CSC2231: DHTs

http://www.cs.toronto.edu/~stefan/courses/csc2231/05au

Stefan Saroiu

Department of Computer Science

University of Toronto

DHTs today

- **Active area of research for over 3 years now**
- **Ongoing work at almost every major university and lab.**
  - over 20 DHT proposals; as many for DHT applications
  - IRIS
    - DHT-based, robust infrastructure for Internet-scale systems.
    - 5 year, $12M, NSF-funded project
- **Large, and growing, research community**
  - theoreticians, networks and systems researchers

- **Good research topic to stay away from!**
  - I'm working on a paper on DHTs!

Today’s Discussion

- **How do DHTs work?**
- **What properties do DHTs have?**
- **What are P2P systems (as opposed to DHTs)?**
  - Why are DHTs appealing to P2P designs

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

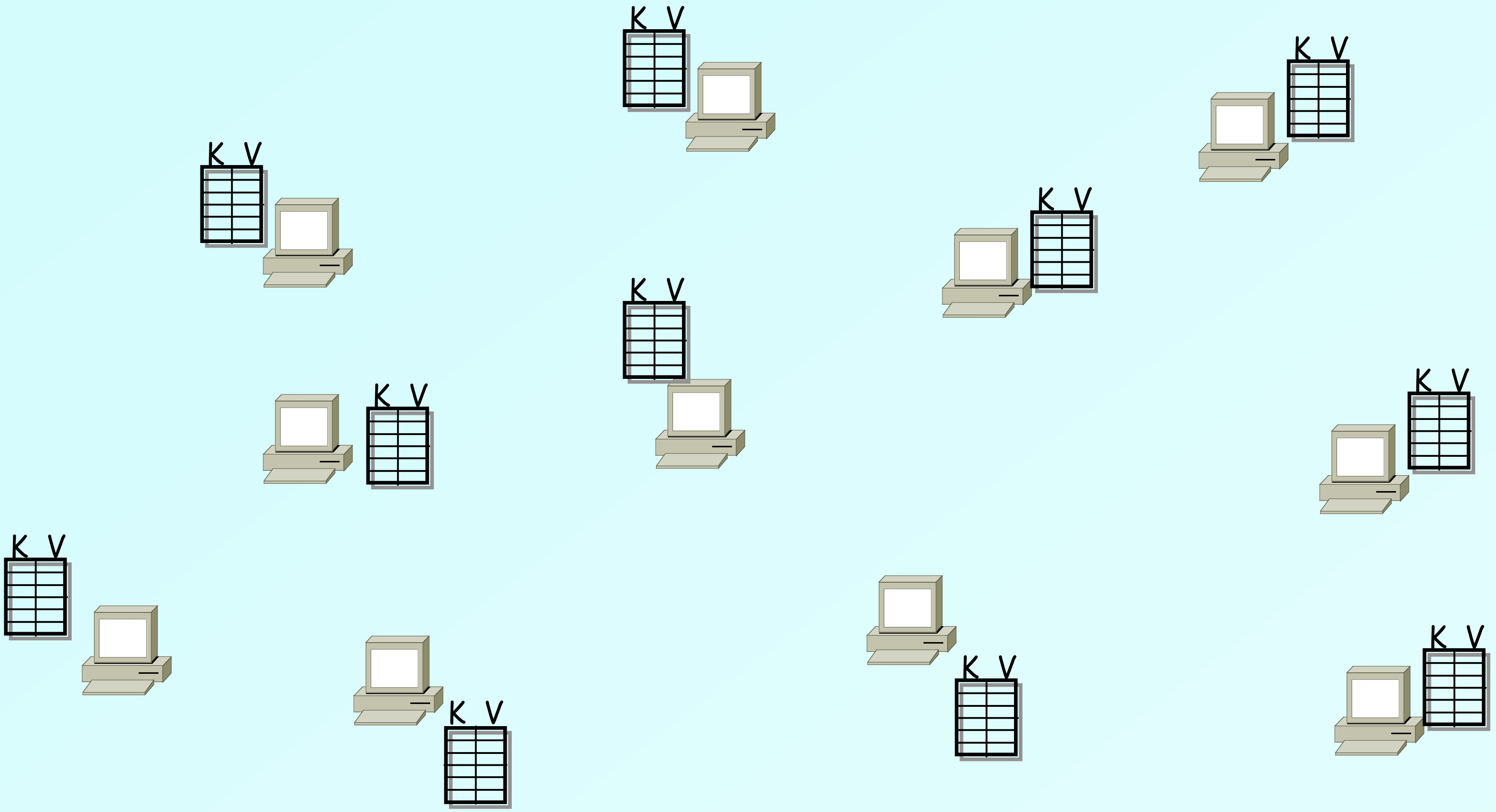
- **Interface**
  - insert(key, value)
  - lookup(key)

How do DHTs work?

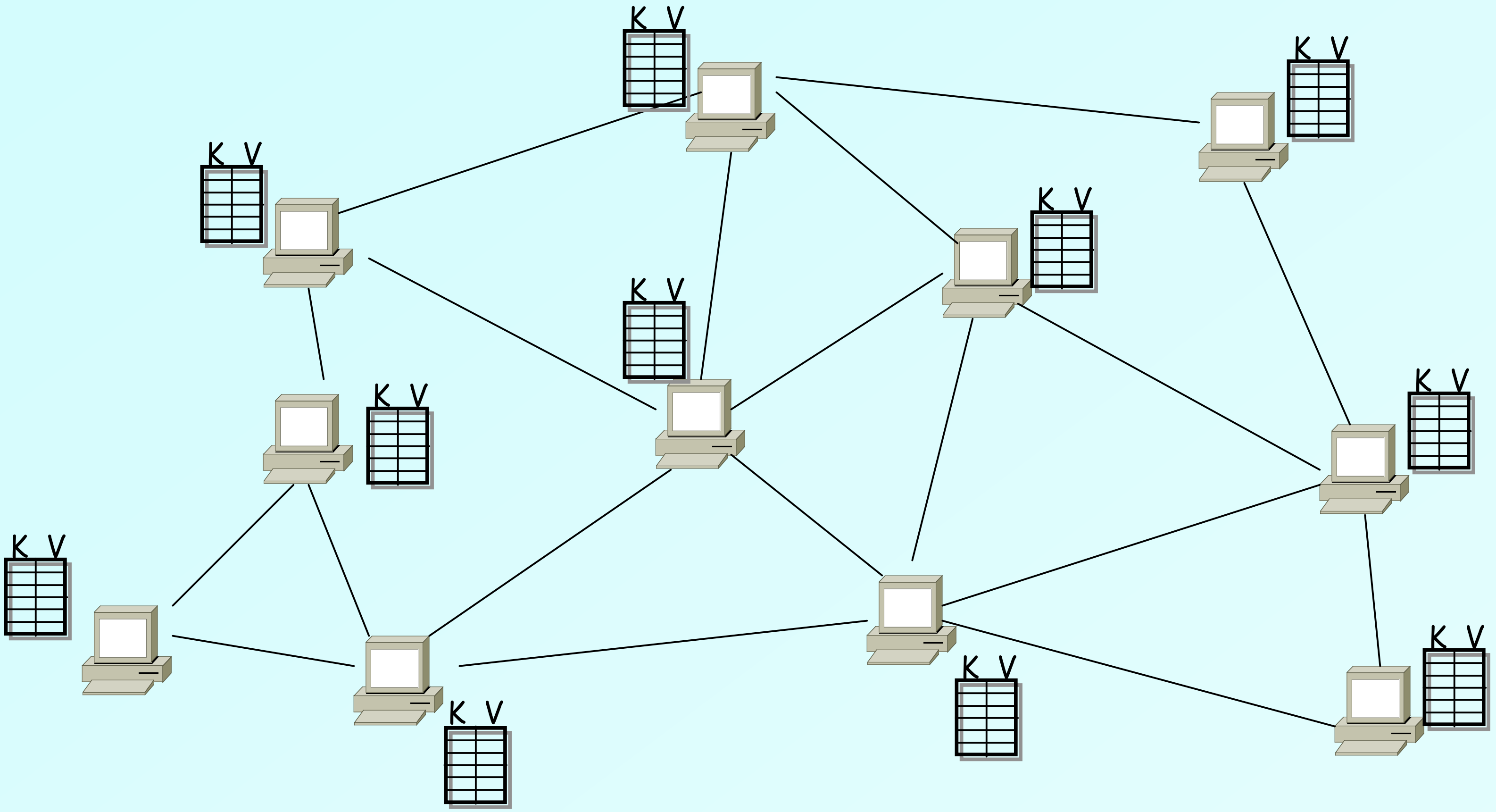
Every DHT node supports a single operation:

