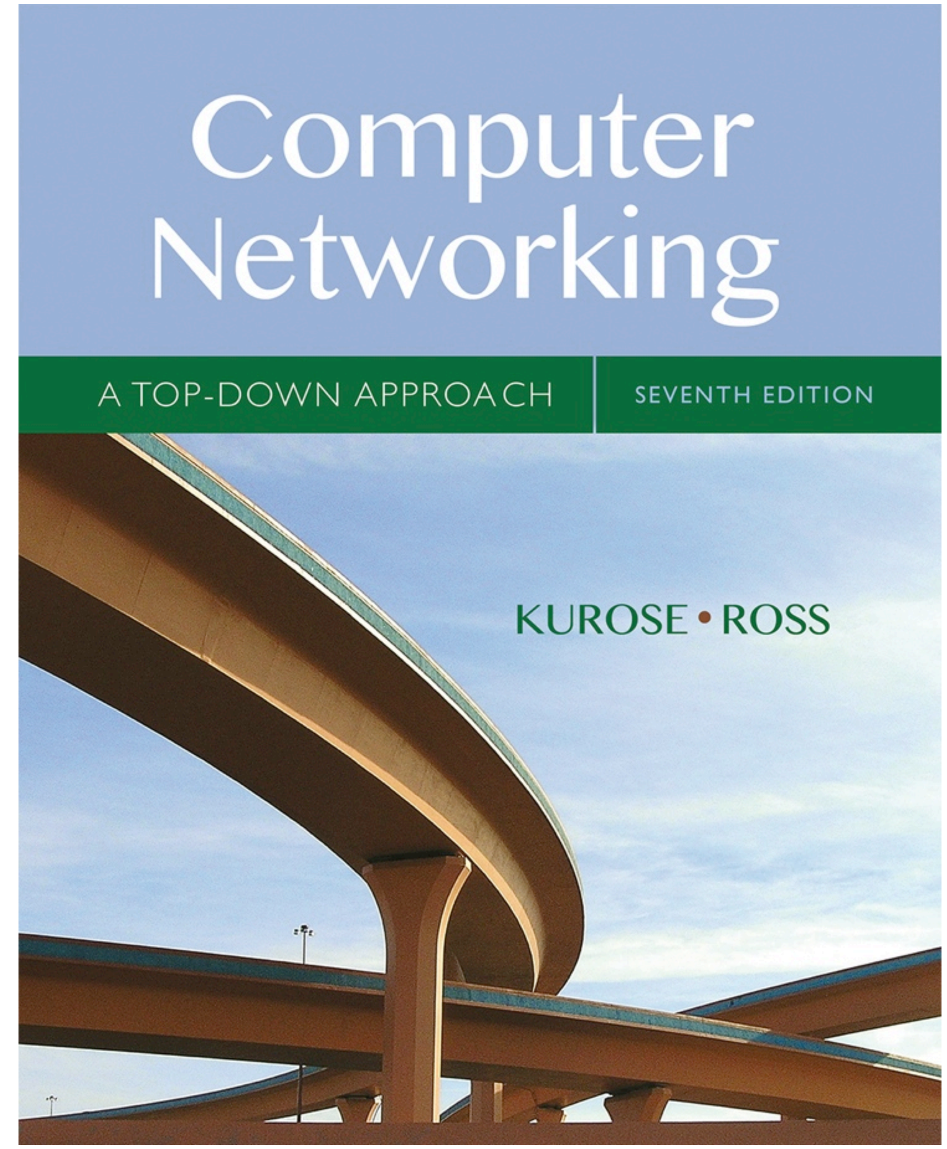
CN-Advanced
L35

Distributed Hash Table

Dr. Ram P Rustagi
rprustagi@ksit.edu.in
http://www.rprustagi.com
https://www.youtube.com/rprustagi

