MASSACHUSETTS
INSTITUTE OF
TECHNOLOGY

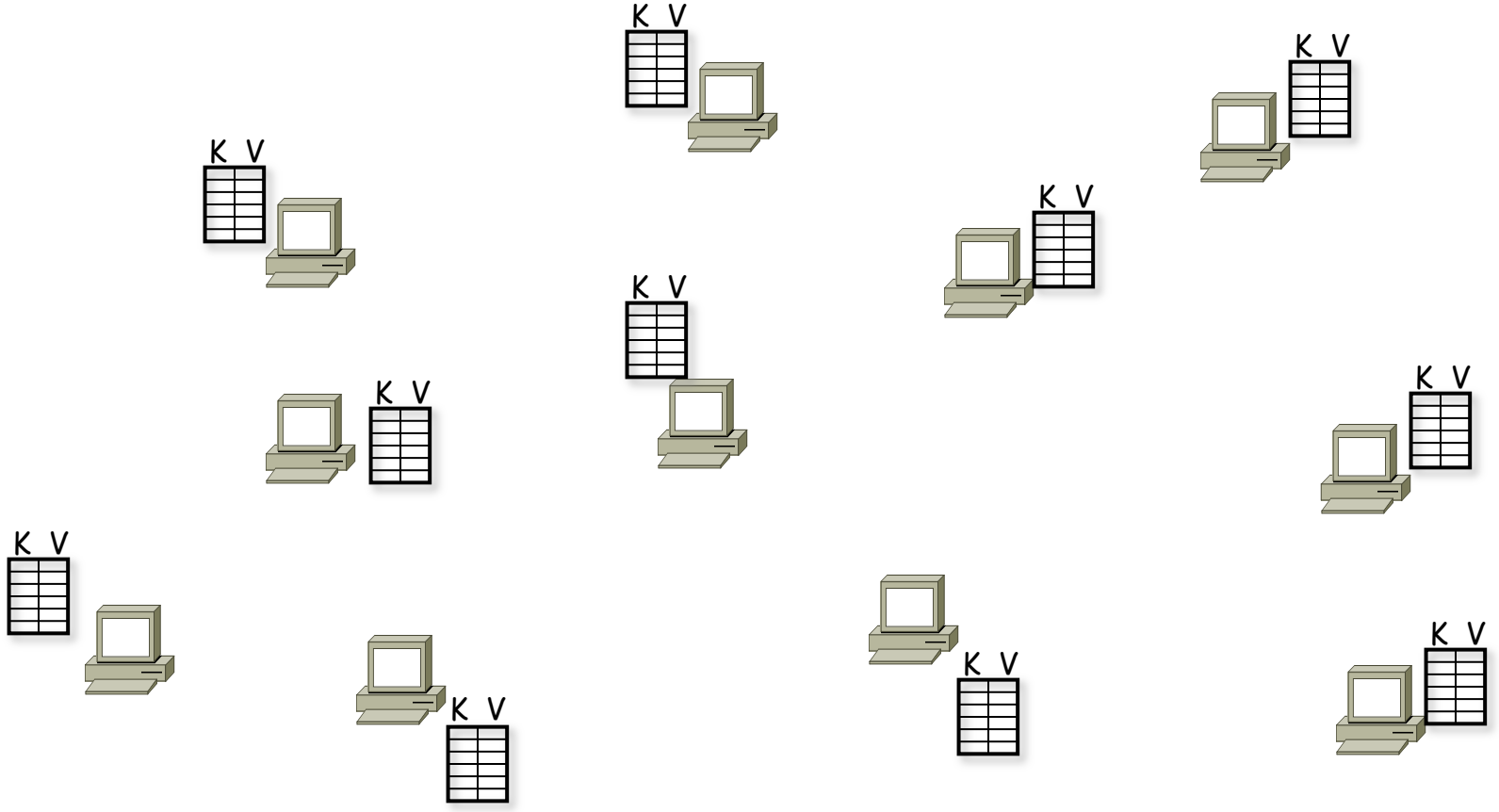
Peer-to-peer systems

6.033 Lecture 13

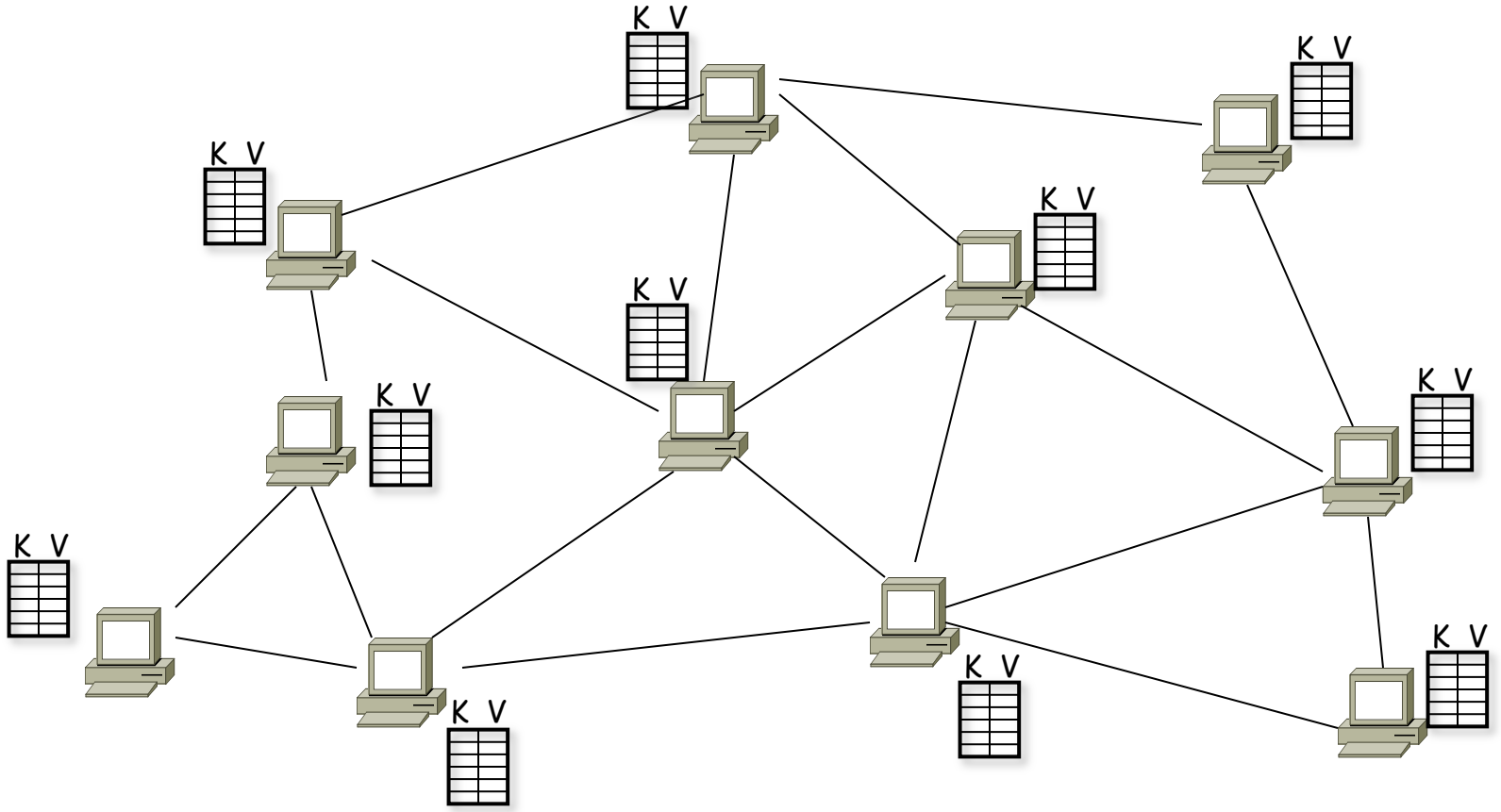
