CSC458

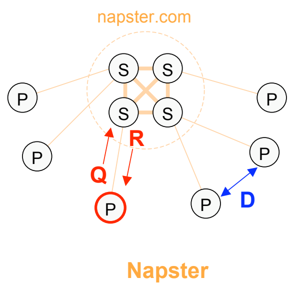
Peer-to-Peer Systems

This Time

- P2P systems
- Focus
  - How do P2P systems work?
- Topics
  - Unstructured P2Ps
  - Structured P2Ps: Distributed Hash Tables
  - Comparison of two distributed systems: WWW and a DHT (Chord)

Unstructured P2Ps

Napster (circa 2002)

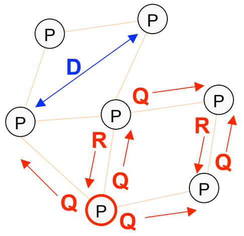


Napster Architecture

- Dedicated, well-provisioned servers
- Peers connect to one server
- Servers index the content of all peers
- To query:
  - Send query to server, wait for reply
- To download:
  - Over HTTP from one peer (uploader) to another (downloader)
  - No server involvement

- Bottom line: Napster was order to shut-down in 2002

Gnutella (circa 2002)



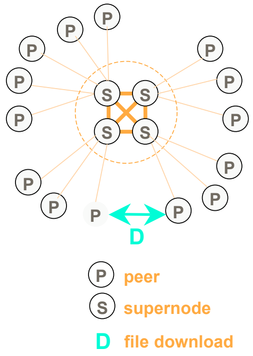
Gnutella

Gnutella ver. 0.4 Architecture

- No servers, only peers
- Upon joining, peer needs to know at least one other peer
  - Through out-of-band mechanisms
- Once joining one peer, send a Ping broadcast
  - Discover additional peers to connect to them <-- flooding!!
- To query:
  - Send a Query broadcast <-- flooding!!
- To download:
  - Over HTTP from one peer (uploader) to another (downloader)

- Bottom line: Unscalable protocol (based on flooding)

Kazaa (today)



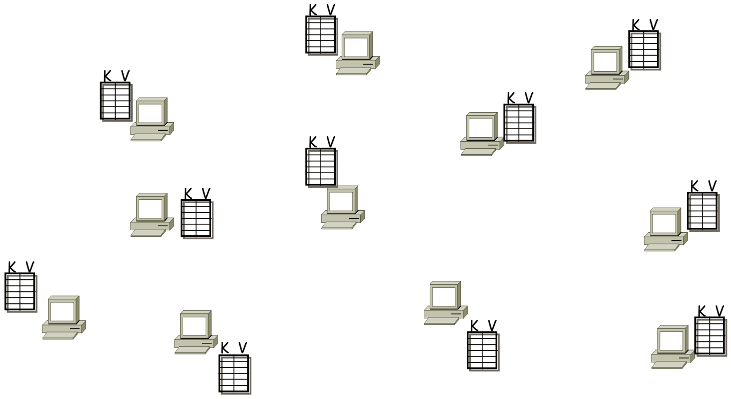
Kazaa Architecture

- Some peers act as servers (aka ultrapeers/supernodes)
- Upon joining, peer needs to know at least one supernode
  - Through out-of-band mechanisms
- Supernodes index the content of all their peers (like Napster)
- To query:
  - Send query to supernode, wait for reply
- To download:
  - Over HTTP from one peer (uploader) to another (downloader)
  - No supernode involvement

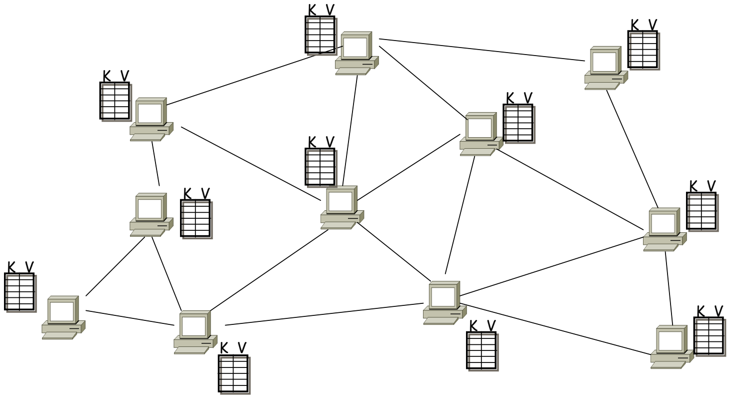
- Bottom line: more scalable architecture, servers are not dedicated!!

