Kadabra: Adapting Kademlia for the Decentralized Web

The Science of Blockchain Conference 2024

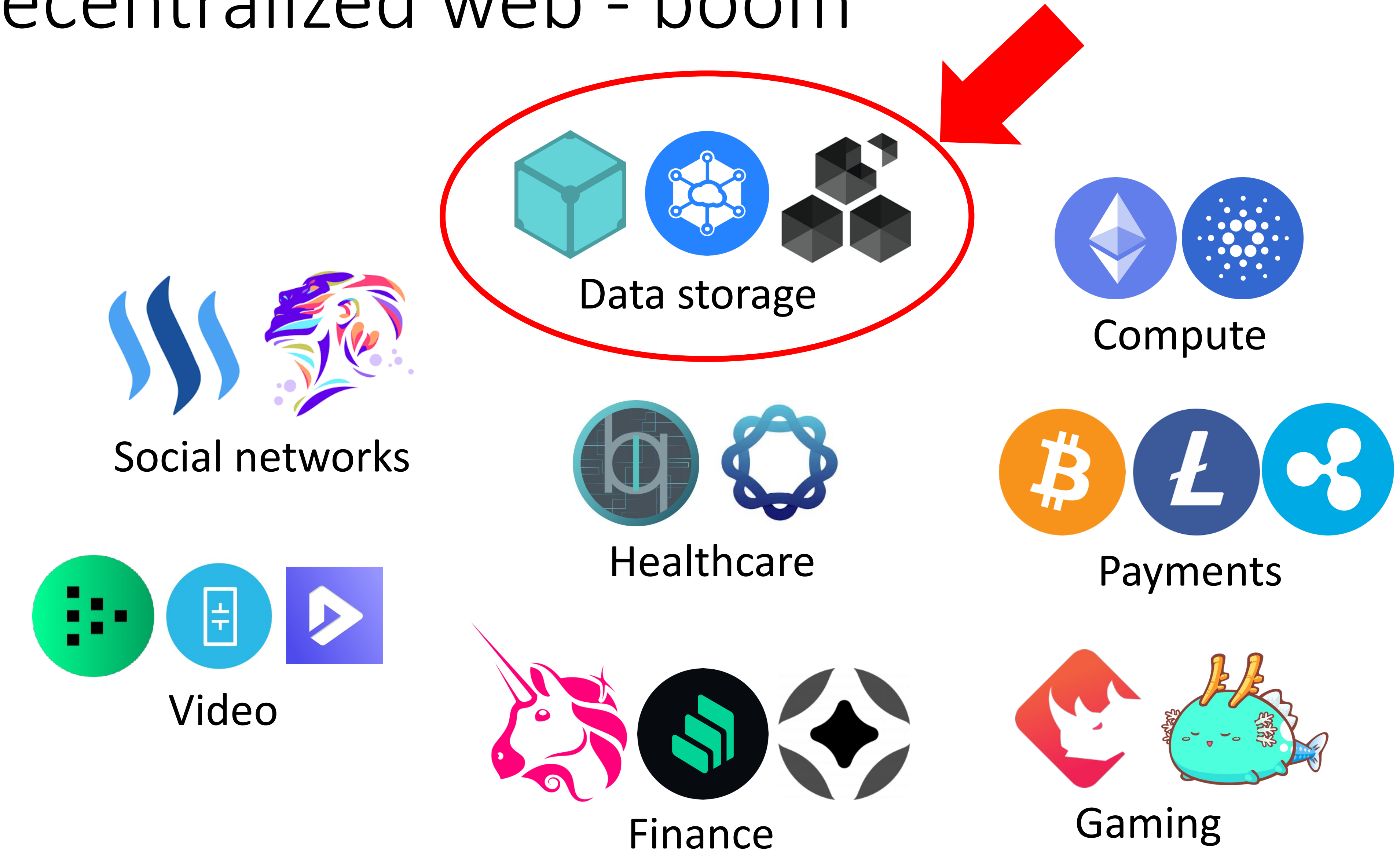
Aug 7-9, 2024 New York, NY, USA

Yunqi Zhang

Shaileshh Bojja Venkatakrishnan

The Ohio State University

Decentralized web - boom



Data storage

Compute

Social networks

Healthcare

Payments

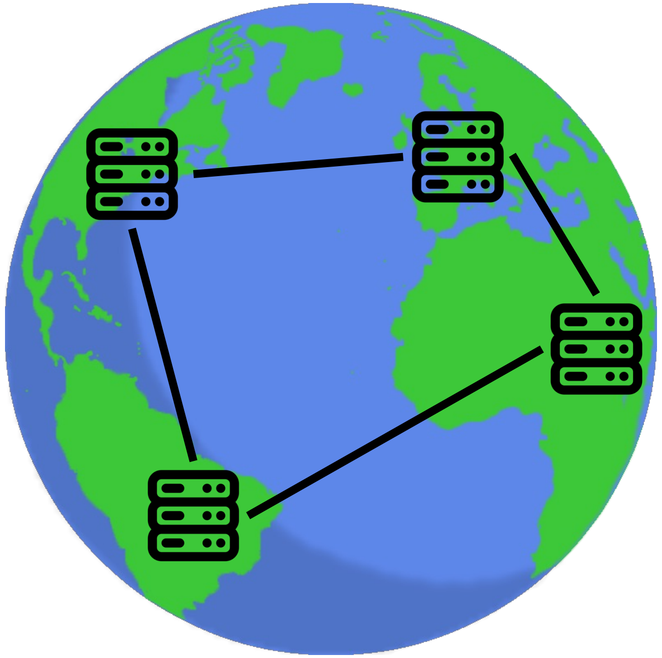
Video

Finance

Gaming

Data storage and retrieval (centralized)

- CDN and Cloud



Akamai

CLOUDFLARE

StackPath

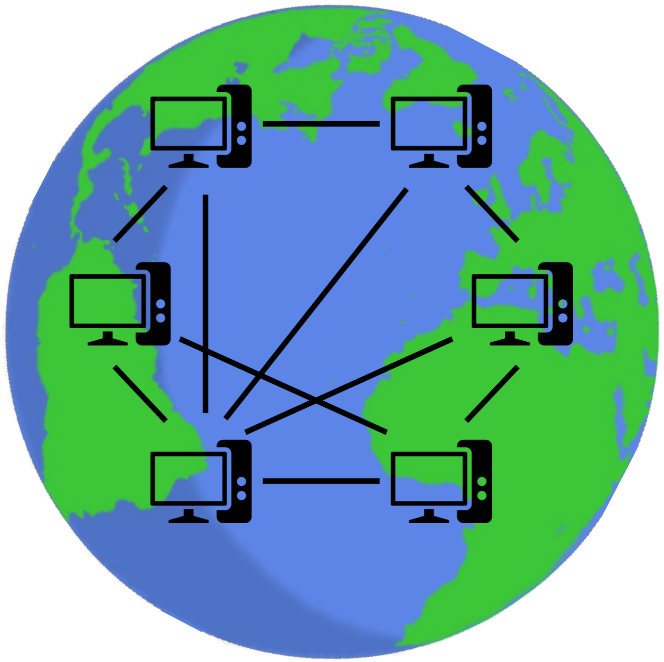
Google Cloud

aws

Microsoft Azure

Data storage and retrieval (decentralized)

- Peer-to-peer (P2P)



STORJ

Storj active nodes have doubled since mid 2021
Storj capacity has increased 6X since mid 2021 [1]

