Seminário: InterPlanetary FileSystem (IPFS)

Pedro Magalhães
pmagalhaes@inf.puc-rio.br

Índice

Introdução

- O que é?
  - Sistema distribuido (P2P) de arquivos
  - Endereçamento por conteúdo (diferente do tradicional endereçamento por caminho)
    - Quero o arquivo “x” (consulta por conteúdo)
    - quero o arquivo “/caminho/para/x” (consulta por caminho)
  - Usa hash distribuído para busca de conteúdo
  - Armazenamento permanente com versionamento, uma vez que o arquivo está na rede ele não pode ser deletado. Podemos apenas versiona-lo. (Muito embora ele possa ser inacessível se nenhum nó possuir uma cópia)
  - Paper
  - https://ipfs.io/

Introdução

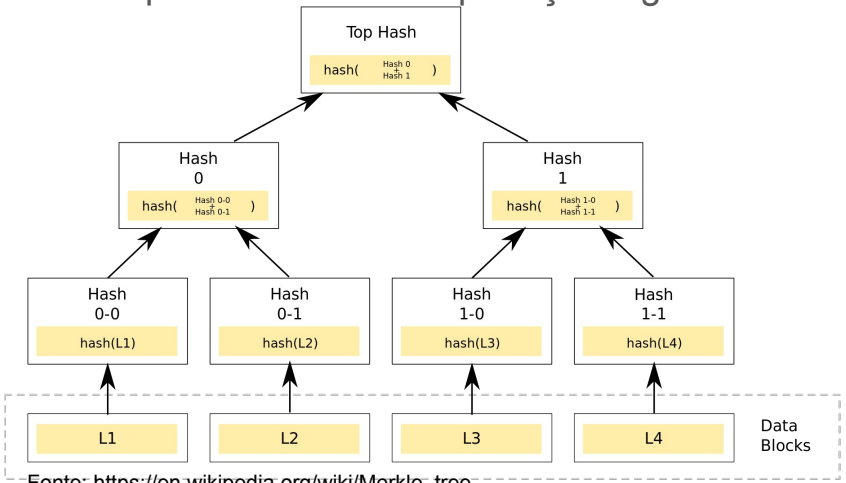
- Porque?
  - Independência de servidores centralizados***
  - Dificulta a censura pois não há um servidor central que pode ser bloqueado.
    - Ex: Turquia bloqueou o acesso à Wikipédia e uma cópia foi feita no IPFS
  - Teoricamente pode tornar a latência menor quando os arquivos estão mais próximos dos usuário.
  - Distribuição dos dados (P2P)

Objetos

- Os objetos são imutáveis
  - Não há preocupação com cache e cópias desatualizadas
- Objetos podem conter links para outros objetos
  - representar por exemplo um diretório com outros diretórios e arquivos,
  - ou um arquivo muito grande que foi quebrado em pedaços menores
- Publicando objetos
  - Primeiro temos que adicionar a chave do objeto ao DHT, entrar na rede com um nó servindo o objeto e passar a chave para outros usuários
  - Os objetos só são distribuídos na rede (fora dos nós que publicaram) quando existe demanda para esses objetos

