CSCI-1680
DHT

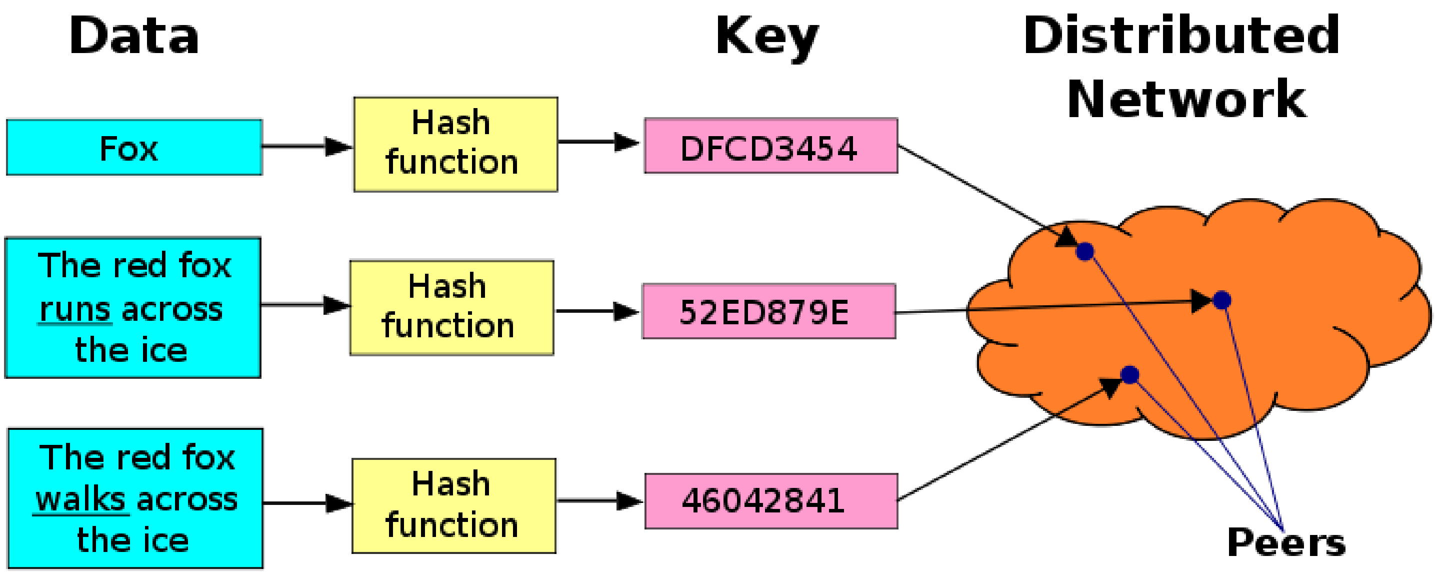
Chen Avin

Based partly on lecture notes by Scott Shenker and John Jannotti

Last time

- DNS
- Today: DHT

DHT - Distributed hash table



P2P

- Database has (key, value) pairs;
  - key: ss number; value: human name
  - key: domain name; value: IP address
- Peers query DB with key
- DB returns values that match the key
- Peers can also insert (key, value) peers

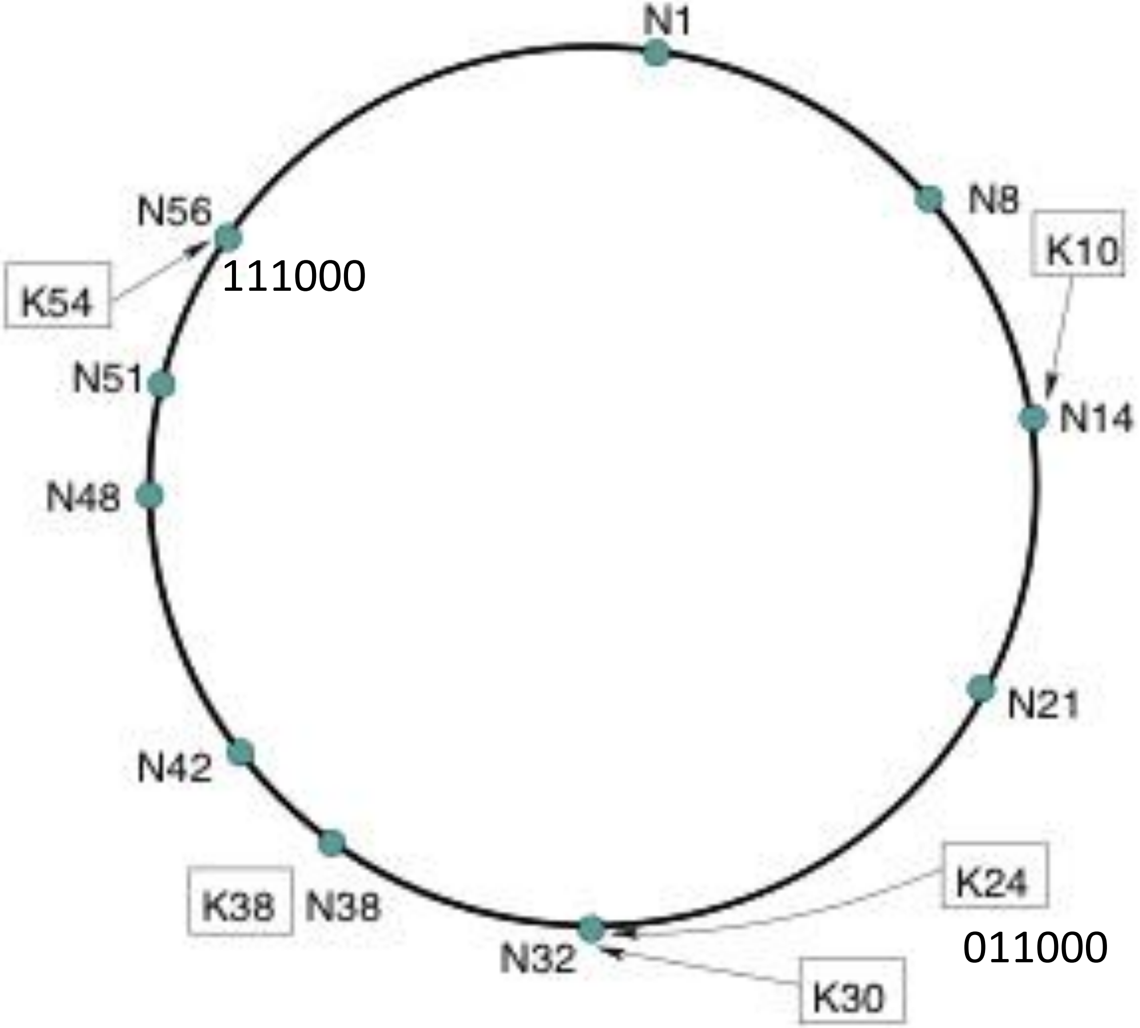
DHTs

- **IDs from a *flat* namespace**
  - Contrast with hierarchical IP, DNS
- **Metaphor: hash table, but distributed**
- **Interface**
  - Get(key)
  - Put(key, value)
- **How?**
  - Every node supports a single operation:

