Distributed Hash Tables

What is a DHT?

- Hash Table
  - data structure that maps “keys” to “values”
  - essential building block in software systems
- Distributed Hash Table (DHT)
  - similar, but spread across many hosts
- Interface
  - insert(key, value)
  - lookup(key)

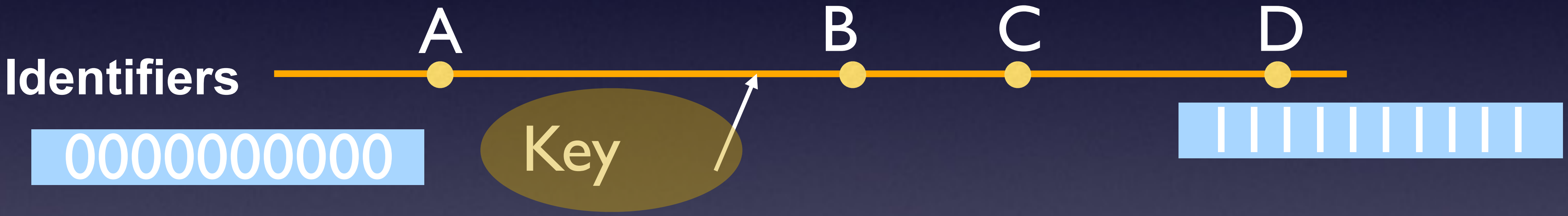
How do DHTs work?

Every DHT node supports a single operation:

- Given key as input; route messages to node holding key
- DHTs are content-addressable

Fundamental Design Idea I

- Consistent Hashing
  - Map keys *and* nodes to an *identifier* space; implicit assignment of responsibility