IPFS

IPFS traffic has increased 10X since late 2021 [2]

swarm

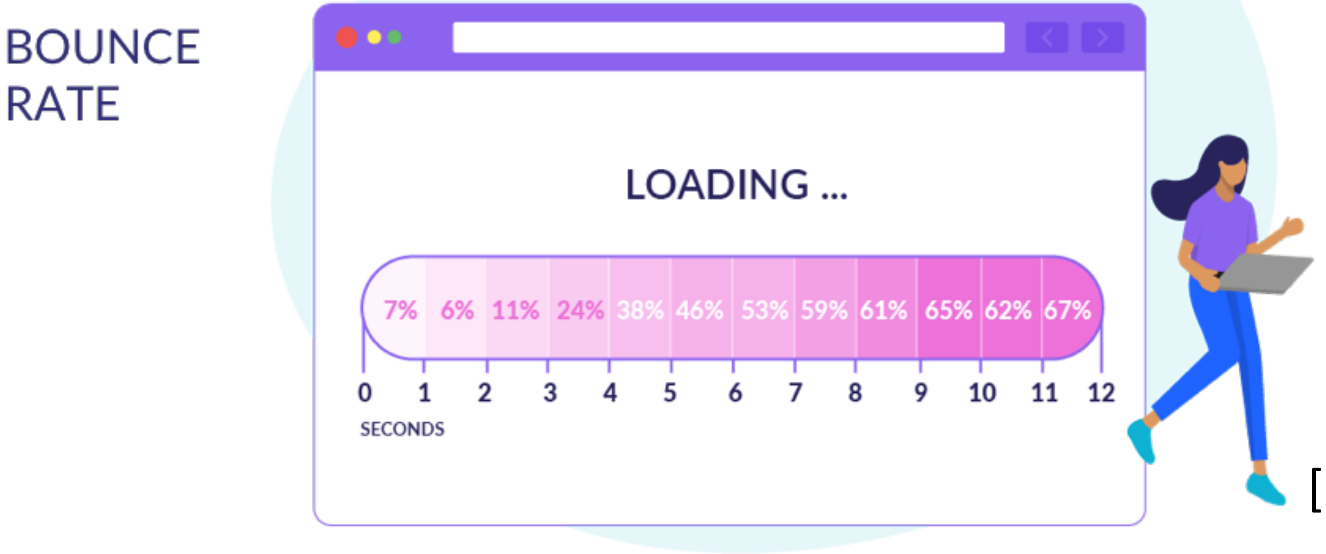
Swarm has 5k daily active nodes
>600 new nodes join Swarm each month [3]

[1] Storj official website. [2] Palo Alto Networks. [3] Swarm Scan.

Key challenge: locate and retrieve data

Important to locate and retrieve data efficiently

- 25% users abandon a website if loading time >4s
- 46% users do not revisit poorly performing websites
- 1s delay would reduce customer satisfaction by 16% [4]

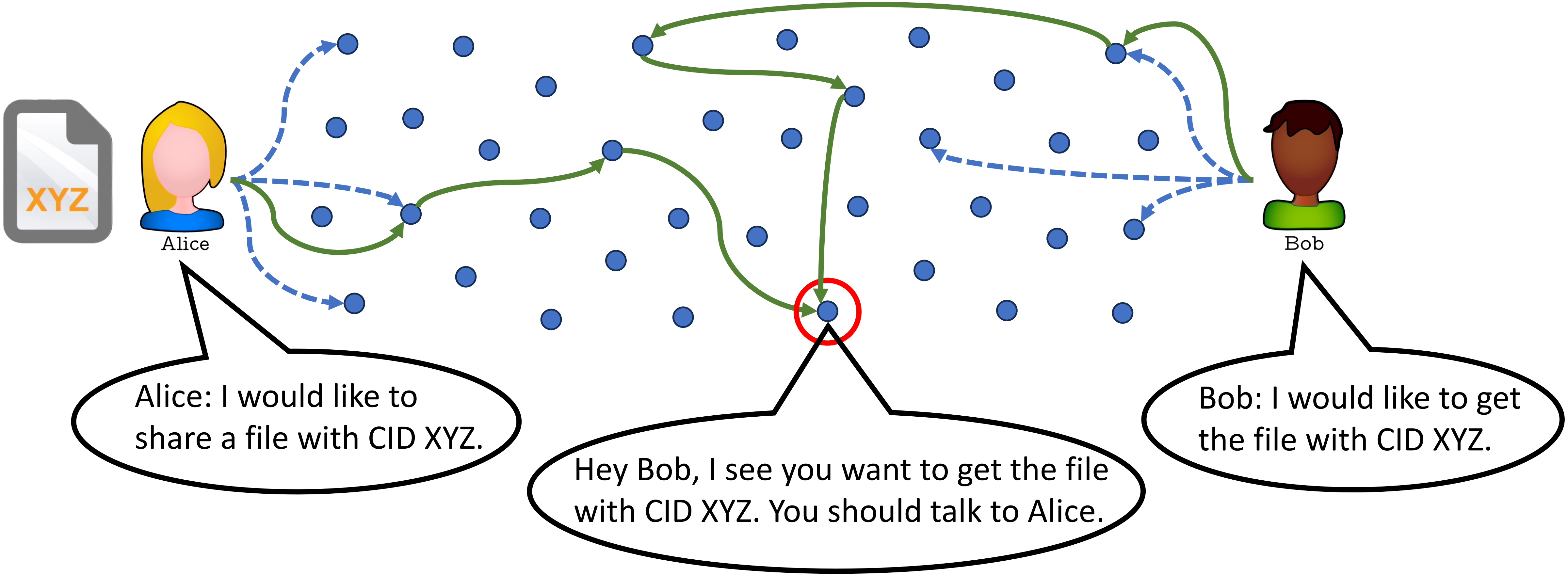


[5]

[4] LoadStorm and Econsultancy. [5] Pingdom.

P2P storage and distributed hash table (DHT)

- Most P2P storage systems are based on a DHT called Kademlia



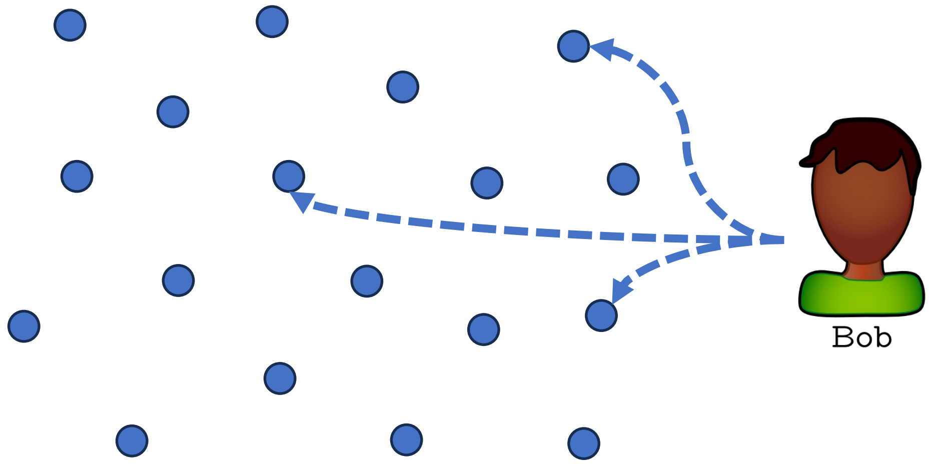
Alice: I would like to share a file with CID XYZ.

Hey Bob, I see you want to get the file with CID XYZ. You should talk to Alice.

Bob: I would like to get the file with CID XYZ.