    Given a *key*, route messages to node holding *key*

Example: The Chord Ring

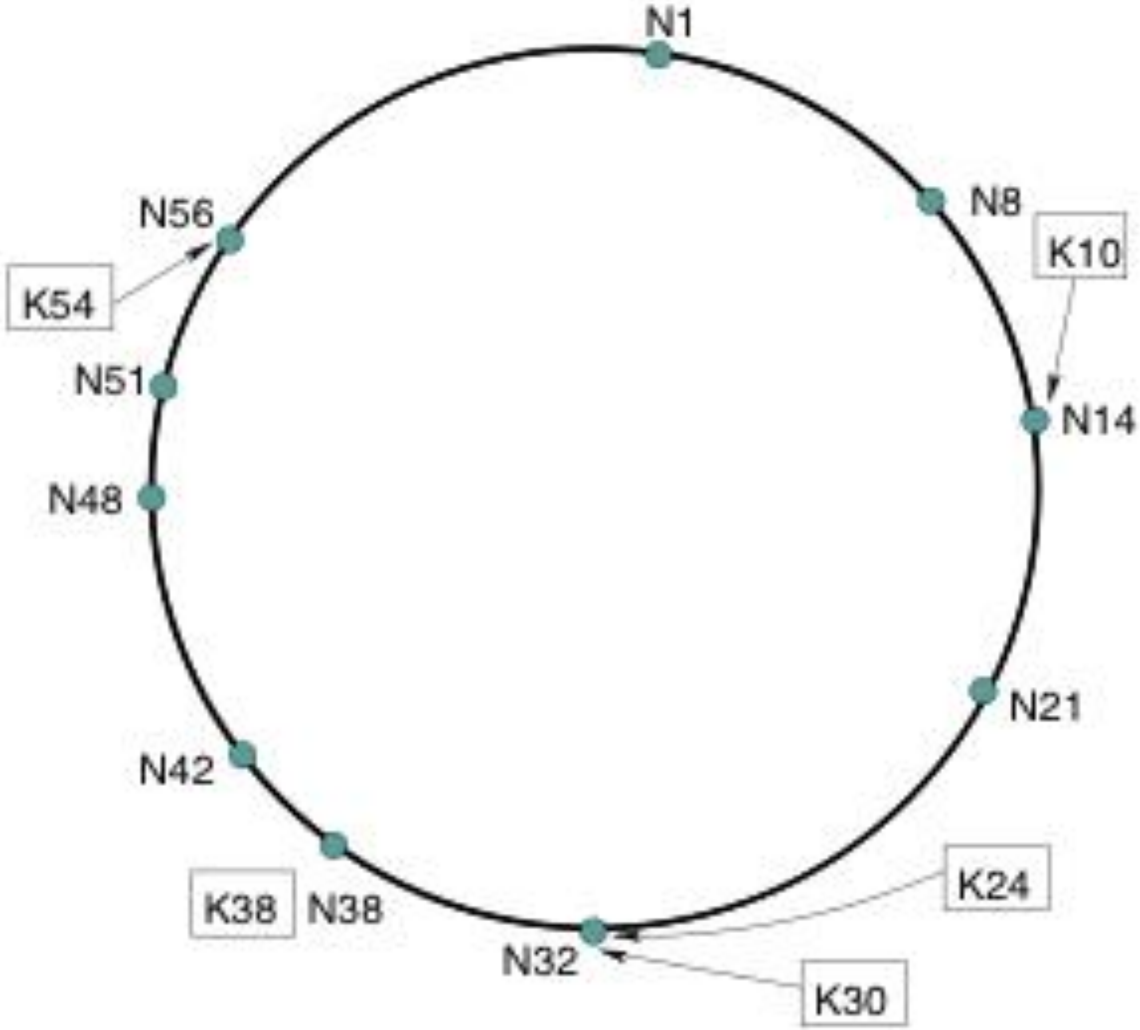
- m bits identifiers
- identifier circle
- Node's identifier by hashing IP
- Key identifier by hashing the key
- successor(k)
- N - # of nodes
- K - # of keys
- Overlay Network



Identifier to Node Mapping Example

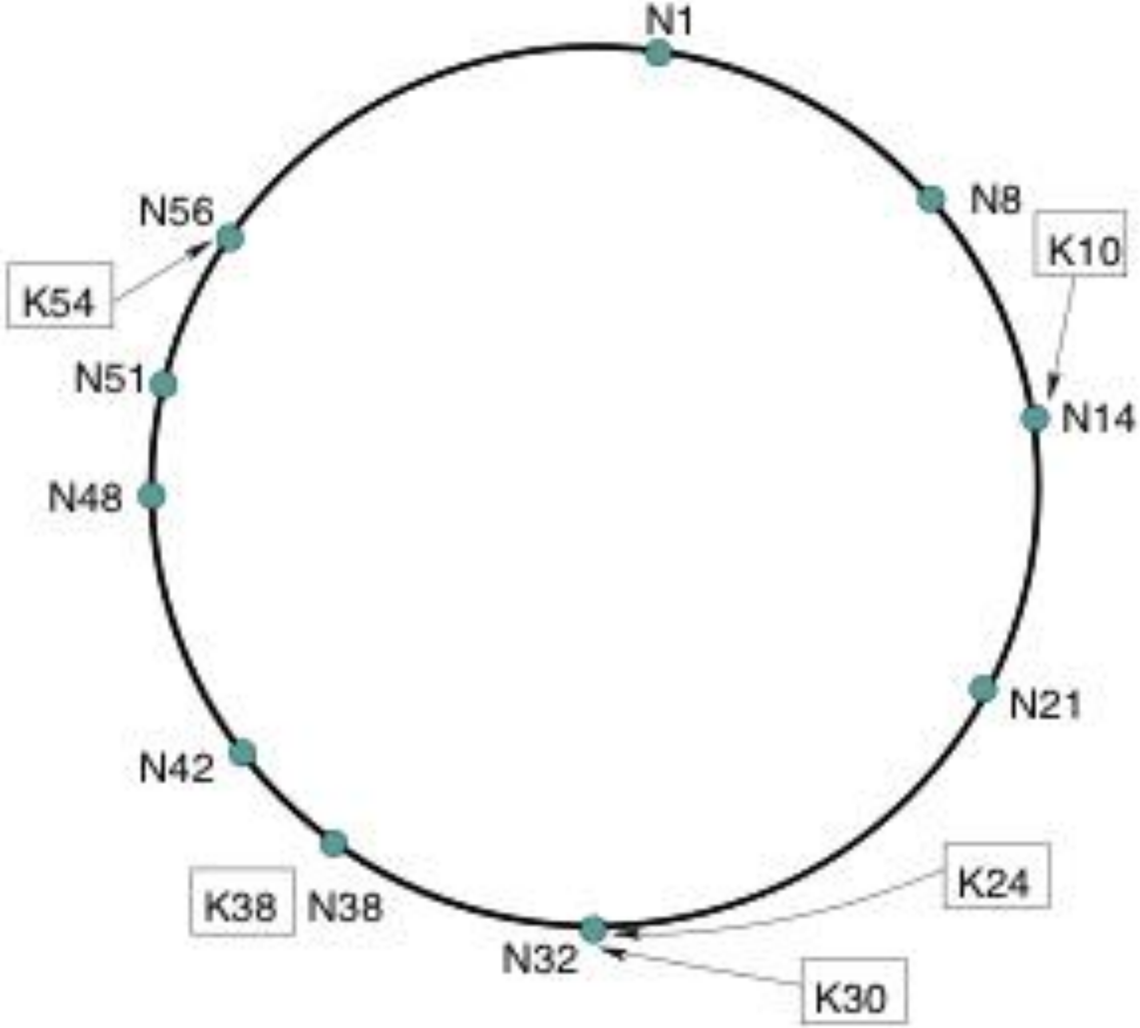
- **Node 8 maps [2,8]**
- **Node 14 maps [9,14]**
- **Node 21 maps [15, 21]**
- **…**
- **Node 1 maps [57, 0]**