Frans Kaashoek

DP1: deadline Thursday 5p

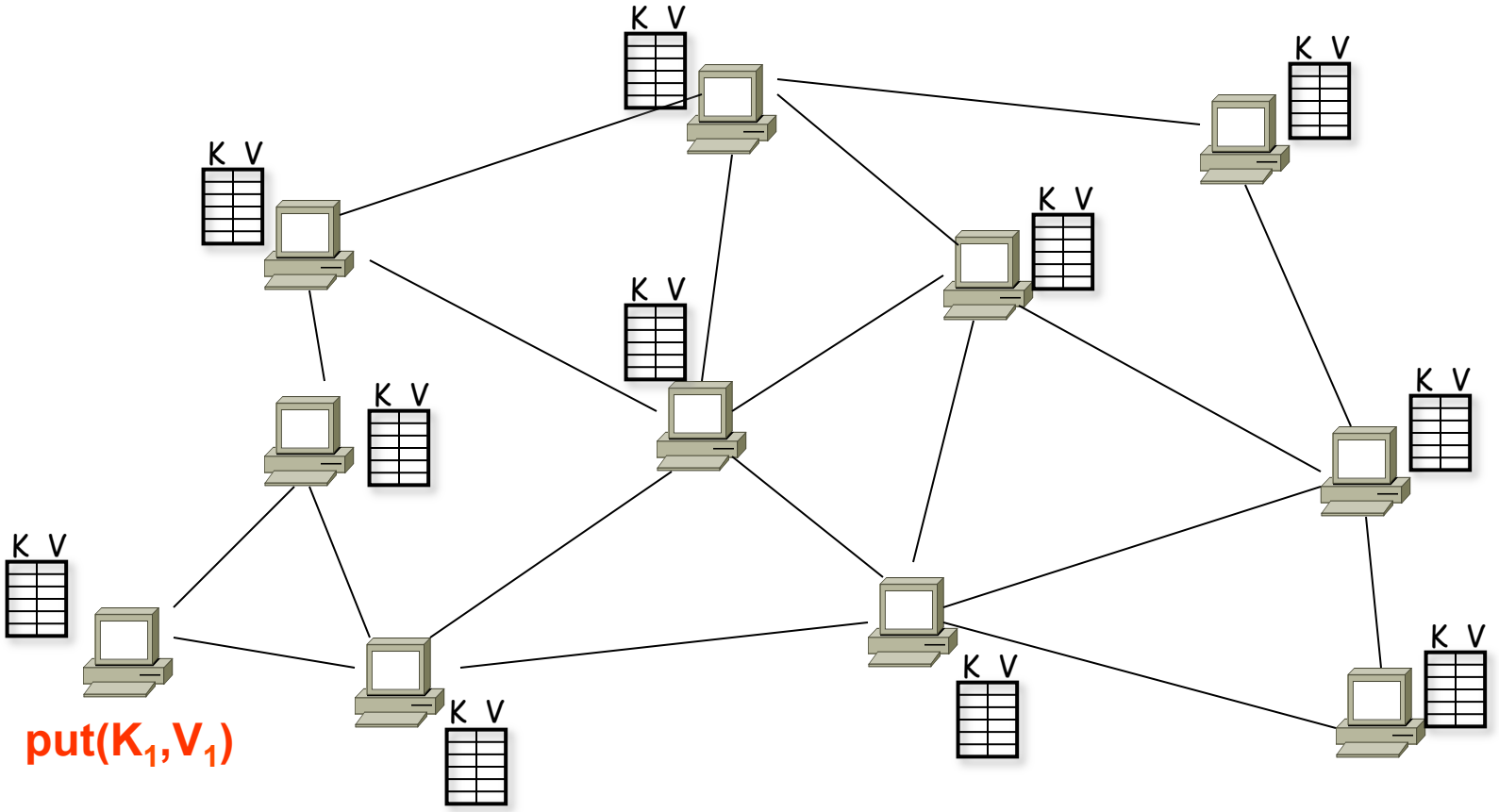
A DHT in Operation: Peers



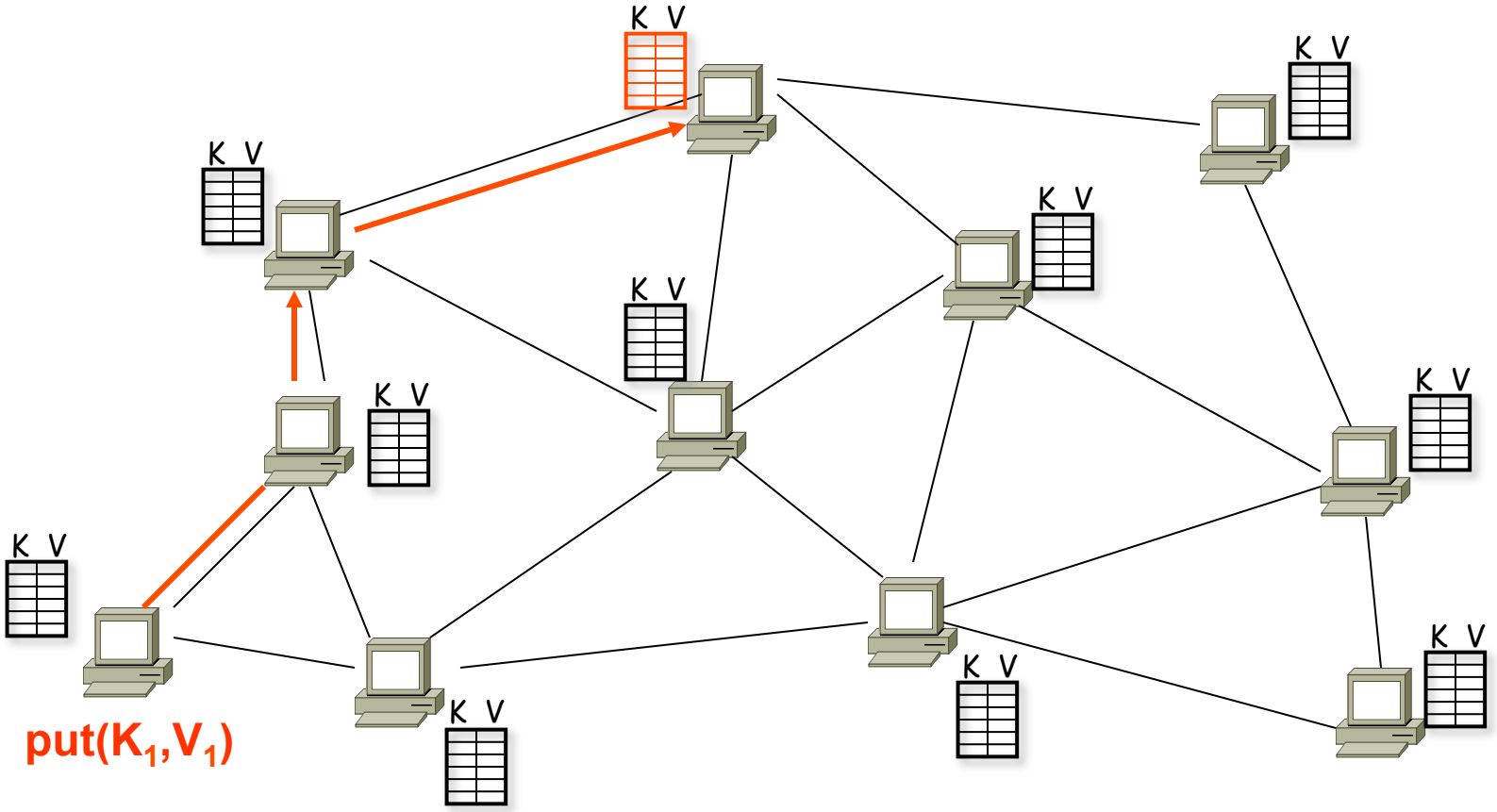
