DHT Routing Table Health

Our DHT is in good shape!

Guillaume Michel
@guissou

ProbeLab,
Protocol Labs

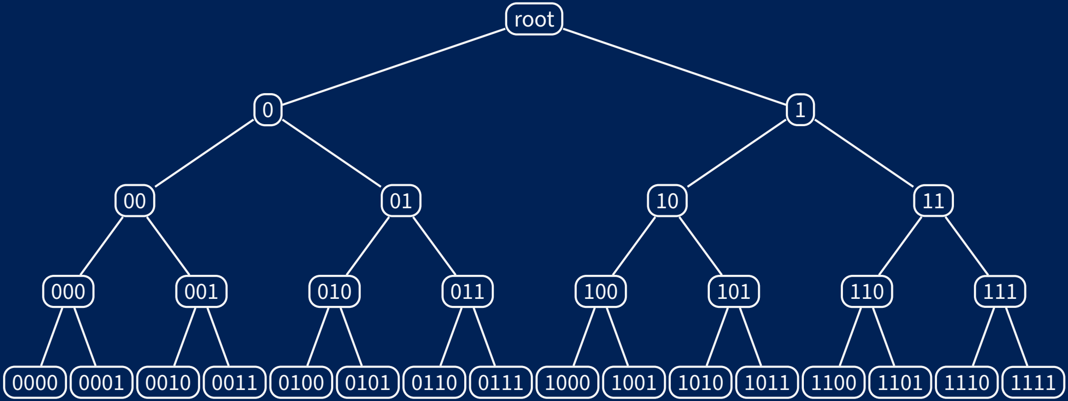
IPFS þing
15th July 2022

Protocol
Labs

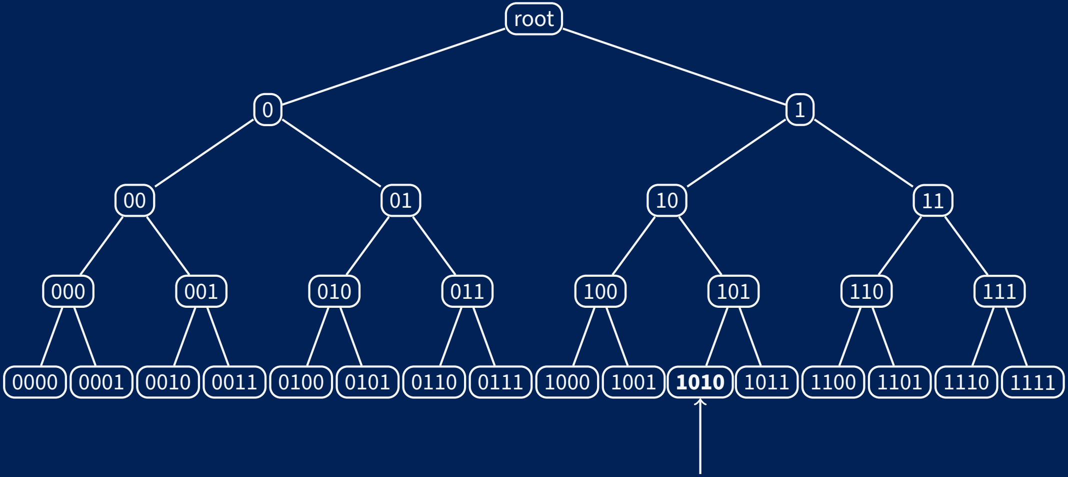
Kademlia DHT Routing Table

- A Distributed Hash Table (DHT) is a decentralized overlay network
- Each node has to know some other peers to be connected to the network, this set of peers is the node's Routing Table
- Kademlia keeps peers in `k-buckets` sorting the `peer_id` by XOR distance (or Common Prefix Length). Each bucket is capped at 20 peers