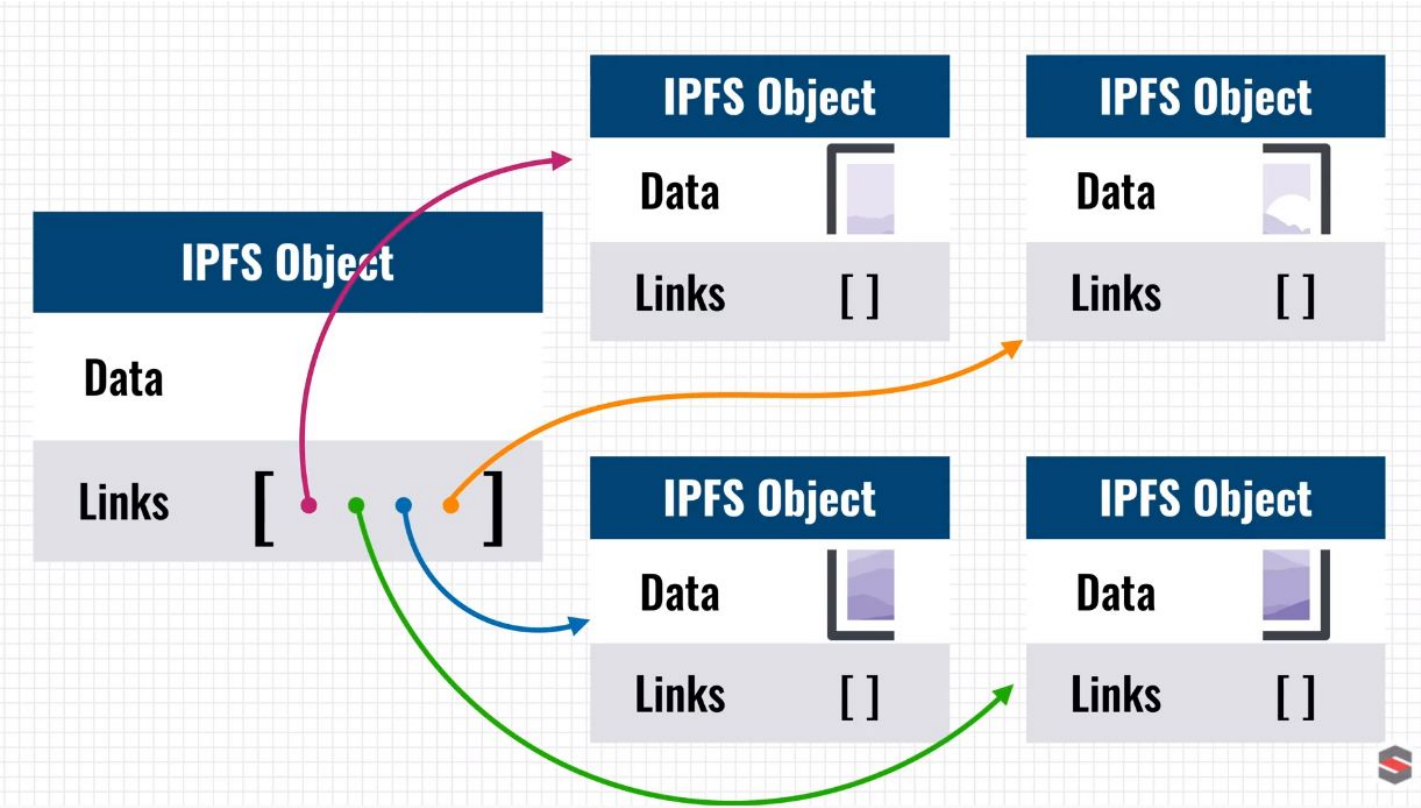
Objetos

- Merkle DAG
  - Todo objeto guardado da rede é identificado de forma única através do “multihash” do seu conteúdo
  - A rede detecta se um arquivo foi alterado ou corrompido
  - Objetos duplicados são identificados e viram uma cópia do dado original
  - Cada objeto é decomposto em diversos pedaços e gera um “Merkle DAG”