Acknowledgements

Chapter Multimedia Networking



Computer Networking: A Top Down Approach

7th edition
Jim Kurose, Keith Ross
Pearson/Addison Wesley
April 2016

P2P Networking

- Need a database of information which peer contains which information.
  - e.g. list of peers having movie-X
    - List could be IP address of peers
  - Information is generally <key, value> pair
- Should it be centralized or distributed
  - Centralized: locating info is easier
  - Distributed: locating info is complex
  - Each have its pros and cons

Distributed Hash Table (DHT)

- Centralized database
  - <key, value> pair in one central DB, e.g.
    - USN, Name
    - Name, phone number(s)
  - Query the DB with key, get the value(s)
- Challenges with Centralized DB
  - Scaling of DB, Performance,
  - Network congestion
- Solution: Decentralized DB (in P2P version)
  - Each peer holds part of the information.
  - Any peer can query DHT for any info.

Distributed Hash Table (DHT)

- DHT: a *distributed P2P database*
- Databse has (key, value) pairs; e.g.:
  - Key: Adhaar number; value: human name
  - Key: movie title; value: IP address
- Distribute the (key, value) pairs
  - over the (millions of peers)
  - A peer may have only key/value info, not actual info
- A peer queries DHT with key
  - DHT returns values that match the key
  - Note: It can’t query an individual peer
- Peers can also insert (key, value) pairs
- Peer can leave too at random

Use of DHT in P2P Context

- Consider Peers A and B have Linux distro.
- The DHT DB will have: (Linux, IP_A), (Linux, IP_B)
- Consider peer C maintains this key/value pair info.
  - C does not have Linux distro though
  - It only has the info where this distro is available
- Assume that D wants to get this Linux distro.
- D queries the DHT with "Linux" as the key
- DHT decides that C has this key.
- DHT contacts C, and obtains key/value pair.
- This info (Linux, IP_A), (Linux, IP_B) is given to D
- D contacts either A or B to get the distro.

Q: how to assign keys to peers?

- Central issue:
  - Can we keep all <key,value> pairs in one place
- Distributed implementation
  - How to assign (key, value) pairs to peers?
  - Can these be randomly distributed among peers?
  - Each peer need to know & query every peer
  - How to distribute?
- Basic idea:
  - Convert each key to an integer (Identifier)
  - Assign integer (Identifier) to each peer
  - Put (key,value) pair in the peer closest to the key
    - Number of peers doesn't cover full number range