Kademlia keyspace

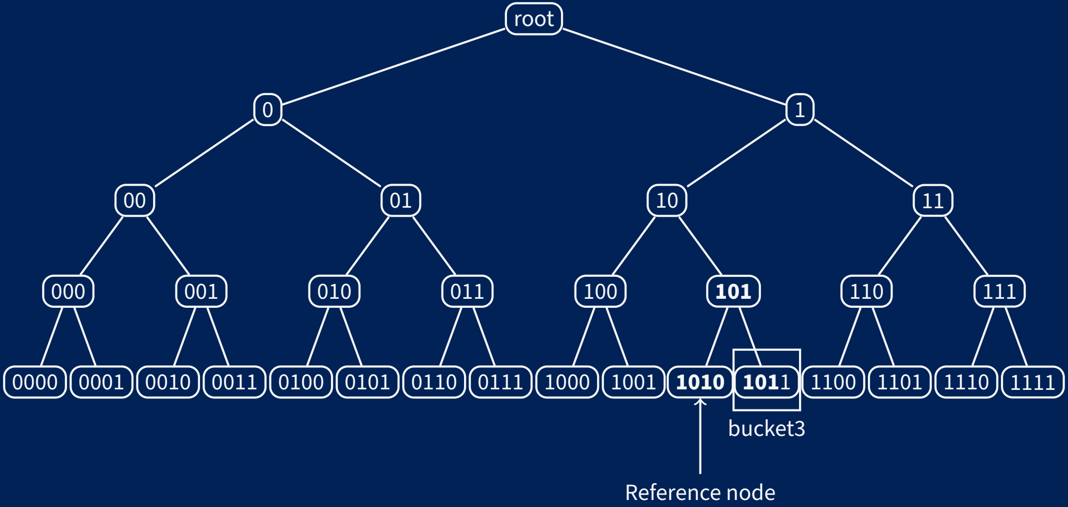


Kademlia keyspace

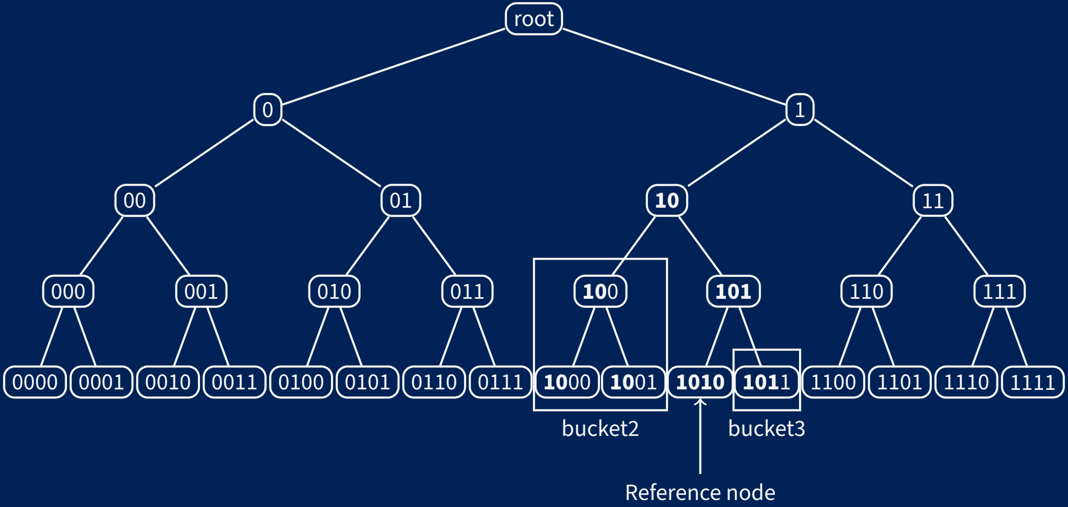


Reference node

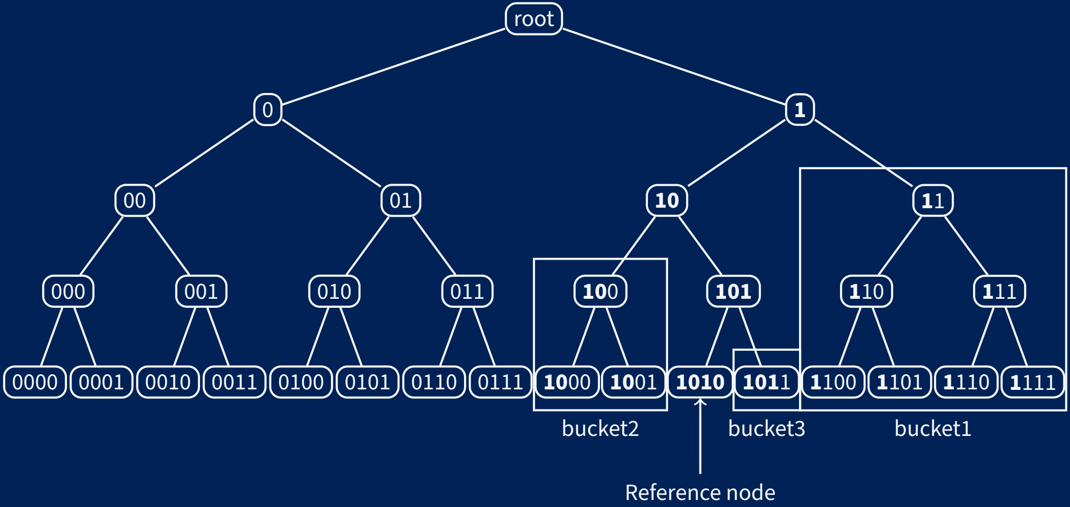
Kademlia keyspace