A DHT in Operation: Overlay



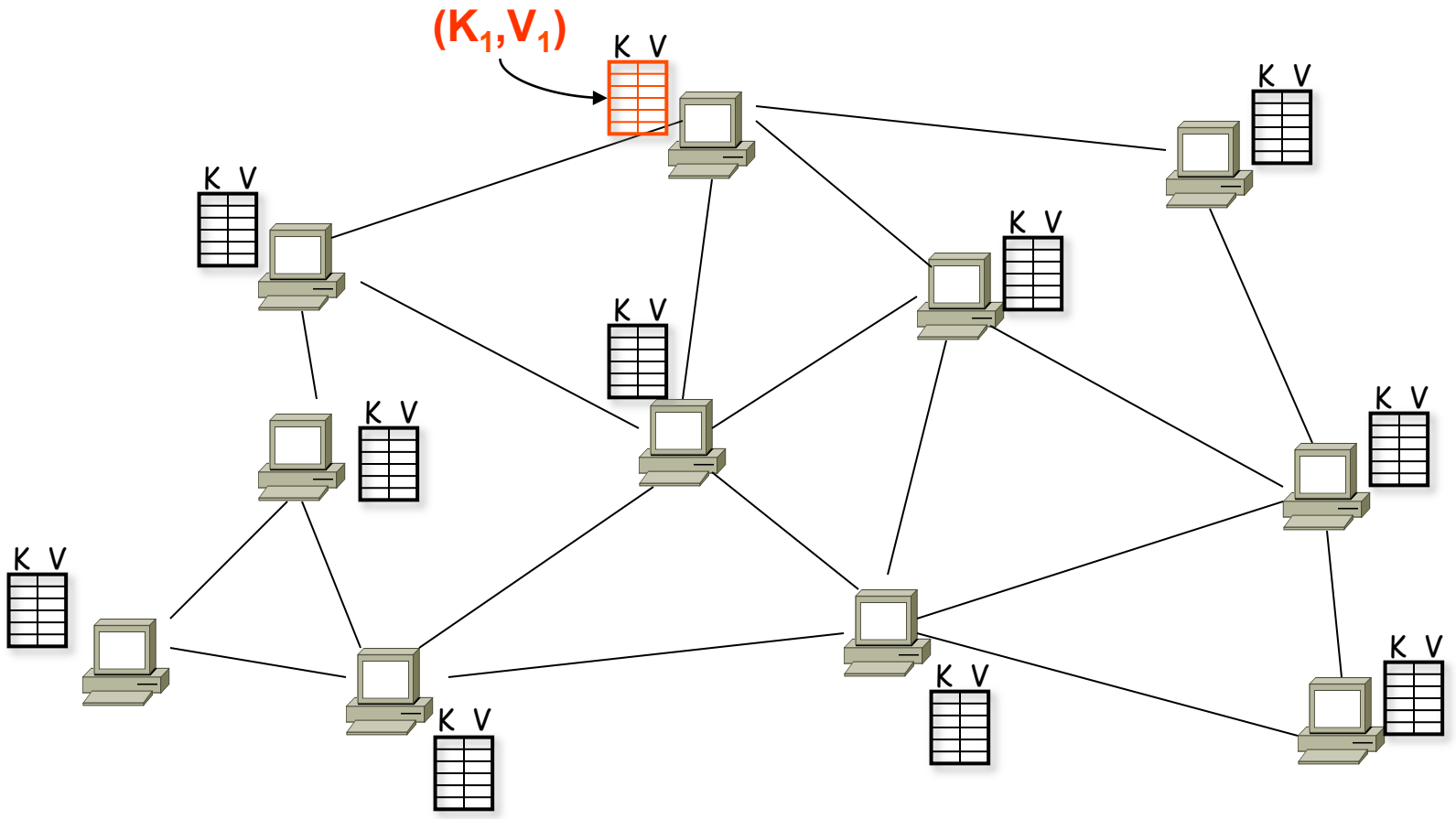
A DHT in Operation: put()



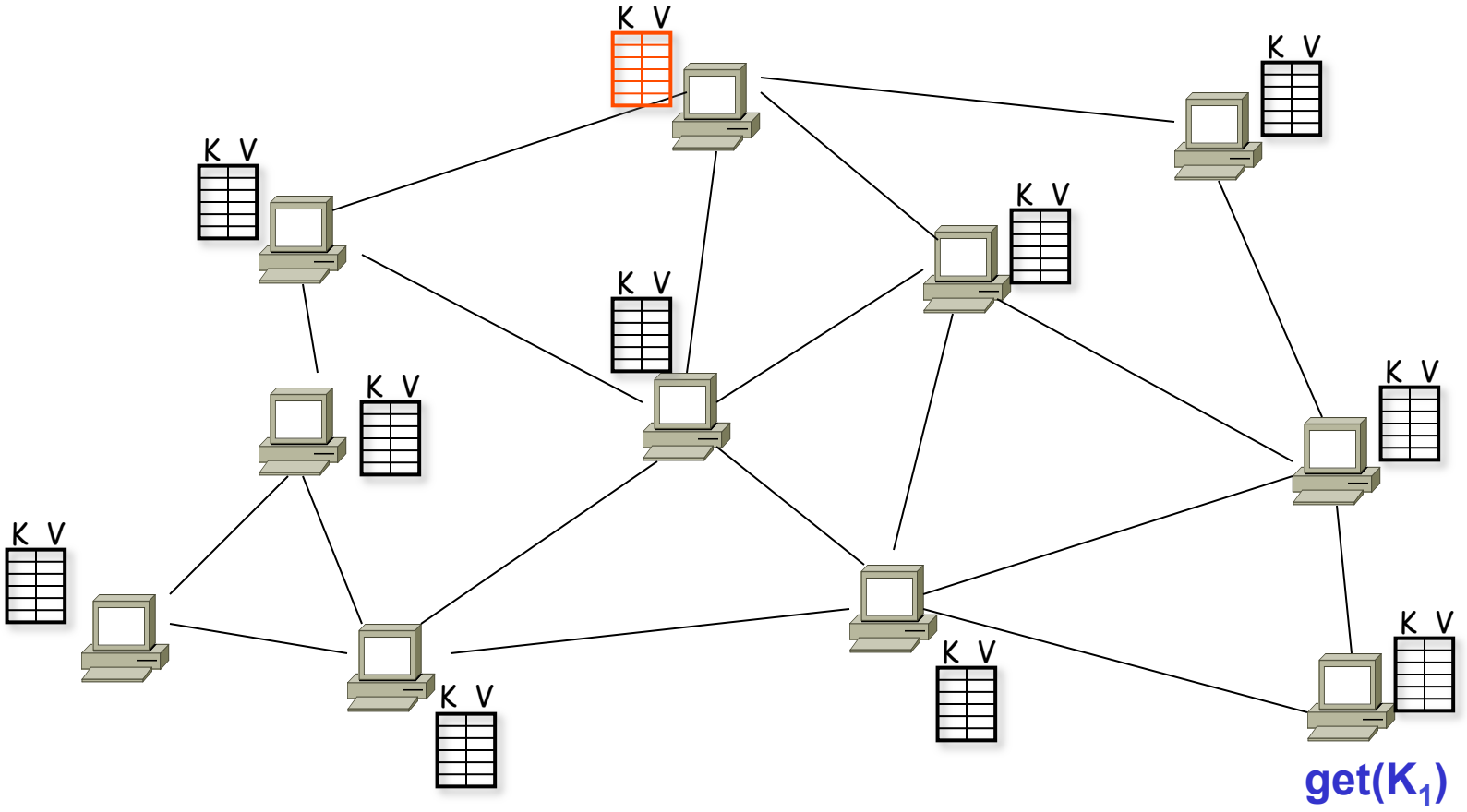
A DHT in Operation: put()