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

DHT: basic idea

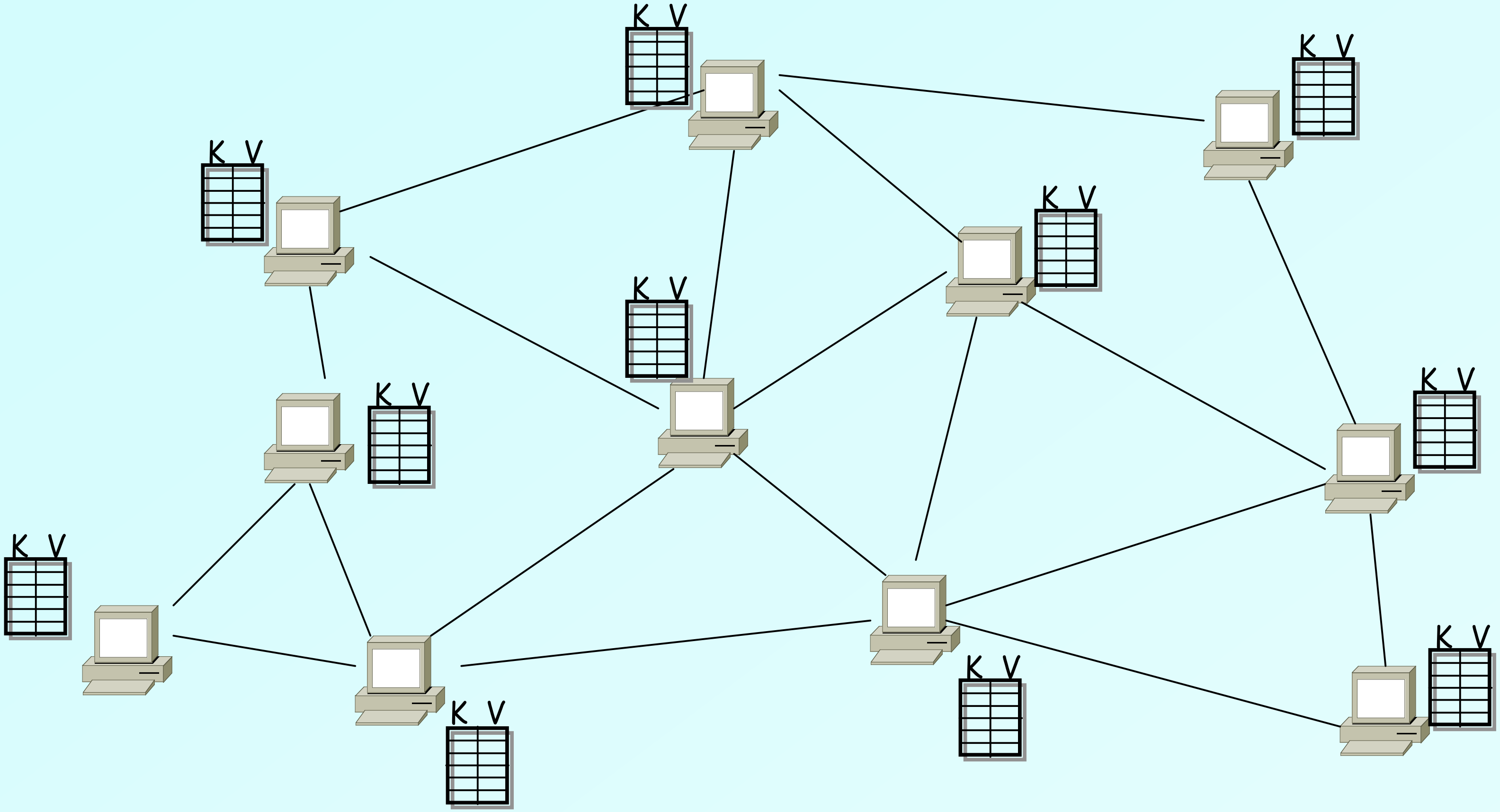


DHT: basic idea



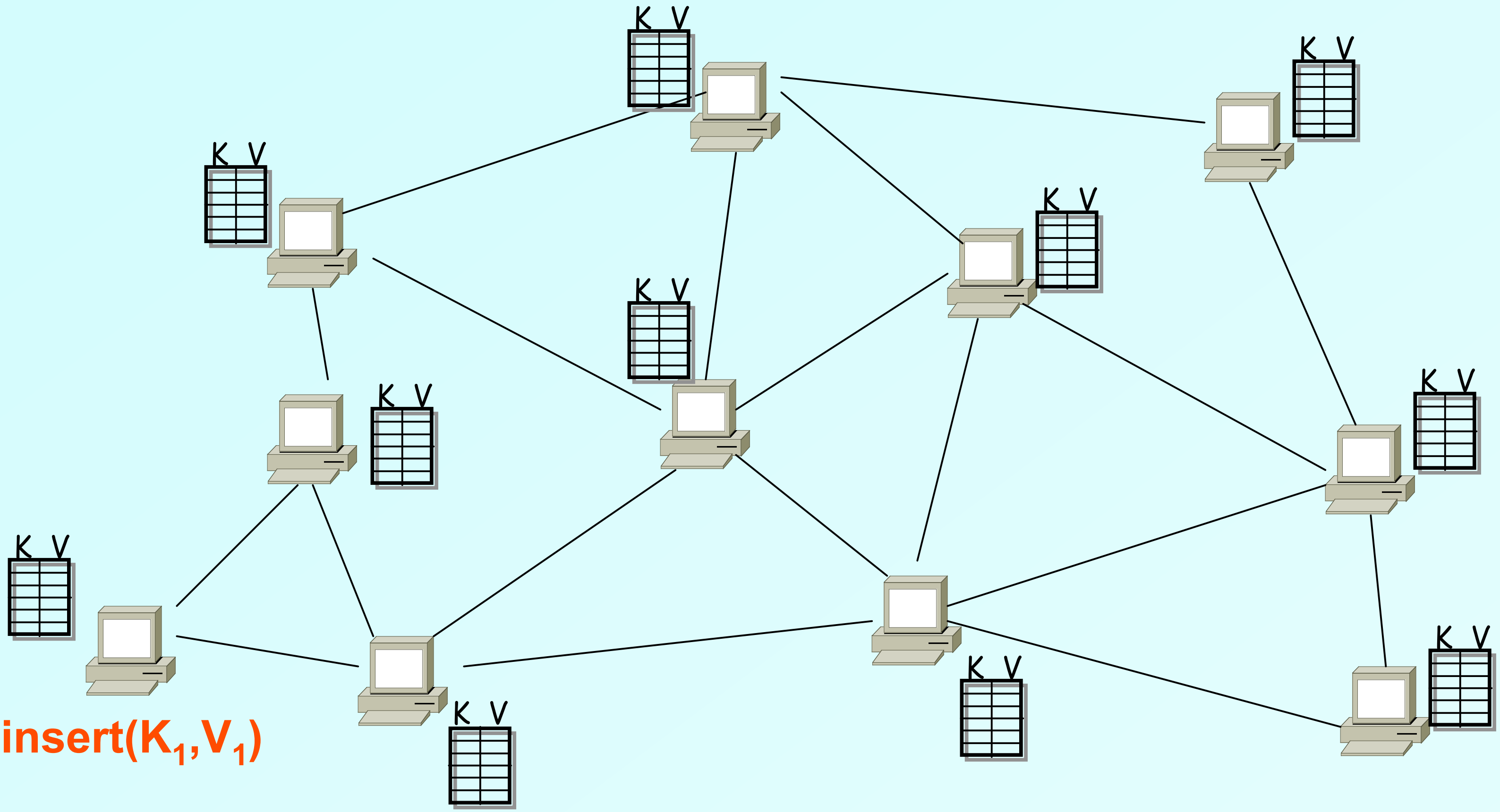
