DHT Routing Table Health

Our DHT is in good shape!

Gui Michel
@guissou

**ProbeLab,
Protocol Labs**

**IPFS Camp
28th October 2022**

IPFS Camp

Protocol Labs

What is a Distributed Hash Table?

IPFS Camp

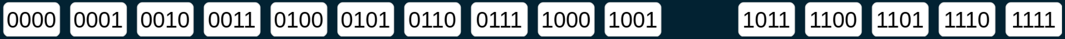
- Computer Overlay Network
- Distributed key-value store
- Keyspace is flat: content can be found at the location of its hash
- State is `Log(n)`
- Lookup is `Log(n)`

Kademlia DHT in IPFS

IPFS Camp

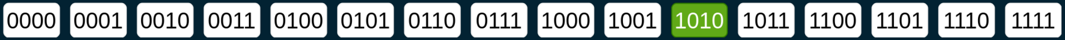
- Keyspace is 256-bits
- Each peer has a `PeerID` in the keyspace
- Locality between peers is based on XOR distance
- The DHT doesn’t store the content but **pointers** to the content: **Provider Records**
- Each peer keeps track of other peers in `k-buckets`, and sorts their `PeerIDs` by XOR distance / Common Prefix Length

Joining the DHT



0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1011 1100 1101 1110 1111

Joining the DHT



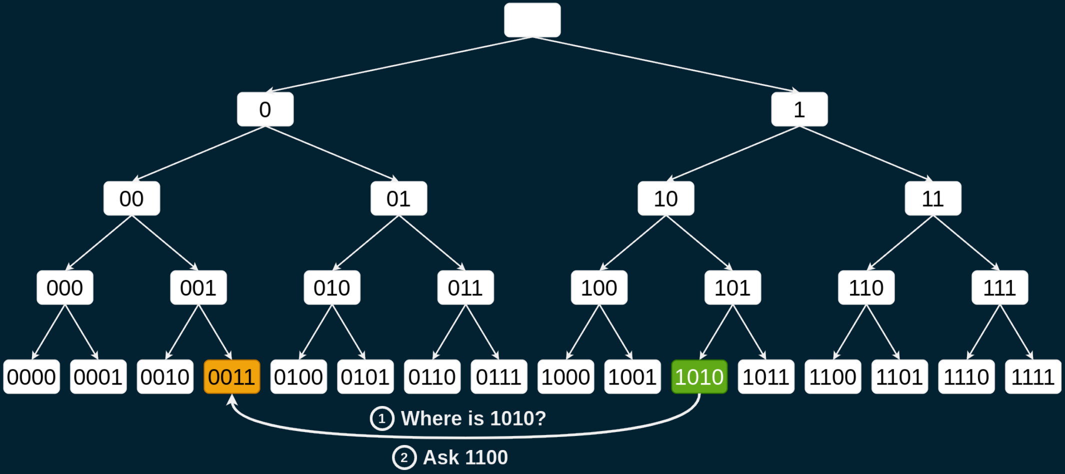
0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 **1010** 1011 1100 1101 1110 1111