Fonte: https://en.wikipedia.org/wiki/Merkle_tree

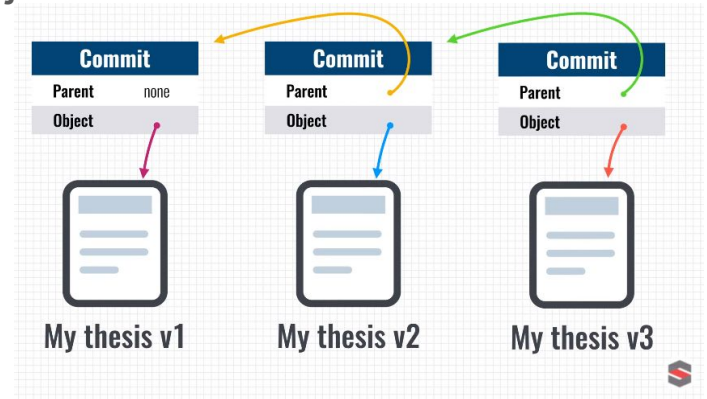
Objetos



Fonte: https://www.youtube.com/watch?v=5Uj6uR3fp-U

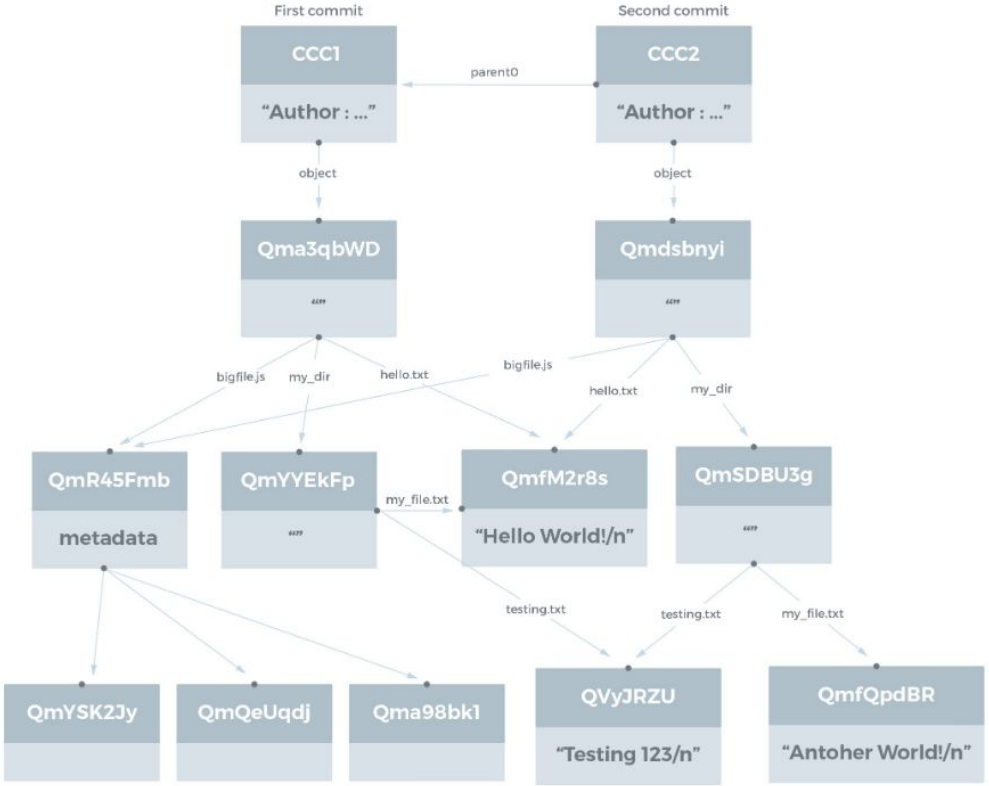
Objetos - Arquivos

- Uso como um sistema de arquivos comum (podemos salvar pastas)
- Segundo o artigo e algumas referências podemos fazer o versionamento de arquivos e pastas, mas não encontrei muitas de como funciona, só que é similar ao git pois usa uma estrutura do DAG de merkle parecida.
- Podemos usar “path” comuns a partir do hash de uma pasta. meuHash/foo.txt, vai procurar o foo.txt no objeto com hash “meuHash”



Fonte: https://www.youtube.com/watch?v=5Uj6uR3fp-U

Objetos - Versionamento de arquivos



Fonte: https://medium.com/@ConsenSys/an-introduction-to-ipfs-9bba4860abd0

Design - Identidade

- Nós
  - Cada nó possui uma chave pública e uma privada. O nó é identificado por um “NodeId” que é o hash de sua chave pública
  - Quando um nó se conecta a outro eles trocam suas chaves públicas e cada um checa se o NodeId do “peer” está correto, se não a conexão é terminada.
  - Para entrar na rede um nó precisa de o endereço de nós que já fazem parte da rede
  - Cada nó inicializa sua tabela local de roteamento usando os nós que conseguiu se conectar e vai atualizando de acordo com consultas que faz/recebe e nova conexões

Design - Roteamento

- Para roteamento o IPFS usa uma implementação do hash distribuído (DHT) Kademlia com algumas otimizações (S/Kademlia e Coral):
  - Apenas objetos de até 1k são armazenados diretamente na tabela, se for maior são guardadas referências para peers onde o objeto está guardado.
  - Com o Coral há uma hierarquia de DHT's que formam clusters levando em conta o tamanho e a região, assim um nó ao procurar um conteúdo verifica primeiro nos nós mais próximos.
- Como todos os objetos são imutáveis, quando um nó faz um "get" em uma chave ele não precisa percorrer todos os nós envolvidos no "lookup" da DHT, se nós no caminho possuírem o objeto eles podem transferir