- **Each node maintains a pointer to its successor**



Consistent Hashing

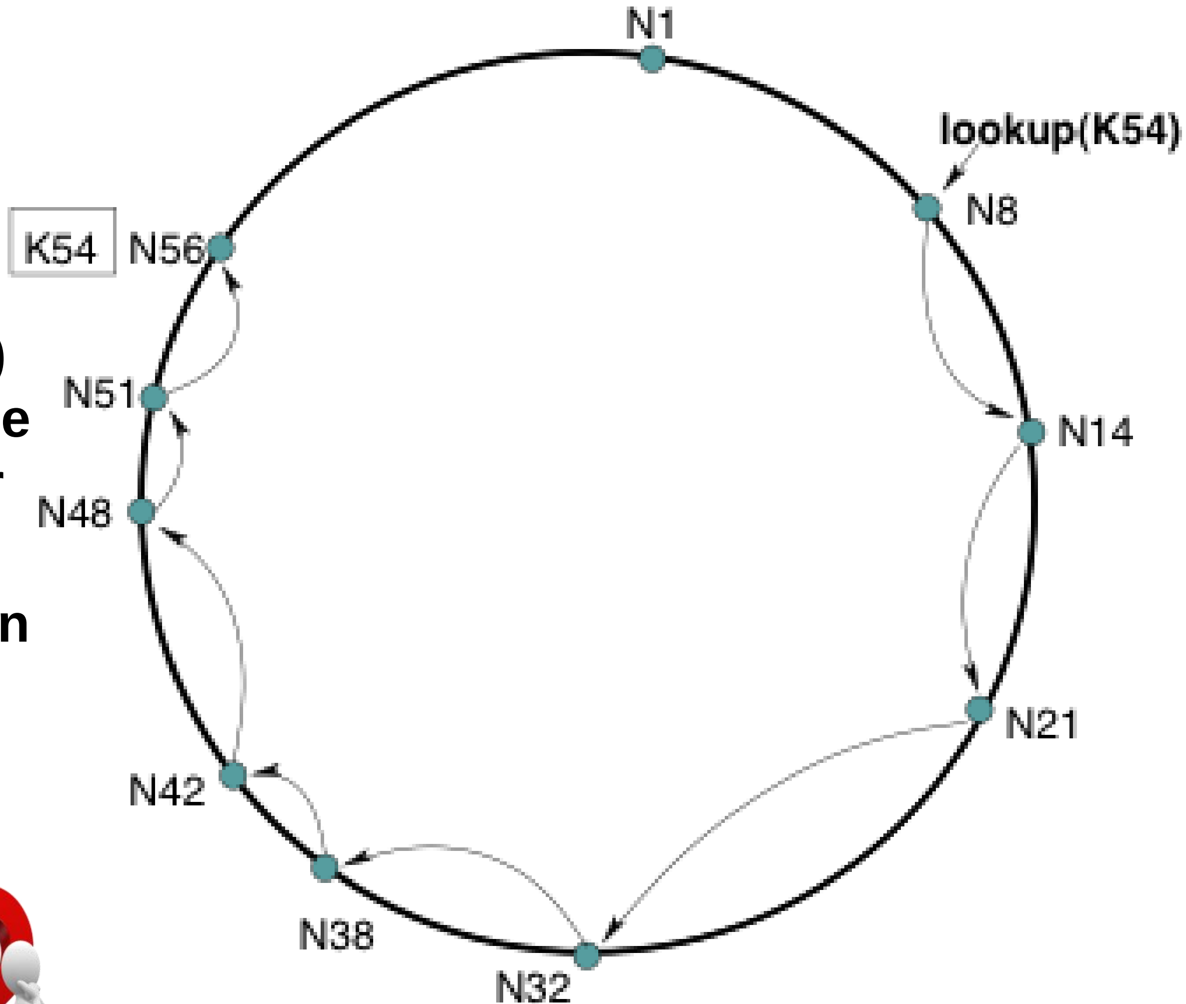
- N nodes
- K keys
- Each node responsible for about K/N keys
- When nodes join/leave about K/N keys change location