Joining the DHT

0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111

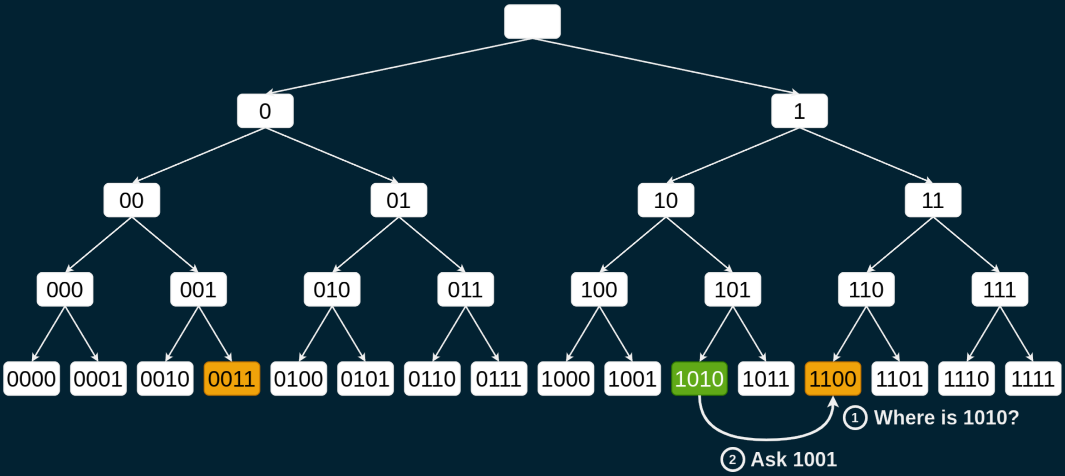
Joining the DHT: self lookup

IPFS Camp



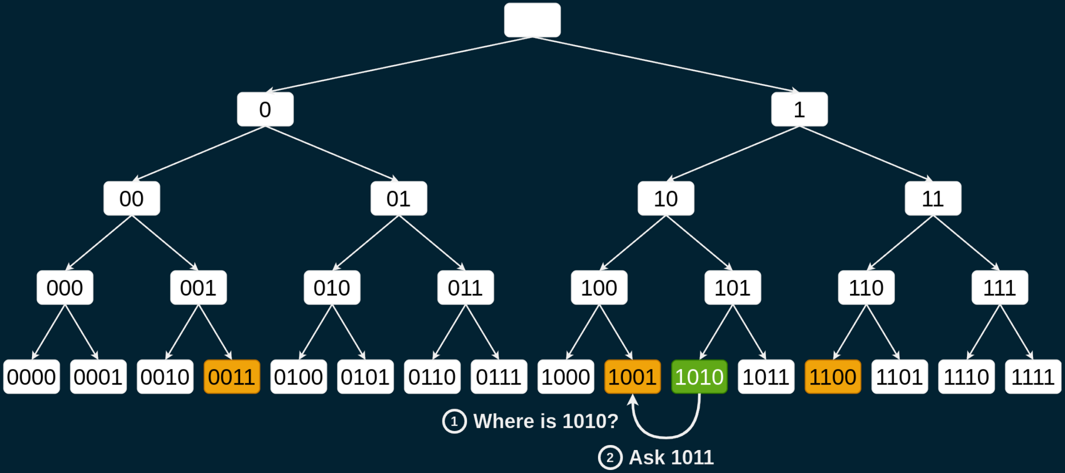
Joining the DHT: self lookup

IPFS Camp



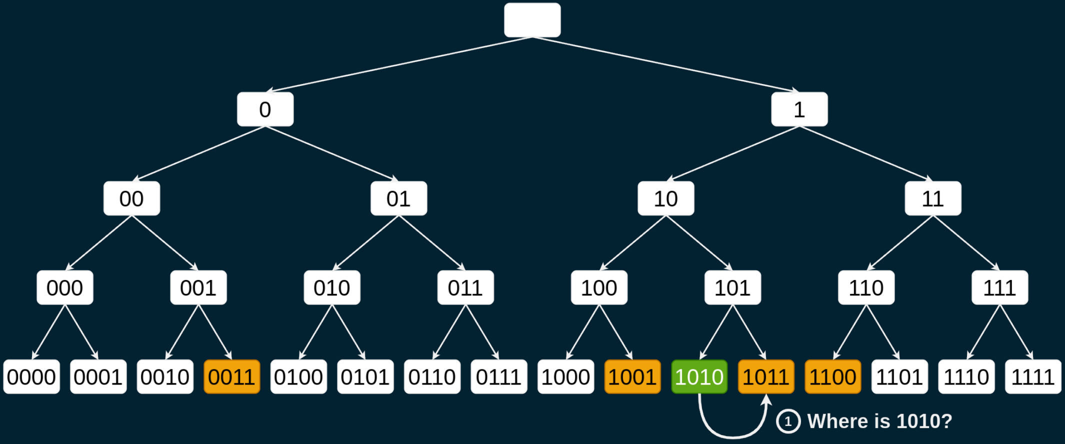
Joining the DHT: self lookup

IPFS Camp



Joining the DHT: self lookup

IPFS Camp



① Where is 1010?