Neighboring nodes are "connected" at the application-level

DHT: basic idea



Operation: take *key* as input; route messages to node holding *key*

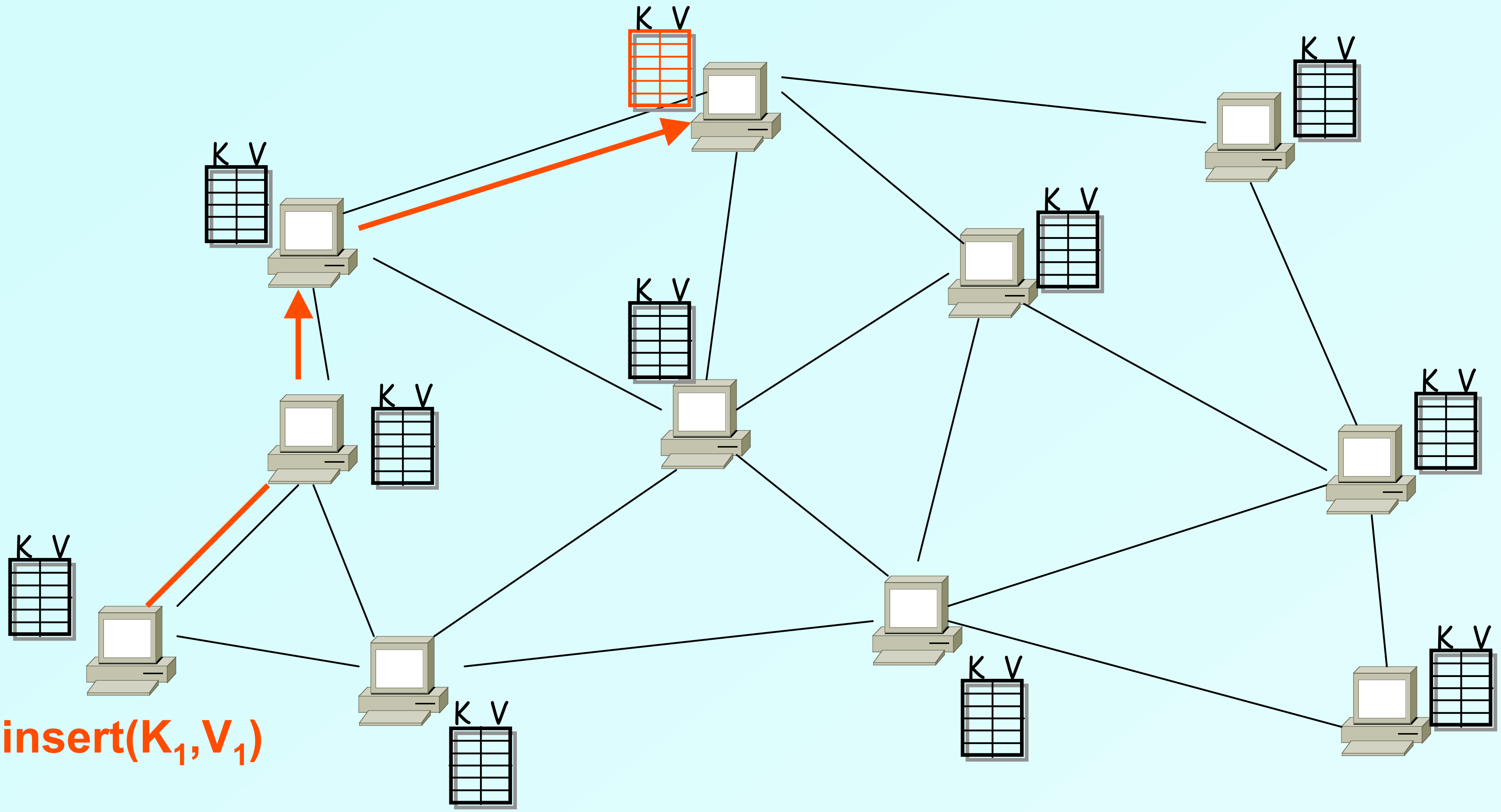
DHT: basic idea



insert(K_1,V_1)