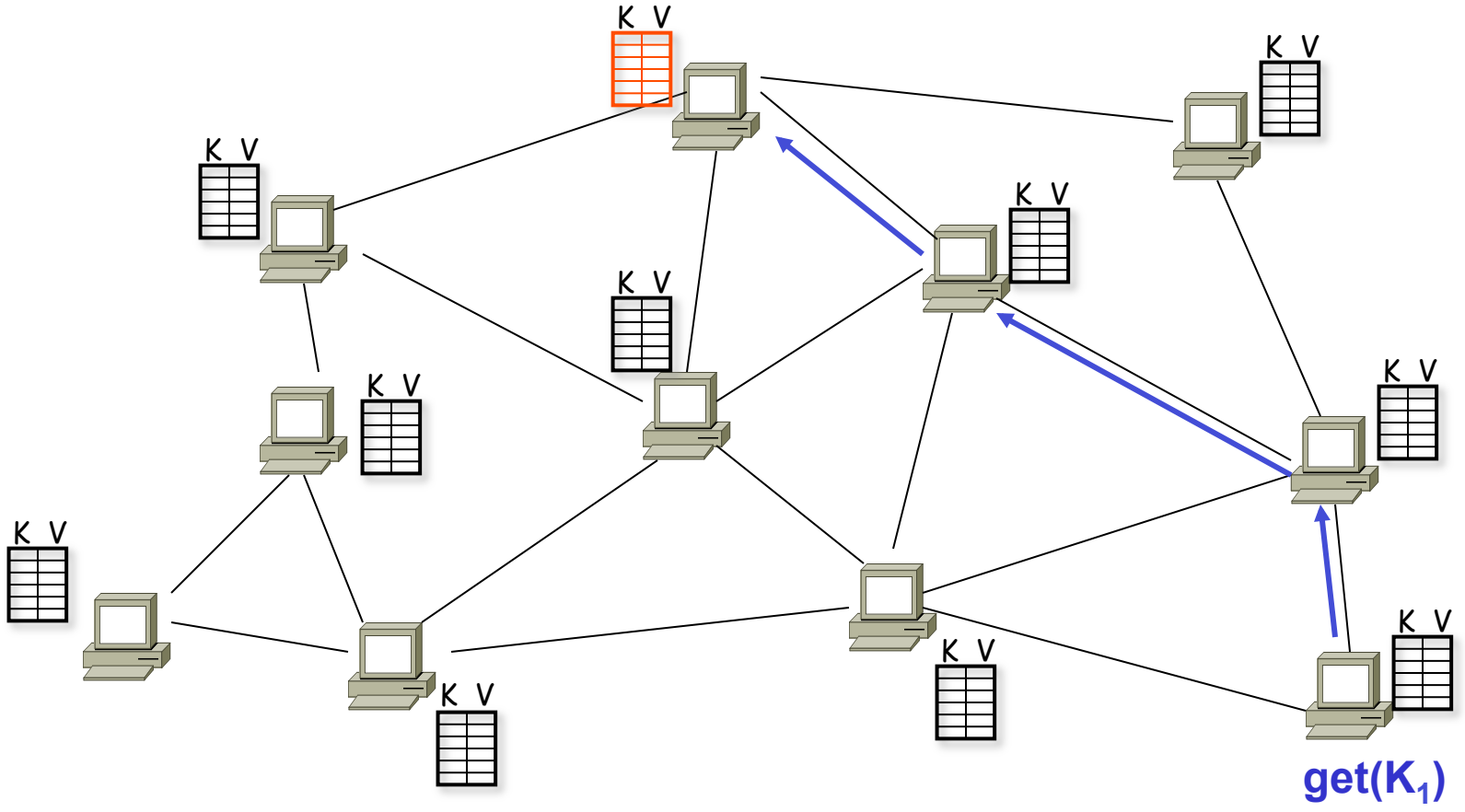
A DHT in Operation: put()



A DHT in Operation: get()



A DHT in Operation: get()



Challenge: nodes join and leave

Chord properties

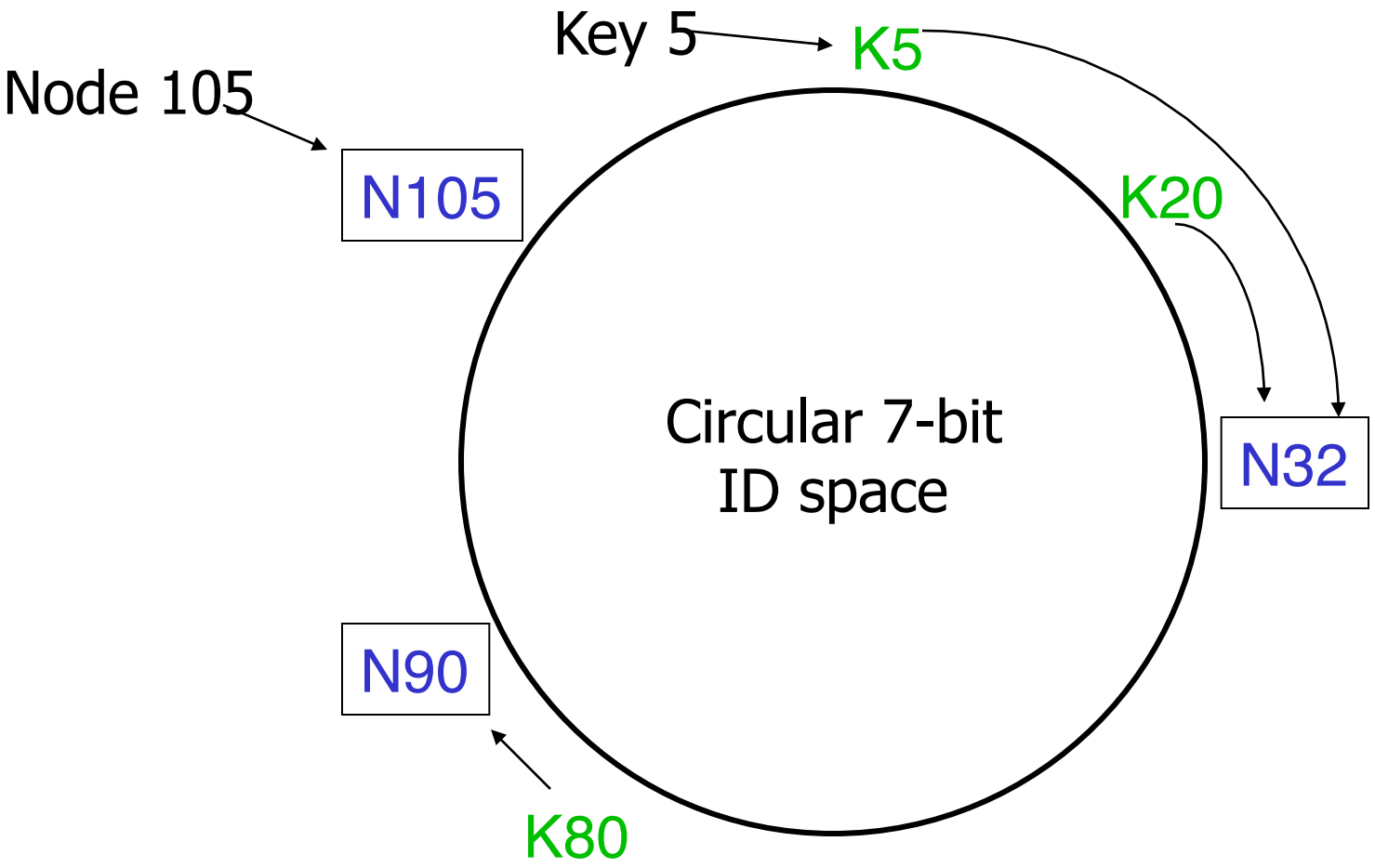
- Efficient: O($log(N)$) messages per lookup
  - N is the total number of servers
- Scalable: O($log(N)$) state per node
- Robust: survives massive failures