② I don't know anyone closer

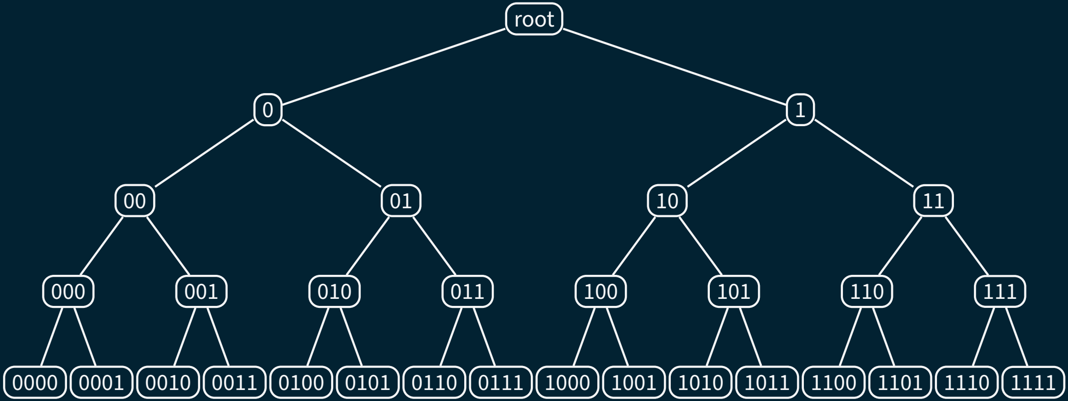
Kademlia details

- Each node periodically looks up for its own `PeerID`
- Multiple requests happen concurrently for the same key
- Upon request a peer returns the 20 closest peers it knows to the requested key
- When a Provider Records is published to the DHT it is stored on the 20 closest peers to its key
- Buckets are capped at 20 peers