Design - Roteamento - Kademlia

- lookup eficiente a procura por uma chave leva log2(N)
- A tabela de roteamento de cada nó é composta de buckets de “K” elementos que representam de elementos com uma determinada característica da chave. Cada nó tenha muita informação sobre os nós “próximos” dele (distancia XOR)
- Muito utilizada por aplicações P2P, como exemplo Gnutella e BitTorrent

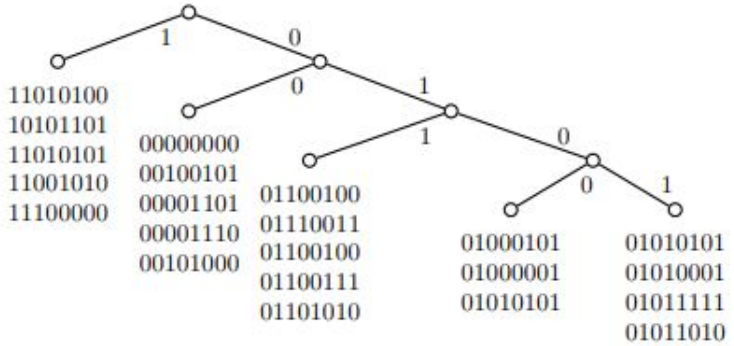
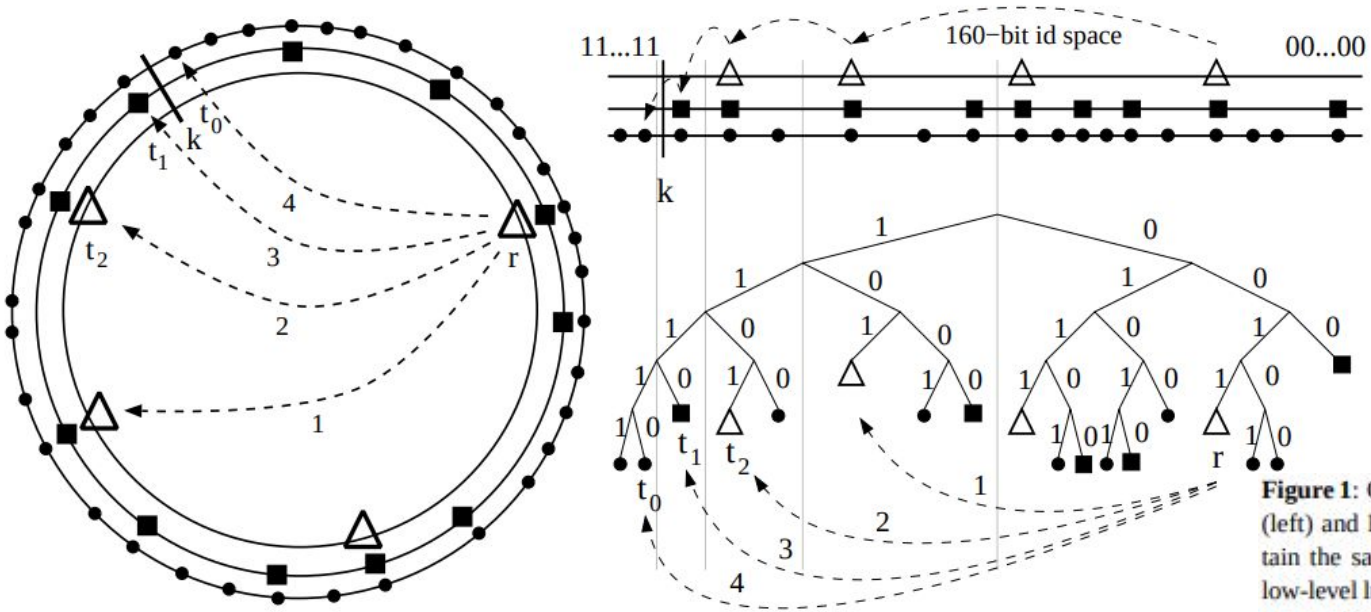