Chord IDs

- Key identifier = SHA-1(key)
- Node identifier = SHA-1(IP address)
- Both are uniformly distributed
- Both exist in the same ID space

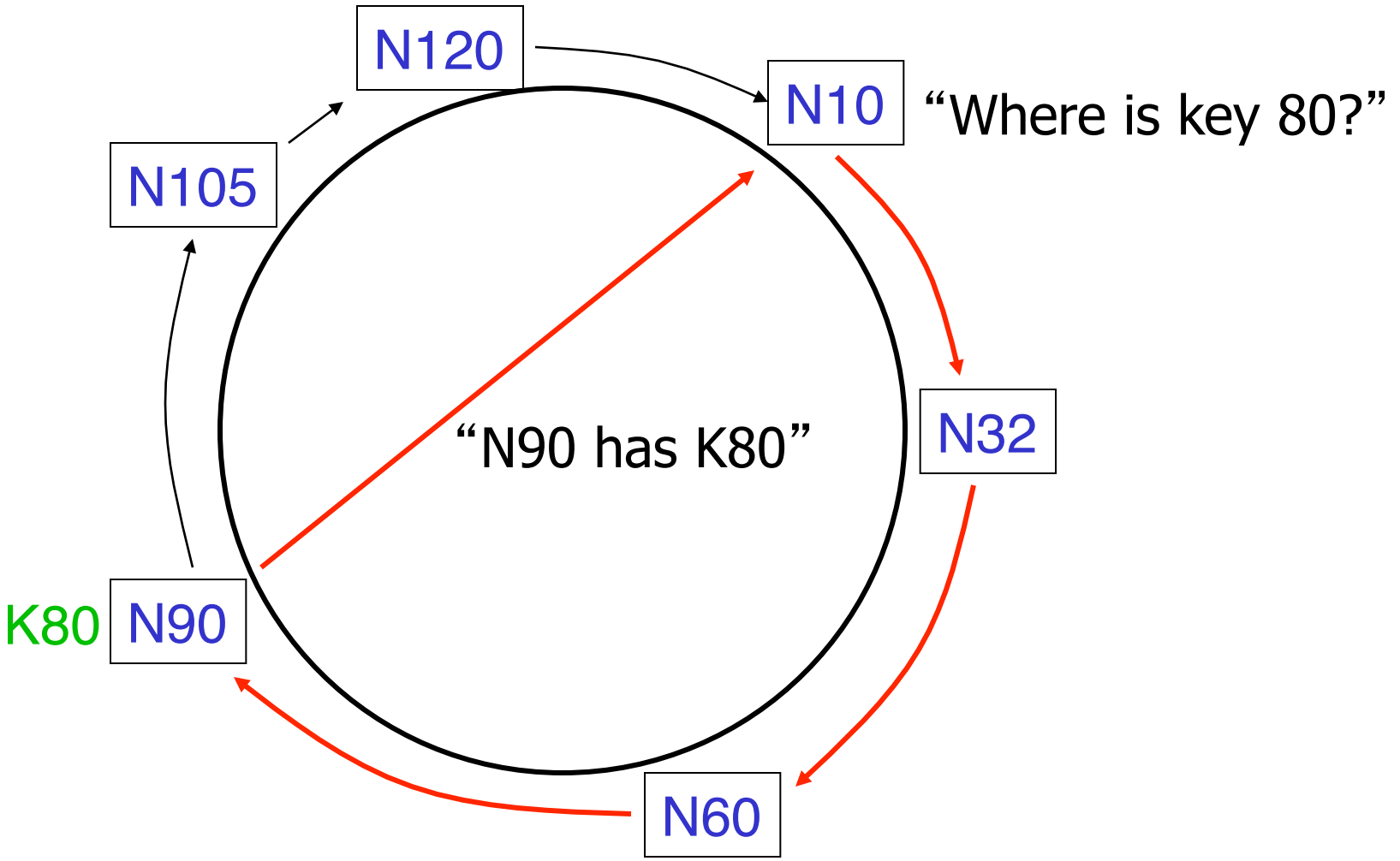
- How to map key IDs to node IDs?