- **These constants don’t need to be the same value!**

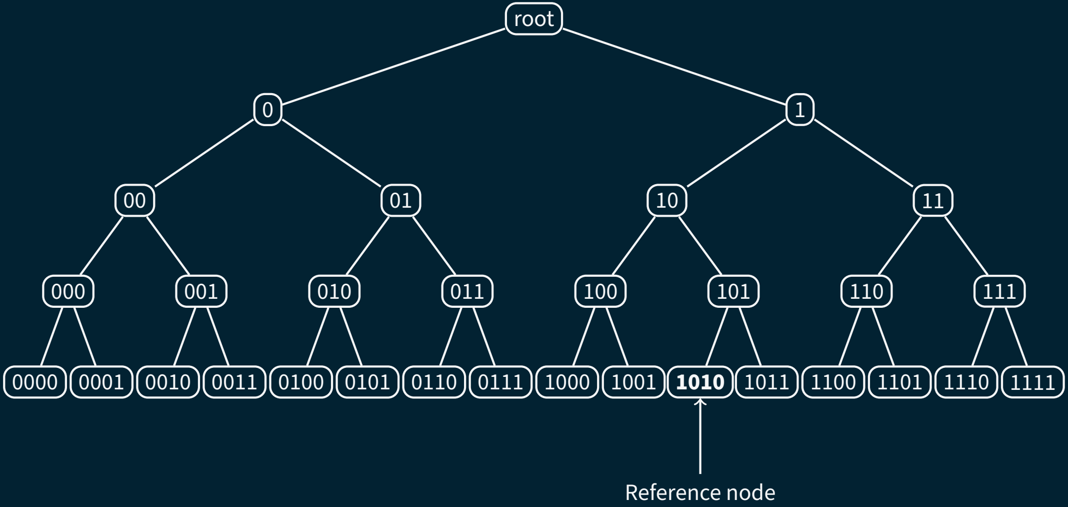
Kademlia keyspace

IPFS Camp

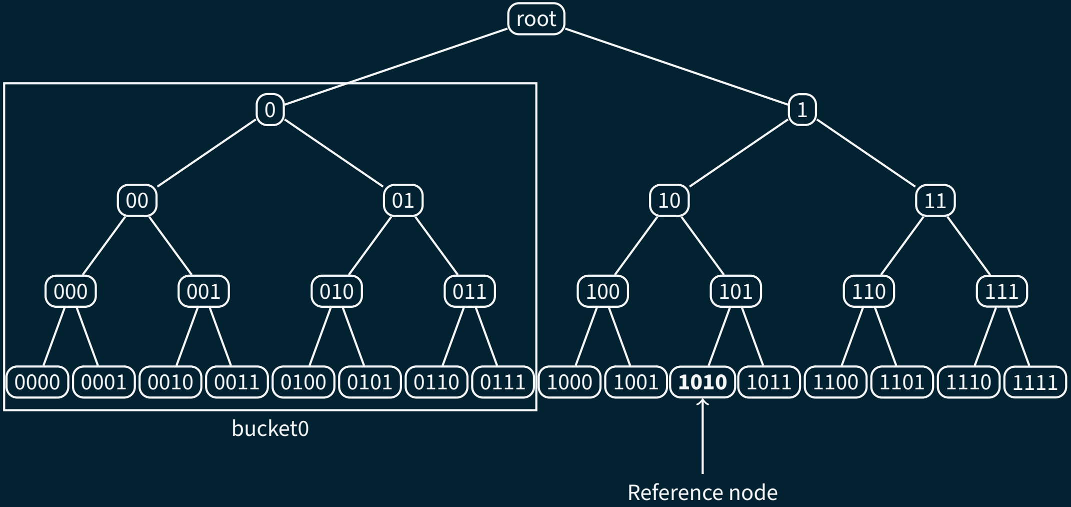


Kademlia keyspace

IPFS Camp

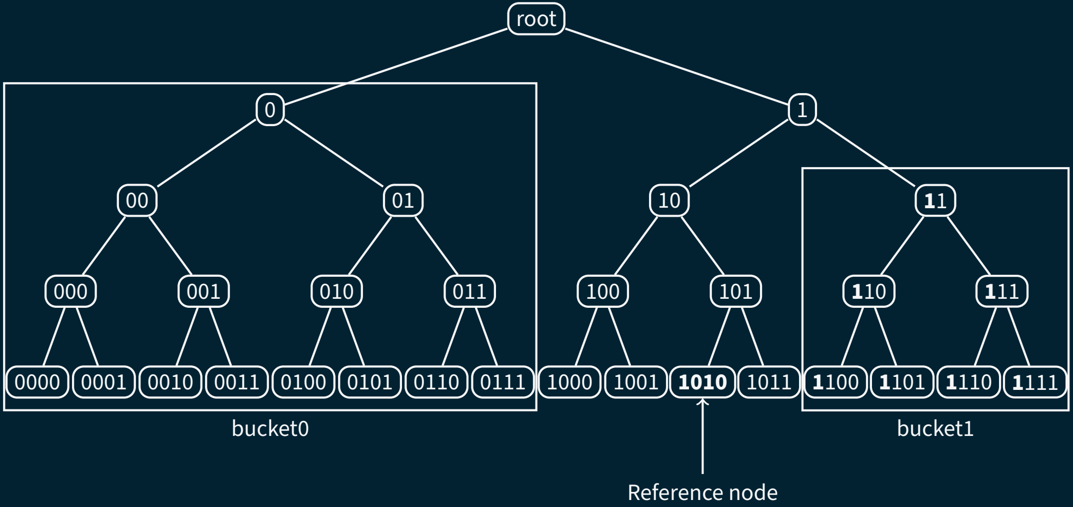


Kademlia keyspace



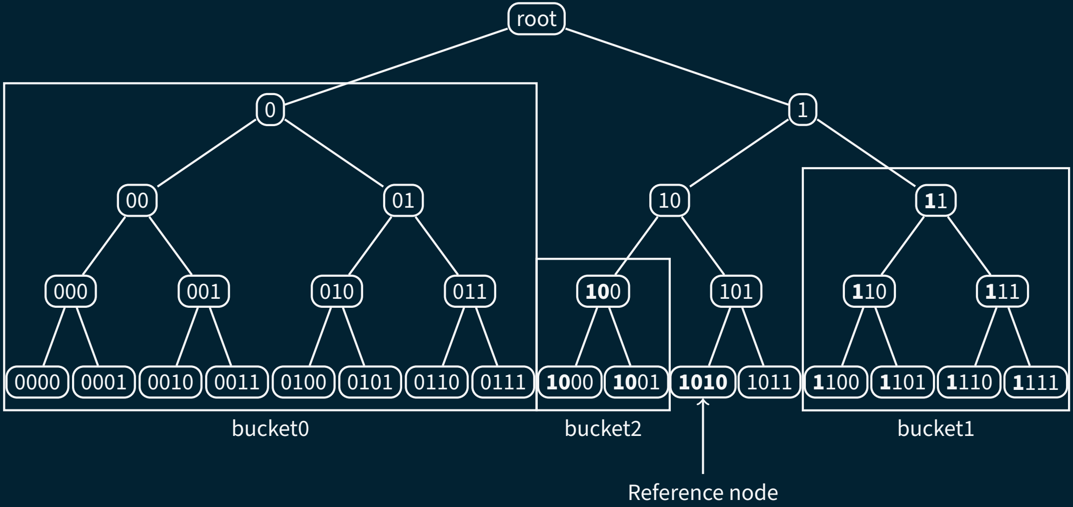