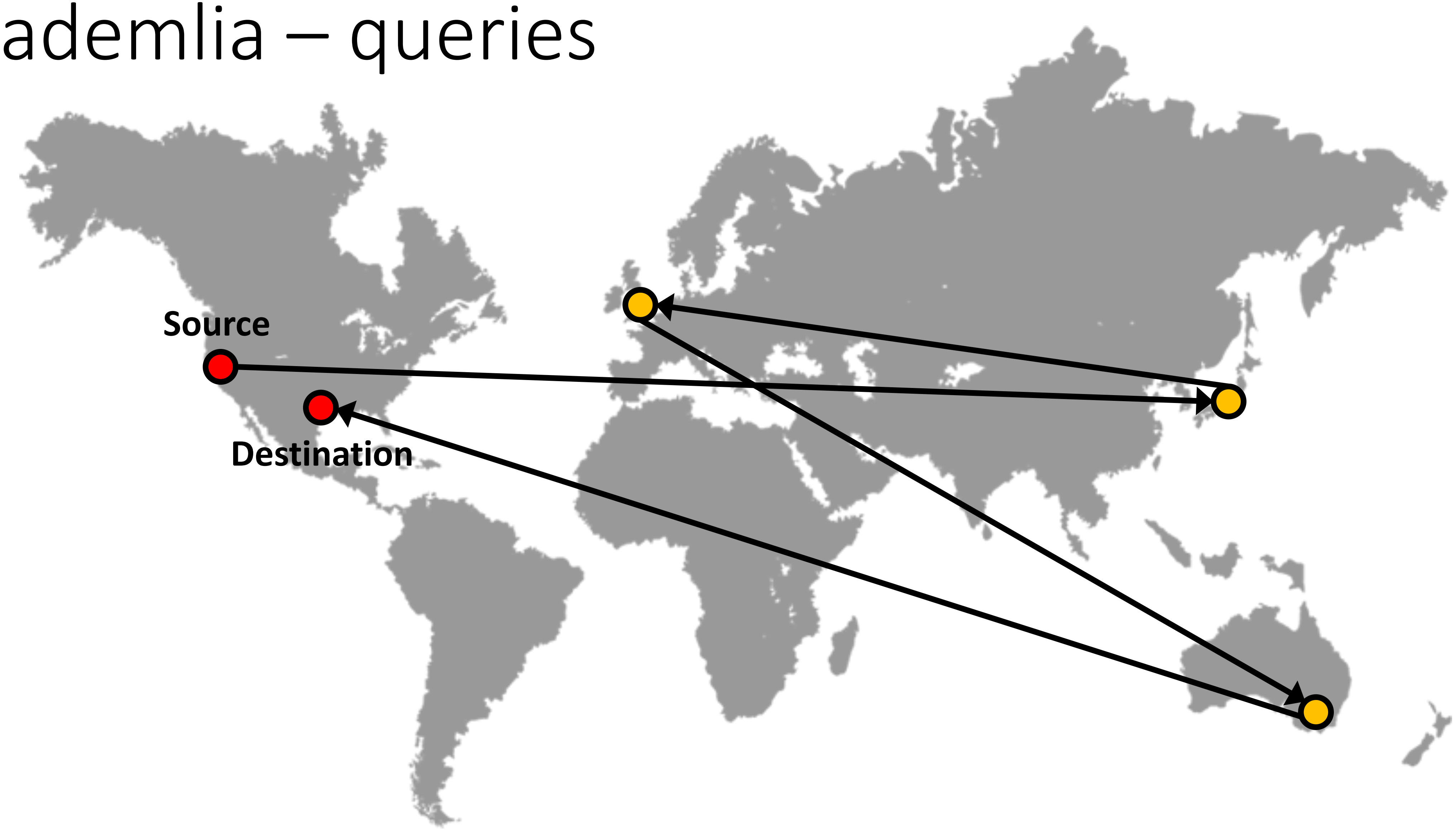
Kademlia – problem

- Many factors can affect the routing efficiency of a node
  - Locations, computing resources, network bandwidths, …
- Kademlia’s routing tables are randomly chosen

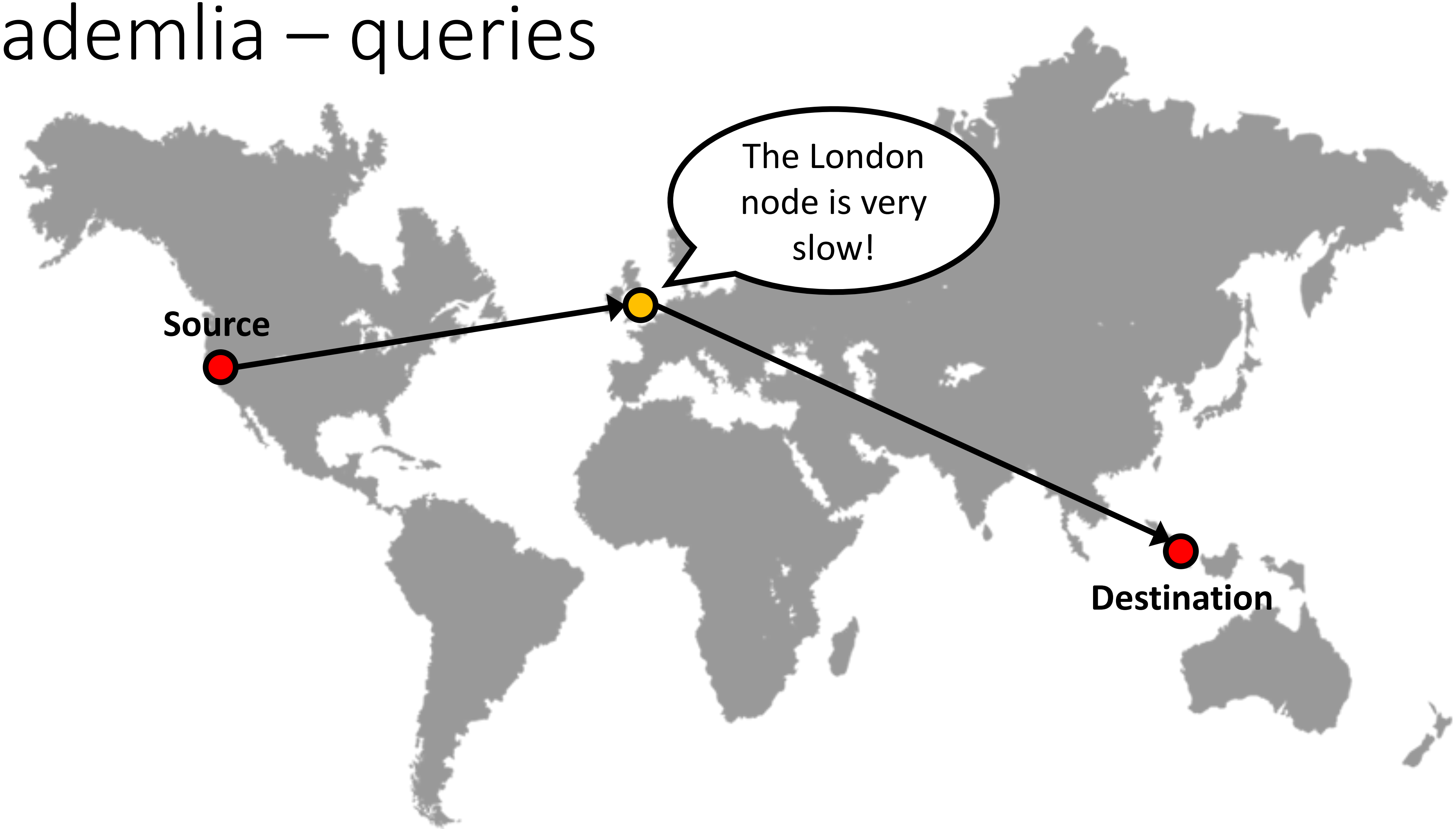


Nodes populate their routing tables randomly, completely agnostic of the environment!

