Kademlia – a P2P DHT

SSD 2022/23

FCUP-UP

Introduction

- Kademlia is a peer-to-peer (P2P) distributed hash table (DHT)
- Provides an overlay for building highly distributed applications
- Resilient to faults and attacks with no central points of failure (CPoF)
  - Including denial of service attacks
- Nodes can enter and leave the systems without compromising or disrupting the system

Kademlia Nodes

- Kademlia treats each node on a network as a leaf on a binary tree
  - Node have a 160-bit Unique Identifier (UId) (based on SHA-1)
  - A Node's position (in the tree) is determined by the shortest unique prefix of its UId
- Kademlia protocol ensures that each node knows of at least one node on each of its sub-trees
  - Thus, a node can locate any other node by its ID
- Node selection for storing a key-value pair relies on a notion of distance between two identifiers
  - Given two 160-bit identifiers, x and y, distance between them is $x \oplus y$ (XOR).

Inter-Node Messaging

- The Kademlia protocol consists of four Remote Procedure Calls (RPCs):
- **PING:** probes a node to see if it's online
- **STORE:** instructs a node to store a key-value pair
- **FIND_NODE:** returns information about the *k* nodes closest to the target id
- **FIND_VALUE:** similar to the FIND_NODE RPC, but if the recipient has received a STORE for the given key, it just returns the stored value

Node Joining

- Joining a Kademlia network requires discovery of a single node
  - Bootstrap node
- A joining node announces itself to the Boostrap node
- The Boostrap node replies with a set of “closest” nodes
  - Collects the NodeID from each response and adds it to its own peer table

- Parallel and asynchronous queries can prevent timeout delays or ‘retrieval hold-ups’
  - Due to failed nodes which have dropped off or left the network

Routing Tables and K-buckets

- The routing table is a binary tree whose leaves are **k-buckets**
  - Nodes maintain detailed knowledge of the address space closest to them
  - Exponentially decreasing knowledge of more distant address space
- **K-buckets** are lists of routing addresses of other nodes in the network
  - Contain the IP address, port, and NodeID of the respective nodes

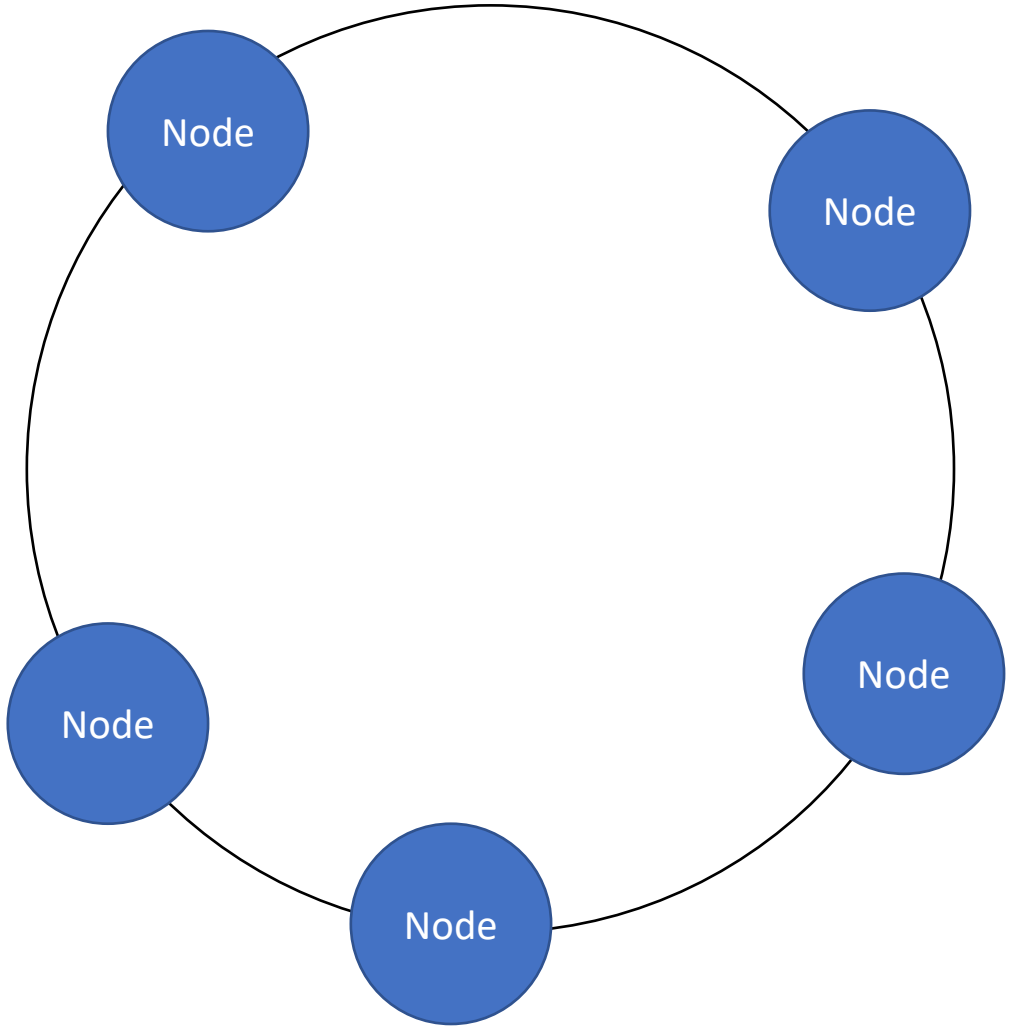
- The routing table size is asymptotically bounded by $O(\log_2(n/k))$
  - Where *n* is the actual number of nodes in the network and *k* is the bucket size
  - Larger buckets slightly reduce the total number of buckets in the routing table.
- Symmetry is useful since it means that each of these closest contacts will be maintaining detailed knowledge of a similar part of the address space
  - Rather than a remote part.