Kademlia keyspace

IPFS Camp



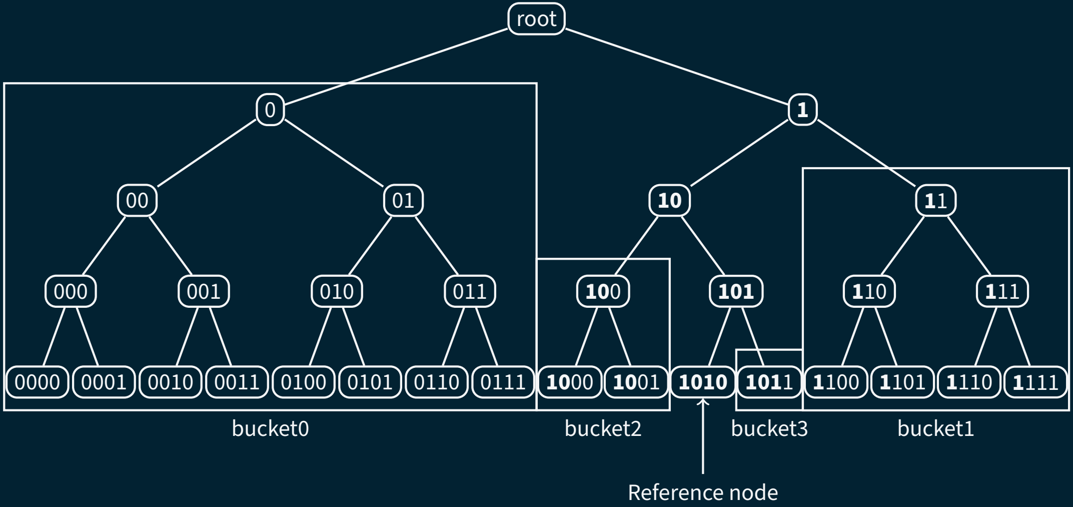
Kademlia keyspace

IPFS Camp



Kademlia keyspace

IPFS Camp



Example: Routing Table of peer 01101000

Bucket 0

1. 11010111
2. 10001011
3. 10101110
4. 11110101
5. 10000010
6. 11010100
7. 11000100
8. …