Kademlia – queries



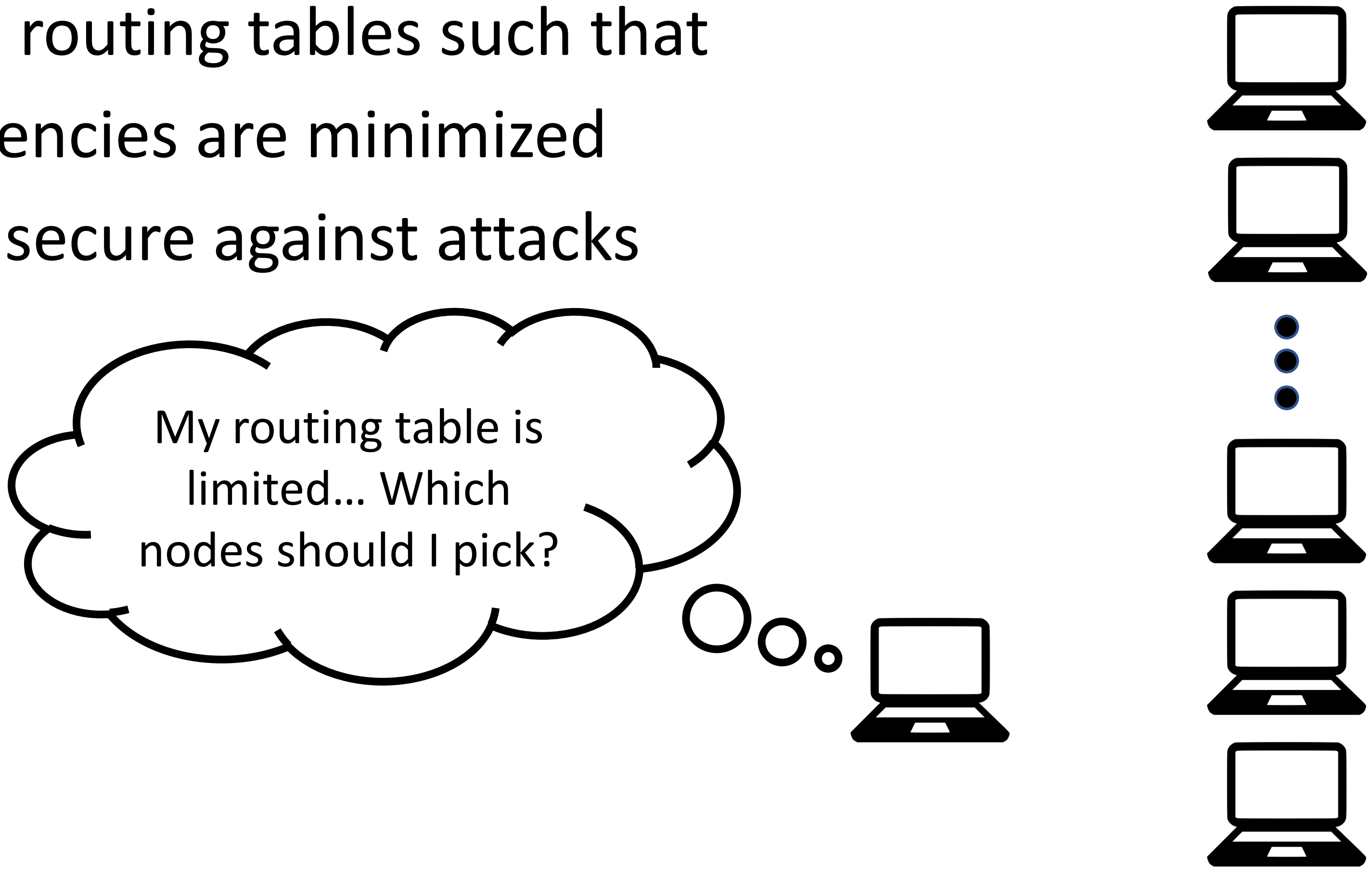
Kademlia – queries



Kadabra – goal

To configure routing tables such that

- Lookup latencies are minimized
- Nodes are secure against attacks



My routing table is limited… Which nodes should I pick?

Kadabra – summary

- The first effort to accelerate DHTs via a **data-driven approach**
- Automatically adapts to **heterogeneity and dynamism** in networks
- Motivated by the **multi-armed bandit problem**
- Employs novel **exploration-exploitation strategy**
- Completely **decentralized** and **robust against sybil attacks**

- Kadabra achieving between **15–50% lower lookup latencies** compared to state-of-the-art baselines

- In a nutshell, Kadabra nodes measure lookup latencies and utilize them to find the most suitable peer configuration for themselves.

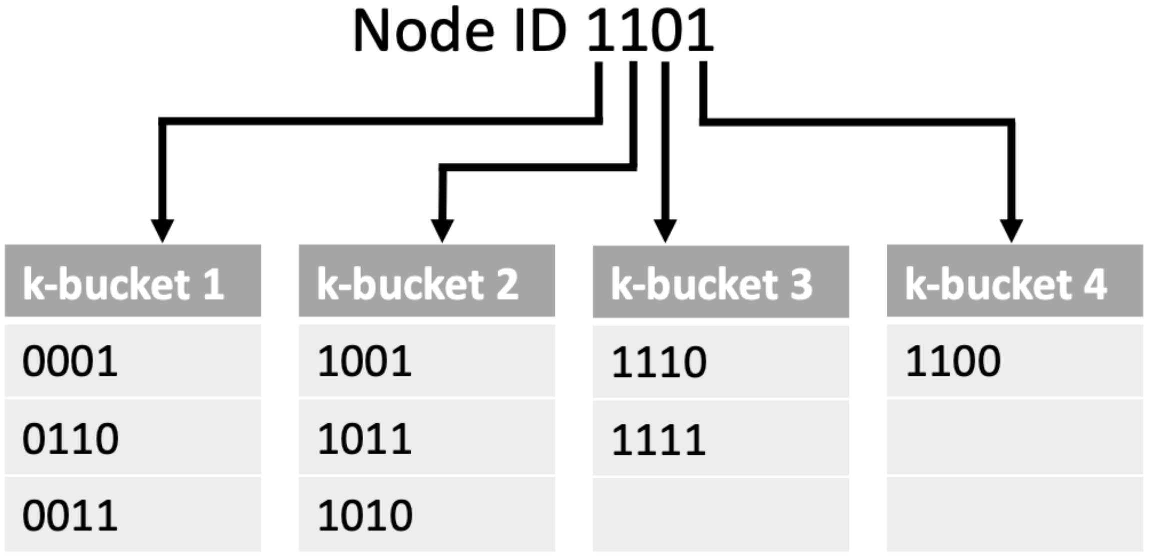
Speed up DHT (prior works)

- Location-awareness: [Castro 2002], [Hildrum 2002], and [Jain 2003]
- Parallel lookups: [Stutzbach 2006] and [Jimenez 2011]
- Recursive routing: [Heep 2010]
- Caching: [Einziger 2016] and [Liang 2009]

- The prior attempts did not completely solve the problem: they are either inefficient or vulnerable.

Kademlia DHT

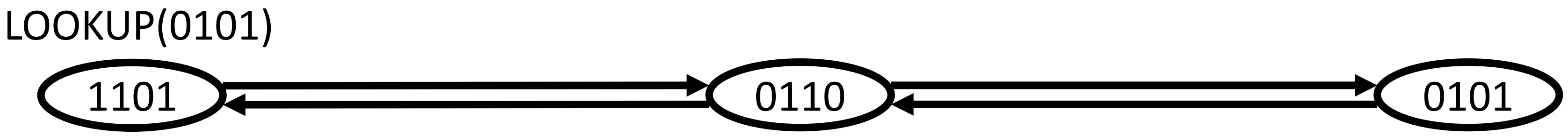
- Each node in a Kademlia network is assigned a 128-bit pseudo-random identifier called node ID.
- Routing tables in a Kademlia network consist of k-buckets. The data stored in k-buckets include node IDs, node IP addresses, ports.



Node ID 1101

k-bucket 1	k-bucket 2	k-bucket 3	k-bucket 4
0001	1001	1110	1100
0110	1011	1111	
0011	1010		

Kademlia DHT



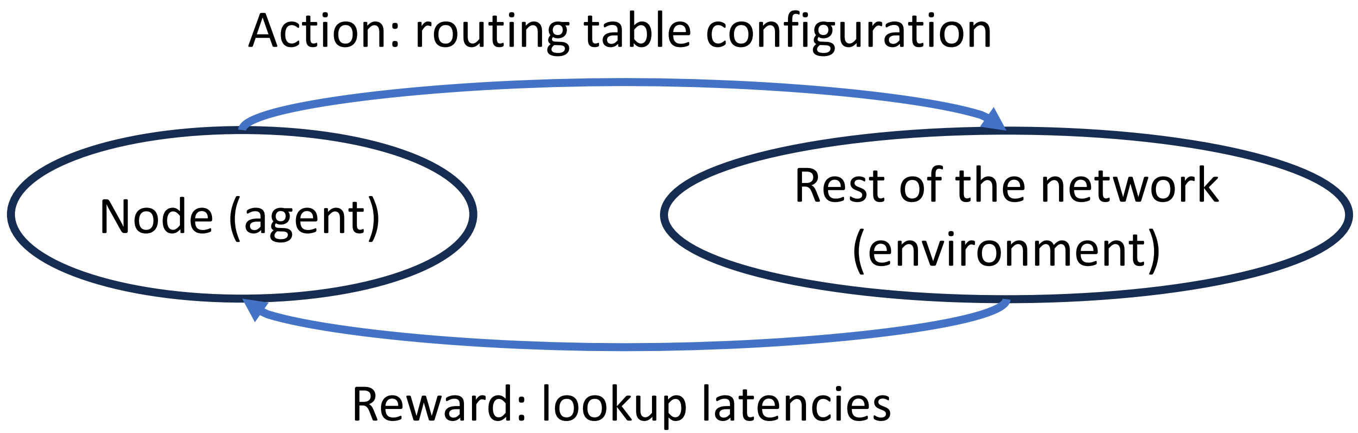
Node 1101 Routing Table	
k-bucket 1	0001, *0110*, 0011
k-bucket 2	1001, 1011, 1010
k-bucket 3	1110, 1111
k-bucket 4	1100

Node 0110 Routing Table	
k-bucket 1	1111, 1011, 1100
k-bucket 2	0001, 0011, 0010
k-bucket 3	*0101*, 0100
k-bucket 4	0111

Node 0101 Routing Table	
k-bucket 1	1101, 1011, 1001
k-bucket 2	0000, 0010, 0001
k-bucket 3	0110, 0111
k-bucket 4	0100

Kadabra – multi-armed bandit

- In Kadabra, we treat the routing table configuration problem as a multi-armed bandit problem



Action: routing table configuration

Node (agent)

