DHT Interoperability

Bridging Networks through Kademlia

Gui Michel
@guissou

Probelab,
Interplanetary Shipyard

IPFS Camp

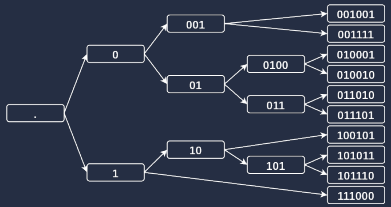
Interplanetary Shipyard

IPFS Camp
12th July 2024

Kademlia DHT

IPFS Camp

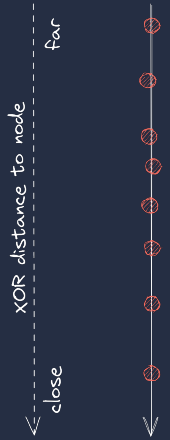
- A Distributed Hash Table (DHT) is a decentralized overlay network used for peer and content routing
- Nodes identified by a binary `Key`, determining nodes location in the Keyspace
- Highly scalable
  - State: log *N*
  - Lookup: log *N*
- Routing table stores nodes into *buckets* based on their distance from the local node's `Key`



Kademlia Binary Trie

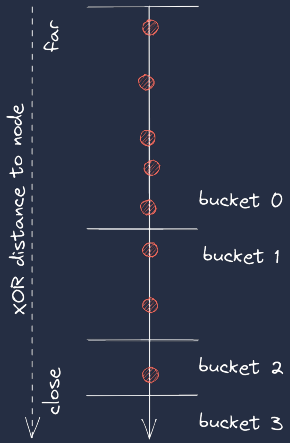
Kademlia: Keyspace & Routing Table

IPFS Camp



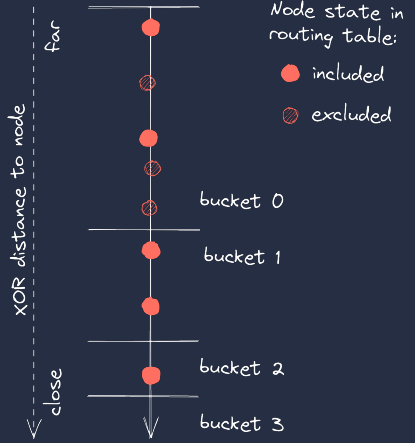
Kademlia: Keyspace & Routing Table

IPFS Camp



Kademlia: Keyspace & Routing Table

IPFS Camp



Routing Table Maintainance

- Routing Table state is $\mathcal{O}(\log N)$ with N being the network size.
- State in memory is small
- Maintenance can be expensive
  - Unresponsive peers should be removed from the table $\Rightarrow$ all peers stored in the routing table must be periodically probed
  - Nodes must periodically lookup their own identity to be aware of their closest neighbors
  - Larger routing table $\Rightarrow$ higher network load

DHT disambiguation

- DHT Protocol → `libp2p kad-dht`, `ethereum discovery`
- DHT Implementation → `go-libp2p-kad-dht`, `@libp2p/kad-dht`
- DHT Swarm → `Amino DHT`, `Filecoin DHT`

What is DHT interop and why we want it?

- Participation in multiple DHT swarms
- Protocol evolution
- Transports compatibility
- Improved network security
- Network bootstrapping

What we DON’T want to do

- Merge existing DHTs
- One DHT to rule them all
- Impose a global standard for all DHTs

Layers within DHT implementations

1. Wire format
2. Transport
3. Encryption
4. Identity
5. Kademlia

Using two DHT networks

- Assume we build a client `Hexo` to retrieve content from IPFS in the Amino DHT and from Bittorrent Mainline DHT ☺
- `Hexo` needs to participate in both Kademlia DHTs
- State is double, maintenance is 2*x* more expensive ☹
- Many open connections ☹

Using two DHT networks

- Assume we build a client `Hexo` to retrieve content from IPFS in the Amino DHT and from Bittorrent Mainline DHT ☺
- `Hexo` needs to participate in both Kademlia DHTs
- State is double, maintenance is 2*x* more expensive ☹
- Many open connections ☹

This can be optimized!