Bucket 1

1. **0**0110101
2. **0**0001000
3. **0**0111011
4. **0**0101101
5. **0**0110100

Bucket 2

1. **01**011101
2. **01**001111
3. **01**010110

Bucket 3

1. **011**11011
2. **011**10001

Bucket 4

1. **0110**0011

`k-bucket` replacement policy

- `Kademlia`: only when a bucket is full and there is a new candidate, least-recently seen and unreachable node gets evicted, but live nodes are never evicted
- `kubo` implementation: periodically ping the nodes in the routing table, and evict the unresponsive ones

Measurements data

IPFS Camp

- The Nebula Crawler crawls the IPFS network and provides all peers in the network along with their routing table for a point in time
- Data taken from 28 crawls over 1 week (4 crawls per day) starting on 2022-04-19

Methodology

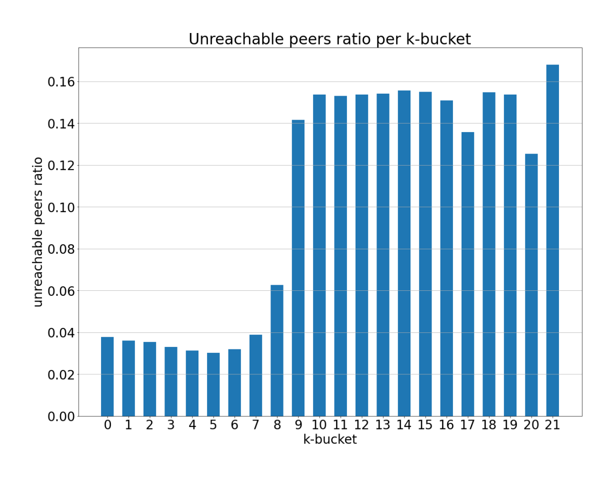
- The Nebula Crawler provides a global snapshot of the network
- We can reconstruct the `k-buckets` of all peers by computing the XOR distance between a `PeerID` and the `PeerID`s of all peers in its routing table
- From the global snapshot we can find the closest peers to every other peer and verify if any peer is missing from a `k-bucket`

Unreachable peers in the Routing Table

IPFS Camp

Unreachable peers may still be referenced in other peers routing tables (stale entries)

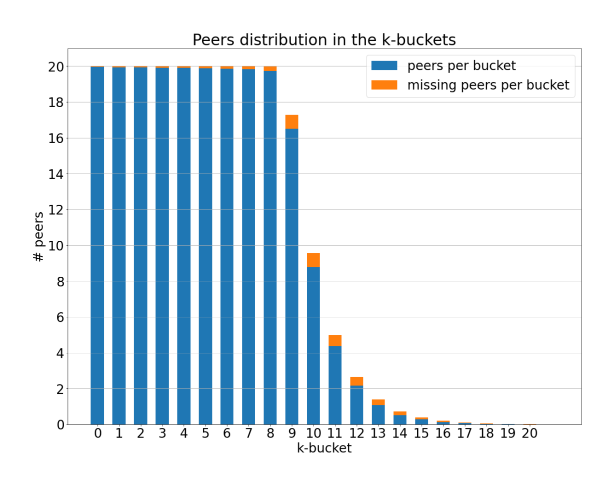
- Average for buckets 0 to 8: 3.8% ~ 0.75 peers
- Average for buckets 9 to 21: 15%



Peers distribution in the k-buckets

IPFS Camp

- Peers distribution in bucket follows an exponential growth, capped at 20
- Buckets 0–8 are missing on average 0.12 peers per bucket
- Buckets 9–14 are missing on average 0.53 peers per bucket