Structured P2Ps or Distributed Hash Tables (DHTs)

DHT: basic idea

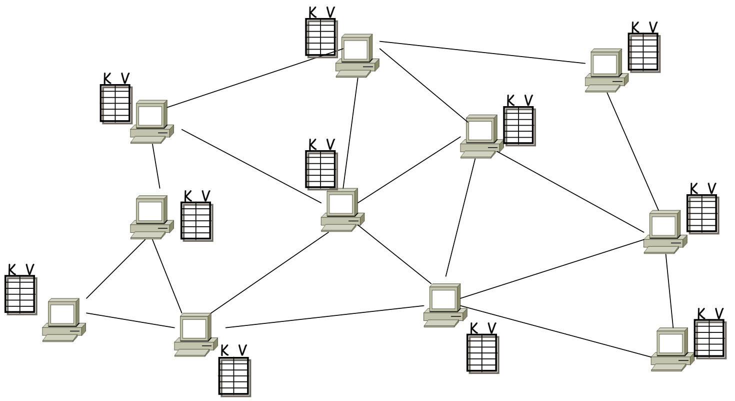


DHT: basic idea



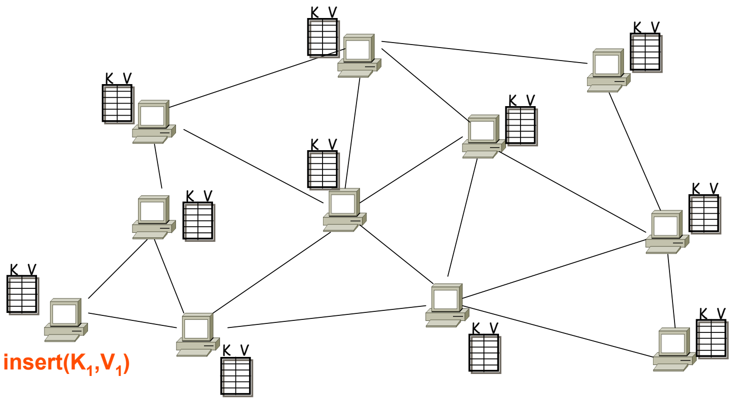
Neighboring nodes are "connected" at the application-level

DHT: basic idea



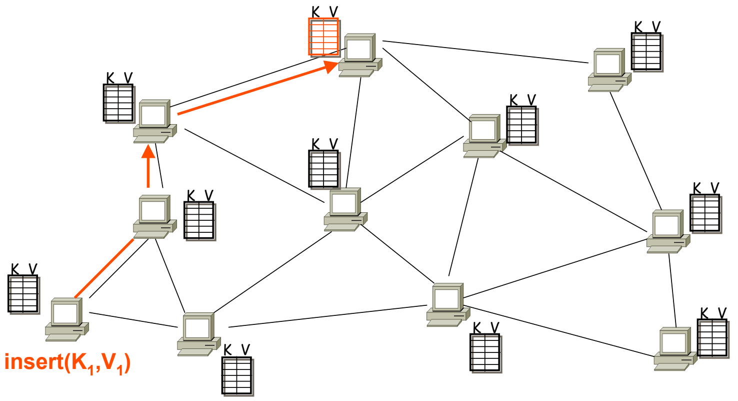
Operation: take *key* as input; route messages to node holding *key*

DHT: basic idea



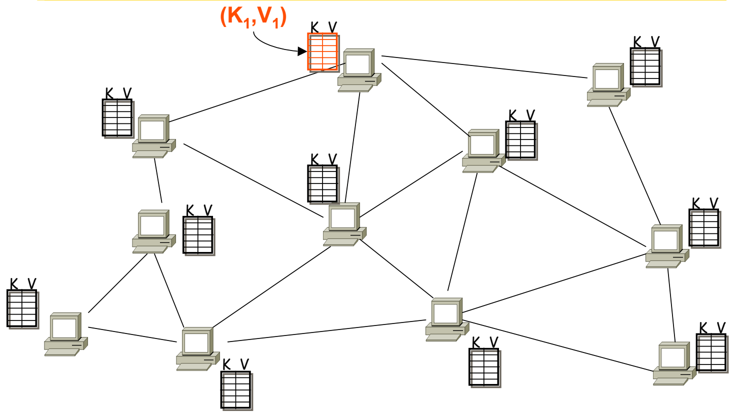
Operation: take *key* as input; route messages to node holding *key*