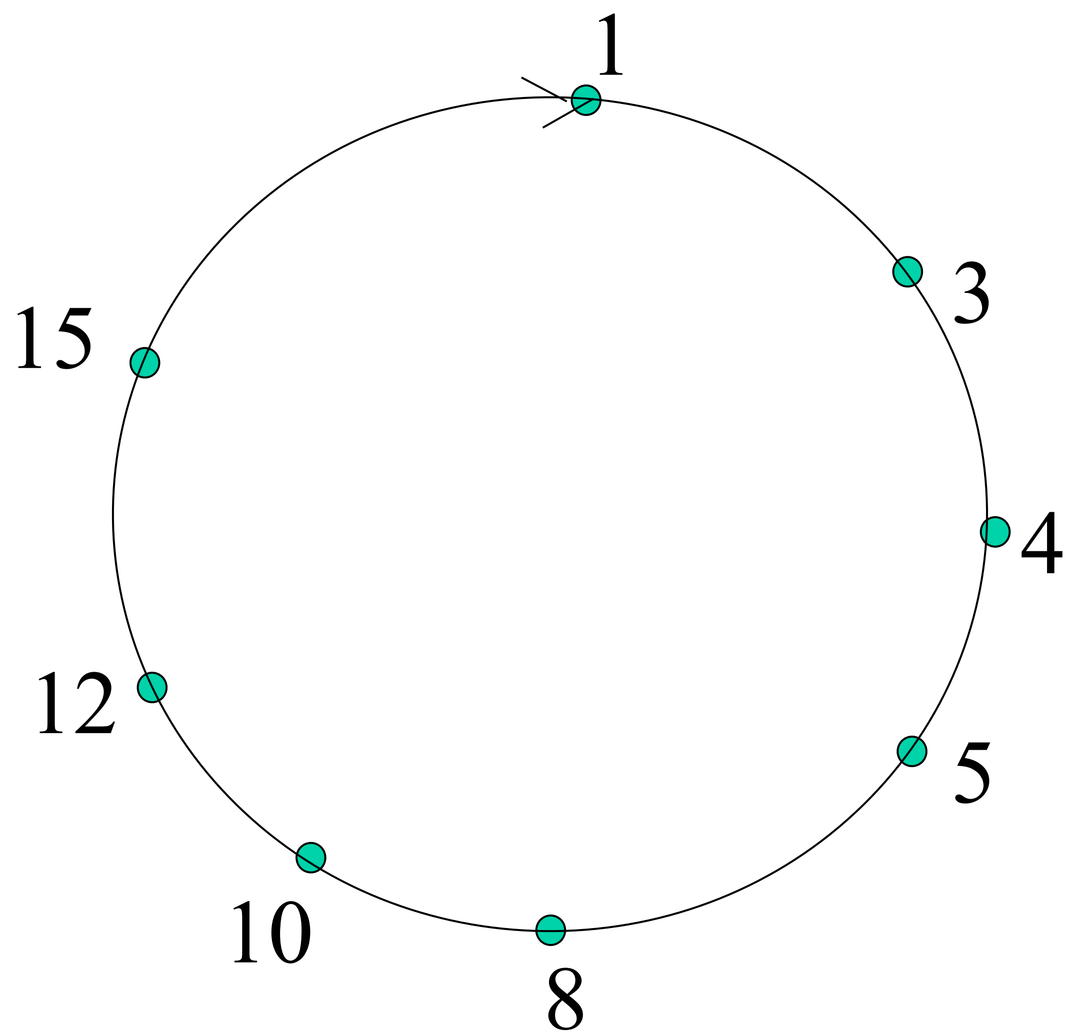
DHT identifiers

- Assign integer identifier to each peer in range $[0, 2^n - 1]$ for some n.
  - Each identifier represented by n bits.
- Require each key to be an integer in same range
  - Convert given key value (e.g. 17CS52) into integer
- To get integer key, hash original key
  - e.g. key = hash("Computer Networks")
  - The reason behind referring it as a *distributed "hash" table*
- *Should each peer know about existence of all other peers?*
  - *Scalability issues?*

Assign keys to peers

- Rule: assign key to the peer that has the *closest* ID.
- Convention:
  - Closest is the *immediate successor* of the key.
- e.g., n=`4`; peers: `1,3,4,5,8,10,12,14`;
  - key = `13`, then successor peer = `14`
  - key = `15`, then successor peer = `1`
- How does one identify the peer having a key?
  - Each maintains info about assignment
    - Benefit: can locally determine
- How does one know about all existing peers?
  - Issues: not scalable, any changes will cause issues

Circular DHT (1)