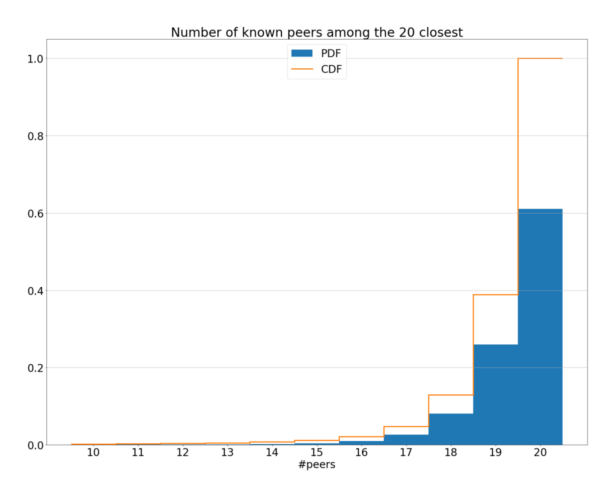


20 closest peers awareness

IPFS Camp

- Probability Density Function (PDF)
- Cumulative Distribution Function (CDF)

- 61.1% of the peers know all their 20 closest peers
- 95.2% of the peers know at least 18 of their 20 closest peers



Diversity in the k-buckets

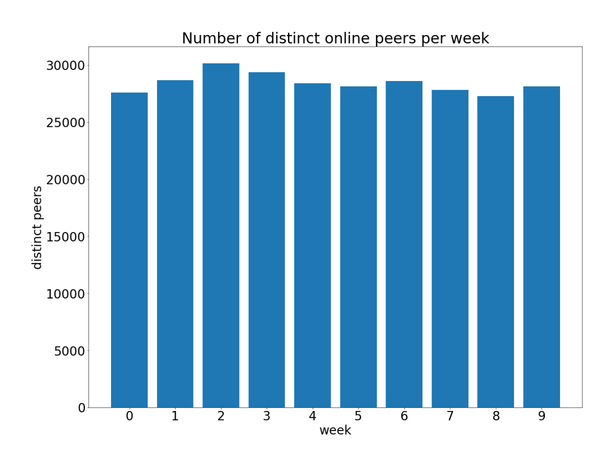
IPFS Camp

- Live peers never get replaced in the `k-buckets` by design
- Eventually, buckets with many candidates (e.g buckets `0-1`) will be filled almost exclusively with a small number of very stable peers
- Routing for content *far away* (in XOR distance) may become centralized on a small set of peers
- Bad for decentralization :(
- Bad for load balancing :(

Diversity measurements

IPFS Camp

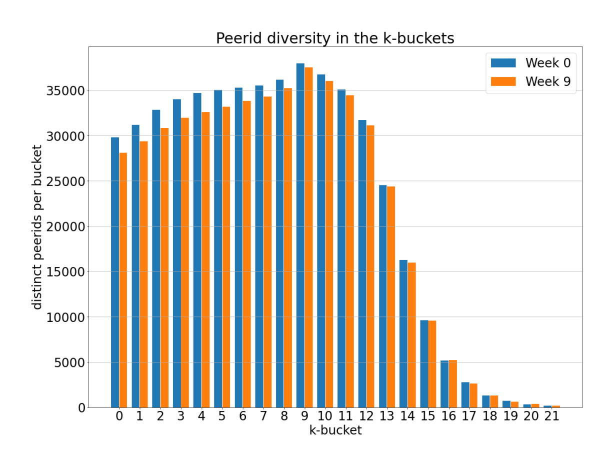
- Measurements for 10 consecutive weeks starting on 2022-02-16
- Each week’s measurements are based on data from 14 crawls (2x/day)
- Diversity in `k-buckets` is measured as the number of distinct `peer_ids` observed in each bucket for all peers



Diversity in each k-buckets

IPFS Camp

- Buckets 10+: non-full buckets → low diversity
- Bucket 9: bucket just full → highest diversity
- Buckets 0–1: many candidates, only the most stable don't get evicted → lower diversity
- We expect diversity in buckets 0–1 to decrease over time