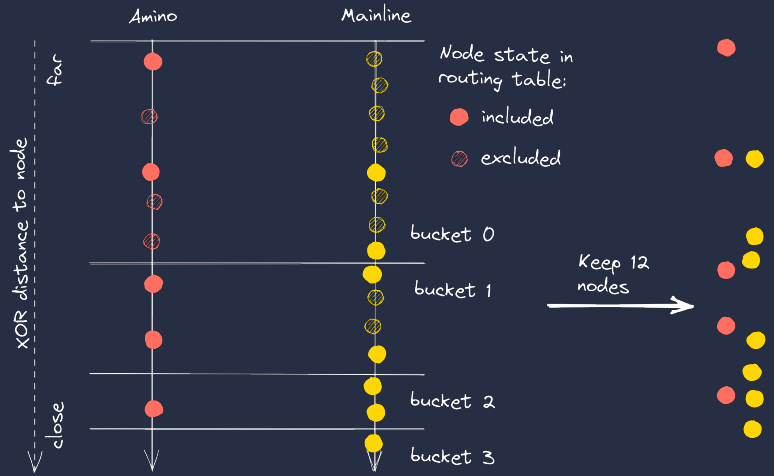
Example: Amino-Mainline Bridge

Example: Amino-Mainline Bridge

IPFS Camp

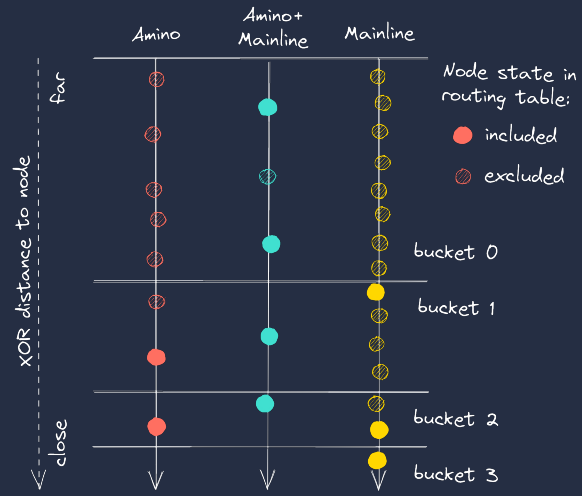
Example: Amino-Mainline Bridge

IPFS Camp



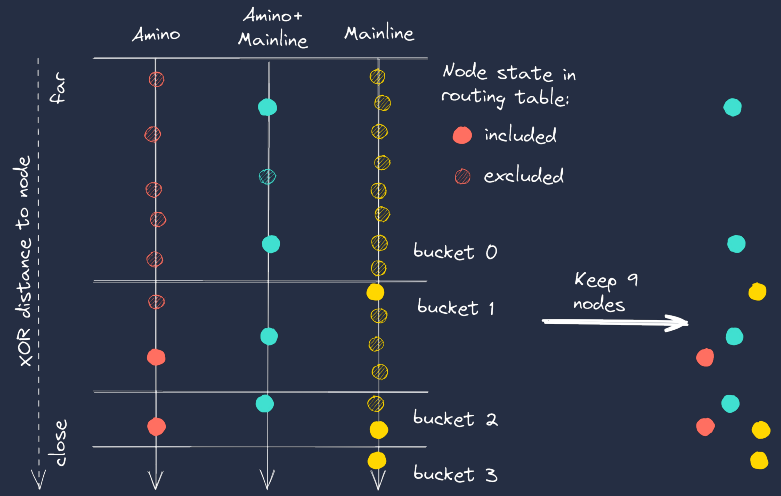
Example: Amino-Mainline Bridge

IPFS Camp



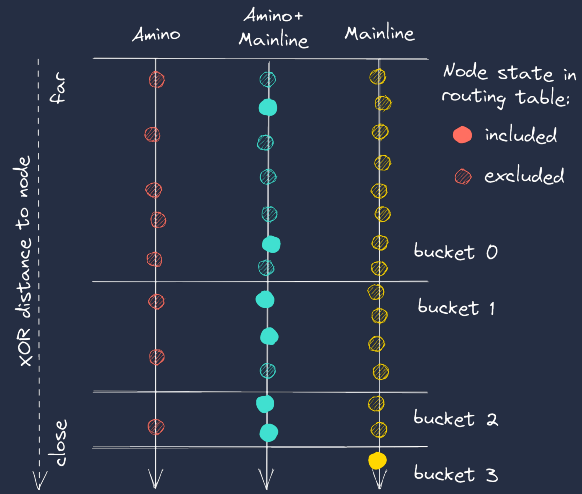
Example: Amino-Mainline Bridge

IPFS Camp



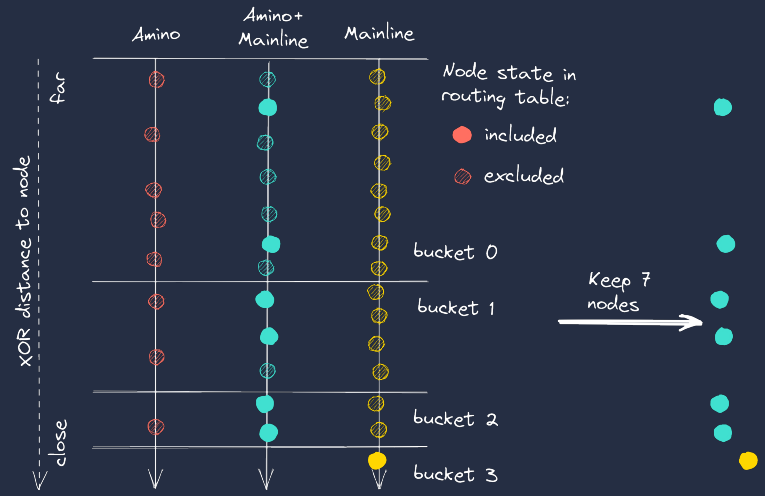
Example: Amino-Mainline Bridge

IPFS Camp



Example: Amino-Mainline Bridge

IPFS Camp



Scaling

- Scaling is great!
- Routing table size ≈ routing table size for largest network ☺
  - Larger overhead when networks are disjoint (small set of peers participating in both networks)