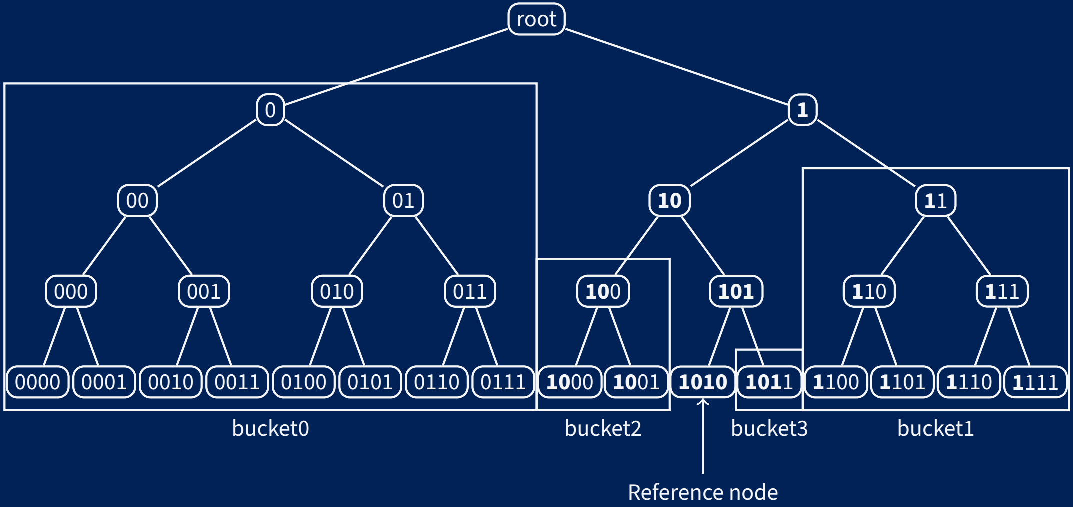
Kademlia keyspace



Kademlia keyspace



Kademlia keyspace



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 nodes that it didn’t hear of recently, and evict them if they don’t respond