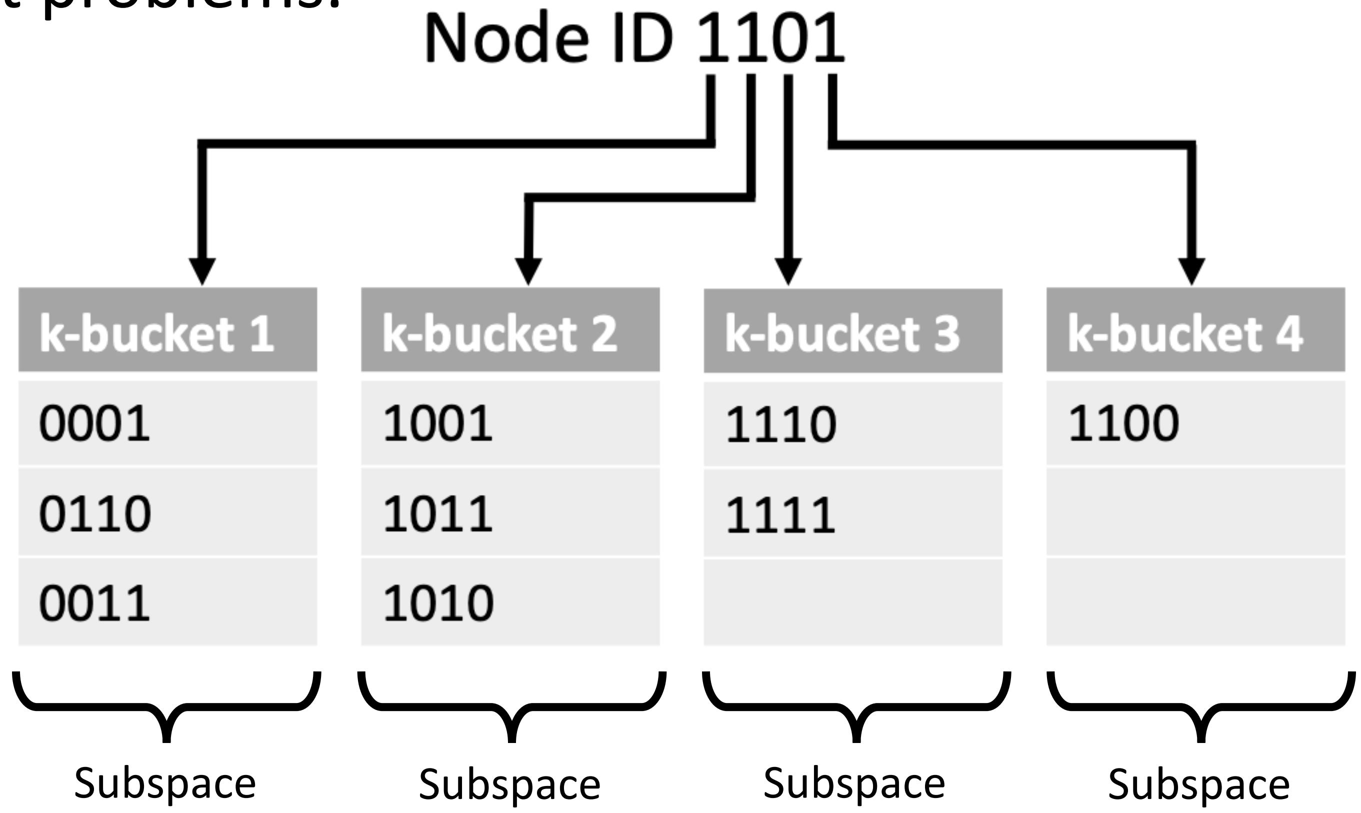
Rest of the network (environment)

Reward: lookup latencies

- In the naïve approach, the action space could be huge. How do we handle the huge action space?

Kadabra – subspaces

- In Kadabra, each k-bucket is a subspace, and we treat subspaces as independent problems.



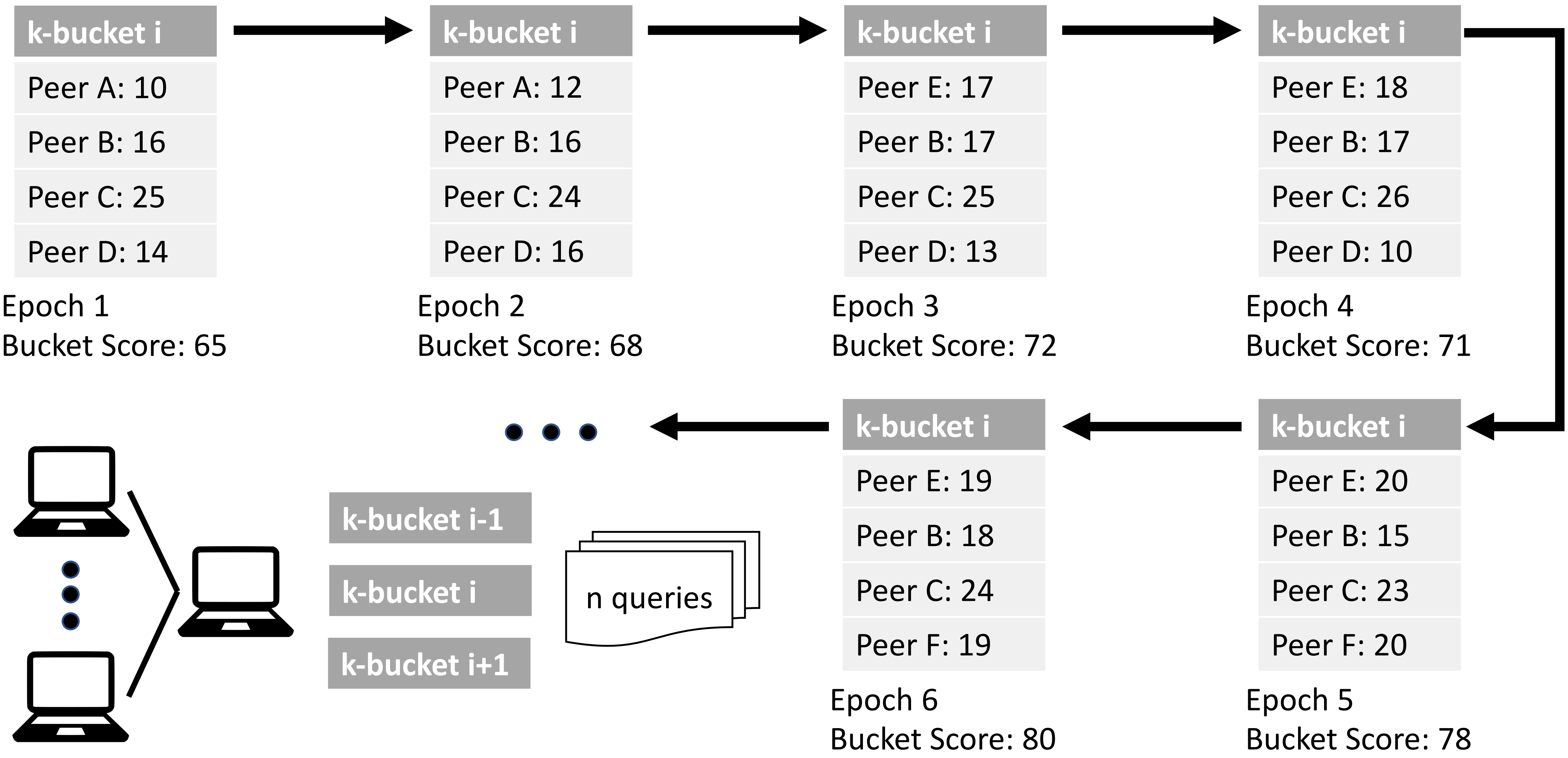
k-bucket 1	k-bucket 2	k-bucket 3	k-bucket 4
0001	1001	1110	1100
0110	1011	1111	
0011	1010		

Kadabra – algorithm

- The k-bucket stores a list of candidate peers $\mathcal{L}$
- At the end of an **epoch**
  - if next epoch is exploitation: the k-bucket runs the epoch with current peer configuration or previous peer configuration depends on which has better bucket score
  - if next epoch is exploration: the k-bucket replaces the worst scored peer with a random peer from $\mathcal{L}$
- The k-bucket keeps track of the bucket **score** of the current and the previous epoch

- When there are queries initiated or routed via the k-bucket, the k-bucket stores data pertaining to unseen peers and potentially add them to $\mathcal{L}$