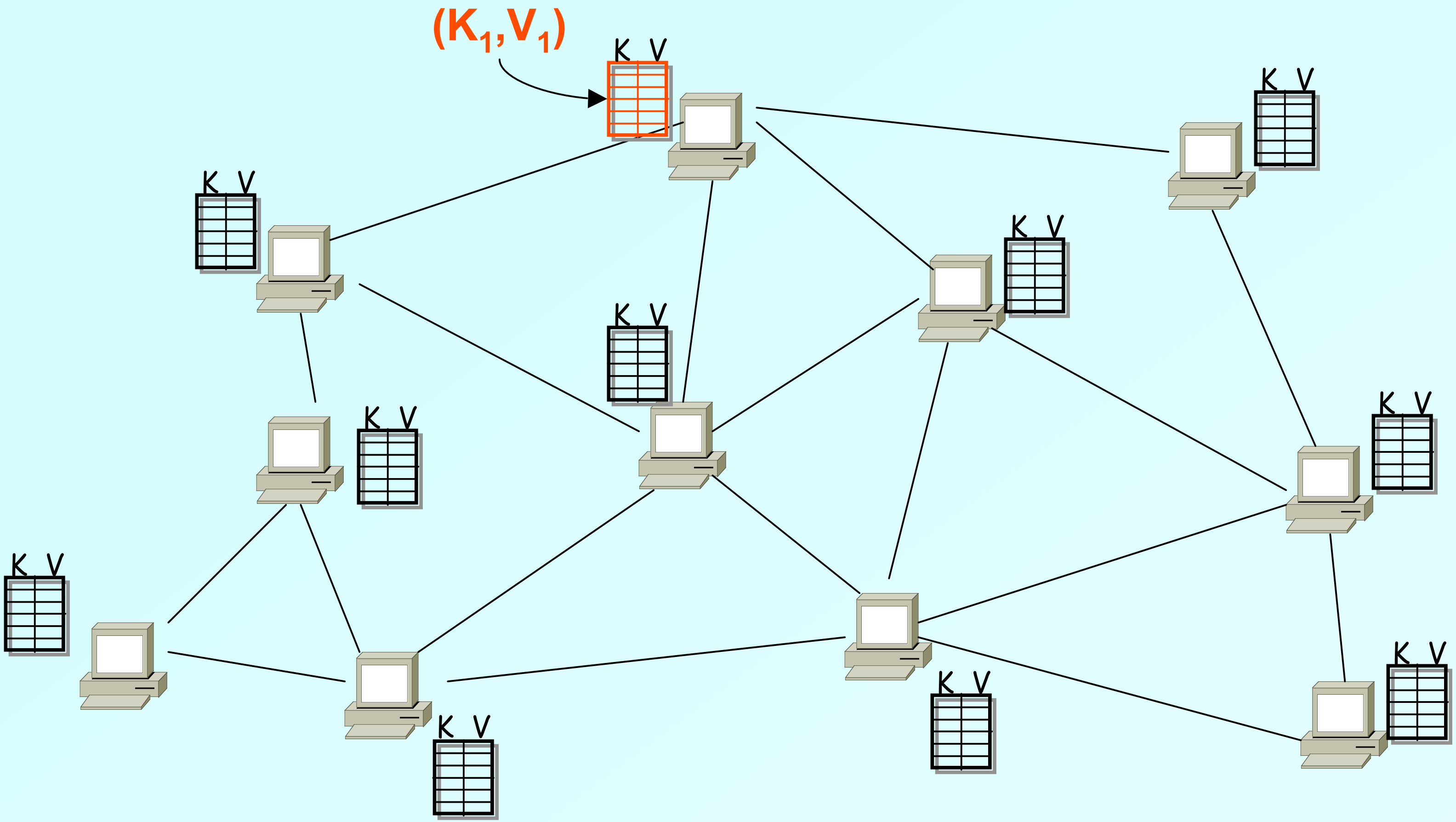
Operation: take *key* as input; route messages to node holding *key*

DHT: basic idea



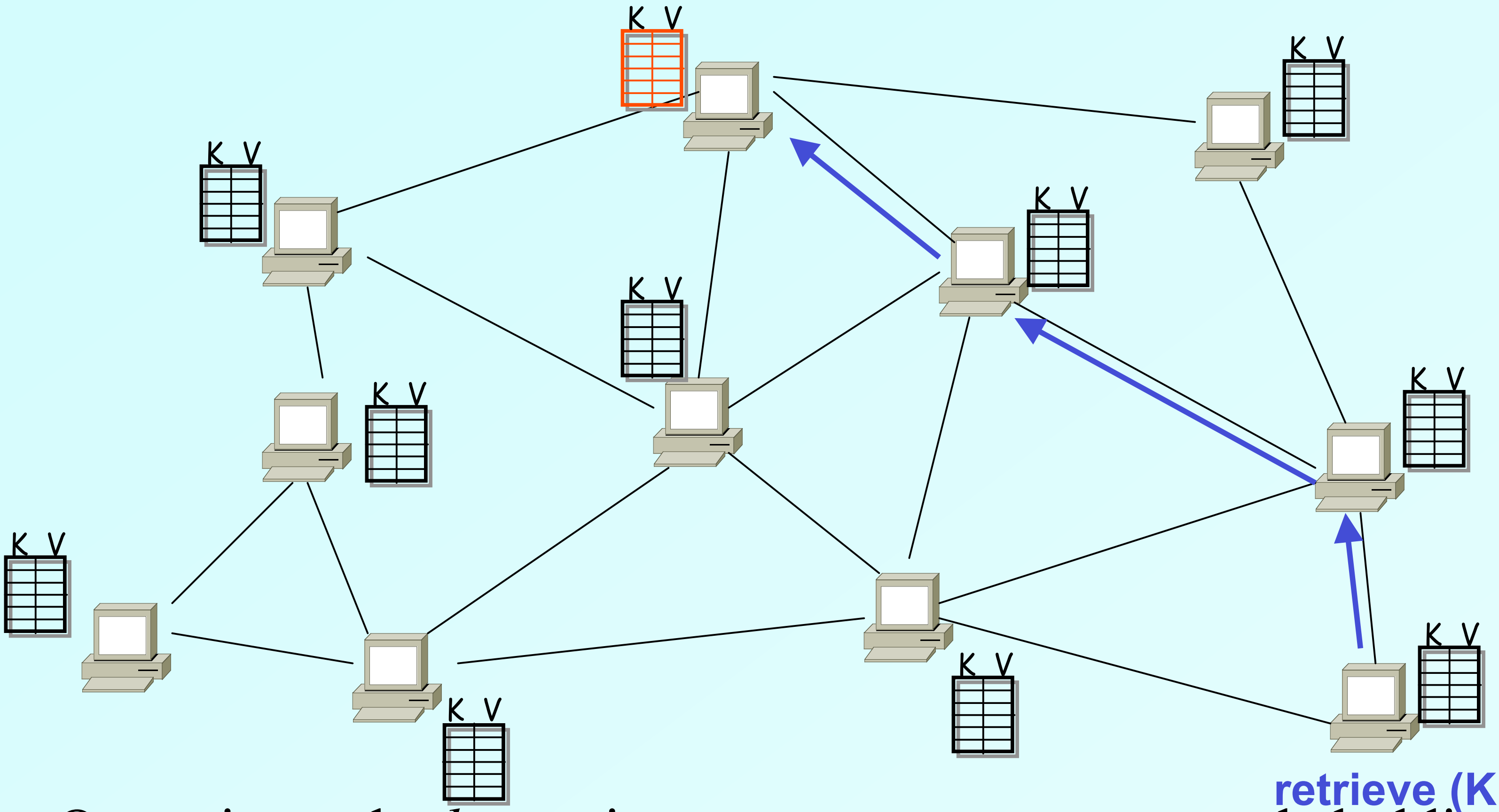
Operation: take *key* as input; route messages to node holding *key*