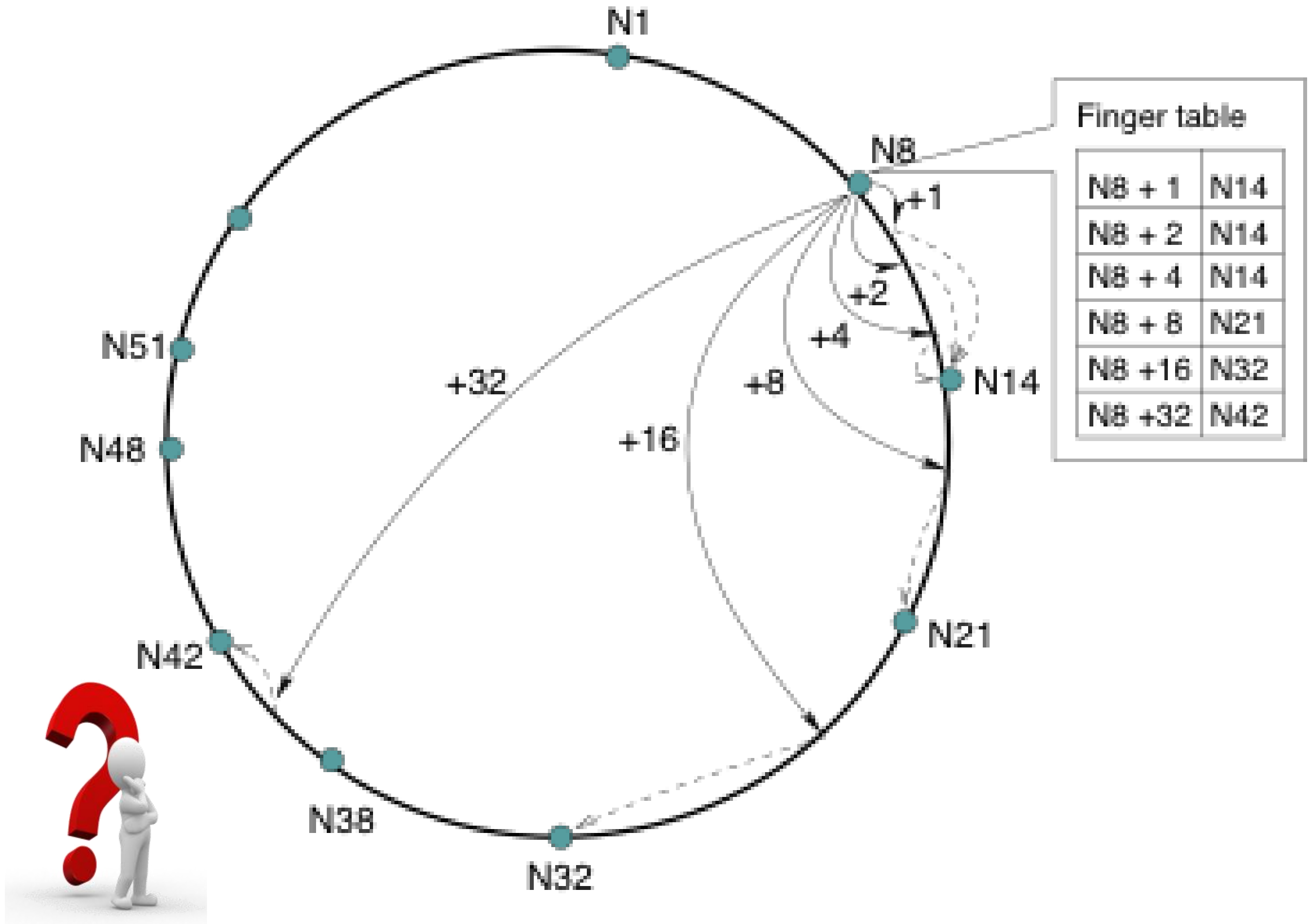
Simple Lookup

- Each node maintains its successor

- Route packet (ID, data) to the node responsible for ID using successor pointers
- How long the search on average?
- O(N) operation per search when there are N peers



Scalable Key Location

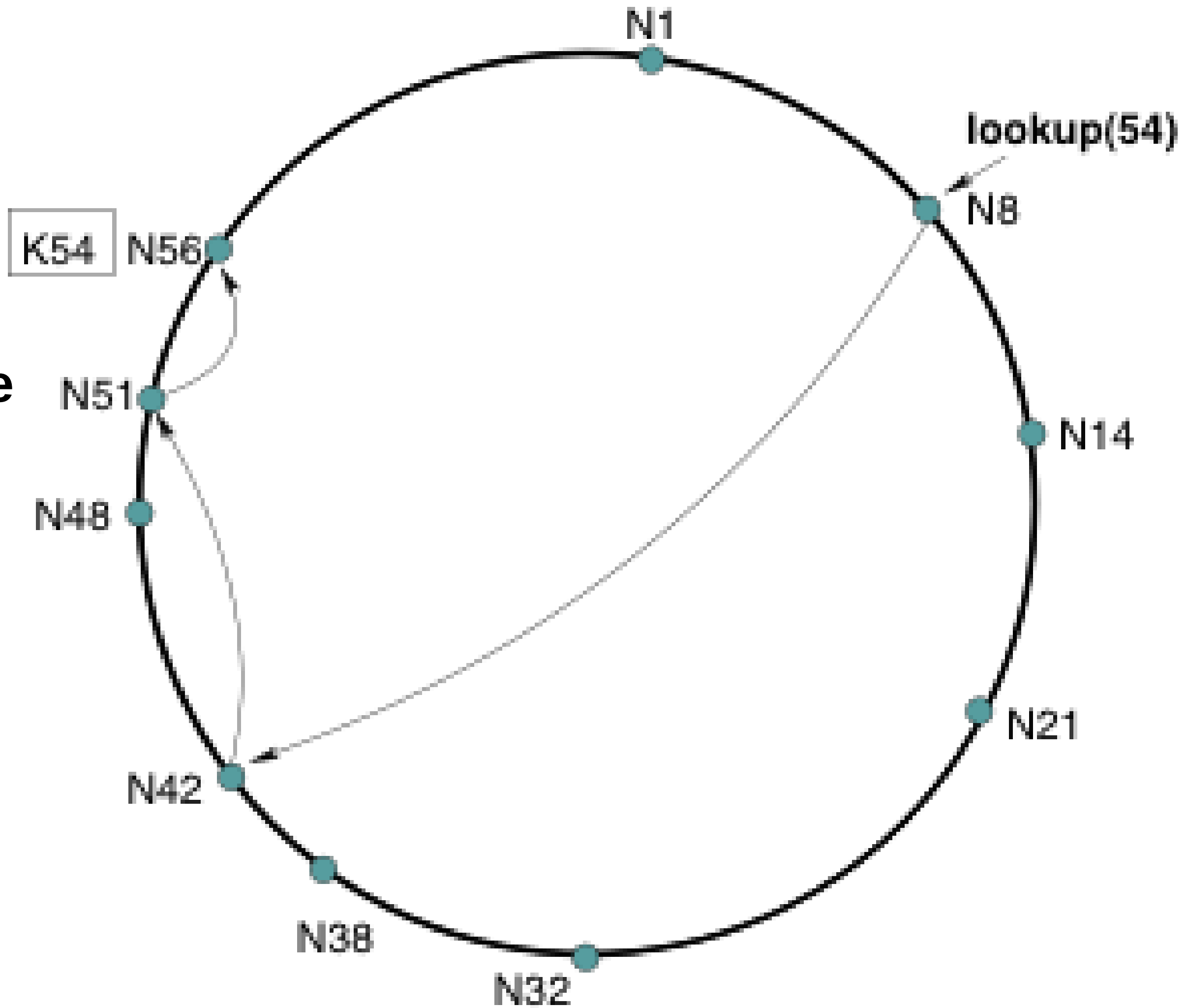


- **Adding** **short cuts**
- **Finger Table**
- **How many entries?**
- **How long the search?**

*i*th entry at peer with id *n* is first peer with id >= $n + 2^i \pmod{2^m}$

Key Lookup

- O(log N) neighbors
- Cutting the Distance by half
- O(log N) operations per search
- Namespace 2^m