Routing Tables and K-buckets

- Routing table is distributed and subject to different versions/visions
  - Each node maintains and manages a mapping for a subset of the nodes on the network in its own routing table
  - No absolute truth exists to where NodeIDs are mapped to their address

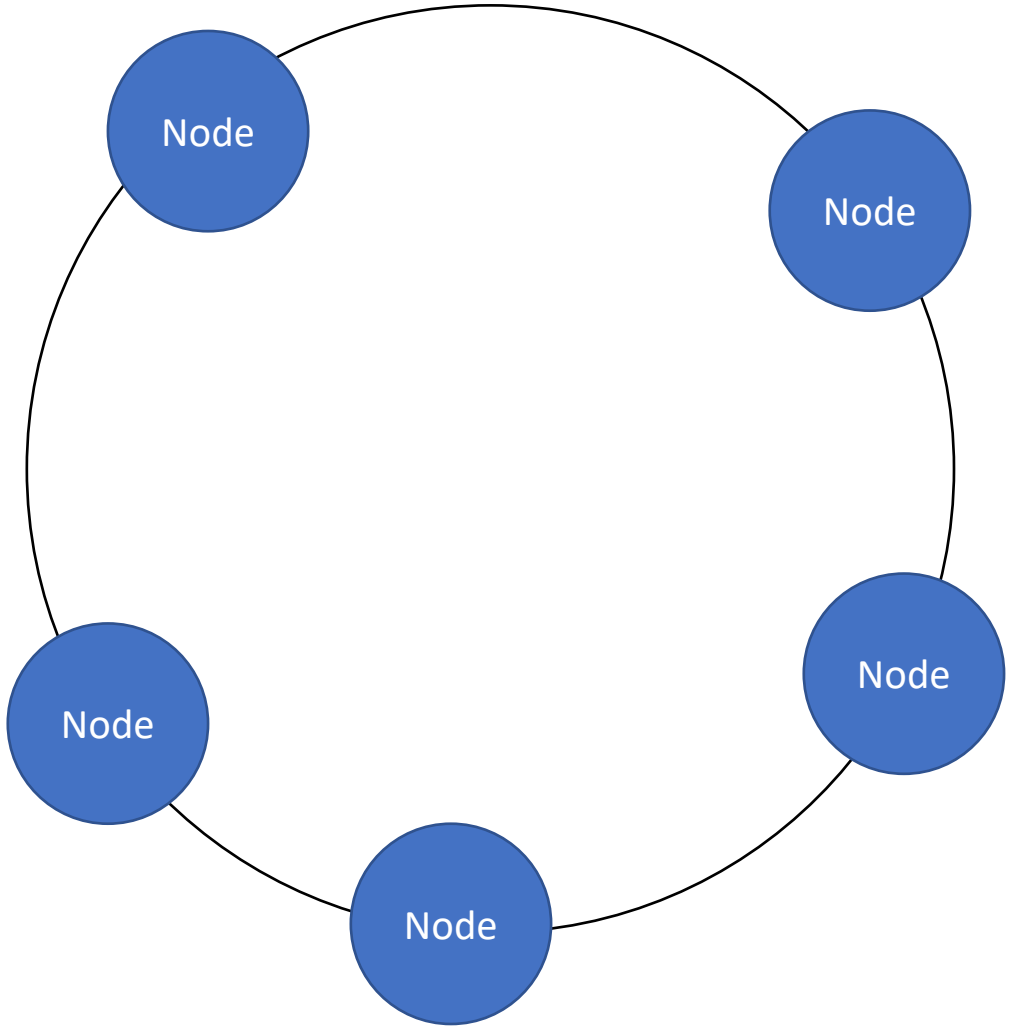
Join Network

- Each node has:
  - IP address
  - Port No
  - Unique ID
  - Tree<Buckets> (routing table)
  - Storage
- Special Nodes
  - Bootstrap Nodes
  - Know all nodes