Consistent hashing



A key is stored at its successor: node with next higher ID

Basic lookup

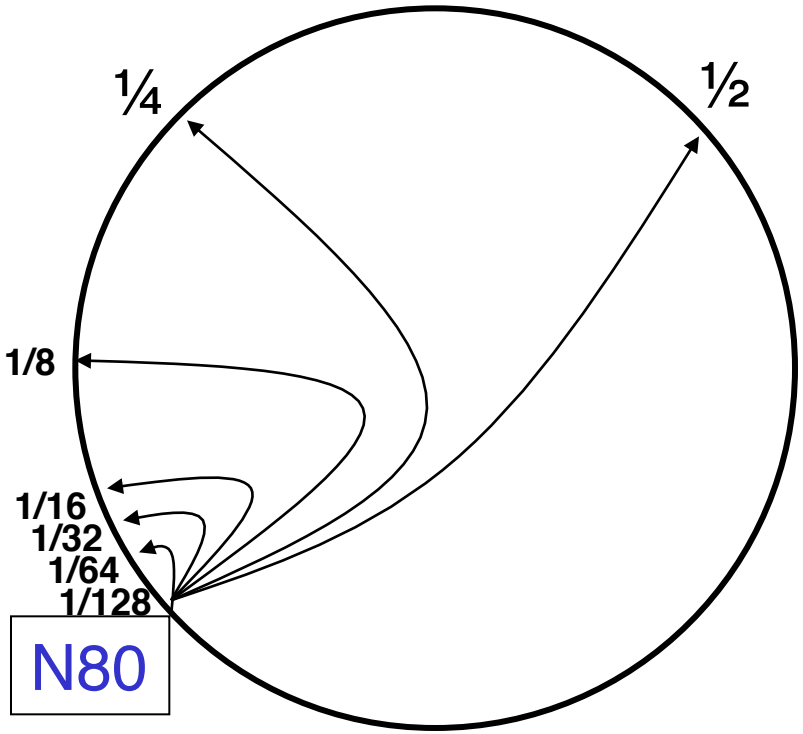


Simple lookup algorithm

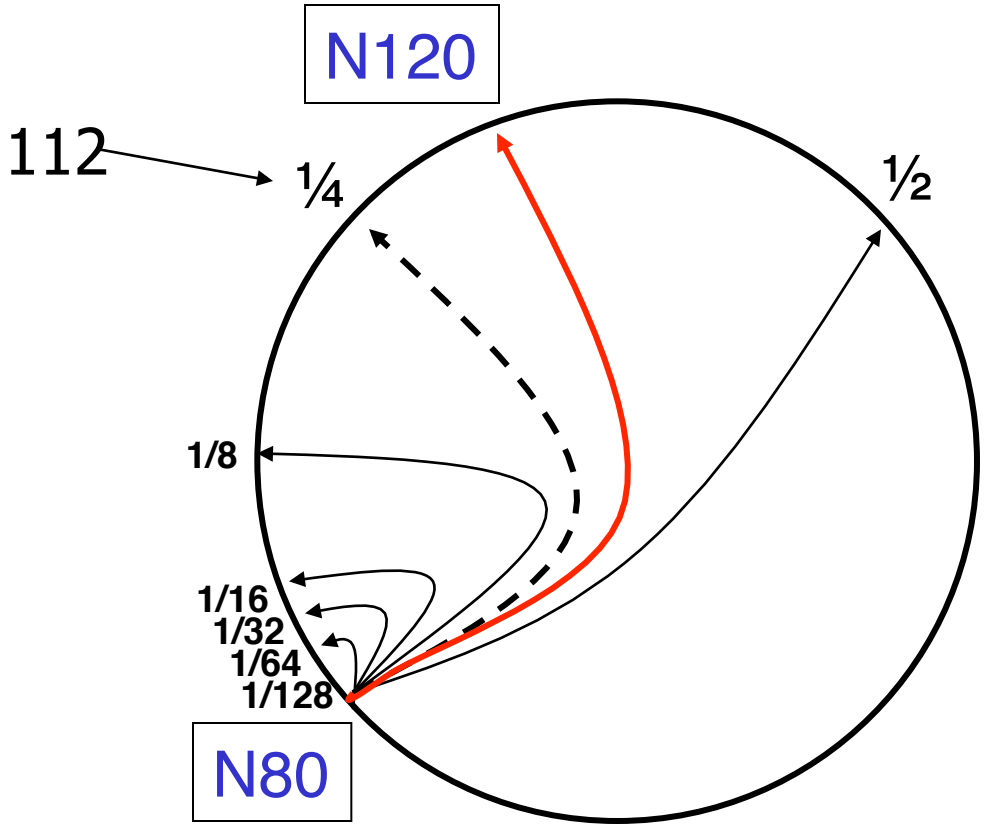
```
Lookup(my-id, key-id)
    n = my successor
    if my-id < n < key-id
        call Lookup(id) on node n    // next hop
    else
        return my successor          // done
```