Measurements data

- 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

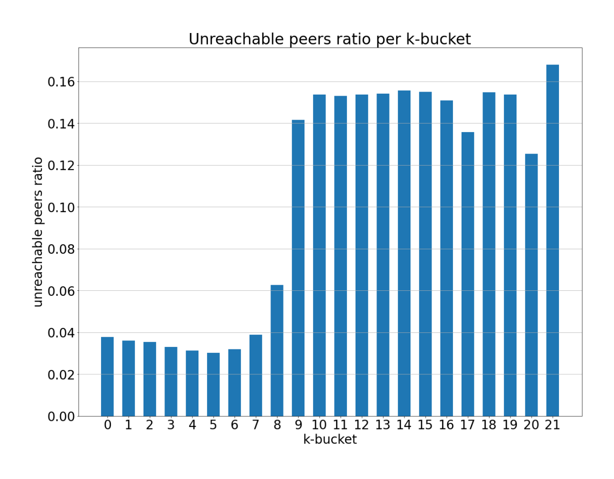
Methodology

- 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 `peer_id` and the `peer_ids` 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`
- Caveat: XOR distance is non-linear! Computationnaly expensive to find the closest peers to a specific `peer_id`. A python Binary Trie implementation was built for this purpose

Unreachable peers in the Routing Table

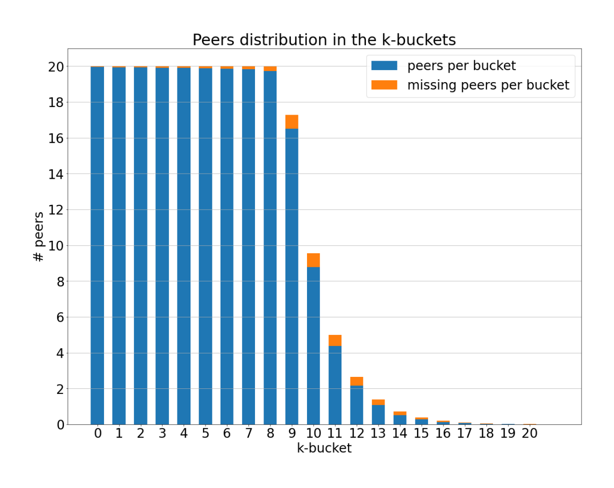
Unreachable peers may still be referenced in other peers routing tables (stale entries)

- Average for buckets 0 to 8: $3.8\% \sim 0.75$ peers
- Average for buckets 9 to 21: 15%



Peers distribution in the k-buckets

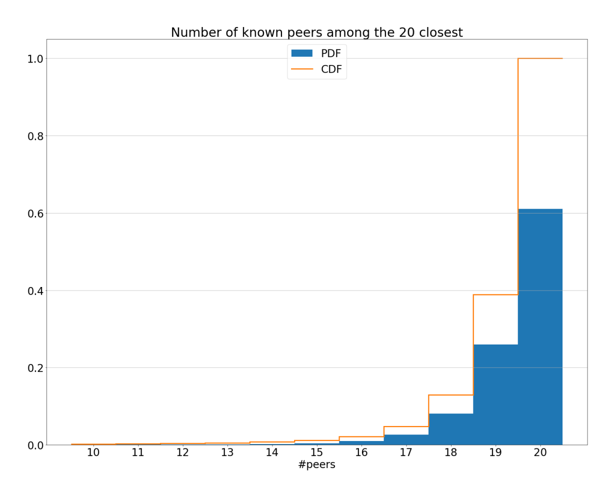
- Peers distribution in bucket follows an exponential growth, capped at 20
- Buckets 0–8 are missing on average 0.12 peers per bucket
- Buckets 9–14 are missing on average 0.53 peers per bucket