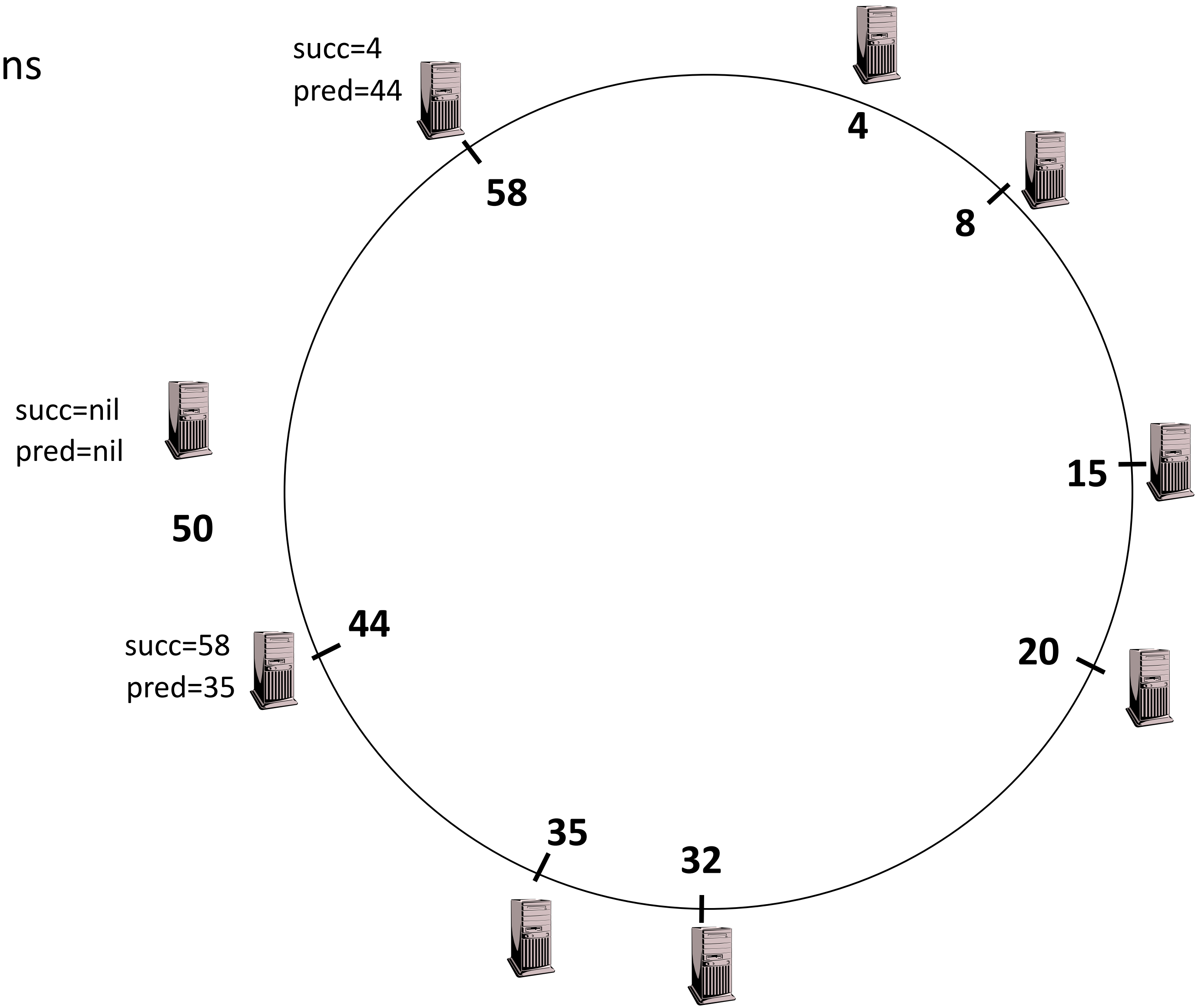
Stabilization Procedure

- **Periodic operations performed by each node N to maintain the ring:**

```
STABILIZE() [N.successor = M]
   N->M: "What is your predecessor?"
   M->N: "x is my predecessor"
   if x between (N,M), N.successor = x
   N->N.successor: NOTIFY()
NOTIFY()
    N->N.successor: "I think you are my successor"
M: upon receiving NOTIFY from N:
    If (N between (M.predecessor, M))
         M.predecessor = N
```

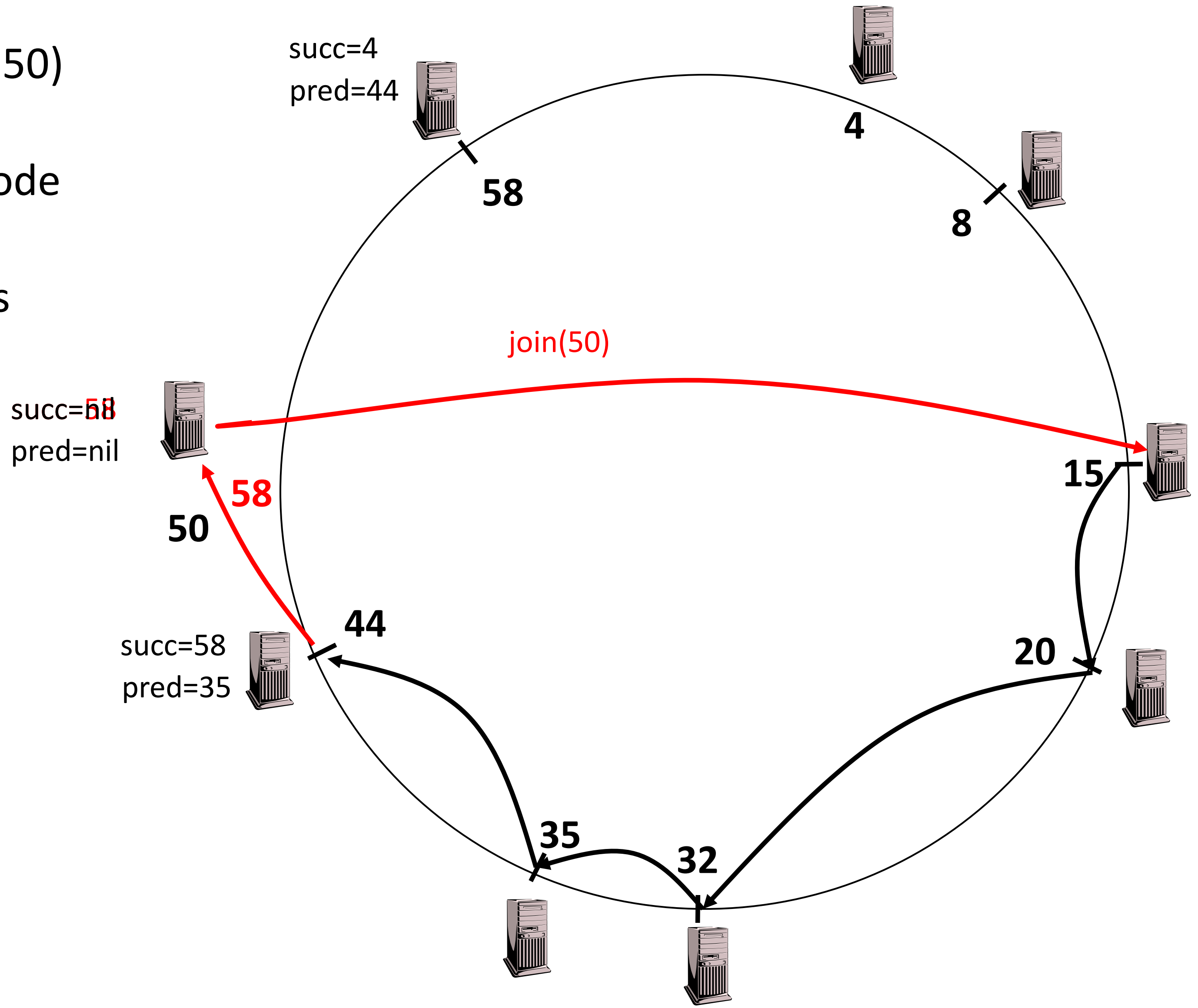
Joining Operation

- Node with id=50 joins the ring
- Node 50 needs to know at least one node already in the system
  - Assume known node is 15