Kadabra – learning



k-bucket i
Peer A: 10
Peer B: 16
Peer C: 25
Peer D: 14

Epoch 1
Bucket Score: 65

k-bucket i
Peer A: 12
Peer B: 16
Peer C: 24
Peer D: 16

Epoch 2
Bucket Score: 68

k-bucket i
Peer E: 17
Peer B: 17
Peer C: 25
Peer D: 13

Epoch 3
Bucket Score: 72

k-bucket i
Peer E: 18
Peer B: 17
Peer C: 26
Peer D: 10

Epoch 4
Bucket Score: 71

k-bucket i
Peer E: 20
Peer B: 15
Peer C: 23
Peer F: 20

Epoch 5
Bucket Score: 78

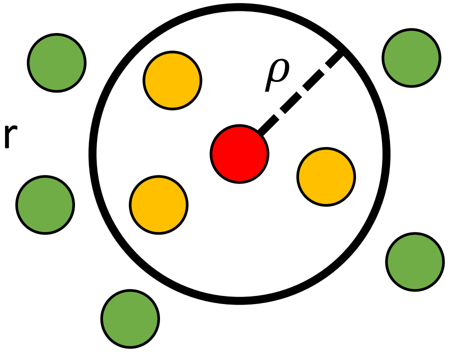
k-bucket i
Peer E: 19
Peer B: 18
Peer C: 24
Peer F: 19

Epoch 6
Bucket Score: 80

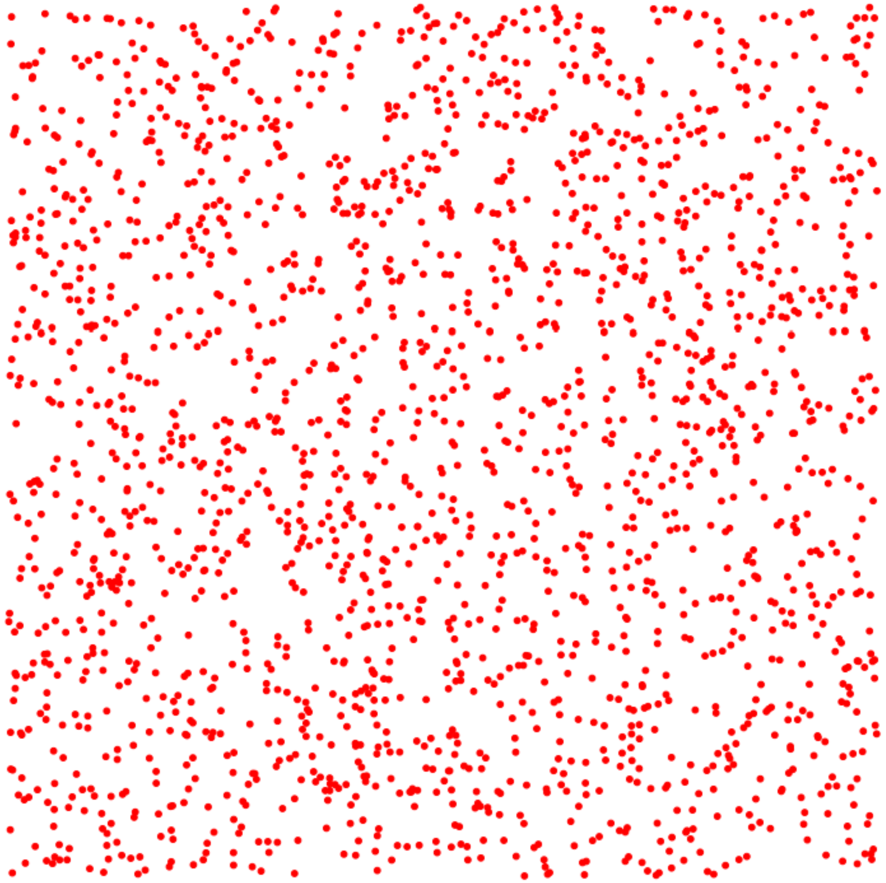
Kadabra – security

- A naïve way of sampling peers from $\mathcal{L}$ would be picking peers with smaller RTTs, just like Proximity Neighbor Selection (PNS)
- The naïve way is bad
  - PNS does not work under some scenarios
  - PNS is vulnerable to sybil attacks
- Kadabra k-bucket has a user-defined security parameter ρ
- Sampled peers must have RTTs > ρ

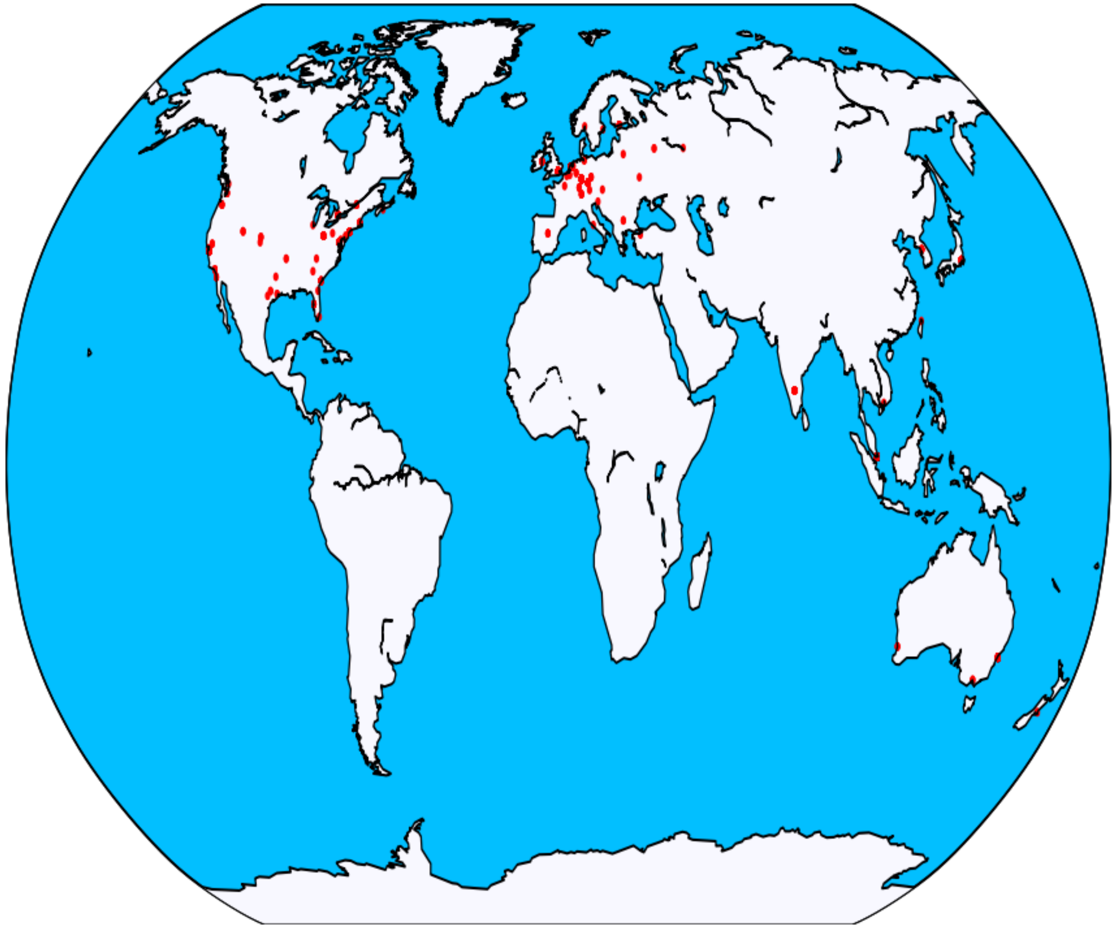
ρ makes Kadabra nodes prefer green peers rather than yellow peers.