- Correctness depends only on successors

“Finger table” allows log(N)-time lookups



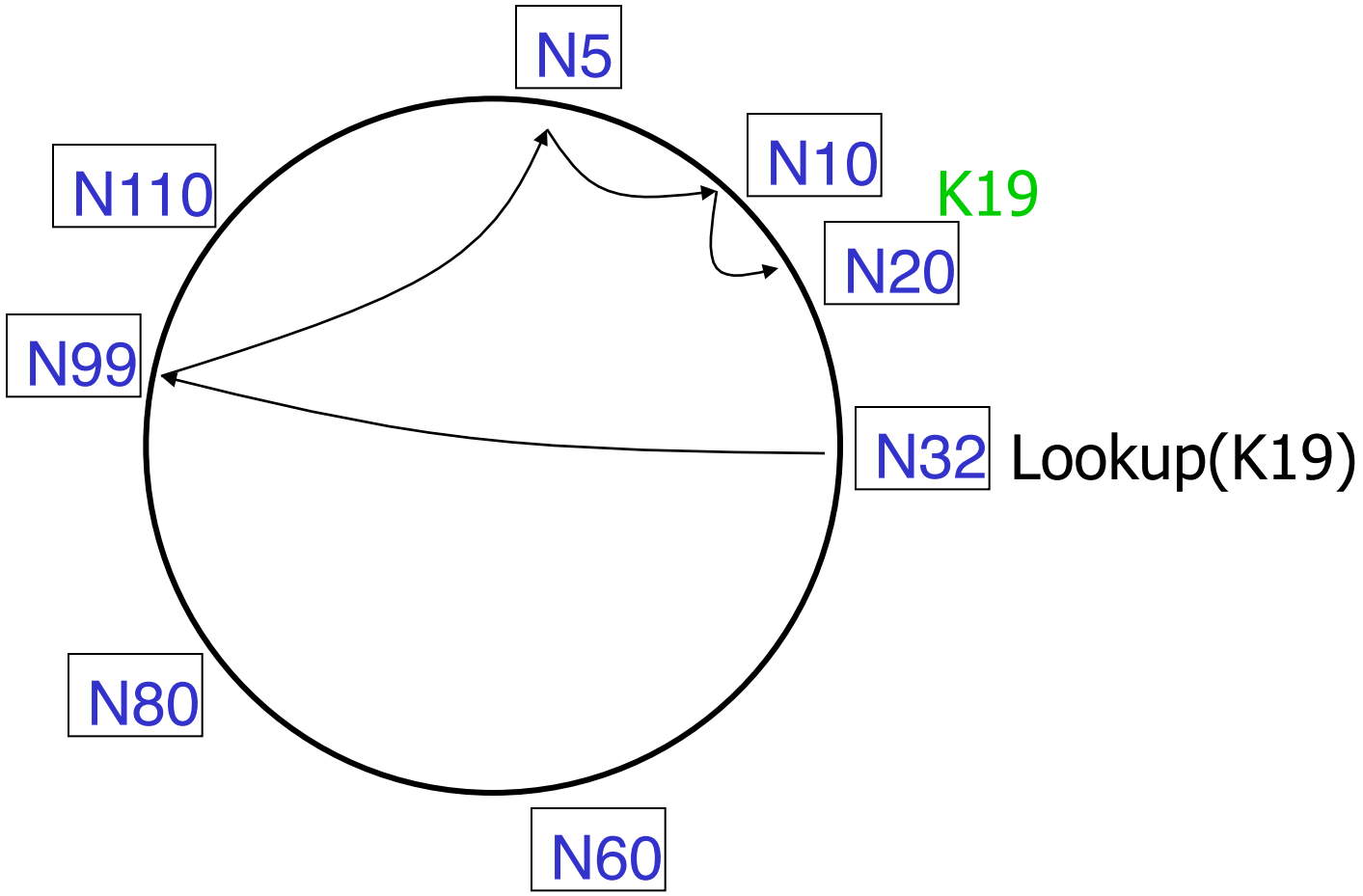
Finger i points to successor of $n+2^i$



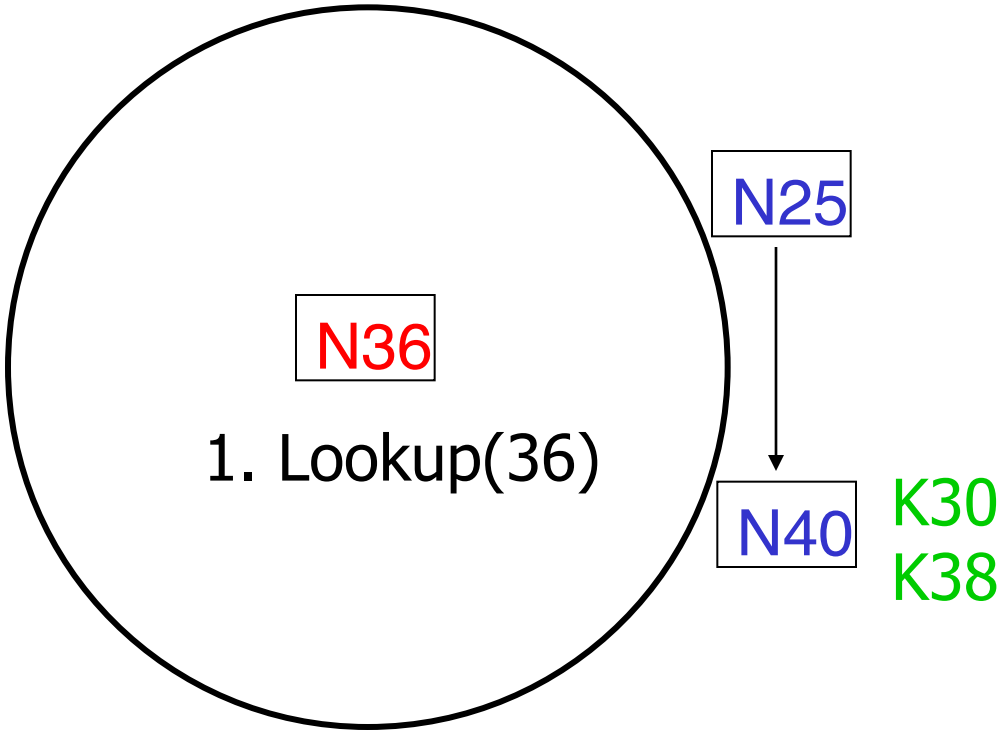
Lookup with fingers

```
Lookup(my-id, key-id)
   look in local finger table for
        highest node n s.t. my-id < n < key-id
   if n exists
        call Lookup(id) on node n      // next hop
   else
        return my successor            // done
```