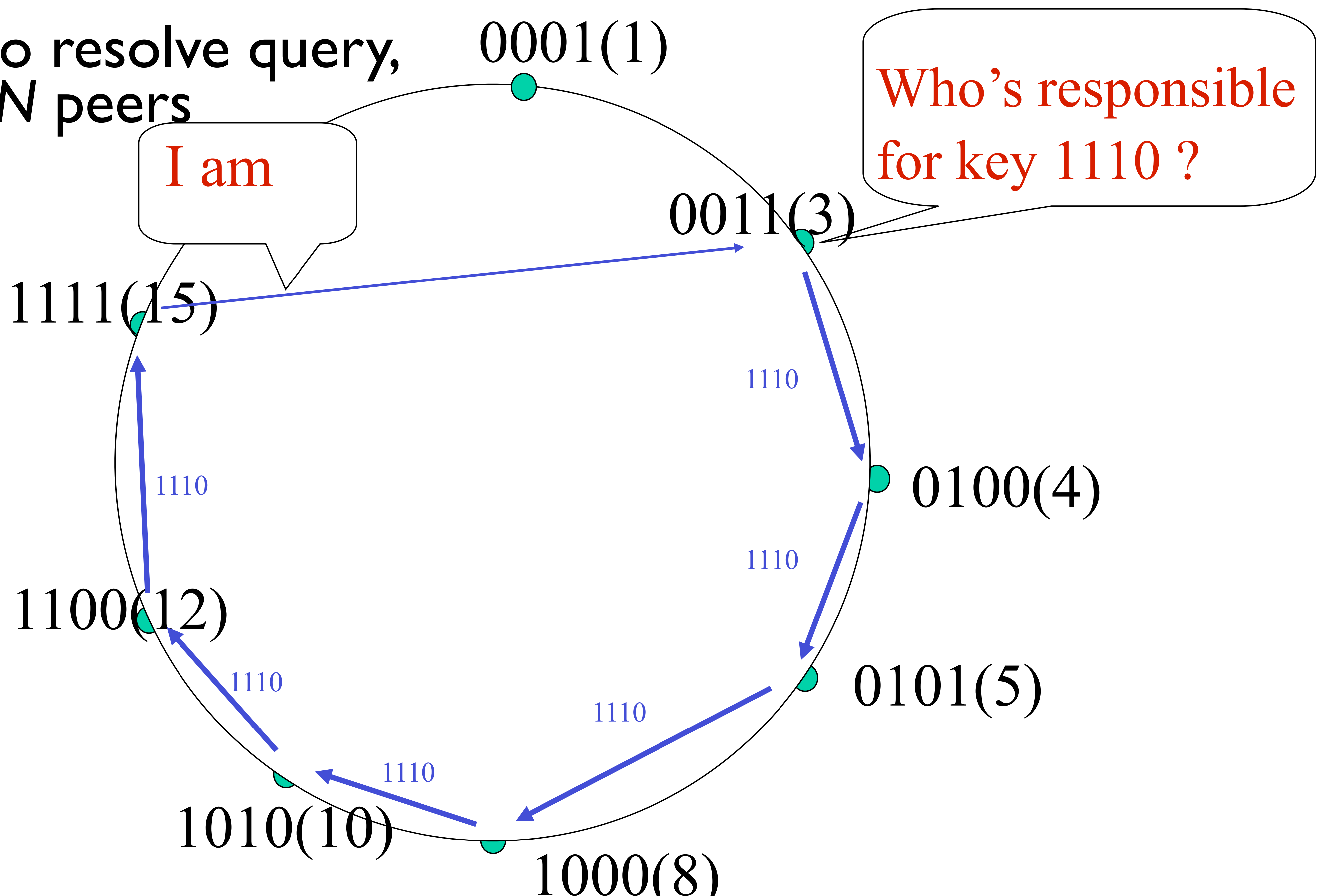


- Each peer *only* aware of immediate successor and predecessor.
- "Overlay network"

Circular DHT (1)