Figura 1: Routing table for node 01010000

Design - Roteamento - Coral



- 3 layers
  - Sub 30 ms (level 0)
  - Sub 100 ms (level 1)
  - above (level-2)
- Busca primeiro nas camadas inferiores, se não encontrar continua a busca a partir do nó mais próximo encontrado na camada inferior

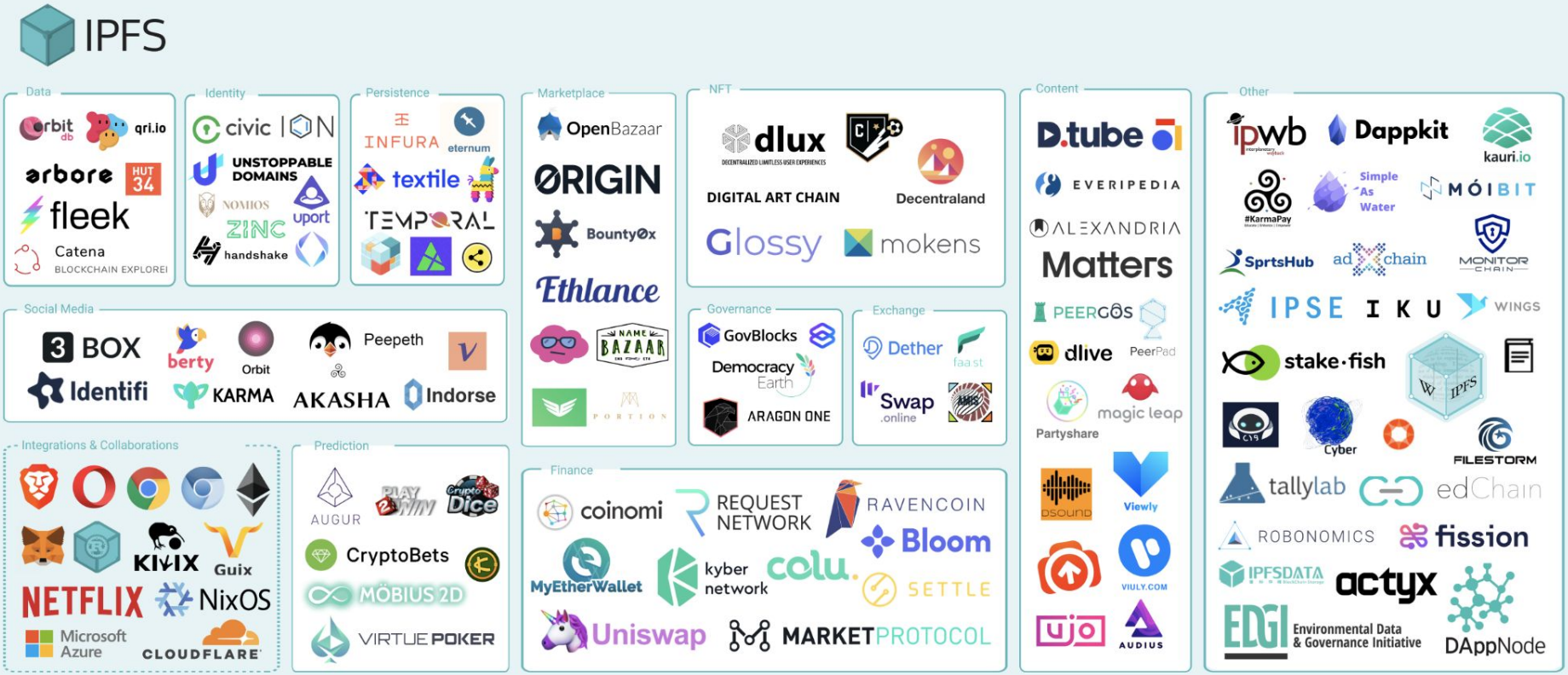
Figure 1: Coral's hierarchical lookup visualized on the Chord (left) and Kademlia (right) routing structures. Nodes maintain the same id in each of their clusters; smaller-diameter low-level lusters are naturally sparser. For a lookup on key k, a node first searches on its lowest cluster. This lookup fails on that level if the node closest to k, node t_2, does not store the key. If this occurs, Coral continues its lookup on a higher-level cluster, having already traversed the id space up to t_2's prefix. Route RPCs are shown with sequential numbering.