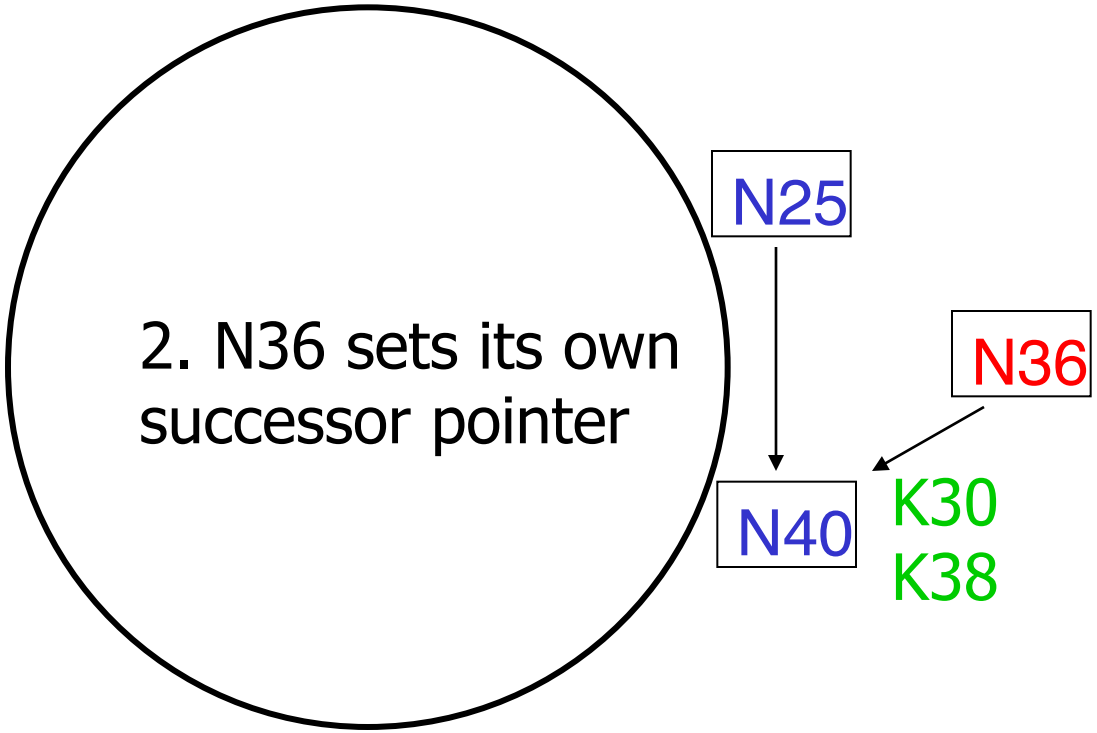
Lookups take O(*log(N)*) hops



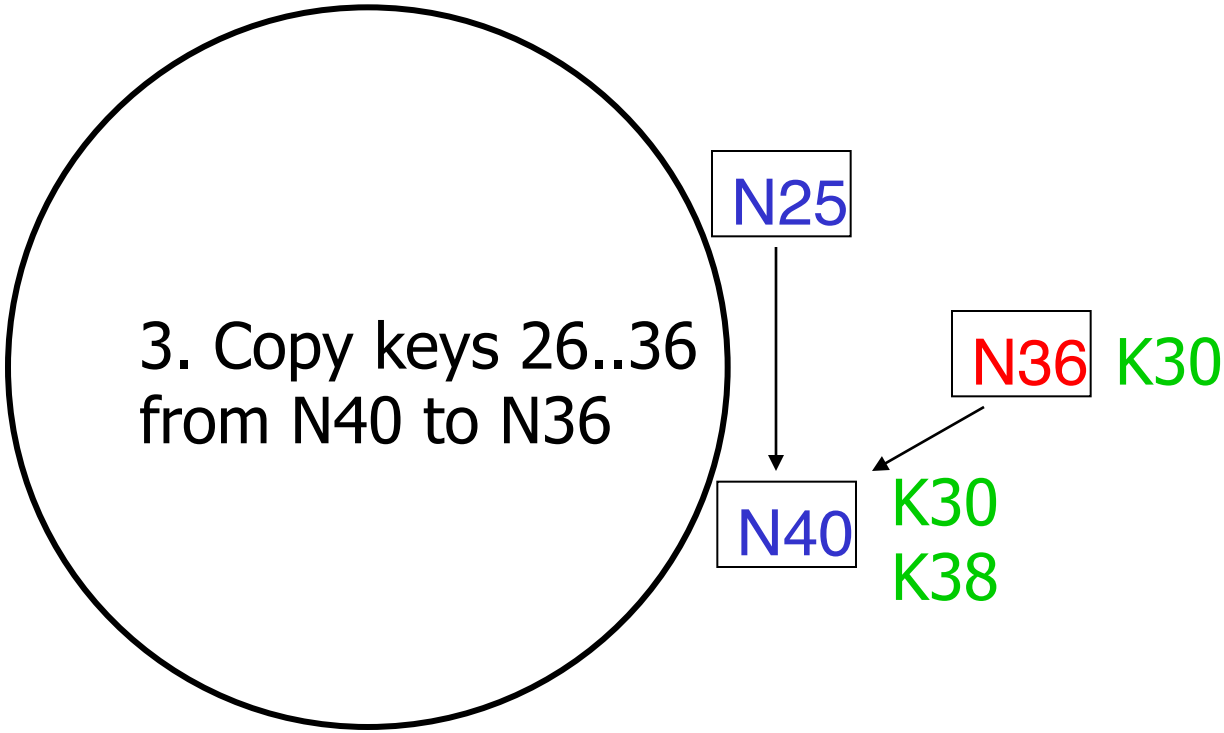
Joining: linked list insert



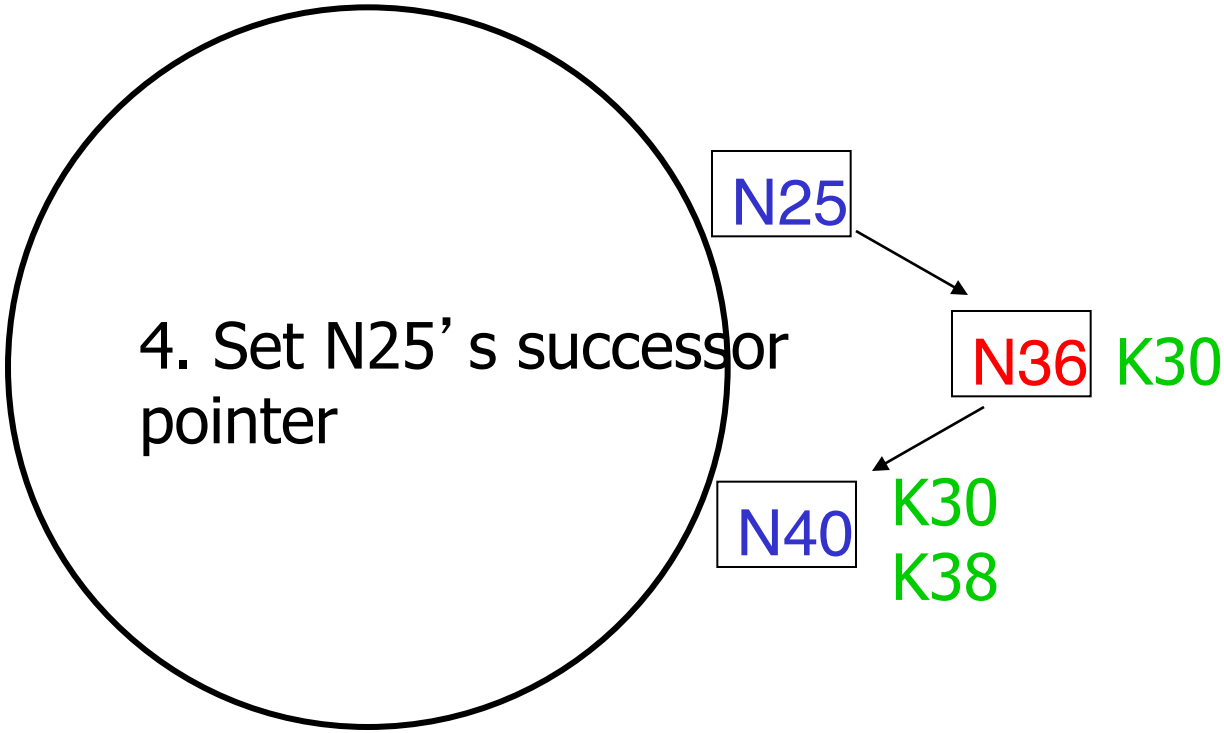
Join (2)



Join (3)

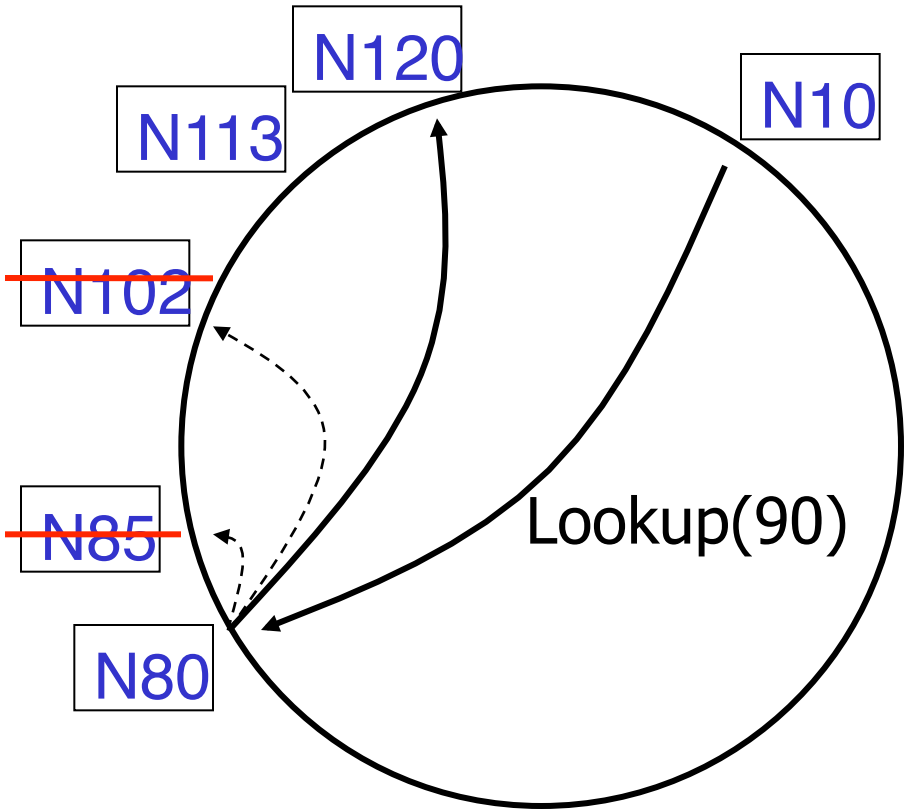


Join (4)



Update finger pointers in the background
Correct successors produce correct lookups

Failures might cause incorrect lookup



N80 doesn't know correct successor, so incorrect lookup

Solution: successor lists

- Each node knows r immediate successors
- After failure, will know first live successor
- Correct successors guarantee correct lookups

- Guarantee is with some probability

Choosing the successor list length

- Assume 1/2 of nodes fail
- P(successor list all dead) = $(1/2)^r$
  - I.e. P(this node breaks the Chord ring)
  - Depends on independent failure
- P(no broken nodes) = $(1 - (1/2)^r)^N$
  - $r = 2log(N)$ makes prob. = $1 - 1/N$

Lookup with fault tolerance

```
Lookup(my-id, key-id)
  look in local finger table and successor-list
      for highest node n s.t. my-id < n < key-id
  if n exists
      call Lookup(id) on node n      // next hop
      if call failed,
          remove n from finger table
          return Lookup(my-id, key-id)
  else return my successor           // done
```

Other design issues

- Concurrent joins
- Locality
- Heterogeneous node
- Dishonest nodes
- …