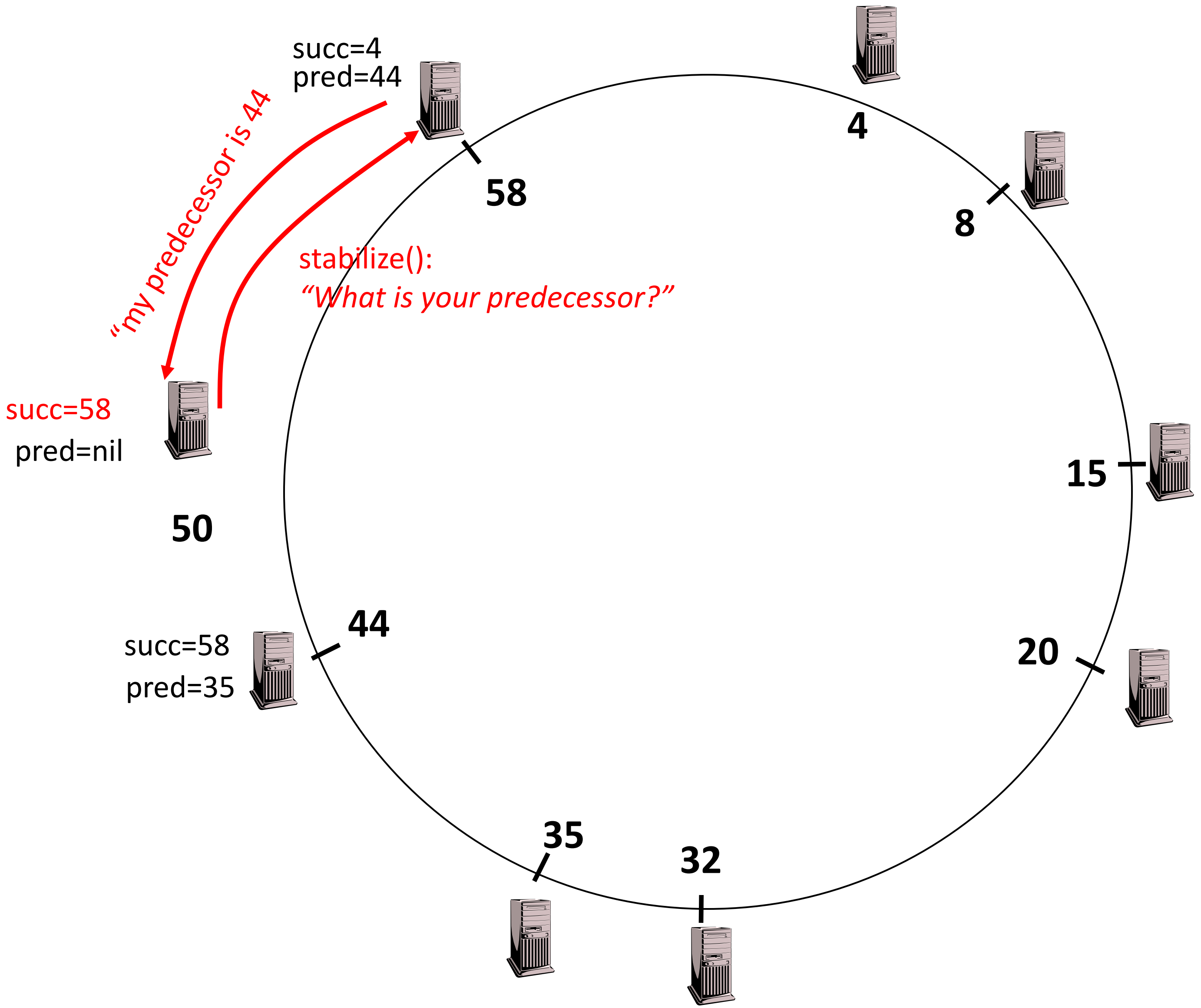
Joining Operation

- Node 50: send join(50) to node 15
- Node 44: returns node 58
- Node 50 updates its successor to 58



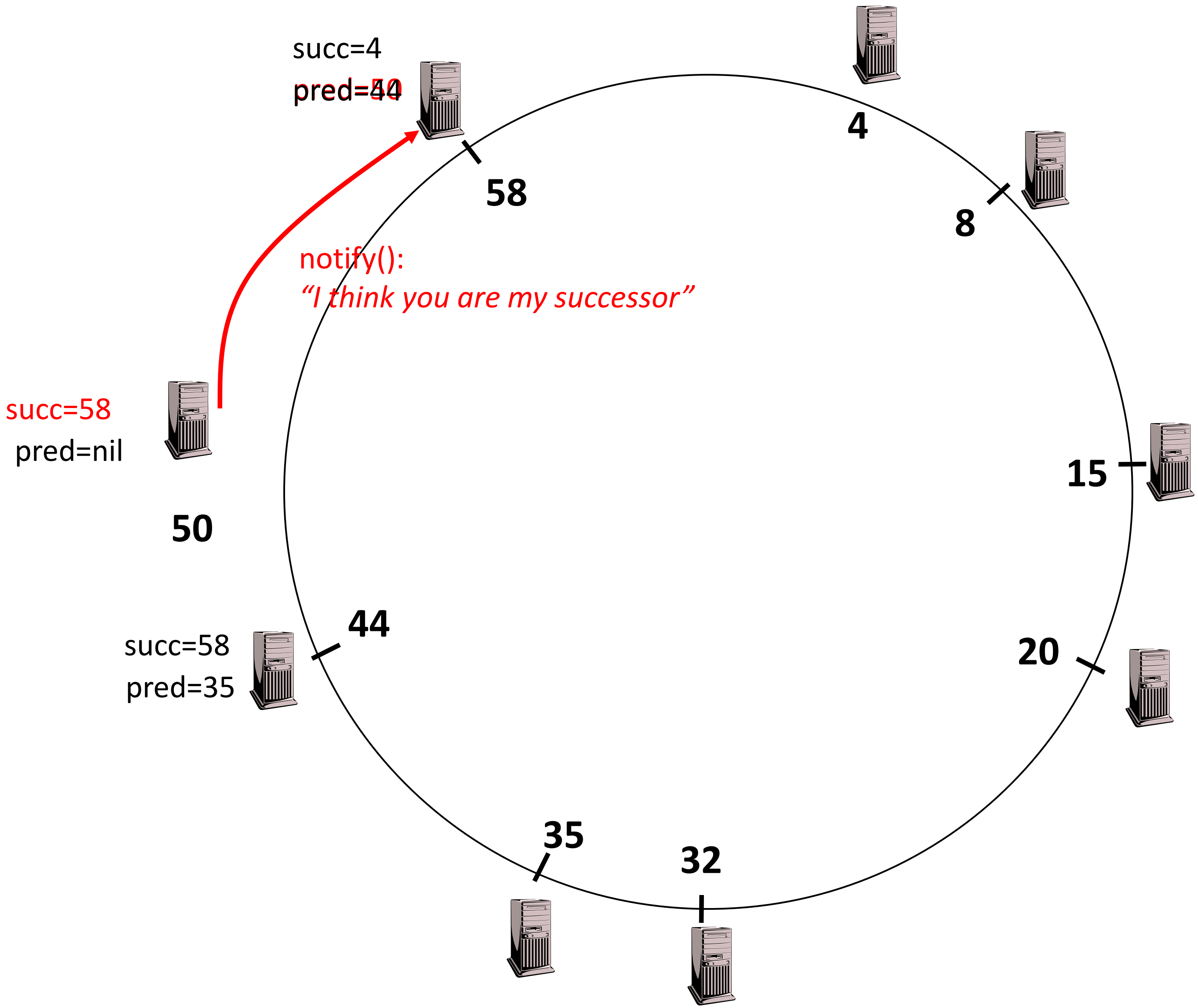
Joining Operation

- Node 50: send stabilize() to node 58
- Node 58:
  - Replies with 44
  - 50 determines it is the right predecessor