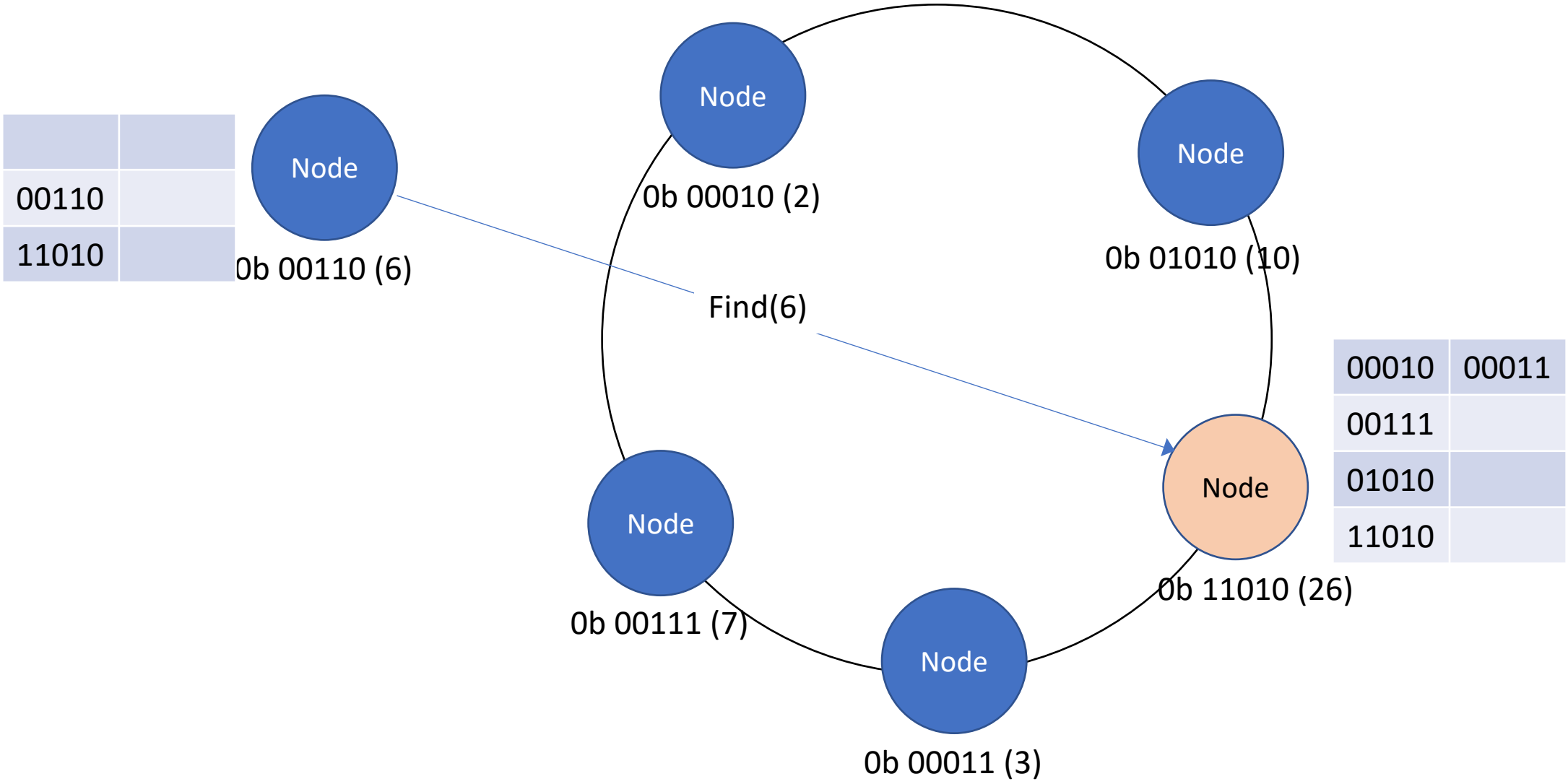


Join Network

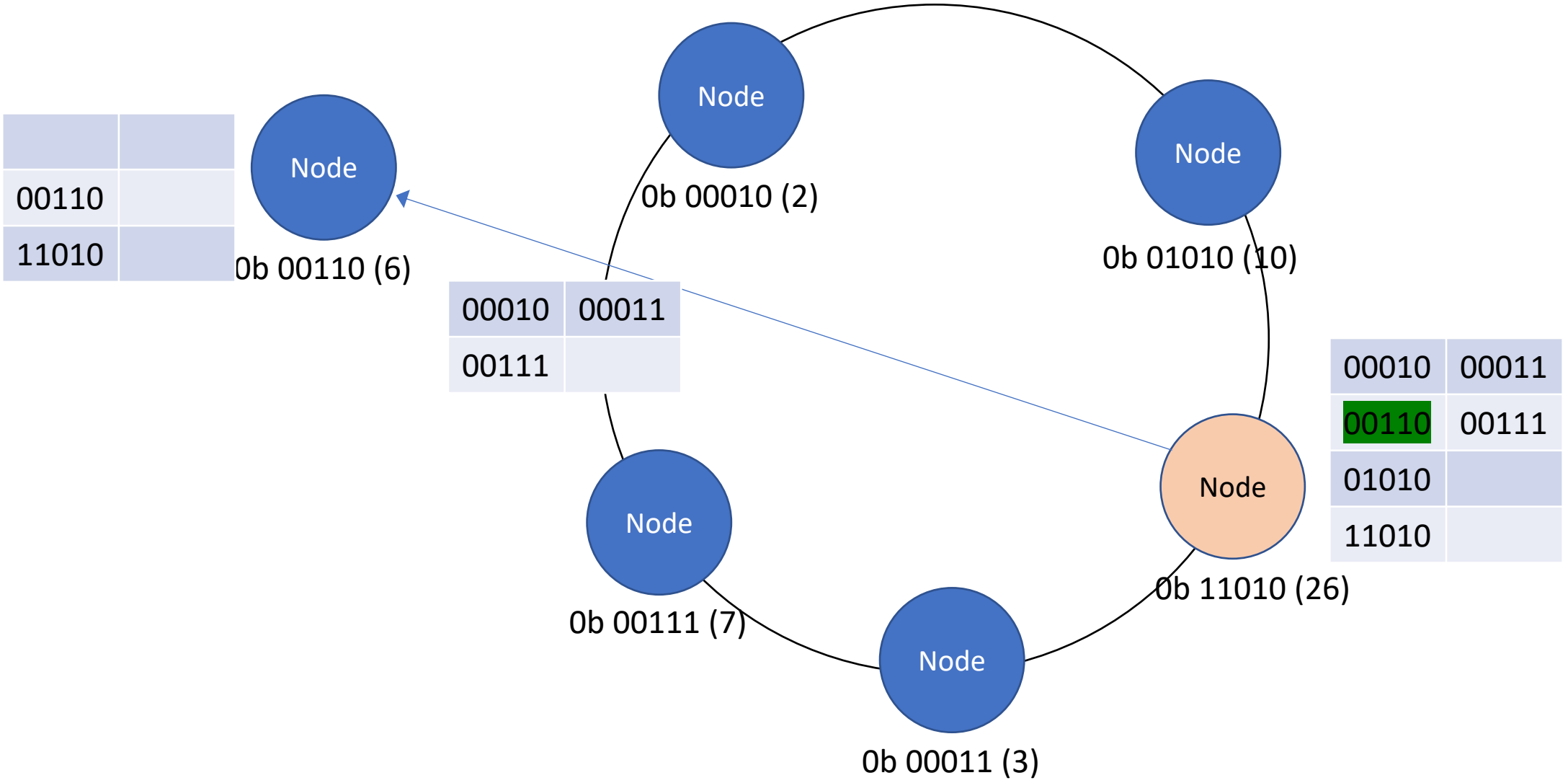
- Entering the group
  - Sends (IP, Port, ID) to a Bootstrap Node
    - Using FIND_NODE
  - Bootstrap Adds the new Node info to its routing table
    - Replies with a set of near nodes
  - Send (IP, Port, ID) to each nodes received
    - Repeat if new near nodes are received