Kadabra – evaluation



Square scenario

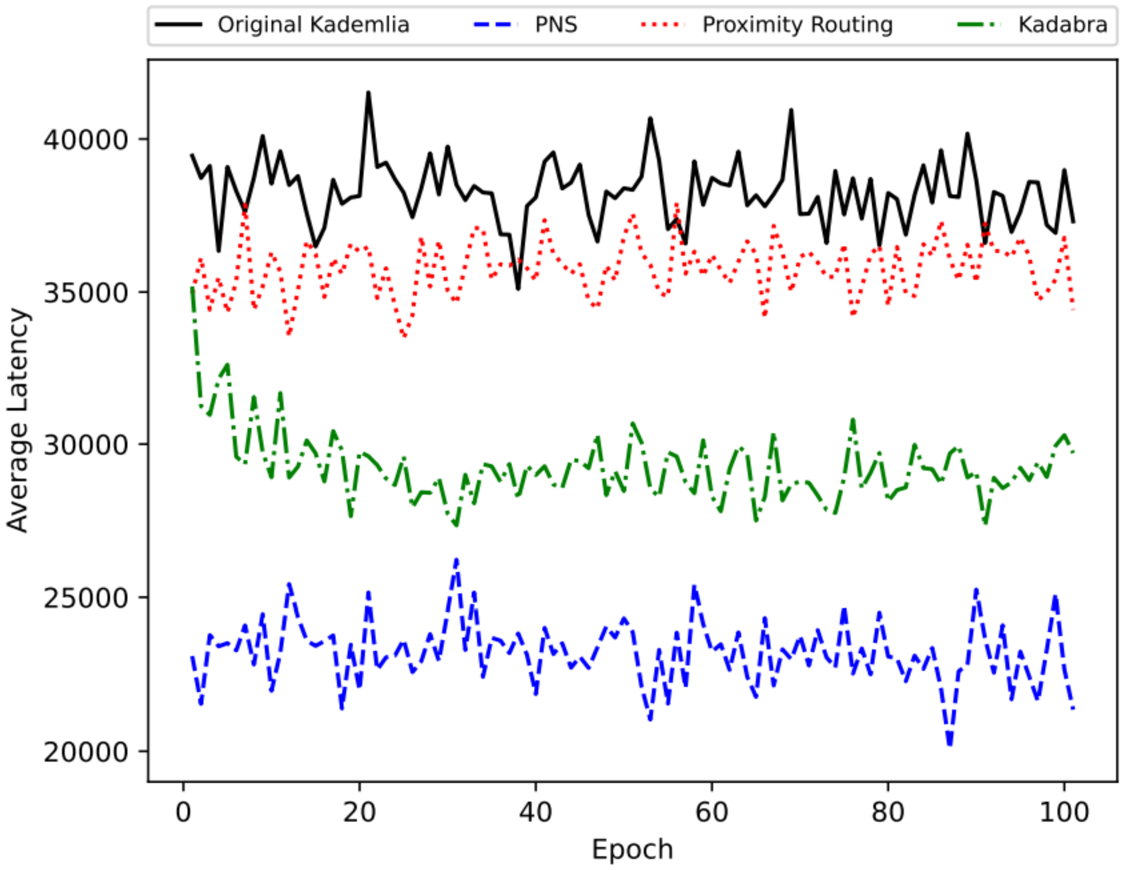


Real world scenario

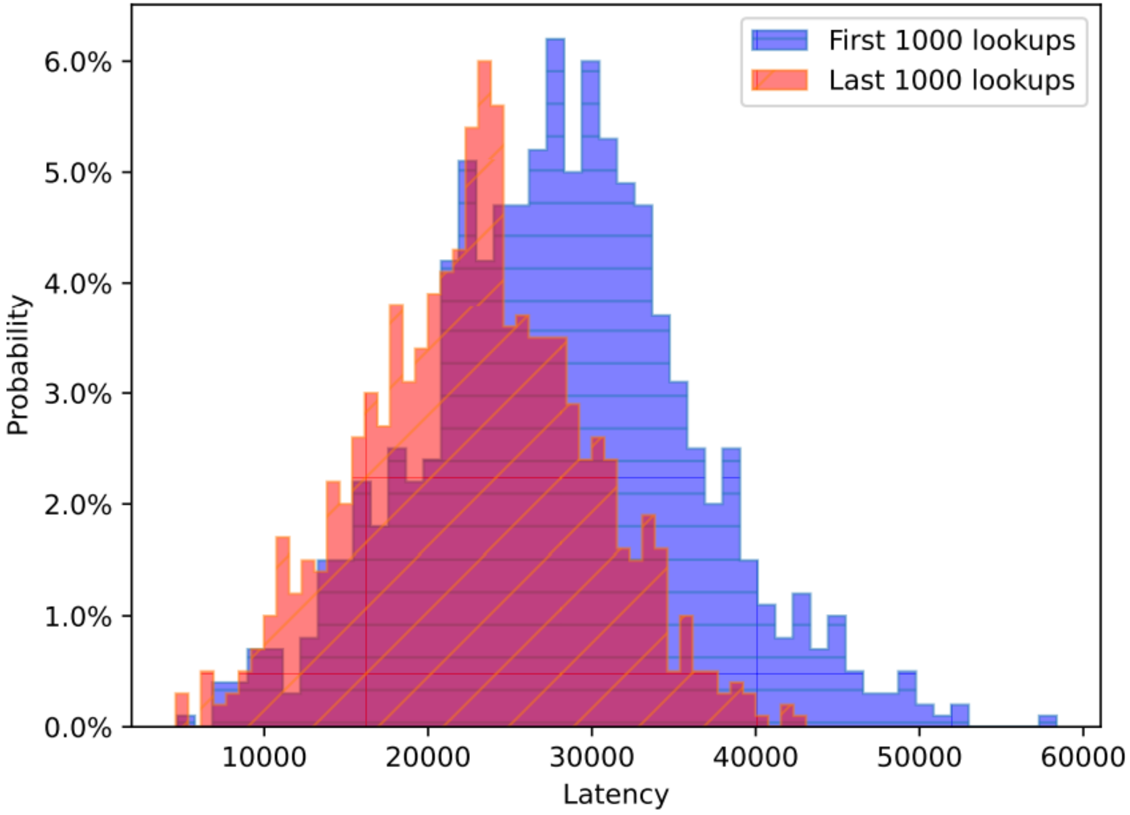
Kadabra – model

- We focus on recursive routing
- For a routing path v, u, and w
  - v: initiator
  - w: destination
- $RTT = l(v,u) + l(u,w) + \delta_w + l(w,u) + \delta_u + l(u,v)$
  - $l(x,y)$: the latency of forwarding a query from x to y
  - δ_x: the latency of uploading value over the Internet at x