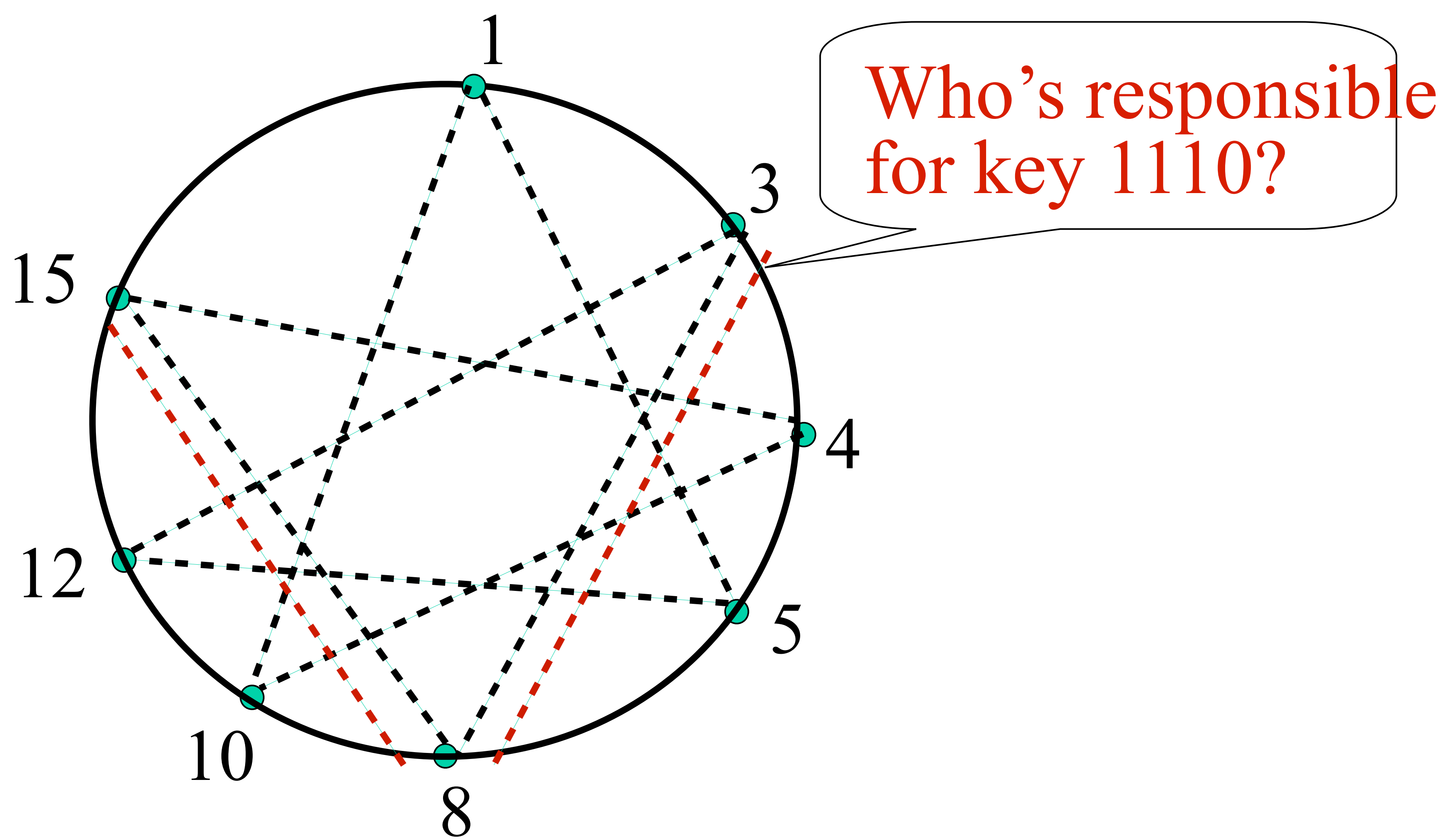
Benefit: Each peer need to know about only 1 peer.