DHT: basic idea



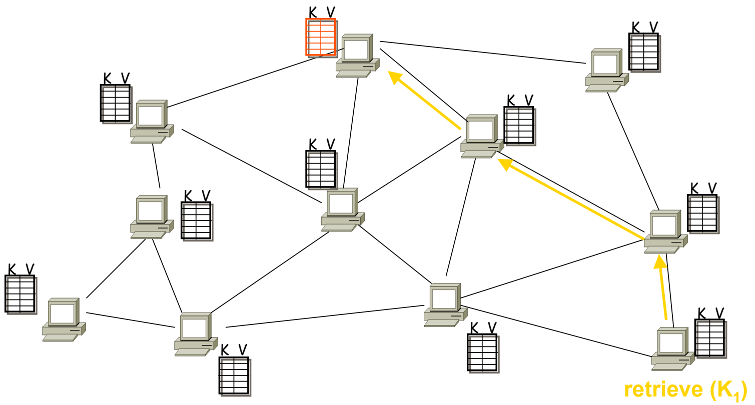
Operation: take *key* as input; route messages to node holding *key*

DHT: basic idea



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

DHT: basic idea



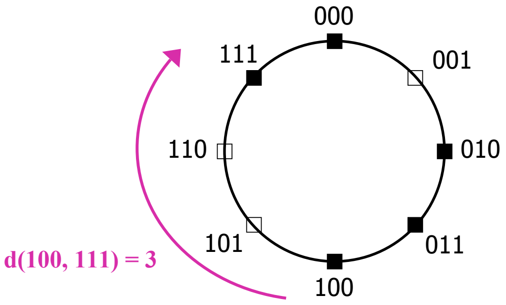
Operation: take *key* as input; route messages to node holding *key*

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?

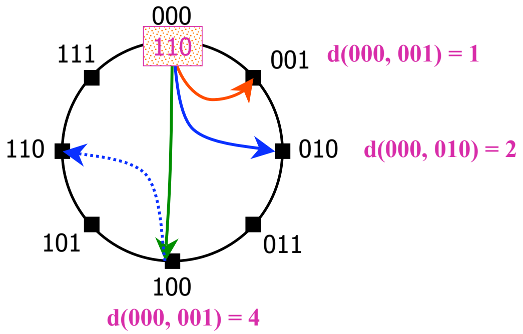
- Various DHT algorithms make different choices
  - Chord, CAN, Pastry, Tapestry, Plaxton, Viceroy, Kademlia, SkipNet, Symphony, Koorde, Apocrypha, Land, ORDI …

State Assignment in Chord DHT



- 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



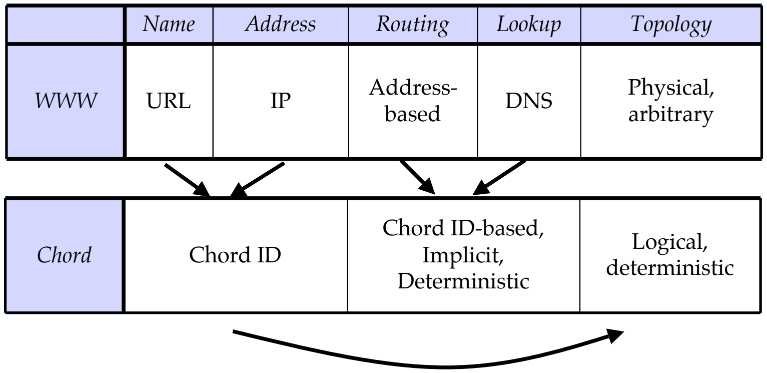
- Neighbor selection: i^{th} neighbor at 2^i distance
- Route selection: pick neighbor closest to destination

Chord Architecture

- N: number of peers in the system
- Each node has O(log n) neighbours
- Each query traverses O(log n) hops to the destination
- Each routing table has O(log n) size

- Very scalable!

What Chord style systems do:



	Name	*Address*	*Routing*	*Lookup*	*Topology*
WWW	URL	IP	Address-based	DNS	Physical, arbitrary
Chord	Chord ID		Chord ID-based, Implicit, Deterministic		Logical, deterministic

Lessons

- Unstructured peer-to-peer systems:
  - Napster: lookup centralized, file Xfer P2P
  - Gnutella: lookup + file Xfer P2P
  - Kazaa: lookup peer to supernode, file Xfer P2P
- DHTs:
  - Chord: lookup based on a distributed hash table
  - Very scalable architecture

- DHTs collapse naming and addressing
- DHTs collapse routing and lookup