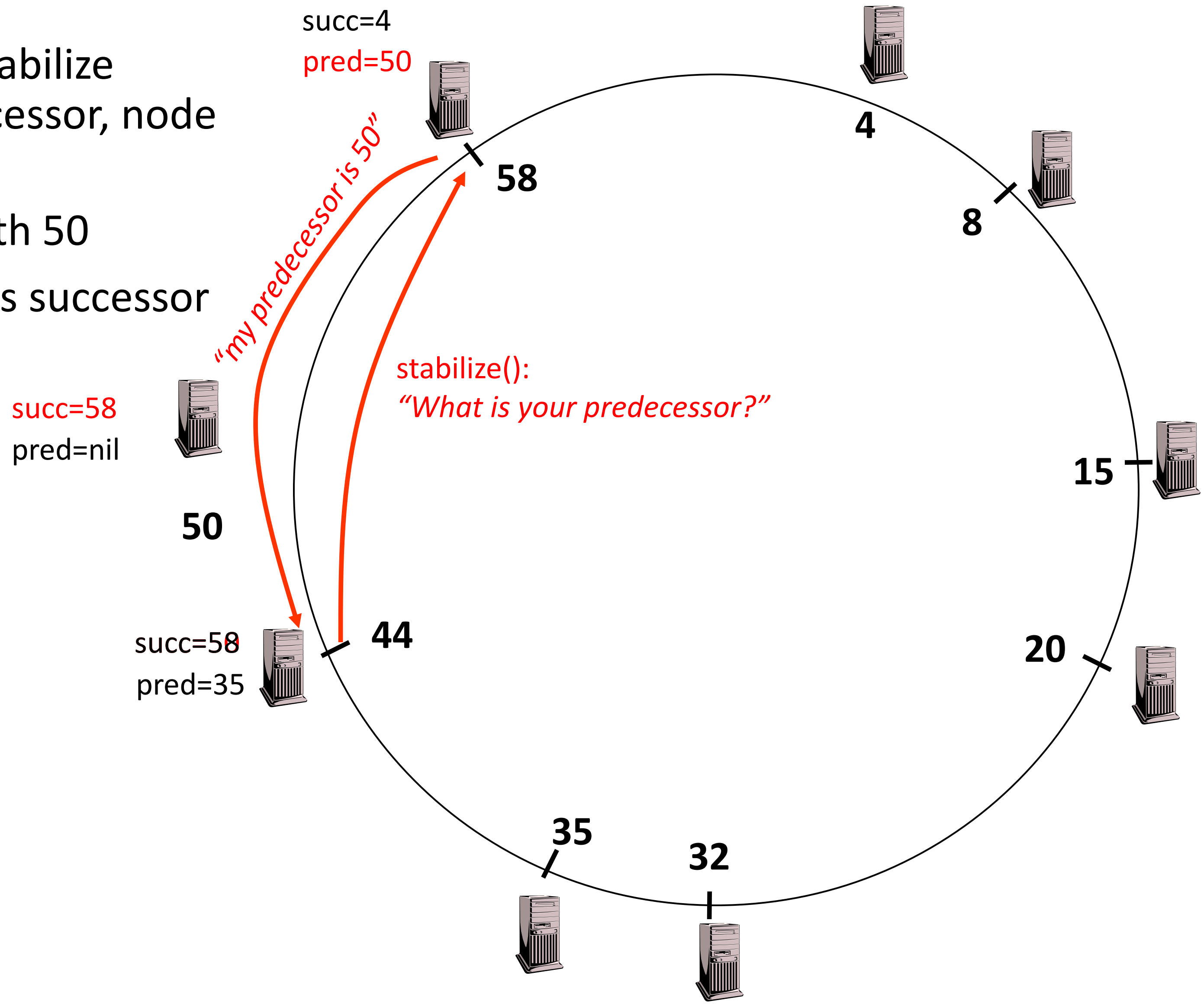
Joining Operation

- Node 50: send notify() to node 58
- Node 58:
  - update predecessor to 50



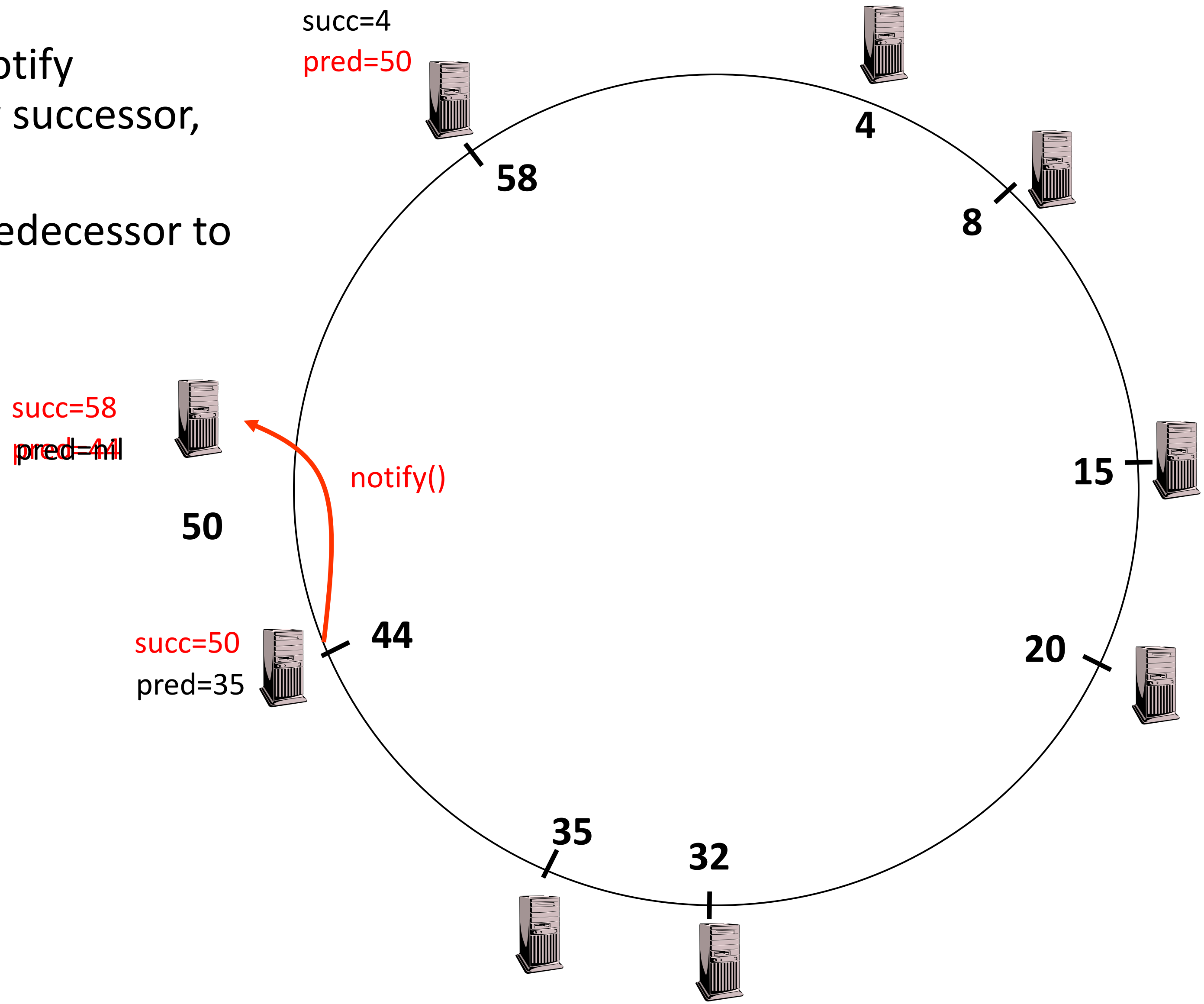
Joining Operation

- Node 44 sends a stabilize message to its successor, node 58
- Node 58 replies with 50
- Node 44 updates its successor to 50



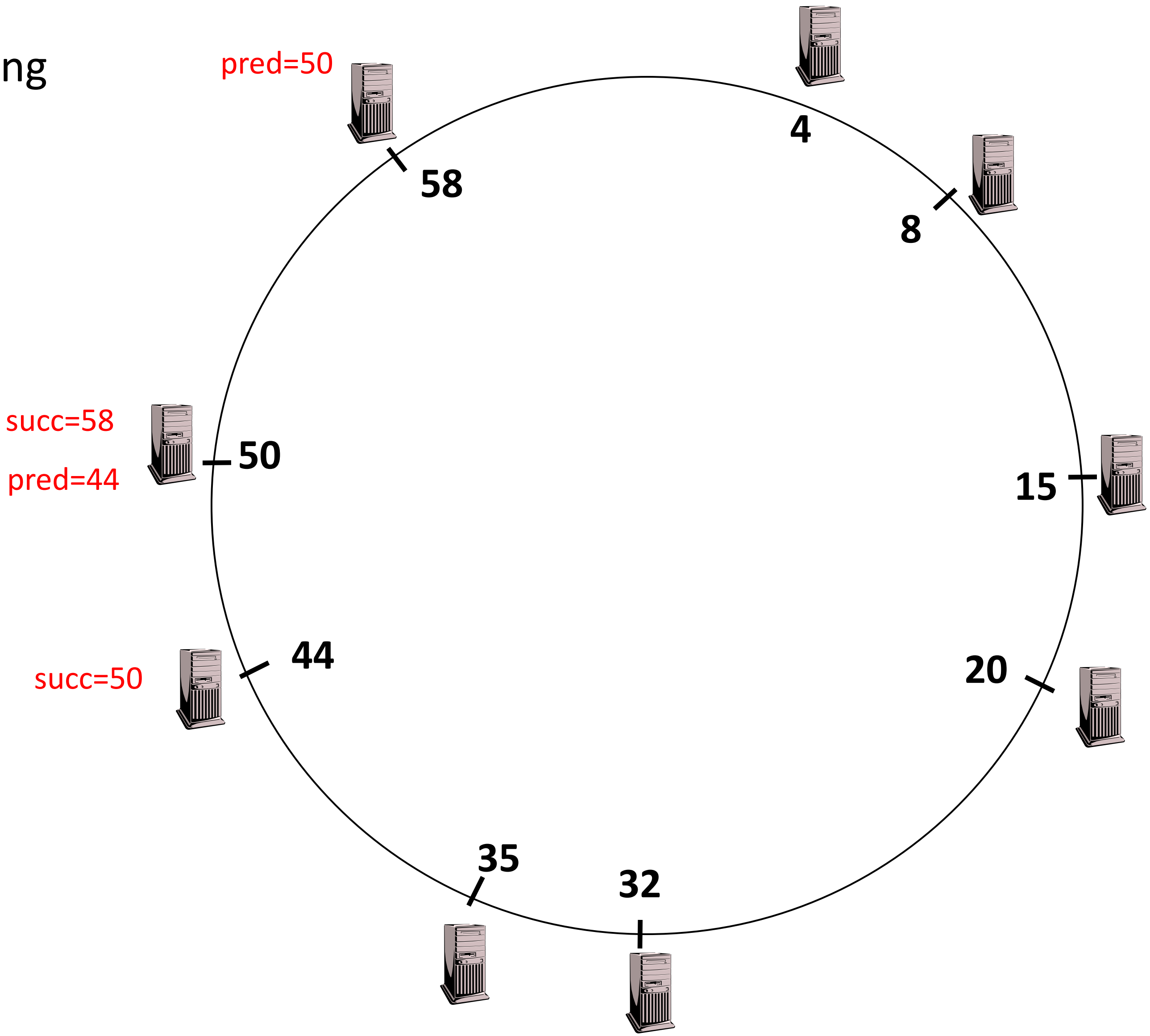
Joining Operation

- Node 44 sends a notify message to its new successor, node 50
- Node 50 sets its predecessor to node 44



Joining Operation (cont’d)

- This completes the joining operation!



Chord

- **There is a tradeoff between routing table size and diameter of the network**
- **Chord achieves diameter O(log n) with O(log n)-entry routing tables**

Many other DHTs

- **CAN**
  - Routing in n-dimensional space
- **Pastry/Tapestry/Bamboo**
  - (Book describes Pastry)
  - Names are fixed bit strings
  - Topology: hypercube (plus a ring for fallback)
- **Kademlia**
  - Similar to Pastry/Tapestry
  - But the ring is ordered by the XOR metric
  - Used by BitTorrent for distributed tracker
- **Viceroy**
  - Emulated butterfly network
- **Koorde**
  - DeBruijn Graph
  - Each node connects to 2n, 2n+1
  - Degree 2, diameter log(n)
- **…**

Discussion

- **Query can be implemented**
  - Iteratively: easier to debug
  - Recursively: easier to maintain timeout values
- **Robustness**
  - Nodes can maintain (k>1) successors
  - Change notify() messages to take that into account
- **Performance**
  - Routing in overlay can be worse than in the underlay
  - Solution: flexibility in neighbor selection
    - Tapestry handles this implicitly (multiple possible next hops)
    - Chord can select any peer between $[2^n,2^{n+1})$ for finger, choose the closest in latency to route through

Where are they now?

- **DHTs allow coordination of MANY nodes**
  - Efficient *flat* namespace for routing and lookup
  - Robust, scalable, fault-tolerant
- **If you can do that**
  - You can also coordinate co-located peers
  - Now dominant design style in datacenters
    - E.g., Amazon's Dynamo storage system
  - DHT-style systems everywhere
- **Similar to Google's philosophy**
  - Design with failure as the common case
  - Recover from failure only at the highest layer
  - Use low cost components
  - Scale out, not up