fonte: https://www.scs.stanford.edu/~dm/home/papers/freedman:coral.pdf

Design - Distribuição de dados

- A distribuição de dados é feita através de troca de blocos, parecido com o BitTorrent.
- Quando um nó se conecta ao outro eles trocam informações passando uma lista de blocos que ele quer (want_list) e a lista que ele já possui (have_list) e tentam se complementar.
  - Quando um nó não possui nada da “want_list” de um outro nó ao qual ele se conectou ele passa a tentar baixar esses blocos de outros nós (com uma prioridade menor)
  - Isso faz com que os blocos sejam disseminados para mais nós mesmo que esses nós não estejam diretamente interessados
- Após receber um bloco/objeto, o nó passa a

Usos


IPFS
Data
orbit db
qri.io
arbore
HUT 34
fleek
Catena
BLOCKCHAIN EXPLOREI
Identity
civic
ION
UNSTOPPABLE DOMAINS
NOMOS
uport
ZINC
handshake
Persistence
INFURA
eternum
textile
TEMPORAL
Marketplace
OpenBazaar
ORIGIN
BountyOx
Ethlance
NAME BAZAAR
PORTION
NFT
dlux
DECENTRALIZED LIMITLESS USER EXPERIENCES
DIGITAL ART CHAIN
Decentraland
Glossy
mokens
Content
D.tube
EVERIPEDIA
ALEXANDRIA
Matters
PEERGOS
dlive
PeerPad
Partyshare
magic leap
dsound
Viewly
VIULY.COM
AUDIUS
Other
ipwb
Dappkit
kauri.io
#KarmaPay
Simple As Water
MÓIBIT
SprtsHub
adchain
MONITOR CHAIN
IPSE
IKU
WINGS
stake·fish
IPFS
Cyber
FILESTORM
tallylab
edChain
ROBONOMICS
fission
IPFSDATA
actyx
EDGI
Environmental Data & Governance Initiative
DAppNode
Social Media
3 BOX
berty
Orbit
Peepeth
Identifi
KARMA
AKASHA
Indorse
Governance
GovBlocks
Democracy Earth
ARAGON ONE
Exchange
Dether
faa.st
Swap .online
Integrations & Collaborations
KIVIX
Guix
NETFLIX
NixOS
Microsoft Azure
CLOUDFLARE
Prediction
AUGUR
PLAY 2 WIN
Crypto Dice
CryptoBets
MÖBIUS 2D
VIRTUE POKER
Finance
coinomi
REQUEST NETWORK
RAVENCOIN
Bloom
MyEtherWallet
kyber network
colu.
SETTLE
Uniswap
MARKETPROTOCOL

Críticas

- Problemas comuns ao P2P
  - Gerenciamento de conteúdo ilegal, como deletar?
  - Como evitar nós “sanguessuga” que só entram na rede para baixar arquivos e saem em seguida
  - Problemas de roteamento quando há muita saída e entrada de nós (vi que eles implementaram um sistema de confiança em que nós mais estáveis são preferidos para roteamento)
- Depende de servidores quando o dado não está bem distribuído na rede. Há serviços de aluguel de servidores para IPFS
  - exemplo: https://pinata.cloud/

Conclusão

- Parece estar crescendo e obtendo cada vez mais suporte
- Ainda em estágio de alfa com muitos bugs (é relatado na própria documentação)
- Poucos peers no Brasil
- Bem fácil de experimentar

Referências principais

IPFS - Content Addressed, Versioned, P2P File System

https://docs.ipfs.io/

https://blog.ipfs.io/2020-07-20-dht-deep-dive/
https://www.scs.stanford.edu/~dm/home/papers/kpos.pdf

https://federwin.sip.ucm.es/sic/investigacion/publicaciones/pdfs/Kademlia%20SpecificationSummary0212.pdf

https://www.scs.stanford.edu/~dm/home/papers/freedman:coral.pdf

https://blog.infura.io/an-introduction-to-ipfs/