20 closest peers awareness

1. Probability Density Function (PDF)
2. 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

- 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 :(

Replacement in action: views of `buckets 0`

Very stable nodes - Stable enough nodes - Unstable nodes

node0

node1
node2
node3
node4
node5

node2

node1
node3
node4
node5
node7

node4

node1
node3
node7
node8
node10

node1

node0
node2
node6
node7
node8

node3

node0
node1
node2
node4
node8

node5

node1
node2
node3
node4
node5

Replacement in action: views of `buckets 0`

Very stable nodes - Stable enough nodes - Unstable nodes

node0

node1
~~node2~~
~~node3~~
node4
node5

~~node2~~

~~node1~~
~~node3~~
~~node4~~
~~node5~~
~~node7~~

node4

node1
~~node3~~
~~node7~~
node8
node10

node1

node0
~~node2~~
~~node6~~
~~node7~~
node8

~~node3~~

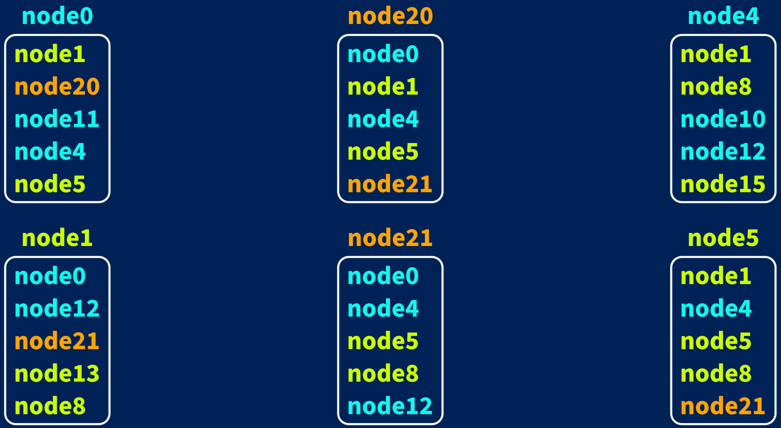
~~node0~~
~~node1~~
~~node2~~
~~node4~~
~~node8~~

node5

node1
~~node2~~
~~node3~~
node4
node5

Replacement in action: views of `buckets 0`

Very stable nodes - Stable enough nodes - Unstable nodes



node0

node1
node20
node11
node4
node5

node20

node0
node1
node4
node5
node21

node4

node1
node8
node10
node12
node15

node1

node0
node12
node21
node13
node8

node21

node0
node4
node5
node8
node12

node5

node1
node4
node5
node8
node21

Replacement in action: views of `buckets 0`

Very stable nodes - Stable enough nodes - Unstable nodes

node0

~~node1~~
~~node20~~
node11
node4
~~node5~~

~~node20~~

~~node0~~
~~node1~~
~~node4~~
~~node5~~
~~node21~~

node4

~~node1~~
~~node8~~
node10
node12
~~node15~~

~~node1~~

~~node0~~
~~node12~~
~~node21~~
~~node13~~
~~node8~~

~~node21~~

~~node0~~
~~node4~~
~~node5~~
~~node8~~
~~node12~~

~~node5~~

~~node1~~
~~node4~~
~~node5~~
~~node8~~
~~node21~~

Replacement in action: views of `buckets 0`

Very stable nodes - Stable enough nodes - Unstable nodes

node0

node4
node11
node23
node32
node35

~~node20~~

~~node0~~
~~node1~~
~~node4~~
~~node5~~
~~node21~~

node4

node0
node10
node12
node23
node32

~~node1~~

~~node0~~
~~node12~~
~~node21~~
~~node13~~
~~node8~~

~~node21~~

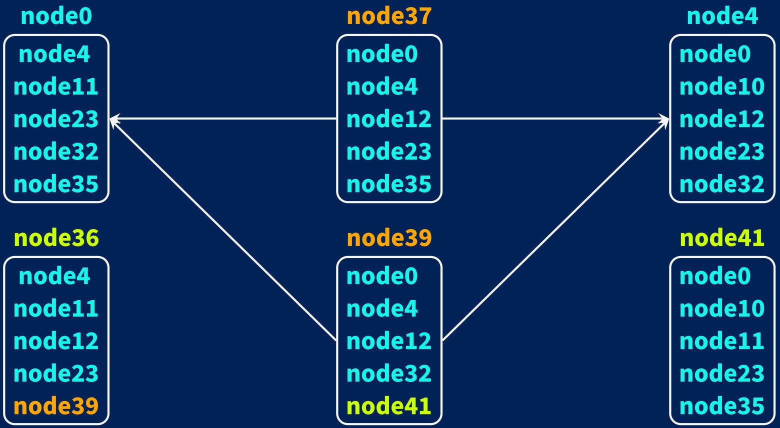
~~node0~~
~~node4~~
~~node5~~
~~node8~~
~~node12~~