DHT: basic idea



Operation: take *key* as input; route messages to node holding *key*

DHT: basic idea

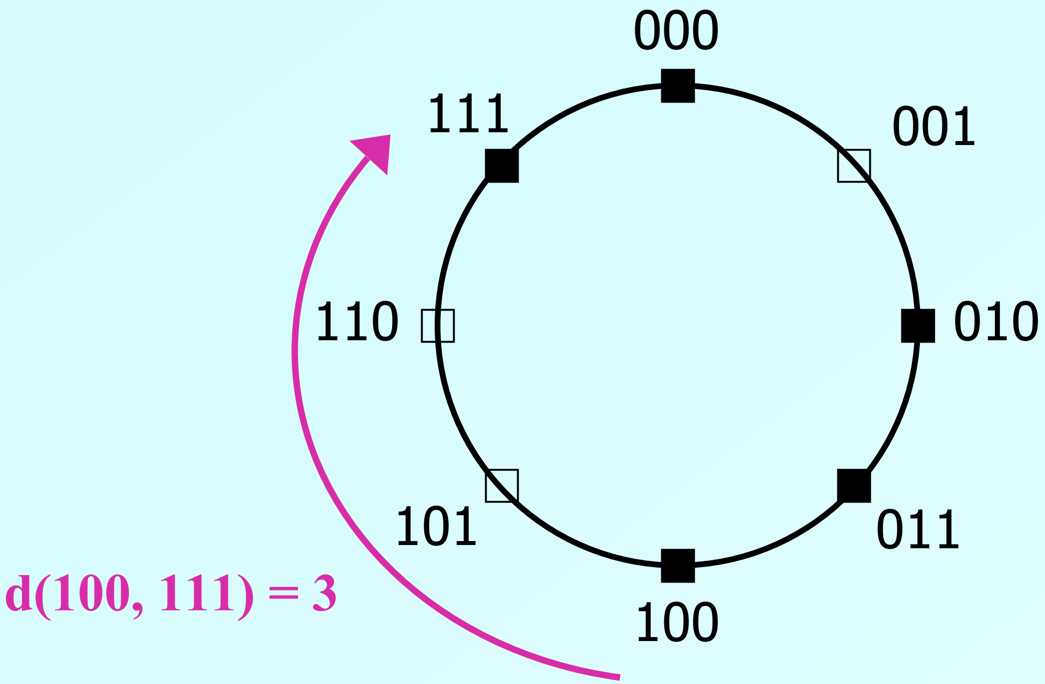


Operation: take *key* as input; route messages to node holding *key*

How to design a DHT?

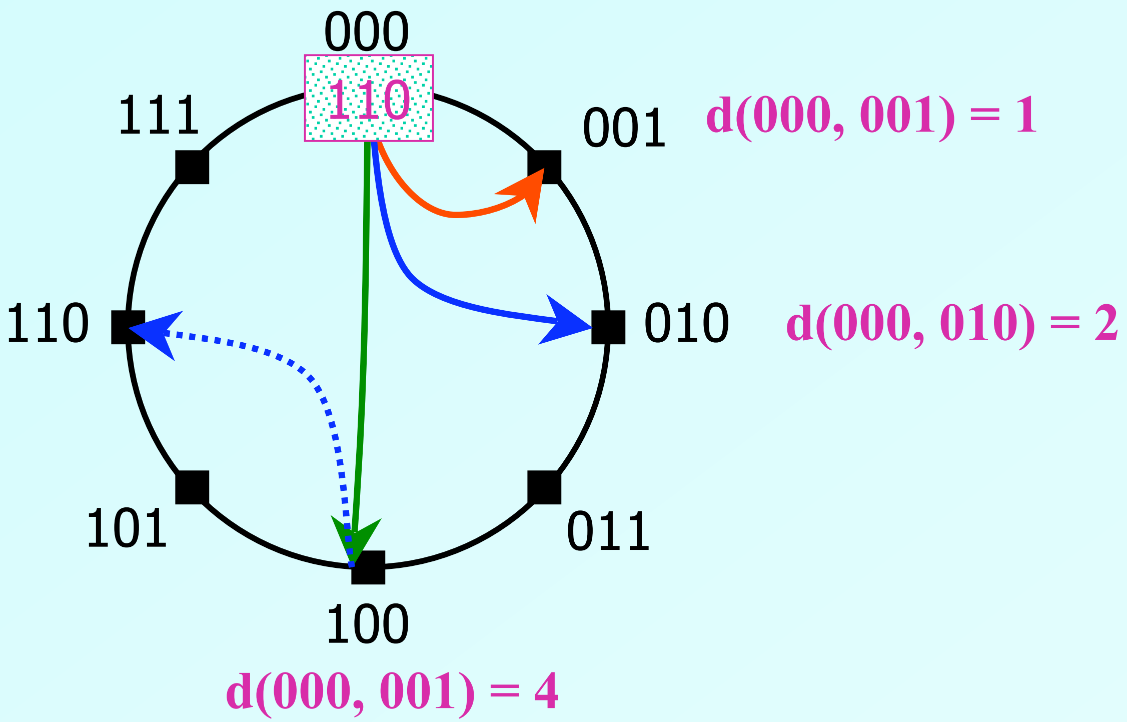
- **State Assignment:**
  - what "(*key, value*) tables" does a node store?
- **Network Topology:**
  - how does a node select its neighbors?
- **Routing Algorithm:**
  - which neighbor to pick while routing to a destination?
- **Various DHT algorithms make different choices**
  - Chord, Pastry, CAN,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

State Assignment in Chord DHT



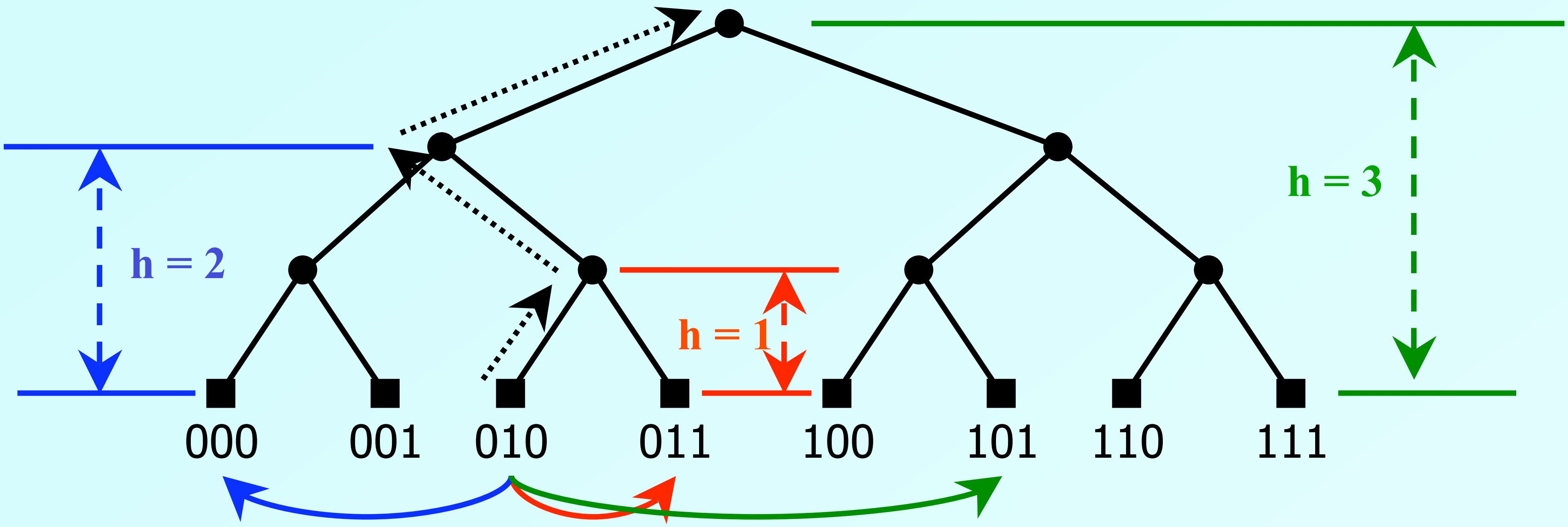
- **Nodes randomly chosen points on a Ring of *values***
- **Each node stores the *values* between itself and predecessor**