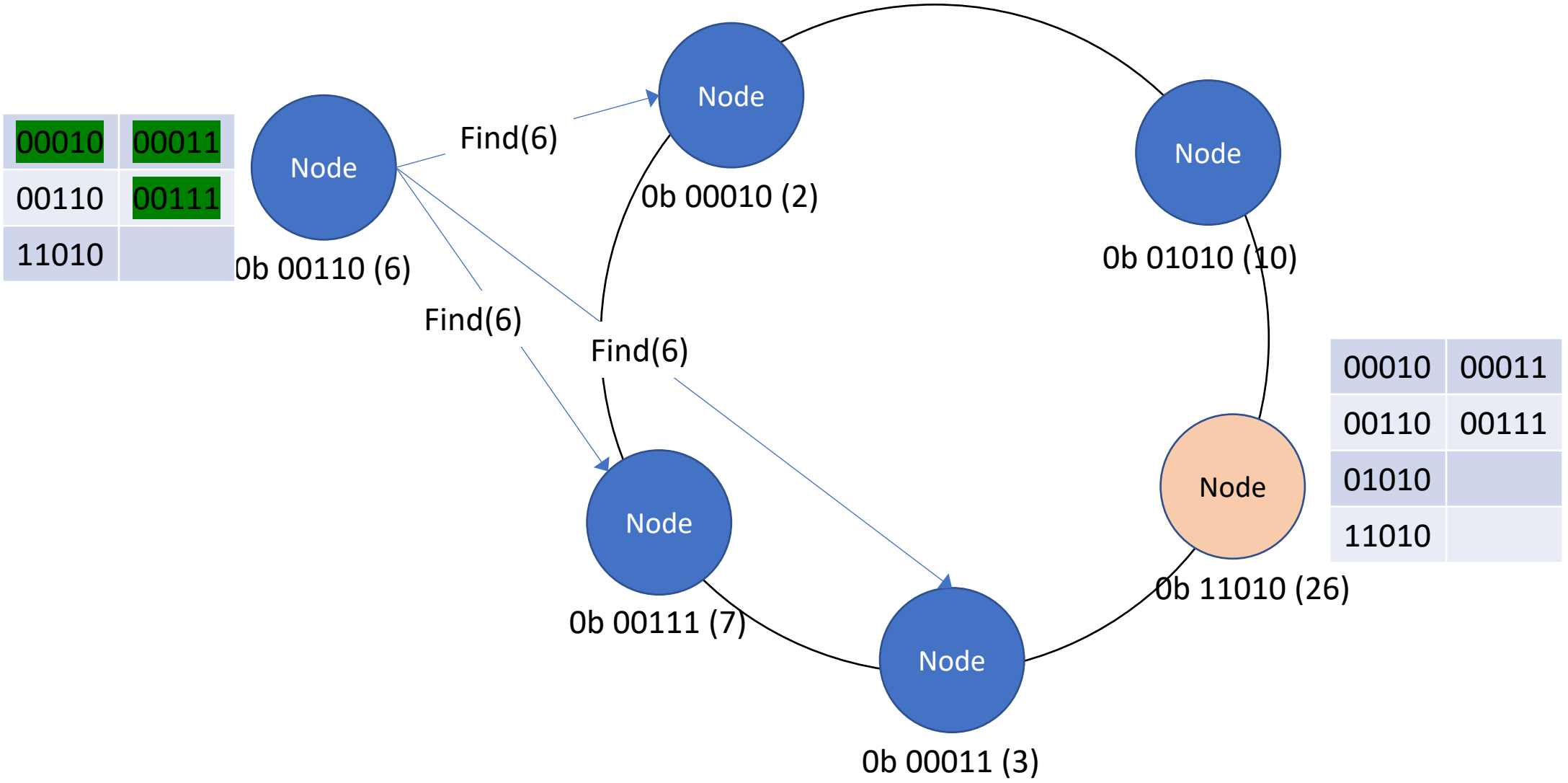
Join Network



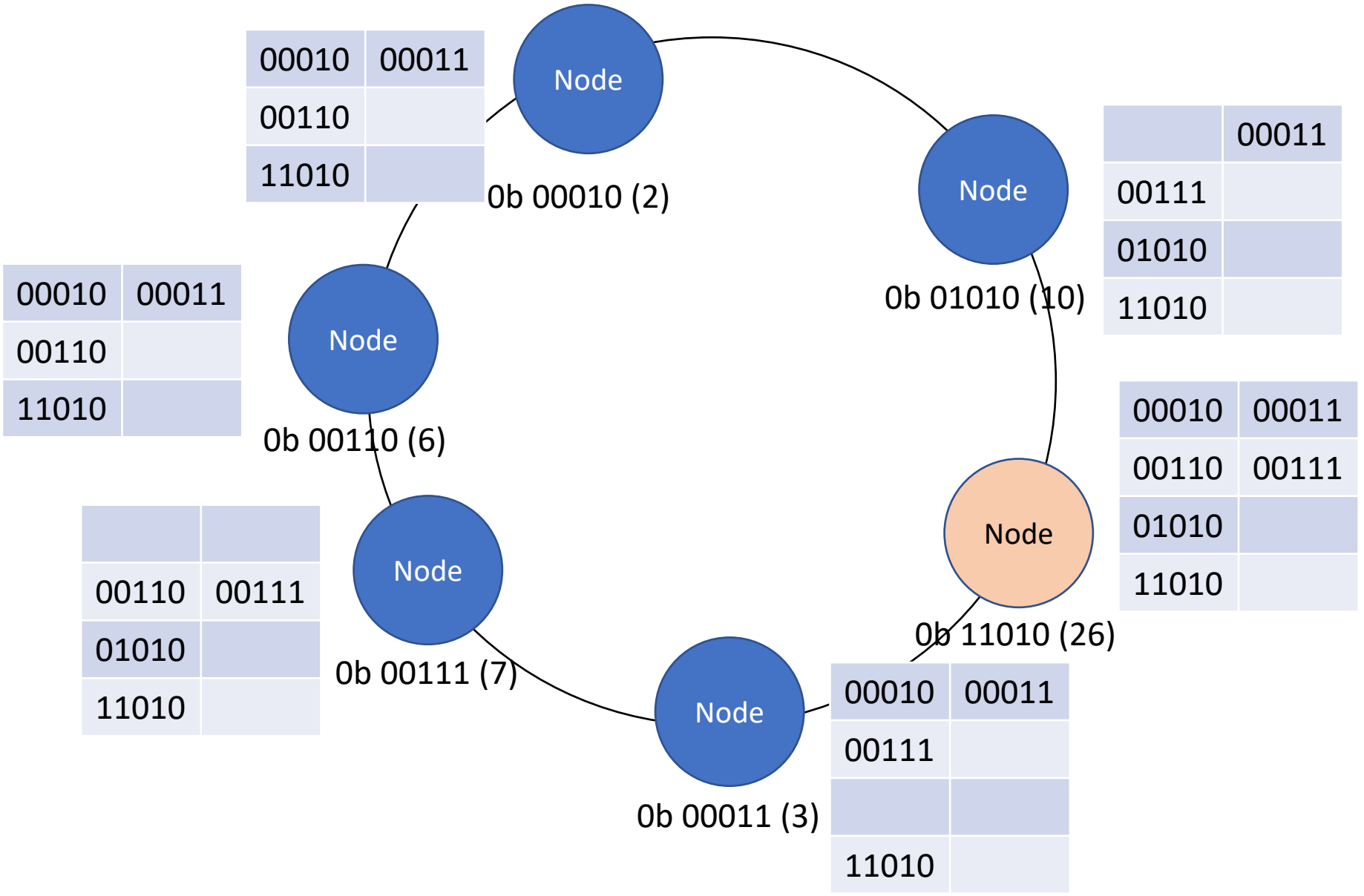
Join Network



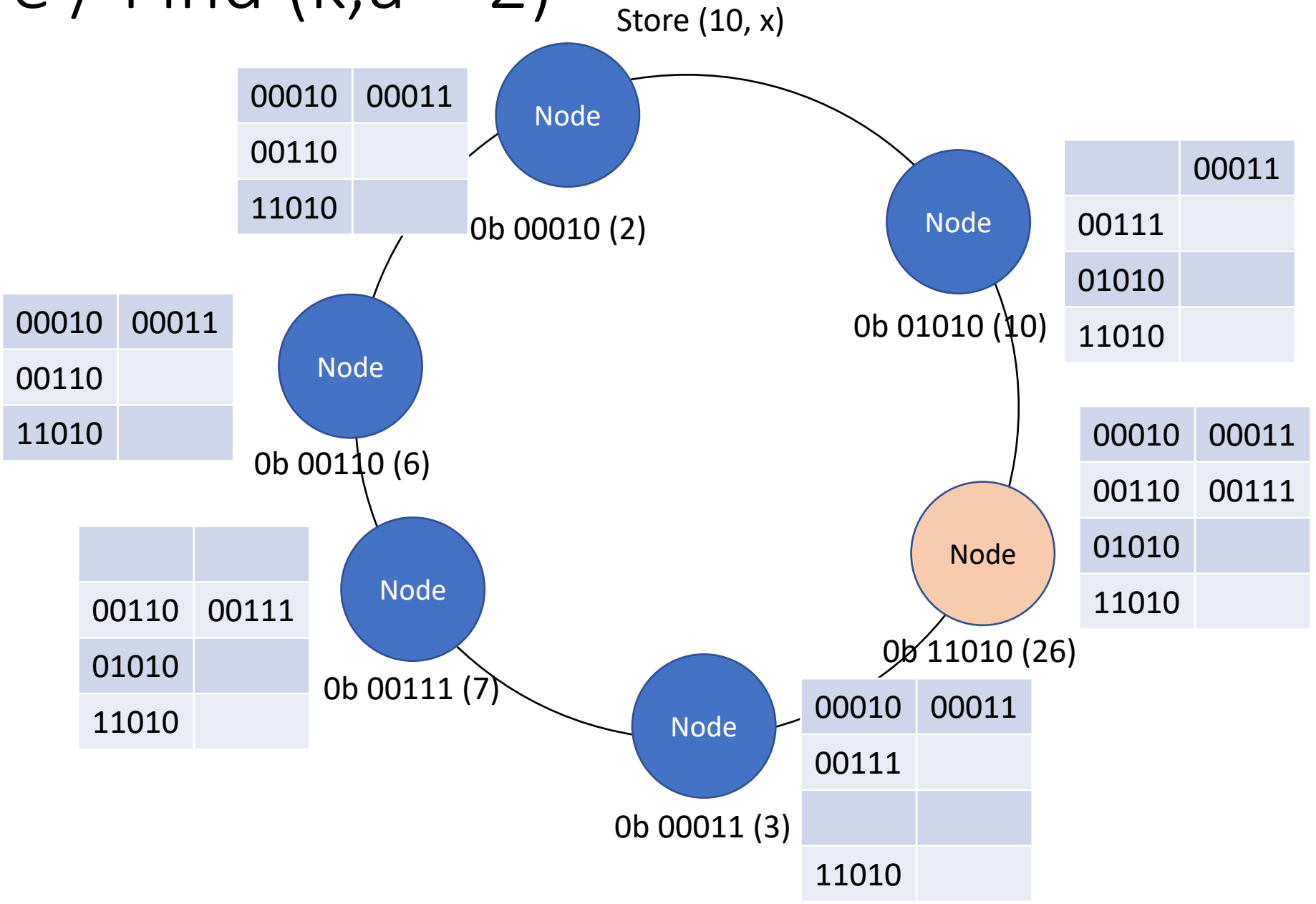
Join Network



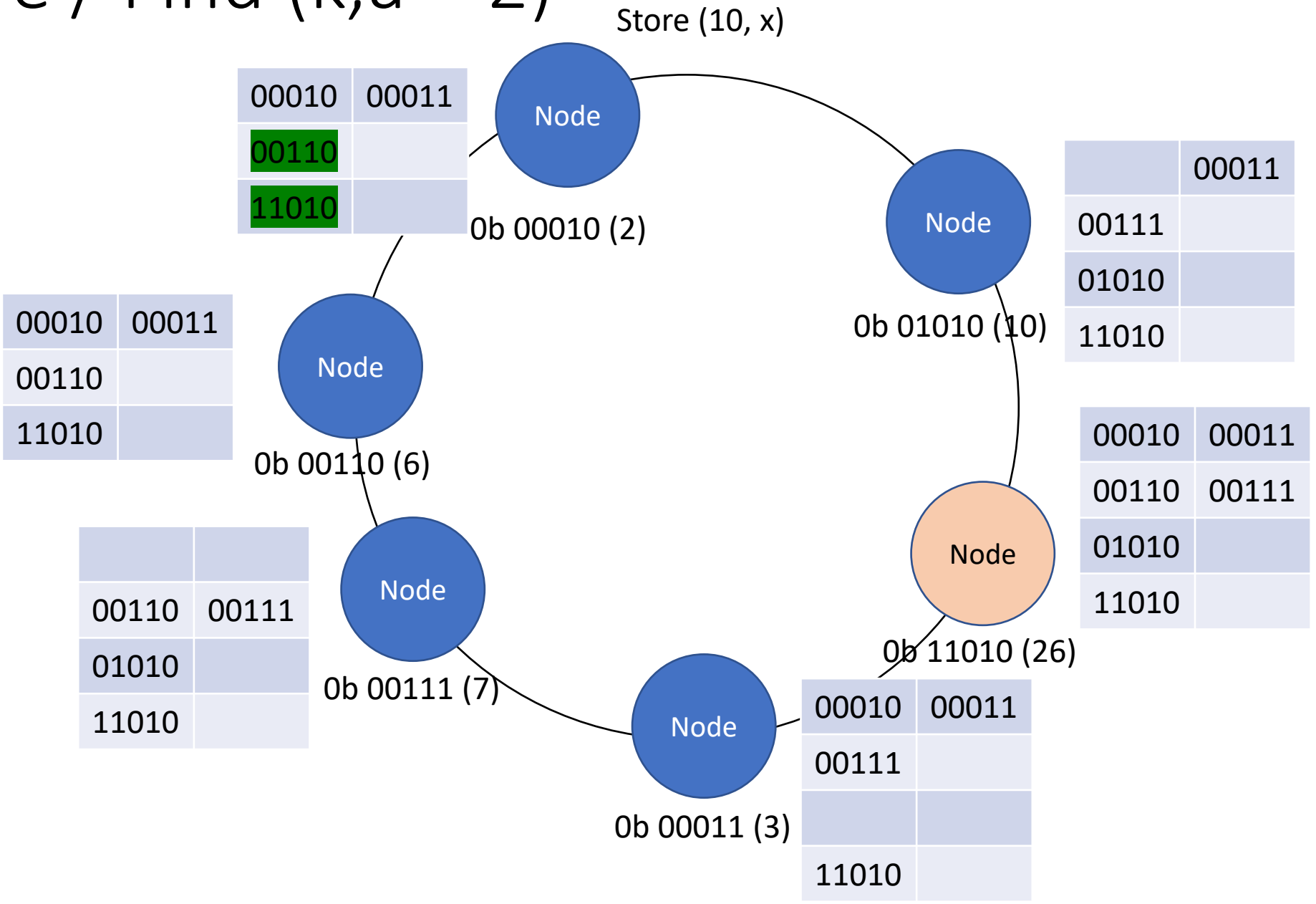
Store / Find


Node
0b 00010 (2)
00010 00011
00110
11010
Node
0b 01010 (10)
00011
00111
01010
11010
Node
0b 00110 (6)
00010 00011
00110
11010
Node
0b 00111 (7)
00110 00111
01010
11010
Node
0b 00011 (3)
00010 00011
00111
11010
Node
0b 11010 (26)
00010 00011
00110 00111
01010
11010