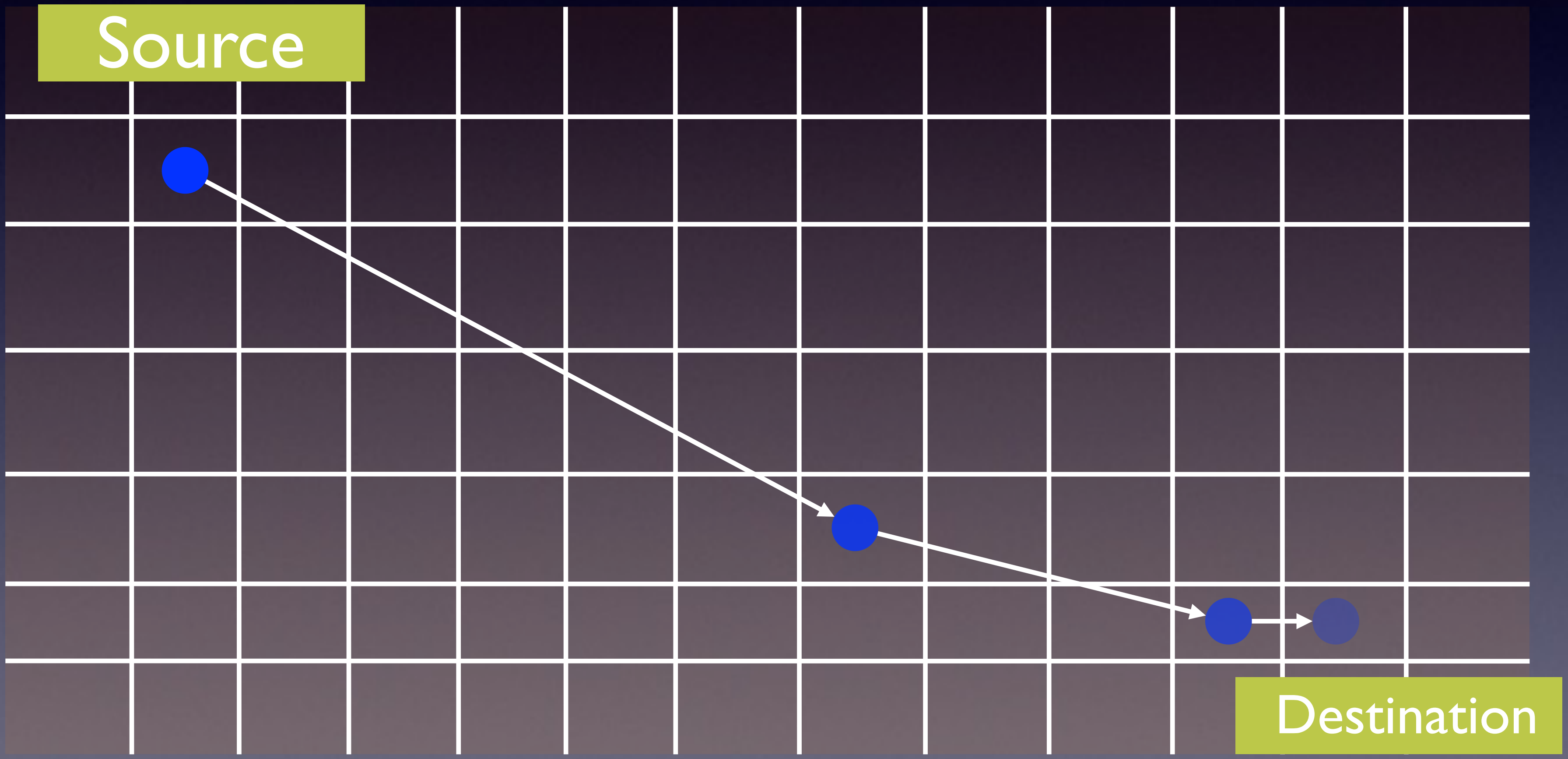


Mapping performed using hash functions (e.g., SHA-1)

- What is the advantage of consistent hashing?

Fundamental Design Idea II

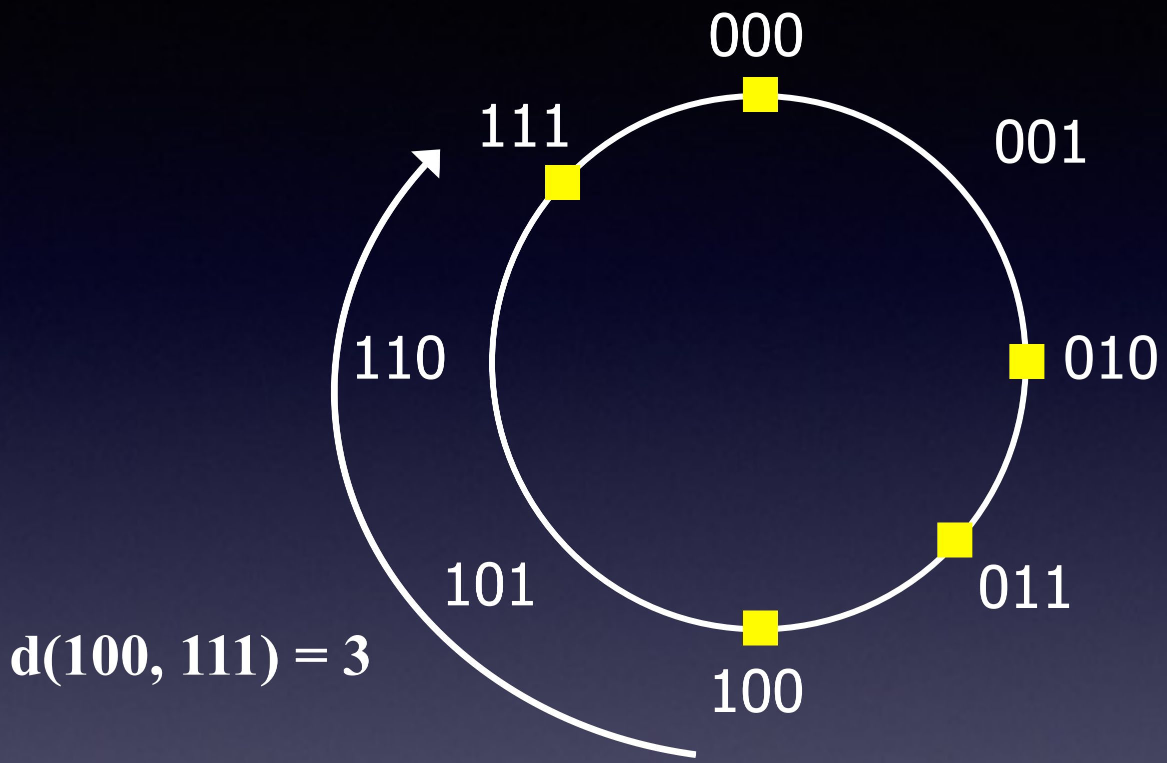
- Prefix / Hypercube routing



How to design a DHT?