~~node5~~

~~node1~~
~~node4~~
~~node5~~
~~node8~~
~~node21~~

Replacement in action: views of `buckets 0`

Very stable nodes - Stable enough nodes - Unstable nodes



node0

node4
node11
node23
node32
node35

node37

node0
node4
node12
node23
node35

node4

node0
node10
node12
node23
node32

node36

node4
node11
node12
node23
node39

node39

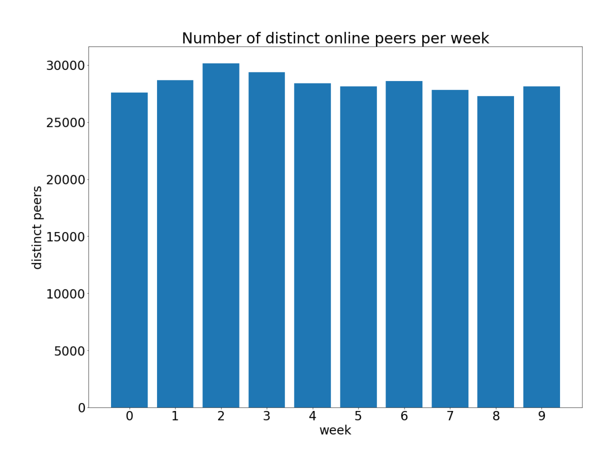
node0
node4
node12
node32
node41

node41

node0
node10
node11
node23
node35

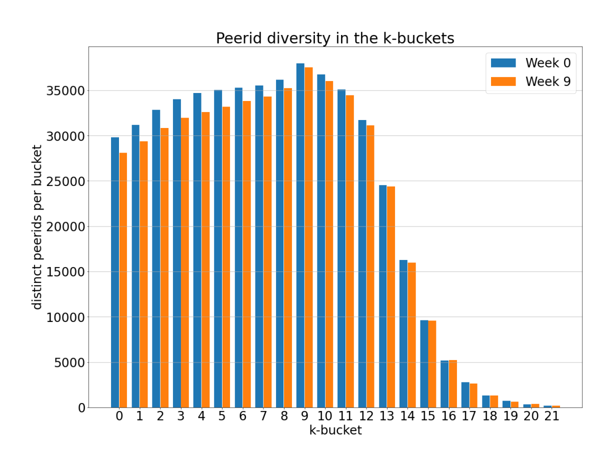
New measurements

- 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

- 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



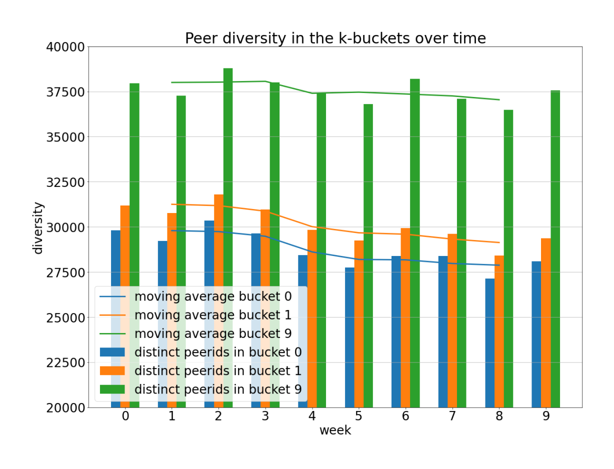
Diversity evolution over time

Moving average difference between week 1 and week 8:

Bucket 0: **-6.9%**

Bucket 1: **-7.3%**

Bucket 9: -2.6%



Conclusion

- 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

- All results of `RFM19` of are available on the `protocol/network-measurements` Github repo

References

1. `RFM19` on the `protocol/network-measurements` Github repo
2. DHT Routing Table Health Notion page
3. Kademlia Paper by Petar Maymounkov and David Mazières
4. Nebula Crawler by Dennis Trautwein
5. Python Binary Trie implementation
6. ProbeLab Notion page