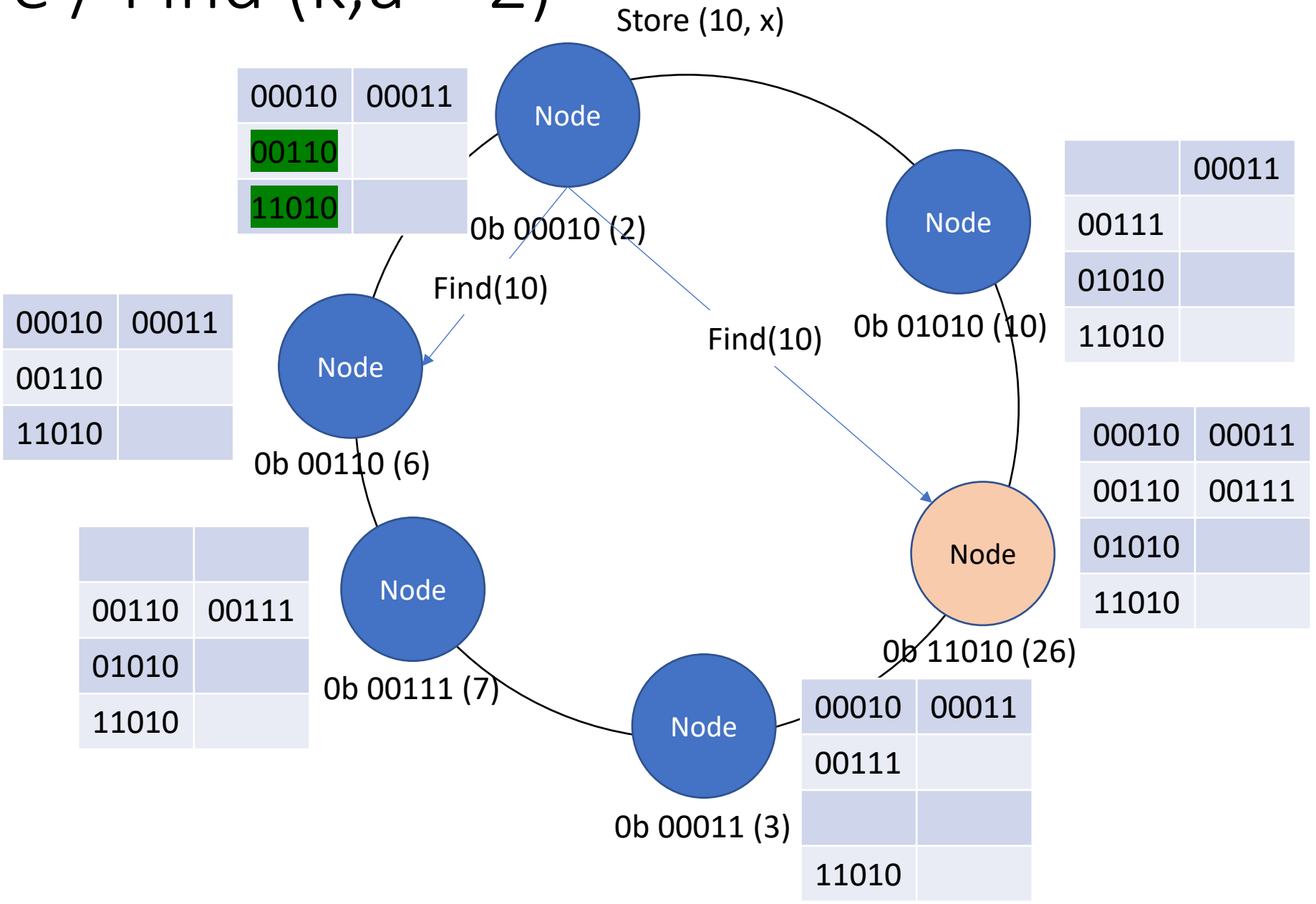
Store / Find (k,a = 2)



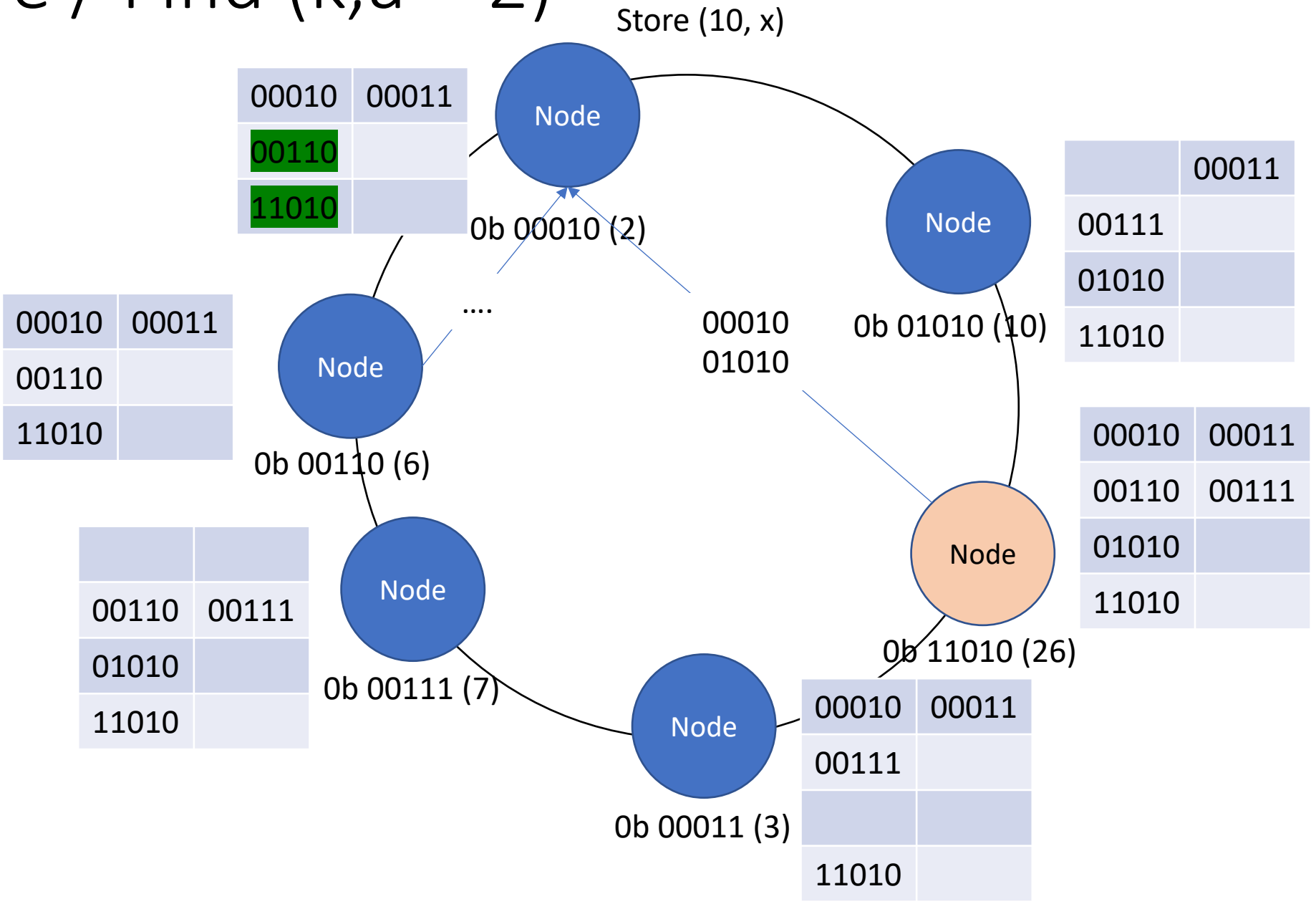
Store / Find (k,a = 2)