- Can be applied for much more than 2 DHT networks ☺

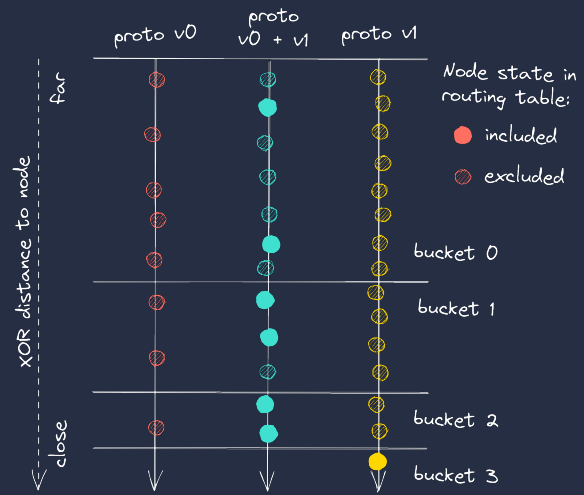
Omnidex: use cases

- Participation in multiple networks, or features (Provider Records, IPNS, etc.)
- Protocol evolution (e.g DHT Reader Privacy Upgrade, S/Kademlia, etc.)
- Transport compatibility

Protocol Evolution

Protocol Evolution

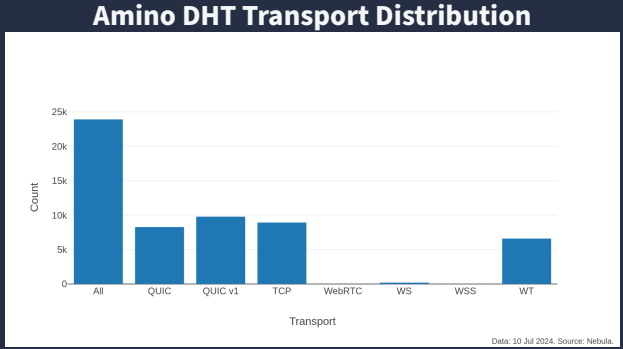
IPFS Camp



Transport compatibility

IPFS Camp

- Some transports are more popular than others.
- Peers using unpopular transports may not be able to reach the closest peers.
- Browser nodes are victim of this problem in the Amino DHT
- IPv4 vs IPv6