Chord Topology and Route Selection



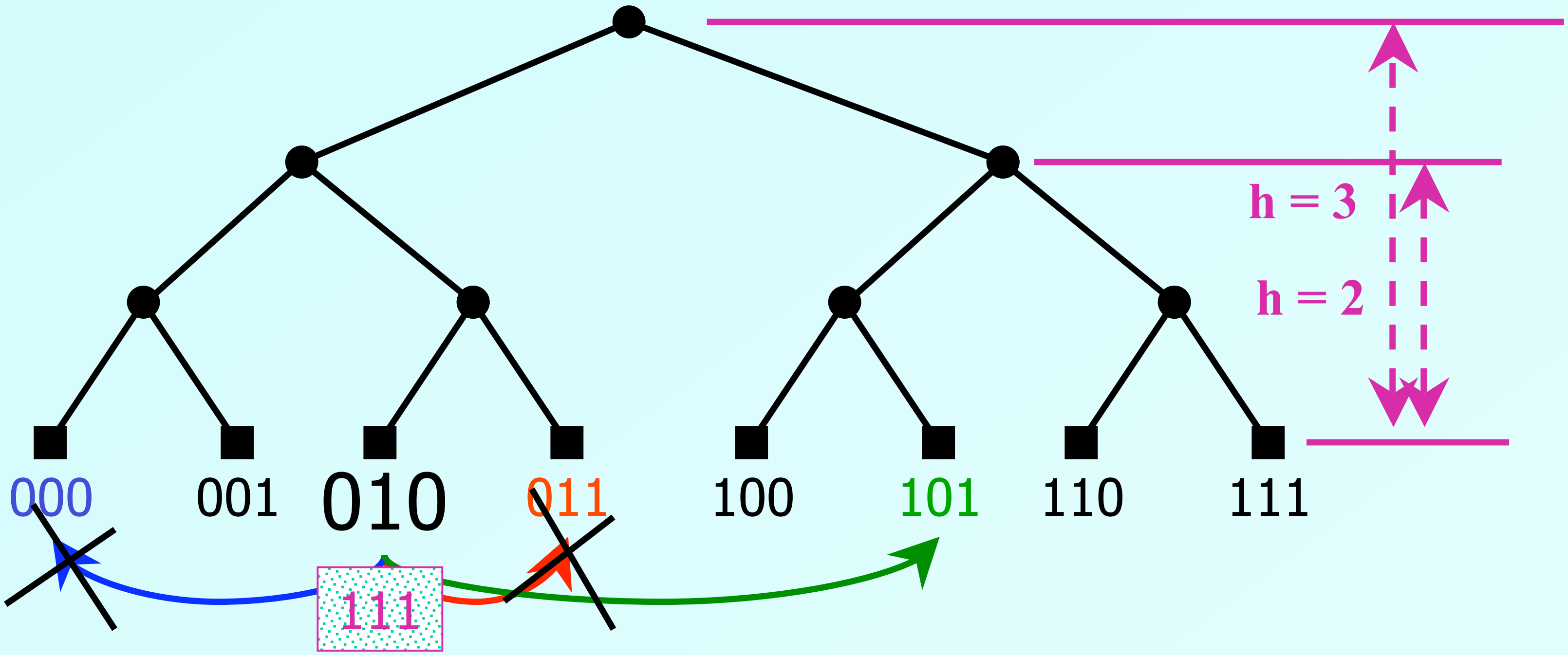
- **Neighbor selection: ith neighbor at 2^{i} distance**
- **Route selection: pick neighbor closest to destination**

State + Neighbor Assignment in Pastry



- **Nodes are leaves in a tree**
- **logN neighbors in sub-trees of varying heights**

Routing in Pastry



- **Route to the sub-tree with the destination**

Today's Discussion

- How do DHTs work?
- **What properties do DHTs have?**
- **What are P2P systems (as opposed to DHTs)?**
  - – Why are DHTs appealing to P2P designs

Properties of DHTs

- **Scalable**
  - each node has O(*logN*) neighbors
- **Efficient**
  - lookup takes *O(logN)* time
- **Completely decentralized and self-organizing**
  - hence highly available
- **Load balanced**
  - all nodes are equal

Are DHTs panacea for building Scalable Distributed Systems?

DHT's Achilles Heel: Heterogeneity

- **DHTs great building blocks for large scale homogeneous systems**
  - Each node has the same role
- **Building heterogeneous systems over DHTs is hard**
  - it often requires careful engineering of the DHT

Today’s Discussion

- How do DHTs work?
- What properties to DHT have?
- **What are P2P systems (as opposed to DHTs)?**
  - Why are DHTs appealing to P2P designs

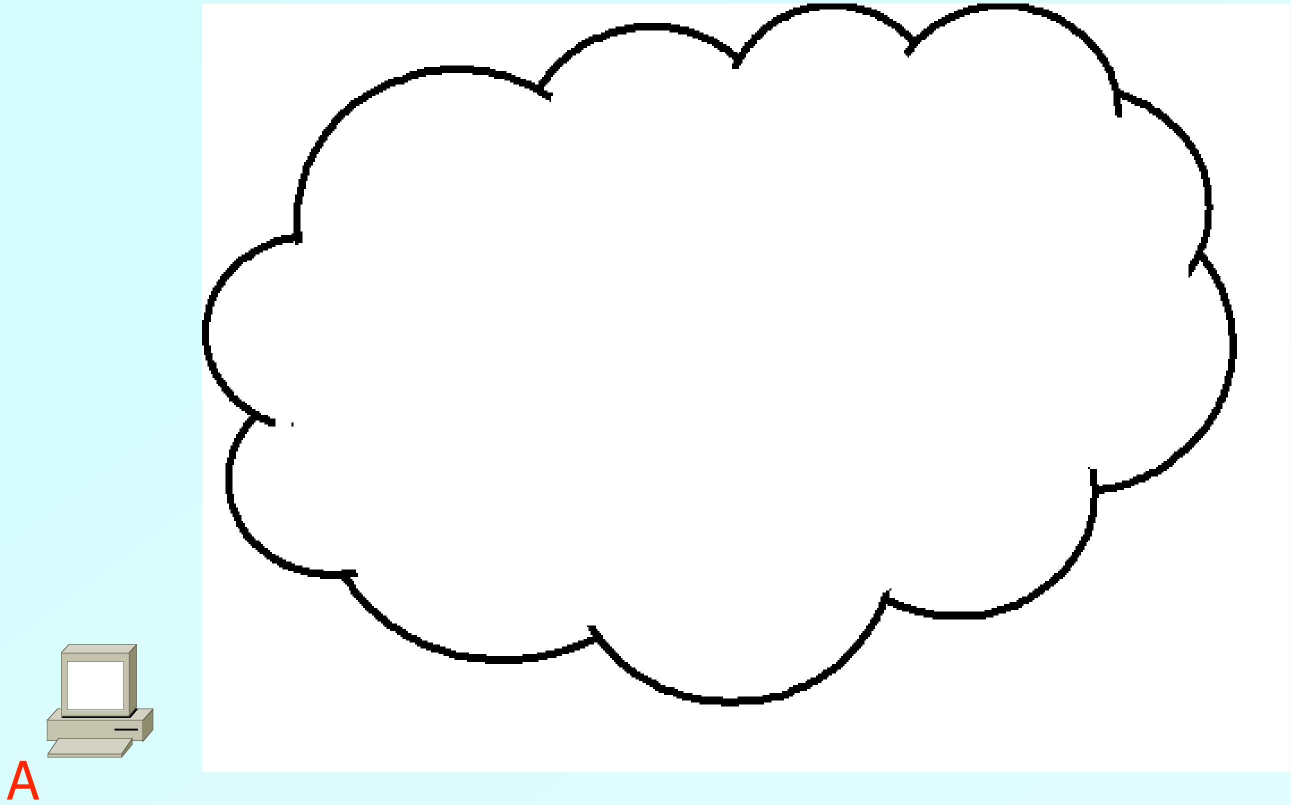
What are P2P systems?

