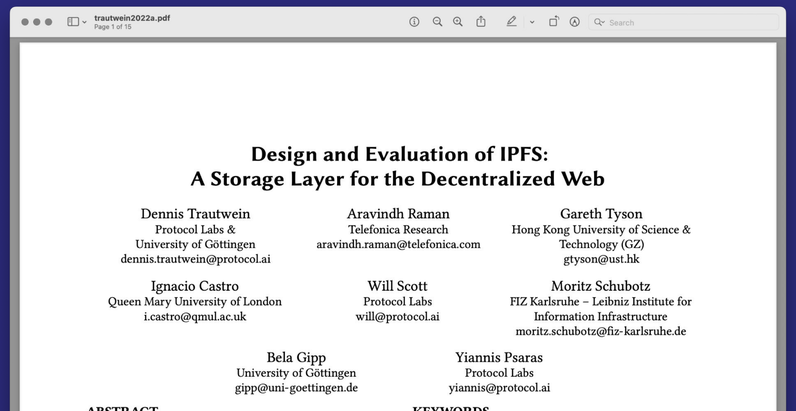
CALL OUTS

Future/Open Projects

Get Involved!

ACM SigCOMM ‘22

QmUvSqPqYsjeab2JgsNc4PjbAGnCzfn5xid6piJgYYzehH


trautwein2022a.pdf
Page 1 of 15

Design and Evaluation of IPFS:
A Storage Layer for the Decentralized Web

Dennis Trautwein
Protocol Labs &
University of Göttingen
dennis.trautwein@protocol.ai

Aravindh Raman
Telefonica Research
aravindh.raman@telefonica.com

Gareth Tyson
Hong Kong University of Science & Technology (GZ)
gtyson@ust.hk

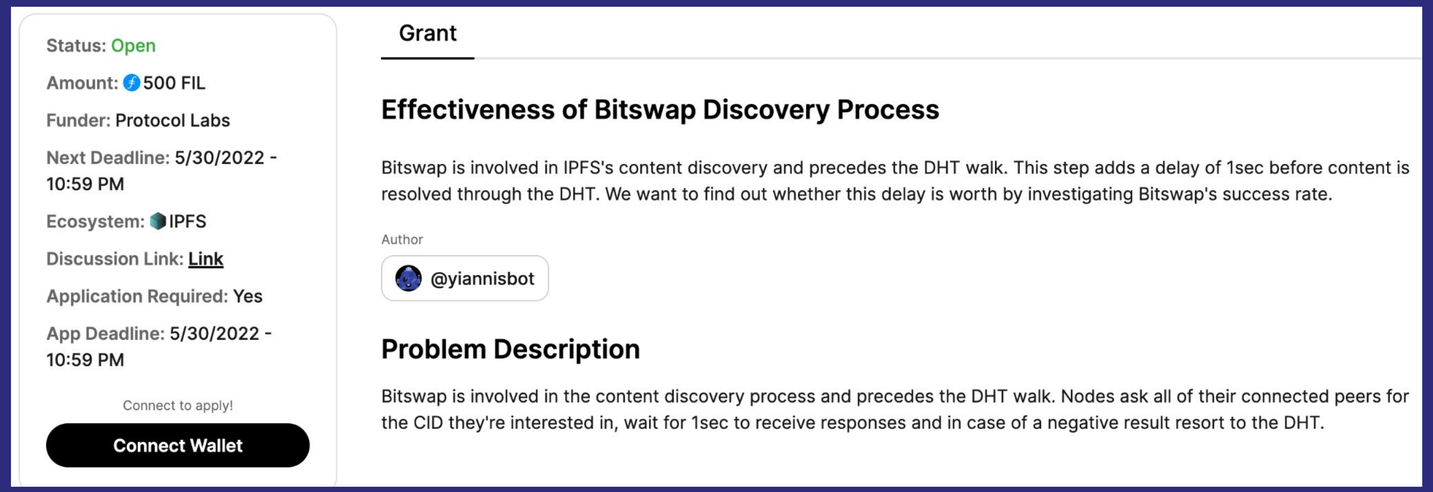
Ignacio Castro
Queen Mary University of London
i.castro@qmul.ac.uk

Will Scott
Protocol Labs
will@protocol.ai

Moritz Schubotz
FIZ Karlsruhe – Leibniz Institute for Information Infrastructure
moritz.schubotz@fiz-karlsruhe.de

Bela Gipp
University of Göttingen
gipp@uni-goettingen.de

Yiannis Psaras
Protocol Labs
yiannis@protocol.ai



- **Funding!**
  - Several grants open at: **https://app.radius.space/**
  - Get your application in!
- **All the action is public!**
  - Check the Network Measurements GH repo: **https://github.com/protocol/network-measurements**
  - More than 15 Requests for Measurements (RFMs)
  - Extra ideas very welcome!

THANK YOU!

@dennis-tra on **GitHub**
@dtrautwein_eu on **Twitter**
https://dtrautwein.eu on the **Web**
dennis@protocol.ai via **Email**

Mutability

IPNS

- InterPlanetary Name System (IPNS)
- Content addressing in IPFS is immutable by nature
- E.g., publishing a website requires mutable pointer