Diversity evolution over time

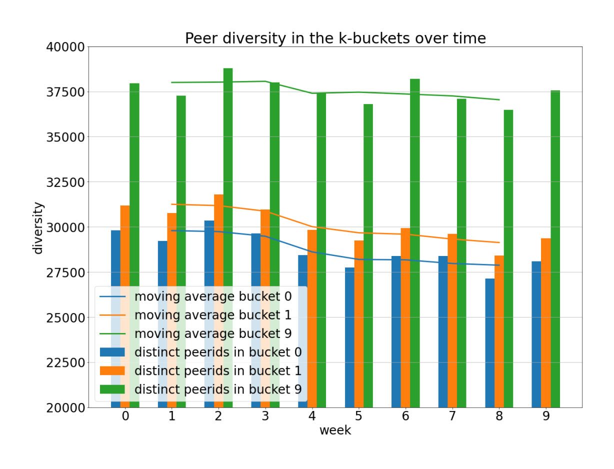
IPFS Camp

Moving average difference between week 1 and week 8:

Bucket 0: **-6.9%**

Bucket 1: **-7.3%**

Bucket 9: -2.6%



Persisting Routing Table States

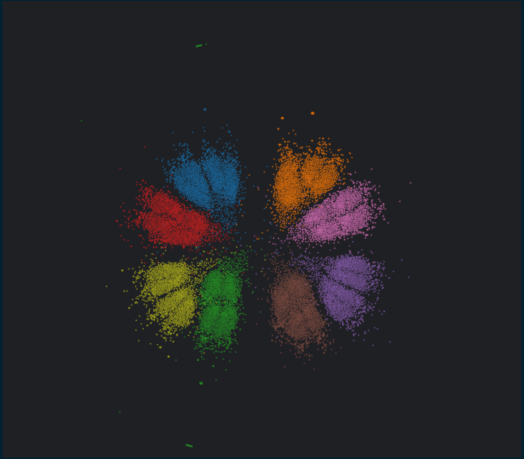
- Routing table currently flushed upon node shutdown
- Routing table has to be repopulated on restart, using bootstrap peers

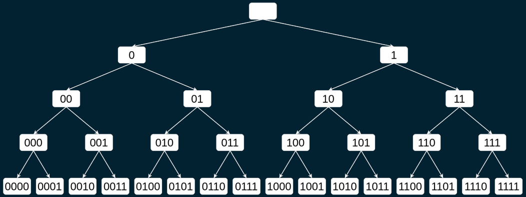
Persisting the state of the routing table would allow:

- Faster convergence
- Less dependence on bootstrap nodes

DHT connection graph

IPFS Camp





0
1
00
01
10
11
000
001
010
011
100
101
110
111
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111

Conclusion

IPFS Camp

- Very low rate of stale entries in the routing table, given high churn
- Peers distributions in the `k-buckets` as expected
- The `k-buckets` are only missing a small number of peers
- `95.2`% of the nodes have at least 18 of their 20 closest peers in their Routing Table
- We observed diversity decreasing over time in low ID buckets, which might become a concern for decentralization

References

Complete report available

https://github.com/protocol/network-measurements

Digression: Bucket quotas

IPFS Camp

Make the bucket replacement policy smarter

- Reduce number of hops
- Reduce hop latency
- More load balancing through diversity
- Make the DHT agile and upgradable
- Keep DHT stability

Current quotas

IPFS Camp

- No more than 3 IP addresses from the same Autonomous System (AS) in the routing table
- No more than 2 IP addresses from the same AS in the same bucket

Quotas example

IPFS Camp

Out of the 20 peers per bucket, if possible we want:

- The 5 nodes with the longest uptime
- 5 nodes whose RTT is below the 30th percentile of this bucket candidates' RTT
- 1 node in each of the 4 sub-buckets (4 in total)
- 4 peers whose DHT version is higher or equal to its own version
- 2 random peers

Nodes from the low RTT, DHT version and random peers get probabilistically pruned and replaced, e.g every 6 hours one node is pruned.

Note: peers can belong to multiple categories at once.

Side effects

IPFS Camp

- Close buckets (high ID) will not change
- Far buckets (low ID) are expected to change
- Stable up-to-date central nodes will be easy to find
- Reaching unstable outdated nodes with high latency may require one extra hop
- Lookup latency (finding 1 of the 20 closest nodes to a key) is expected to significantly decrease
- Provide latency (finding the 20 closest nodes to a key) is expected to slightly decrease