- **Peer-to-Peer as opposed to Client-Server**
- **All participants in a system have uniform roles**
  - they act as clients, servers and routers
  - popular P2P apps: Seti@home, Kazaa, Napster
- **Technological trends favoring P2P**
  - client desktops have more storage, computation power and bandwidth
  - millions of clients connected to the Internet
- **P2P systems leverage the power of these clients**
  - Seti@home leverage computation power
  - Kazaa, Napster leverage bandwidth

Why are DHTs appealing to P2P System Designers?

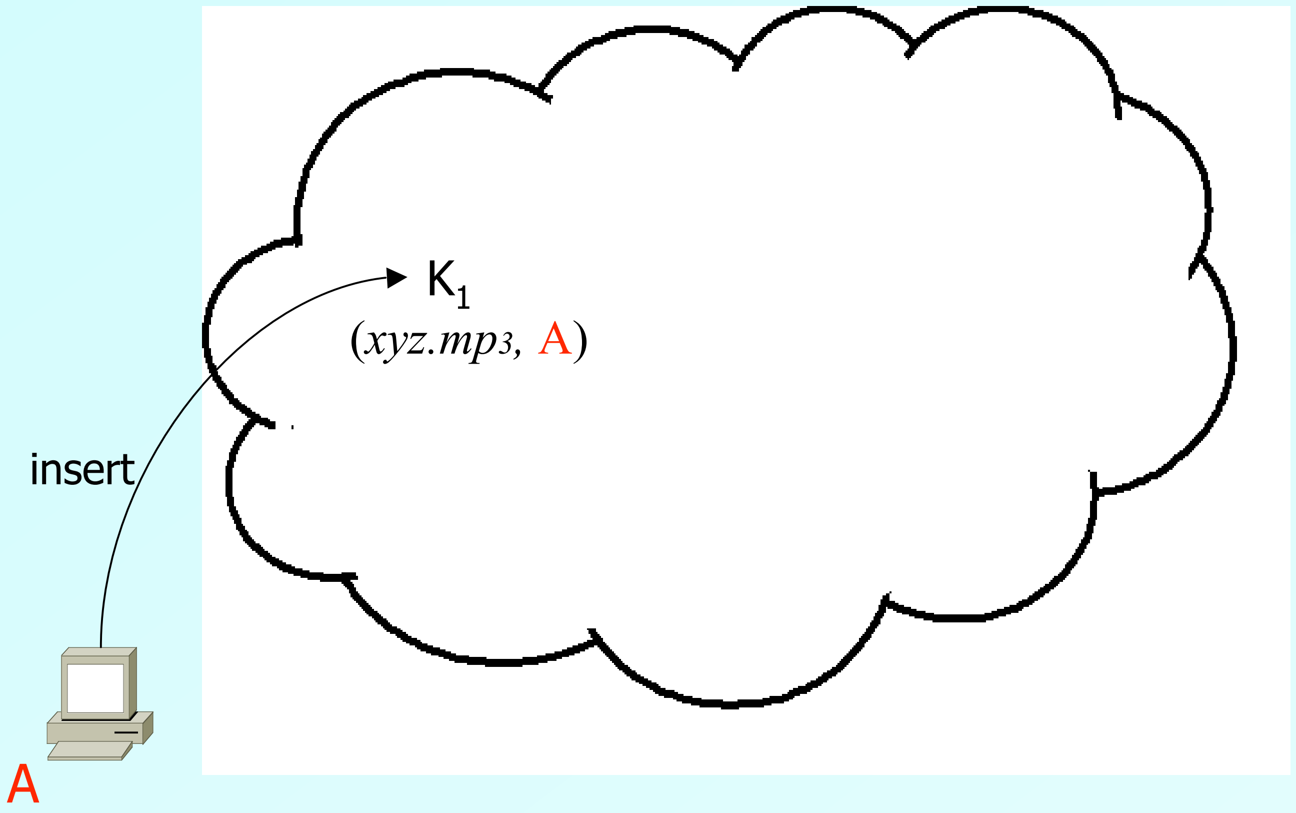
- **Scalable, Load-balanced and Decentralized, Self-organizing**
- **Content-Addressable**
  - Querying is the same as routing (getting to the content)
    - Query does not specify host
  - Internet is host-addressable