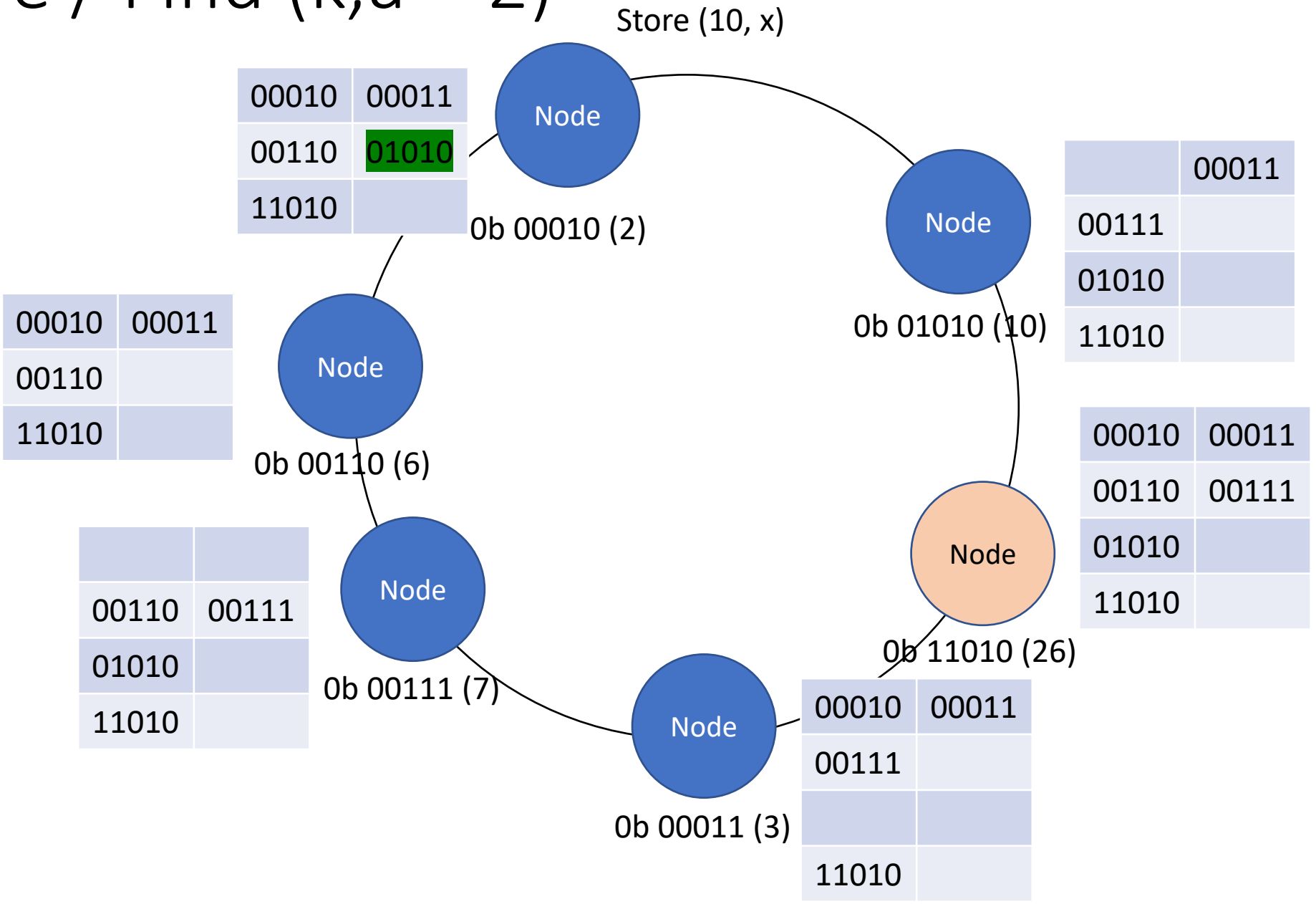
Store / Find (k,a = 2)



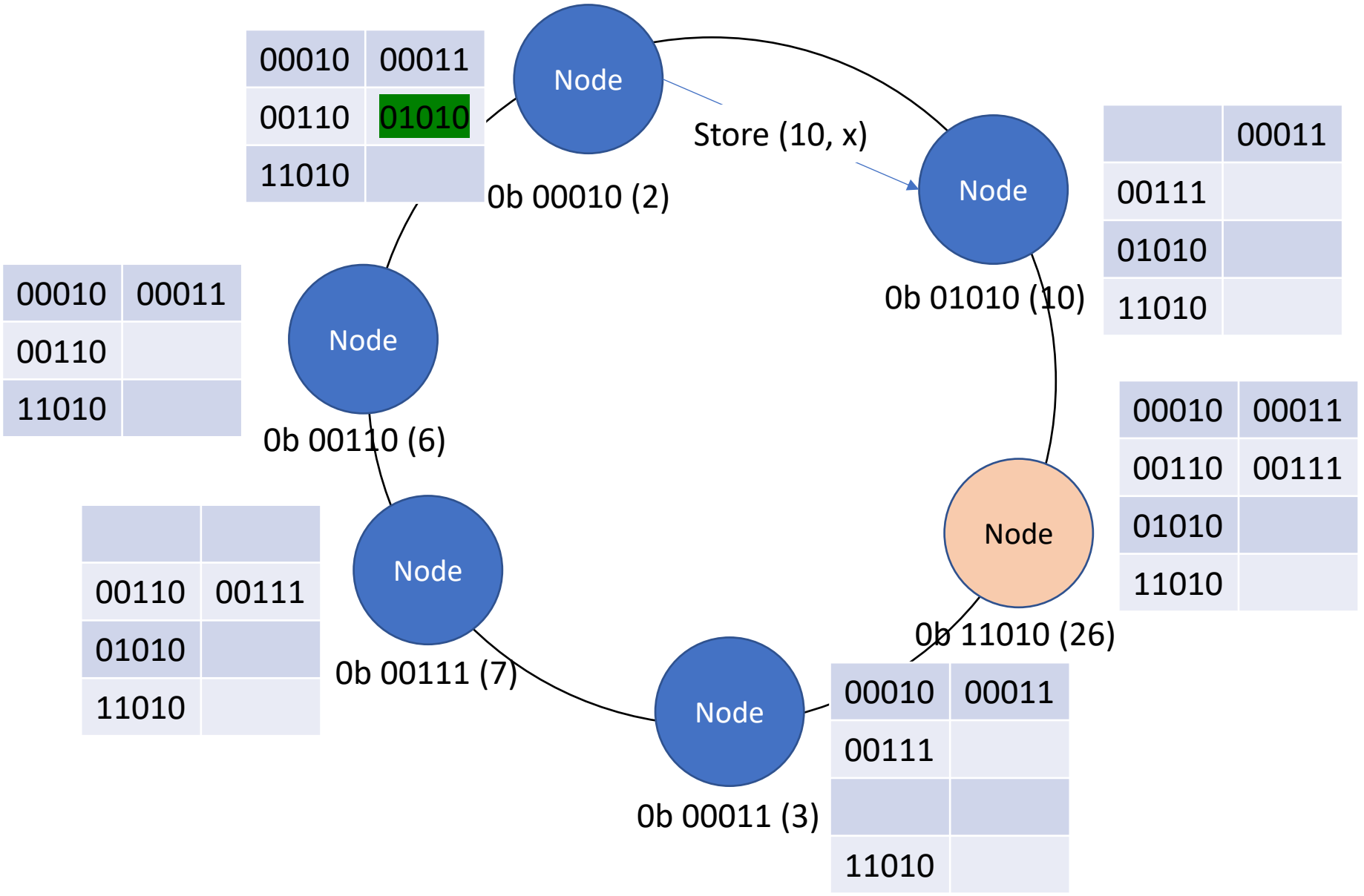
Store / Find (k,a = 2)



