Distributed Hash Table (DHT)

- DHT: a *distributed P2P database*
- database has *(key, value)* pairs; examples:
  - key: ss number; value: human name
  - key: movie title; value: IP address
- Distribute the *(key, value)* pairs over the (millions of peers)
- a peer *queries* DHT with key
  - DHT returns values that match the key
- peers can also *insert* *(key, value)* pairs
Q: how to assign keys to peers?

• central issue:
  • assigning (key, value) pairs to peers.

• basic idea:
  • convert each key to an integer
  • Assign integer to each peer
  • put (key, value) pair in the peer that is closest to the key
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

• to get integer key, hash original key
  • e.g., key = hash(“Led Zeppelin IV”)
    • this is why its is referred to as a distributed “hash” table
Assign keys to peers

• rule: assign key to the peer that has the closest ID.

• convention in lecture: 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
Circular DHT (1)

- each peer *only* aware of immediate successor and predecessor.
- “overlay network”
Who’s responsible for key 1110?

I am

Define closest as closest successor

$O(N)$ messages on average to resolve query, when there are $N$ peers

Circular DHT (1)
Circular DHT with shortcuts

- each peer keeps track of IP addresses of predecessor, successor, short cuts.
- reduced from 6 to 2 messages.
- possible to design shortcuts so $O(\log N)$ neighbors, $O(\log N)$ messages in query

Who’s responsible for key 1110?
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?

handling peer churn:
• peers may come and go (churn)
• each peer knows address of its two successors
• each peer periodically pings its two successors to check aliveness
• if immediate successor leaves, choose next successor as new immediate successor