Content Addressability in a DHT



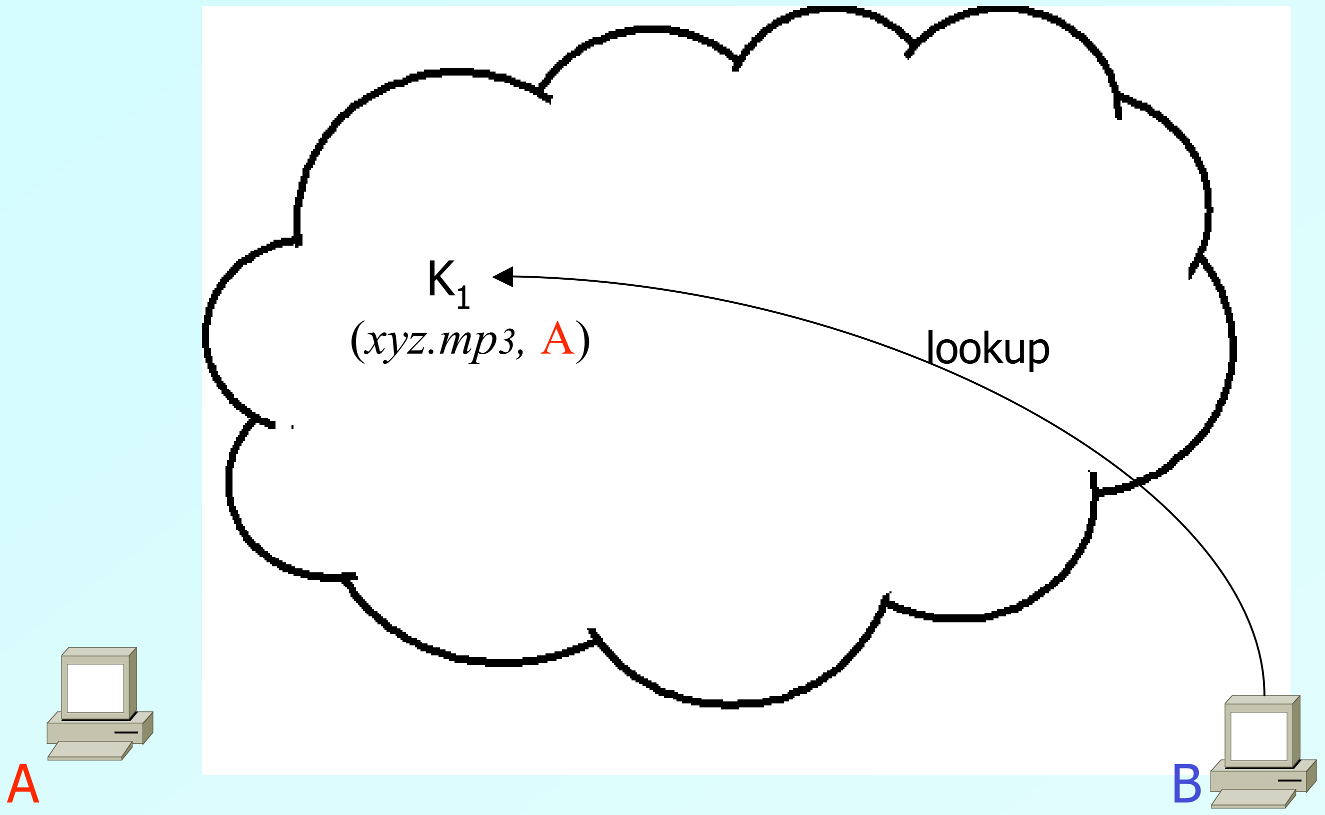
HASH(*xyz.mp3*) = K_1

Content Addressability in a DHT



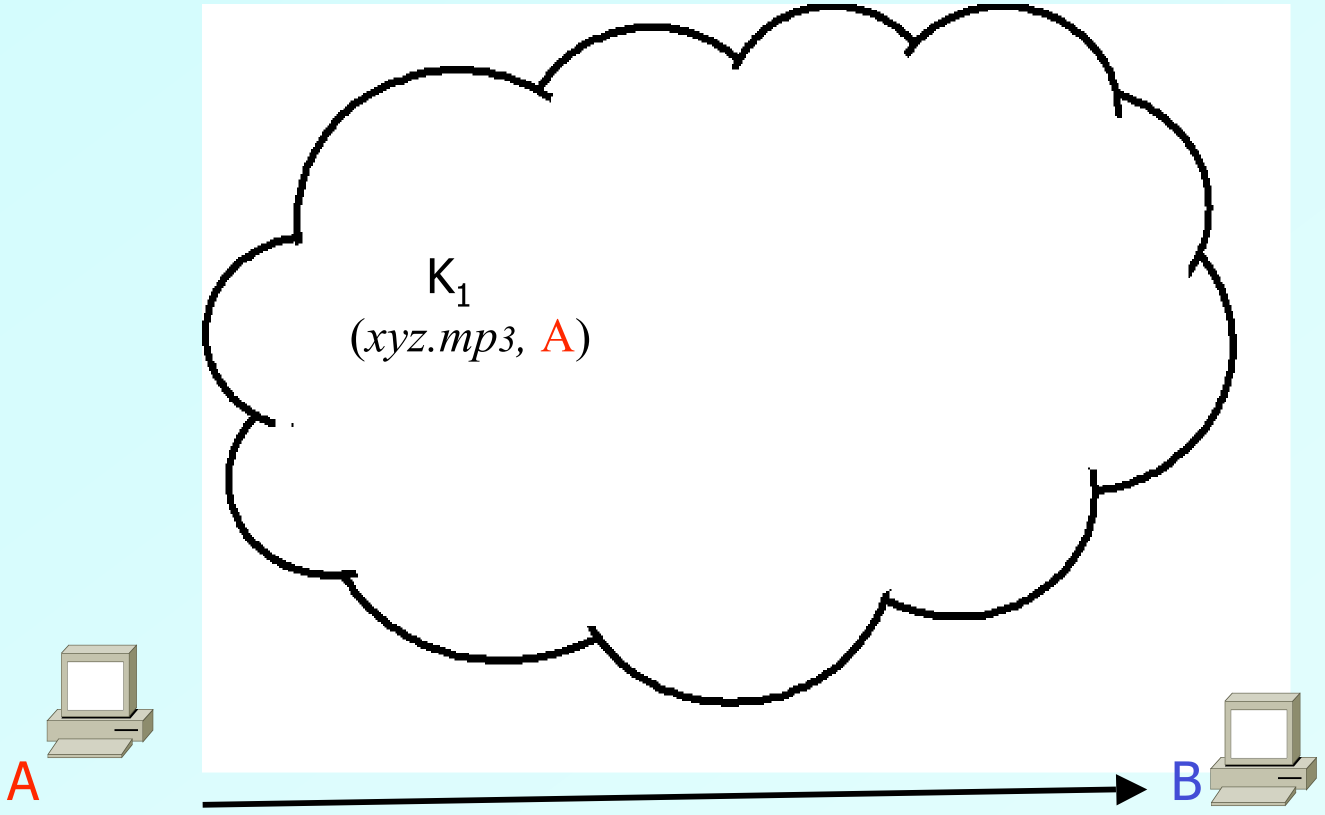
HASH(*xyz.mp3*) = K_1

Content Addressability in a DHT



HASH(*xyz.mp3*) = K_1

Content Addressability in a DHT



Content-addressability: key insight

- **Content-addressability provides a level of indirection between consumers and providers of content/service**

  "Any computer systems problem can be solved by adding a level of indirection"

- **Eliminates need for consumers to know providers & vice-versa**
  - allows a new raft of applications like anycast, multicast, service composition etc.,

Discussion

- **When facing new distributed system design, how do we determine whether DHTs are suitable?**

When should we use DHT?

- **Does system need to scale?**
- **Does system have heterogeneous nodes?**
- **Does system need self-organization?**
- **Does system need fully decentralized solution?**
- **Can system tolerate security risks due to decentralization?**
- **Does system need content addressability?**

The Good, The Bad and The Ugly Application of DHTs

- **The Good**
  - corporation wide file-systems
    - Farsite, GFS, LOCKSS
  - sensor networks and queries over them
    - Pier
  - corporate multicast, video-conferencing
    - Akamai, Scribe
- **The Bad**
  - Wide-area file-sharing
    - Overnet, DHT based Napster
- **The Ugly**
  - Internet wide file-systems, backups
    - CFS, Past, Ivy
  - collaborative spam filtering