Source: https://probelab.io

Transport compatibility

IPFS Camp

Transport compatibility

All combinations of `ip4`, `ip6`, `quic`, `webrtc`

- `/ip4/quic`
- `/ip4/webrtc`
- `/ip6/quic`
- `/ip6/webrtc`
- `/ip4/quic`, `/ip4/webrtc`
- `/ip6/quic`, `/ip6/webrtc`
- `/ip4/quic`, `/ip6/webrtc`
- `/ip6/quic`, `/ip4/webrtc`
- `/ip4/quic`, `/ip6/quic`
- `/ip4/webrtc`, `/ip6/webrtc`
- `/ip4/quic`, `/ip4/webrtc`, `/ip6/quic`
- `/ip4/quic`, `/ip6/webrtc`, `/ip6/webrtc`
- `/ip4/quic`, `/ip6/quic`, `/ip6/webrtc`
- `/ip4/webrtc`, `/ip6/quic`, `/ip6/webrtc`
- `/ip4/quic`, `/ip4/webrtc`, `/ip6/quic`, `/ip6/webrtc`

Transport compatibility

- A node must be able to find the closest nodes given a specific transport
- Each transport has its own sub-DHT
- Nodes try to read & write in all sub-DHTs they participate in
  - This process can be optimized!

DHT Features

Each node must advertise all its transport capabilities and features.

E.g a node using transports `quic` & `webrtc` with `ip4` & `ip6`, participating in `IPNS` & `provs` must advertise the following:

- `/ip4/quic/ipns`
- `/ip4/quic/provs`
- `/ip4/webrtc/ipns`
- `/ip4/webrtc/provs`
- `/ip6/quic/ipns`
- `/ip6/quic/provs`
- `/ip6/webrtc/ipns`
- `/ip6/webrtc/provs`

Multi query

- New RPC: `FIND_NODE_WITH_FEATS(key, []features)`
- Response must contain the `K` closest peers for each feature, along with supported features
  - Overlap is expected to be high
- Records stored `K` times in each sub-DHT

Example: IPFS Provider Records

- A provider will store provider records in all of its transports sub-DHTs
- Any retriever supporting common transport can find the provider record
- Because provider and retriever share at least 1 transport, retriever can open a connection to provider

Trickier with mutable records.

Records consistency

- Records are replicated on `K` nodes in each sub-DHT
- Multiple writers of mutable records with different transports cause consistency issues
- Record holders may not receive the same set of updates
- Sync between nodes using different transports

Sub-DHT requirements

Required RPCs:

- `FIND_NODE`, ideally `FIND_NODE_WITH_FEATS`
- `READ`
- `WRITE`

Notes:

- Transports and features should be decoupled from one another to allow more granularity.
- Each sub-DHT can have its own requirements
  - Secure identity (e.g S/Kademlia)
  - Permissioned networks

Global DHT Layers

- Optional participation
- Global relay & DCUtR nodes
- `FIND_FEATURE` to allow bootstrapping new networks from existing ones
- Global `FIND_NODE` for increased security

Towards standardization

- Transport
- Identity
- RPCs

Going forward

- A first network should implement and deploy this concept
- Ideally in the libp2p codebases
- Other networks can then reuse the code to benefit from the interoperability
- When multiple networks use the same implementation, interop is trivial
- Bridges to other (non-libp2p) networks can be built after the initial implementation (e.g bittorrent mainline, ethereum discovery, etc.)

Conclusion

Omnidex enables:

- Efficient use of multiple DHT networks
- Transport compatibility
- DHT protocol evolution
- Increased security
- Granular participation to DHT base layers (e.g relays)

Q&A

IPFS Camp

Gui Michel (@guissou)

- email: guillaume@ipshipyard.com

Composable DHT: Omnidex

Last updated: August 2023