Peer-to-Peer Systems: Unstructured

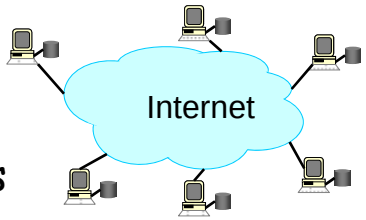
9/30/2009

Admin.

- Programming assignment 1 linked on the schedule page

Recap: Objectives of P2P

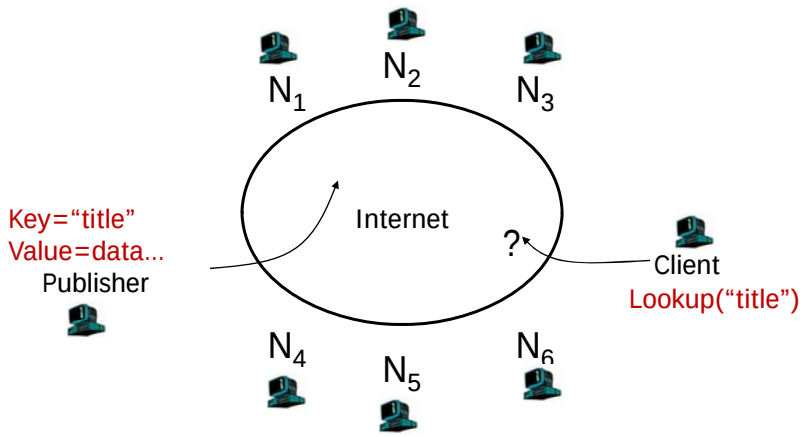
- Share the resources (storage *and* bandwidth) of individual clients to improve scalability/robustness



- Find clients with resources!

3

Recap: The Lookup Problem



find where a particular document is stored

Recap

- Napster (central query server)
- Gnutella (decentralized, flooding)
- Fast track: Modifies the Gnutella protocol into two-level hierarchy
  - Supernodes
    - Nodes that have better connection to Internet
    - Act as temporary indexing servers for other nodes
    - Help improve the stability of the network
  - Standard nodes
    - Connect to supernodes and report list of files
  - Search
    - Broadcast (Gnutella-style) search across supernodes

5

Outline

- Recap
- *P2P*
  - *the lookup problem*
    - Napster (central query server; distributed data server)
    - Gnutella (decentralized, flooding)
    - *Freenet*

6

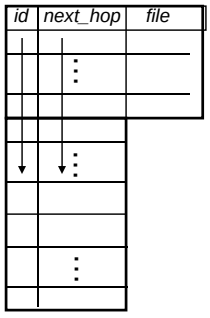
Freenet

- History
  - final year project Ian Clarke , Edinburgh University, Scotland, June, 1999
- Goals:
  - totally distributed system without using centralized index or broadcast (flooding), instead search by routing
  - routing/storing system responds adaptively to usage patterns, transparently moving, replicating files as necessary to provide efficient service
  - provide publisher anonymity, security
  - free speech : resistant to attacks – a third party shouldn't be able to deny (e.g., deleting) the access to a particular file (data item, object)

7

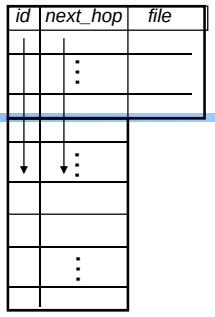
Basic Structure of Freenet

- Each machine stores a set of files; each file is identified by a unique identifier (called key or id)
- Each node maintains a "routing table"
  - *id* – file id, key
  - *next_hop node* – where to search for a file with id' that is similar to id
  - *file* – local copy, if exists, of file with id



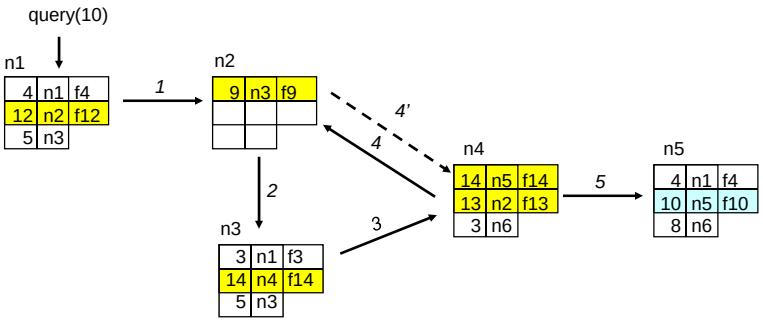
8

Freenet Query



- API: *file* = query(*id*);
- Upon receiving a query for file *id*
  - check whether the queried file is stored locally
  - check TTL to limit the search scope
    - each query is associated a TTL that is decremented each time the query message is forwarded
    - when TTL=1, the query is forwarded with a probability
    - TTL can be initiated to a random value (why random value?)
  - look for the "**closest**" *id* in the table with an unvisited next_hop node
    - if found one, forward the query to the corresponding *next_hop*
    - otherwise, backtrack
      - ends up performing a Depth First Search (DFS)-like traversal
      - search direction ordered by closeness to target
- When file is returned it is cached along the reverse path (any advantage?)