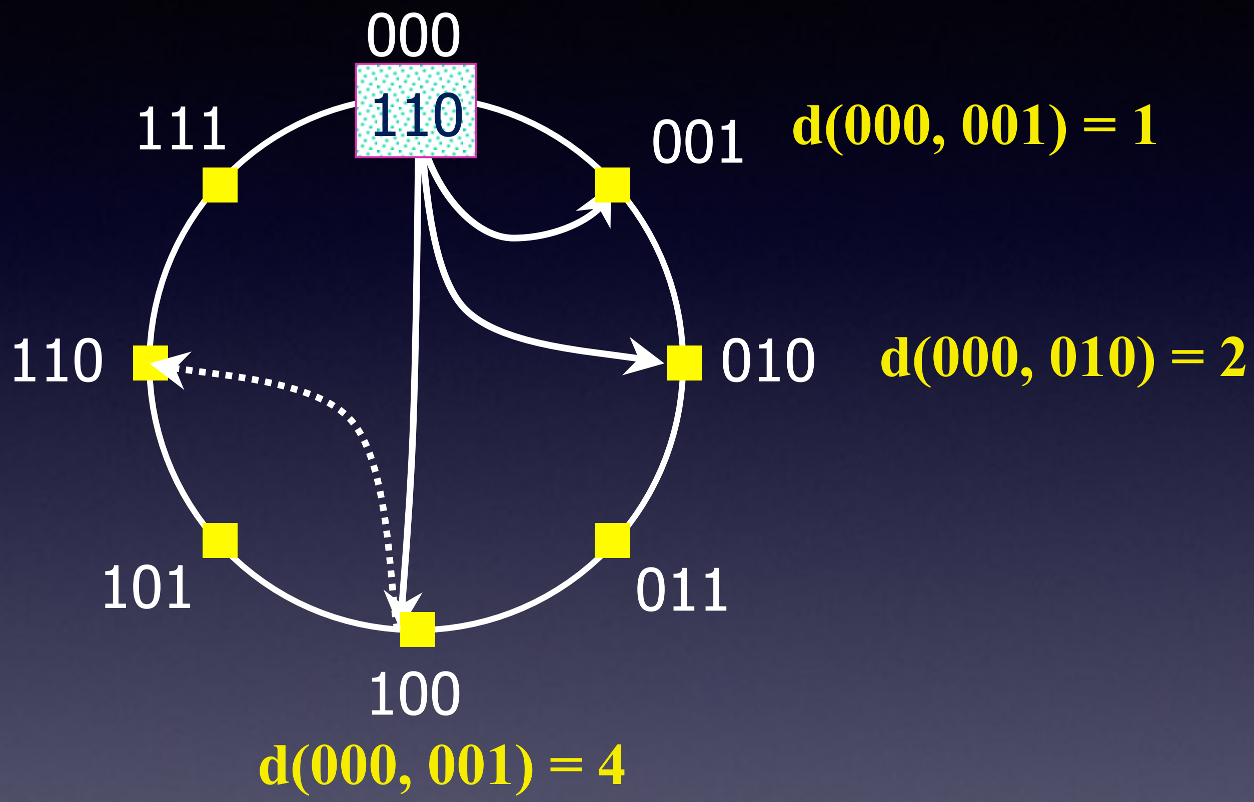
- State Assignment:
  - what "(key, value) tables" does a node store?
- Network Topology:
  - how does a node select its neighbors?
- Routing Algorithm:
  - which neighbor to pick while routing to a destination?
- Various DHT algorithms make different choices
  - CAN, Chord, Pastry, Tapestry, Plaxton, Viceroy, Kademlia, Skipnet, Symphony, Koorde, Apocrypha, Land, ORDI …

State Assignment in Chord



- Nodes are randomly chosen points on a clock-wise ring of values
- Each node stores the id space (values) between itself and its predecessor

Chord Topology and Route Selection



- Neighbor selection: i th neighbor at 2^i distance

- Route selection: pick neighbor closest to destination

Issues

- How do you characterize the performance of DHTs?

Issues

- How do you improve the performance of DHTs?

Issues

- What are the fault tolerance/correctness issues?

Issues

- What are the security issues?

Issues

- What are the load balance issues?