Kadabra – evaluation: square

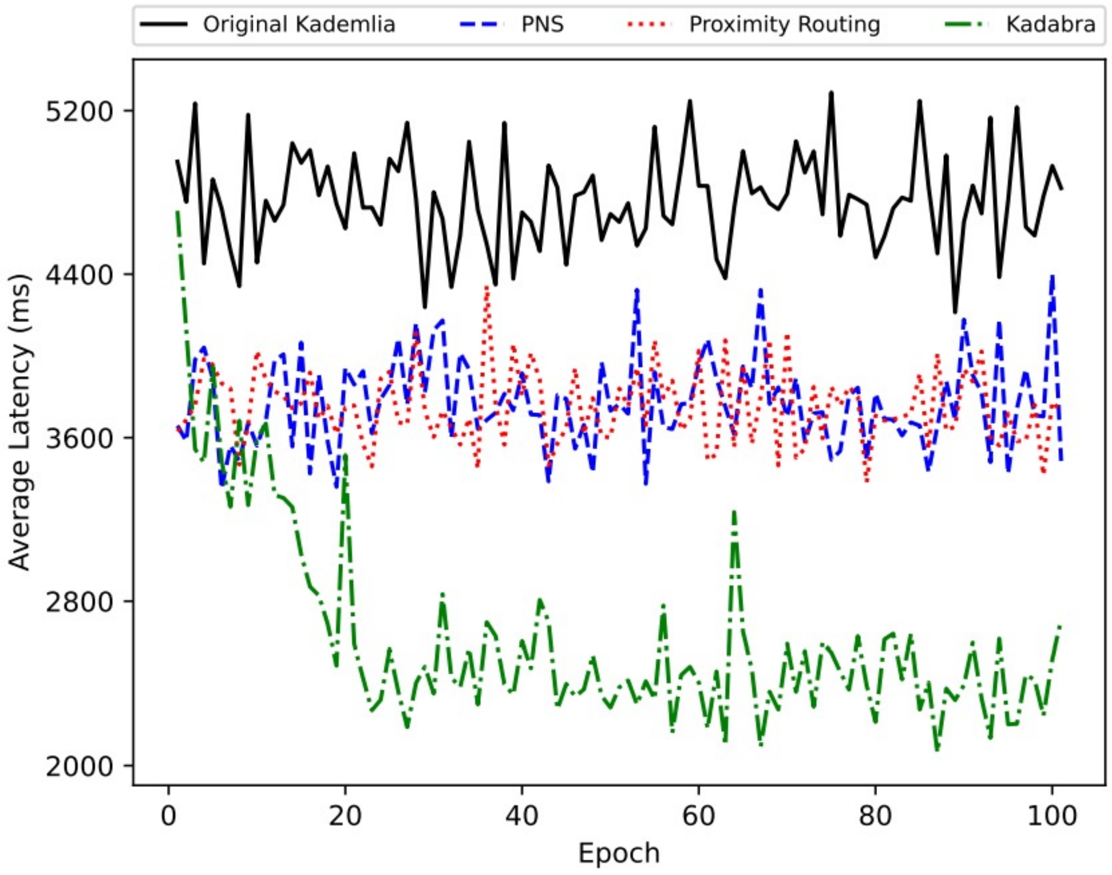


Performance of a random node

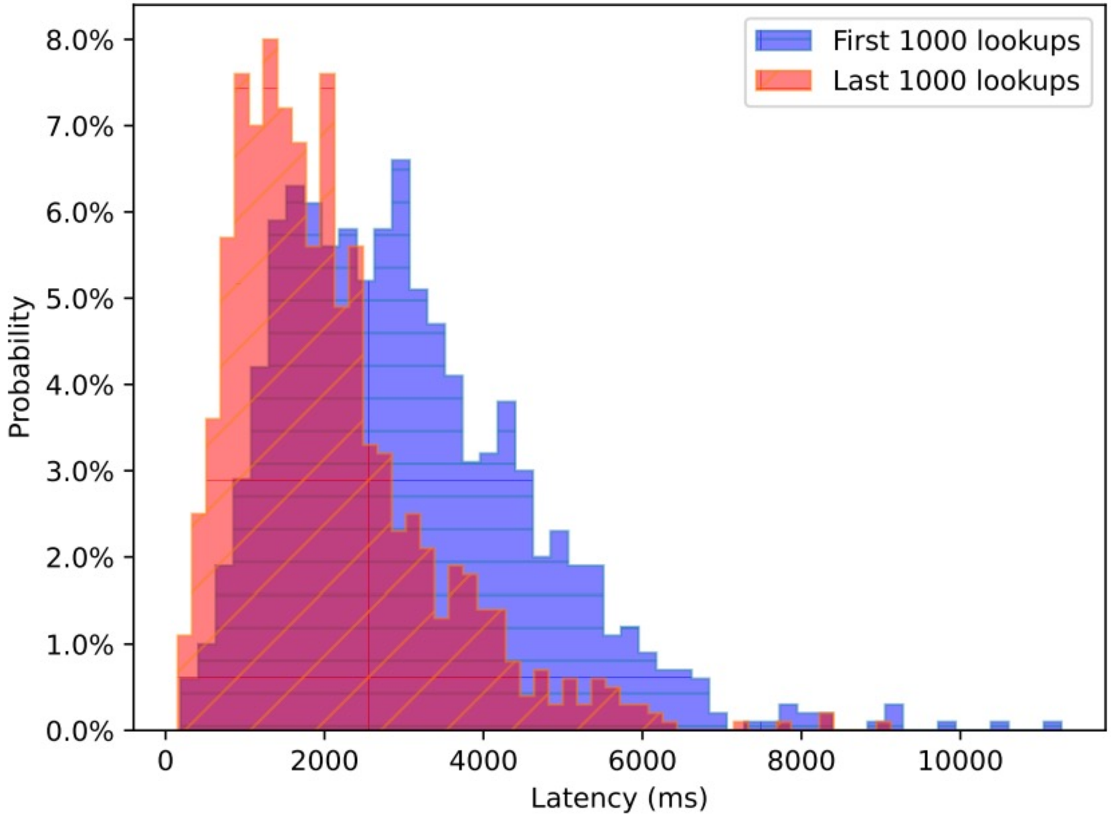


Performance distribution of all nodes

Kadabra – evaluation: real world

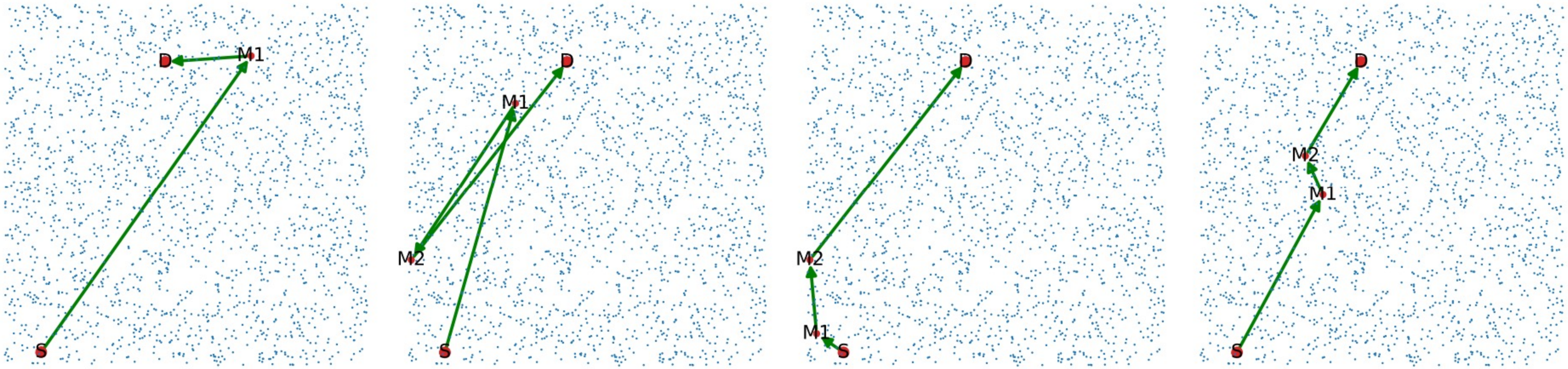


Performance of a random node



Performance distribution of all nodes

Kadabra – evaluation: close-up



(a) Vanilla: 24225 (b) PR: 27124 (c) PNS: 21976 (d) *Kadabra*: 15674

Lookup paths of the same query under different algorithms

Kadabra – evaluation

We evaluated Kadabra in the following situations as well

- KBR and DHT
- Recursive and iterative routing
- Hotspot: some resources are more popular
- Skewed network bandwidth: some nodes are slower
- Instability: introducing a noise to the node latency δ

Our results suggest that Kadabra outperforms the baselines algorithms in most situations and provides better security.

Kadabra – Takeaway

- Kadabra is a simple data-driven algorithm for p2p peer configuration
- Kadabra adapts to heterogeneity and dynamism in networks
- Kadabra is secure against common attacks

Future work

- Performance test in real-world networks
- Thorough analysis of robustness against attacks
- Theoretical understanding of Kadabra
  - We believe this can be done using theories from network potential games
  - Analyze the social cost of Kadabra networks in local connection games

Thank you!

Check out our papers!

Kadabra: Adapting Kademlia for the Decentralized Web

Honeybee: Decentralized Peer Sampling with Verifiable Random Walks for Blockchain Data Sharding

References

[Castro 2002] Castro, M., Druschel, P., Hu, Y.C., Rowstron, A.: Exploiting Network Proximity in Distributed Hash Tables. In: International Workshop on Future Directions in Distributed Computing (FuDiCo). pp. 52–55. Citeseer (2002)

[Hildrum 2002] Hildrum, K., Kubiatowicz, J.D., Rao, S., Zhao, B.Y.: Distributed Object Location in a Dynamic Network. In: Proceedings of the Fourteenth Annual ACM Symposium on Parallel Algorithms and Architectures. pp. 41–52 (2002)

[Jain 2003] Jain, S., Mahajan, R., Wetherall, D.: A Study of the Performance Potential of DHT-based Overlays. In: 4th USENIX Symposium on Internet Technologies and Systems (USITS 03) (2003)

[Stutzbach 2006] Stutzbach, D., Rejaie, R.: Improving Lookup Performance over a Widely-Deployed DHT. In: Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications. pp. 1–12. IEEE (2006)

[Jimenez 2011] Jimenez, R., Osmani, F., Knutsson, B.: Sub-Second Lookups on a Large-Scale Kademlia-Based Overlay. In: 2011 IEEE International Conference on Peer-to-Peer Computing. pp. 82–91. IEEE (2011)

[Maymounkov 2002] Maymounkov, P., Mazieres, D.: Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. In: International Workshop on Peer-to-Peer Systems. pp. 53–65. Springer (2002)

[Heep 2010] Heep, B.: R/Kademlia: Recursive and Topology-aware Overlay Routing. In: 2010 Australasian Telecommunication Networks and Applications Conference. pp. 102-107. IEEE (2010)

[Einziger 2016] Einziger, G., Friedman, R., Kantor, Y.: Shades: Expediting Kademlia's Lookup Process. Computer Networks 99, 37–50 (2016)

[Liang 2009] Liang, G.: An Improved Kademlia Routing Algorithm for P2P Network. In: 2009 International Conference on New Trends in Information and Service Science. pp. 63–66. IEEE (2009)