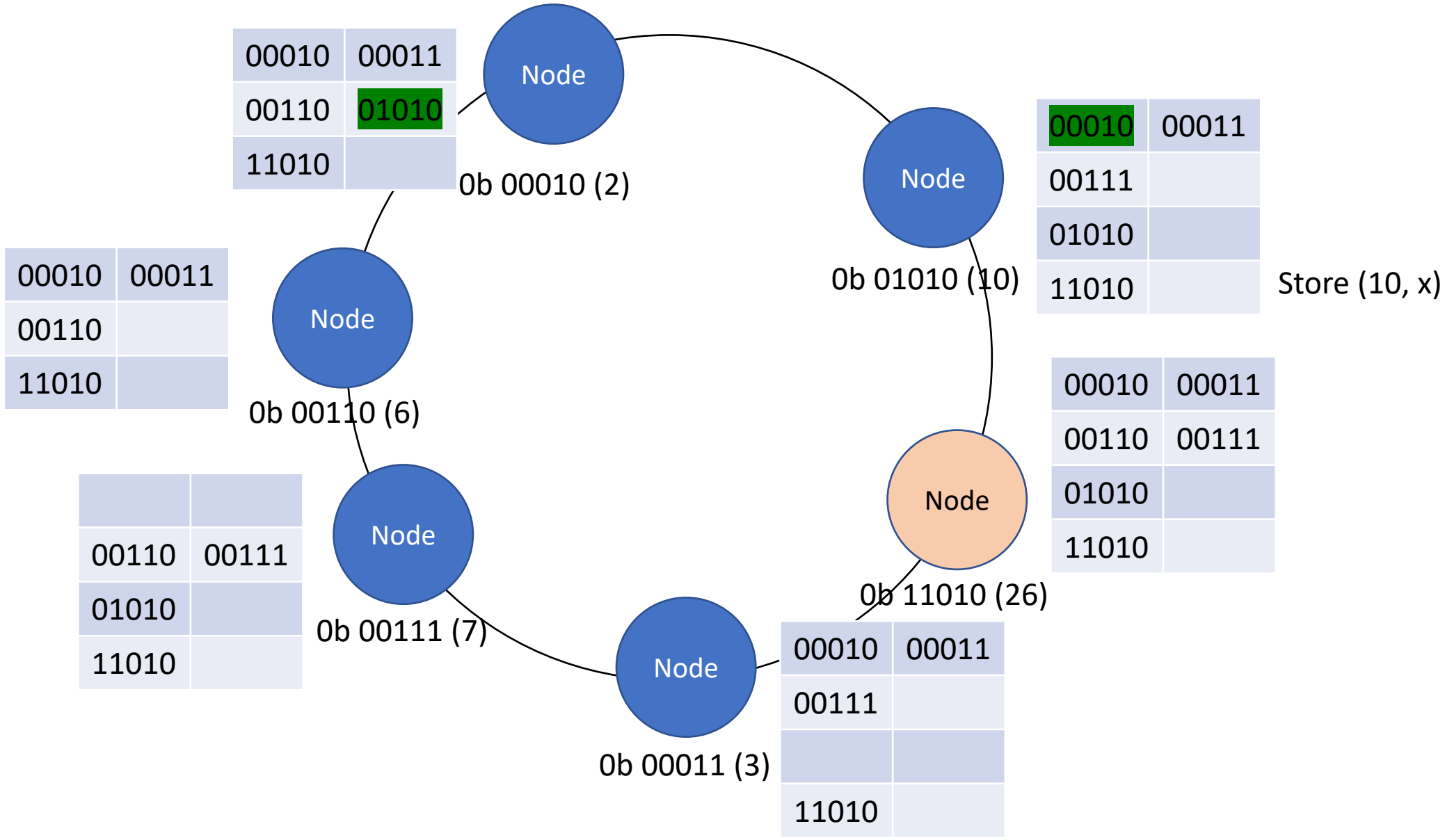
Store / Find (k,a = 2)



Store / Find (k,a = 2)



Store / Find (k,a = 2)



Store / Find (what ifs)

- What if a node fails to respond?
- What if a k-bucket is full?
- What if more than one node have the same distance?
- What if … ?

References

- [0]: Maymounkov, P., & Mazières, D. (2002). Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. In P. Druschel, F. Kaashoek, & A. Rowstron (Eds.), Peer-to-Peer Systems (pp. 53–65). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-45748-8_5

- [1]: Baumgart, I., & Mies, S. (2014). S / Kademlia : A Practicable Approach Towards Secure Key-Based Routing, (June). https://doi.org/10.1109/ICPADS.2007.4447808

- [2]: Freedman, M. J., & Mazières, D. (2003). Sloppy Hashing and Self-Organizing Clusters. In IPTPS. Springer Berlin / Heidelberg. Retrieved from www.coralcdn.org/docs/coral-iptps03.ps

- https://github.com/libp2p/specs/tree/master/kad-dht
- https://codethechange.stanford.edu/guides/guide_kademlia.html
- https://kelseyc18.github.io/kademlia_vis/basics/1/