Challanges:
O(N) messages to resolve query, when there are *N* peers



Define closest as closest successor

Circular DHT with shortcuts



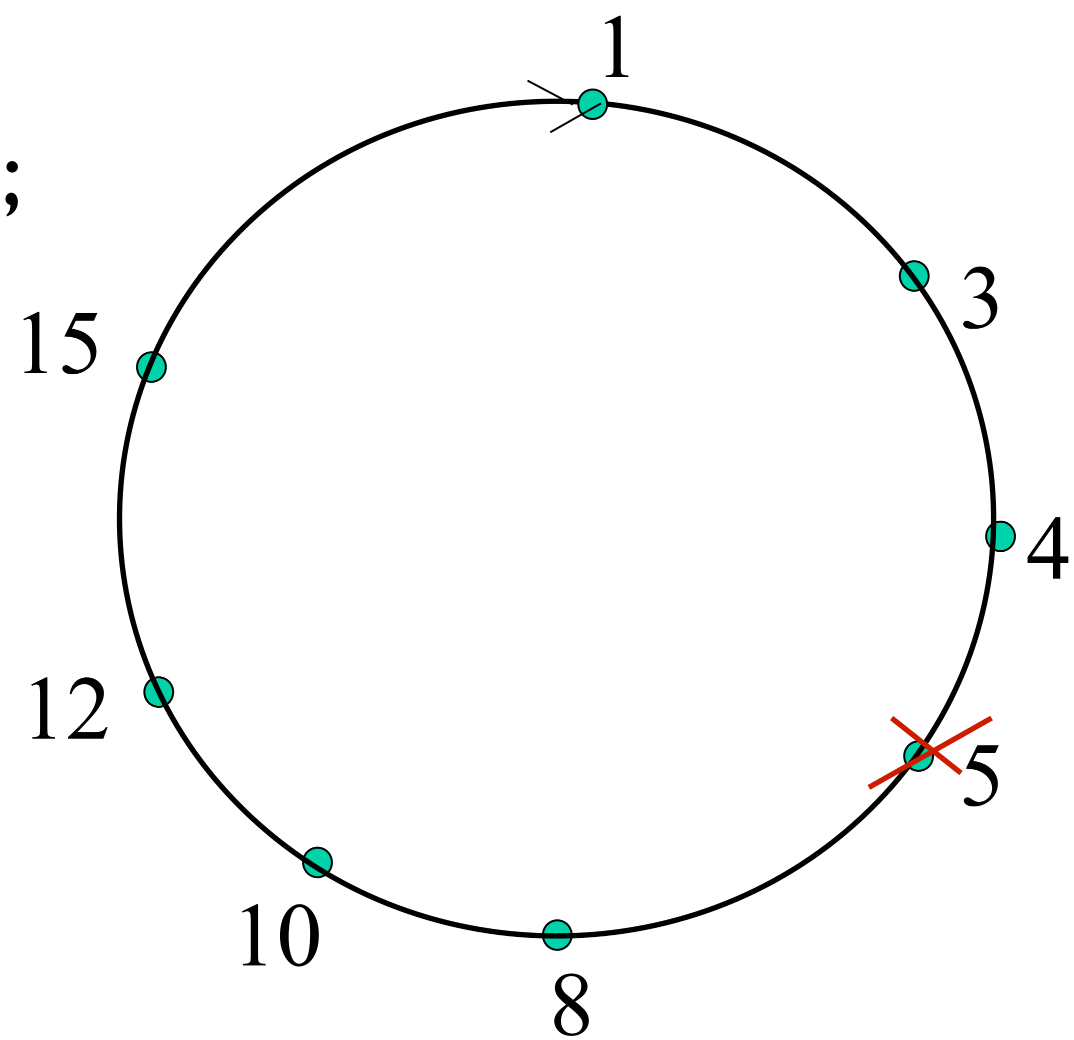
- Each peer keeps track of IP addresses of predecessor, successor, and few other short cuts.
- Reduced from 6 to 2 messages.

Circular DHT with shortcuts

- Trade off between
  - Keeping info about number of peers
  - Number of msgs to communicate for each query
- Two extreme cases
  - Info maintained about all peers
    - communication: 1 msg
  - Info maintained about neighbours
    - communication: N/2 msgs
- Possible to design shortcuts
  - so *O(log N)* neighbours,
  - *O(log N)* messages in query

Peer churn

- *example: peer 5 abruptly leaves*
- Peer 4 detects peer 5 departure;
- makes 8 its immediate successor;
- asks 8 who its immediate successor is;
- makes 8's immediate successor its second successor.



- What if peer `13` wants to join?
  - it only knows peer `1`
  - It sends join req to 1
  - This msg keep getting forwarded till it reaches 12
  - 12 knows that it is going to be predecessor of 13
  - 12 informs 13 and accordingly 13 joins the DHT

Summary

- P2P Distribution
- BitTorrent
- DHT