Query Example



Beside the routing table, each node also maintains a query table containing the state of all outstanding queries that have traversed it → to backtrack

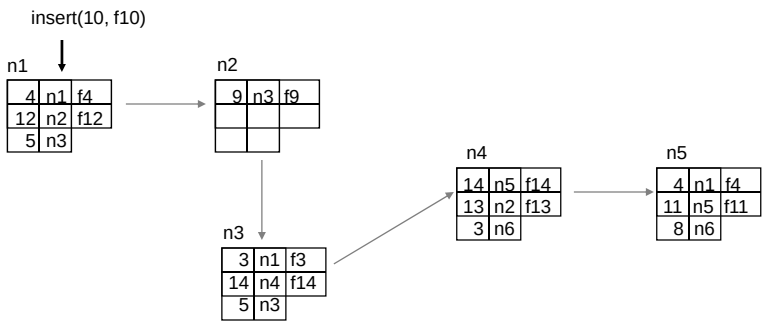
Insert

- API: insert(*id*, *file*);
- Two steps
  - first attempt a "search" for the file to be inserted
  - if found, report collision
  - if not found, insert the file by sending it along the query path (why?)
    - a node probabilistically replaces the originator with itself (why?)

11

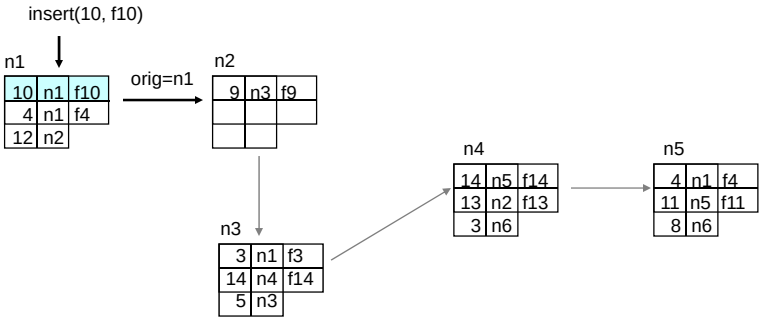
Insert Example

- Assume query returned failure along the shown path (backtrack slightly complicate things); insert f10



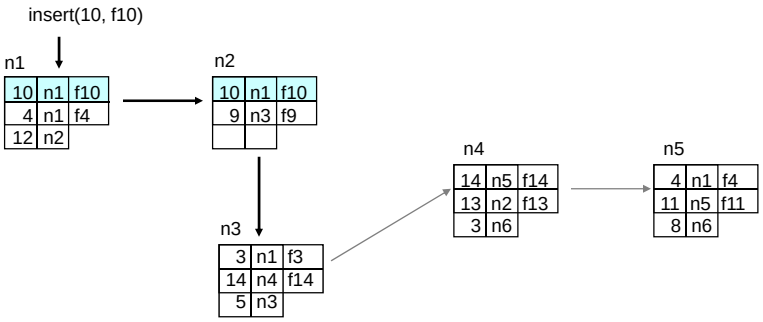
12

Insert Example



13

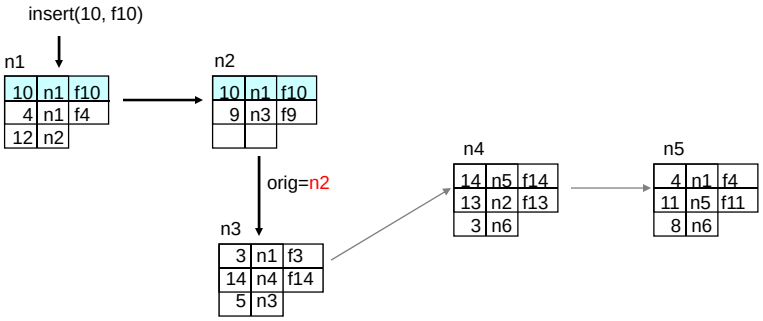
Insert Example



14

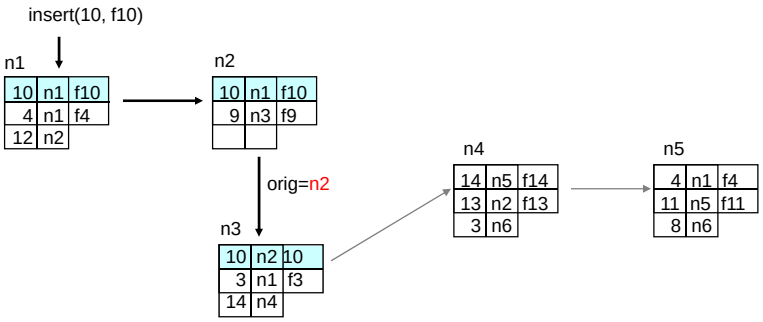
Insert Example

- n2 replaces the originator (n1) with itself



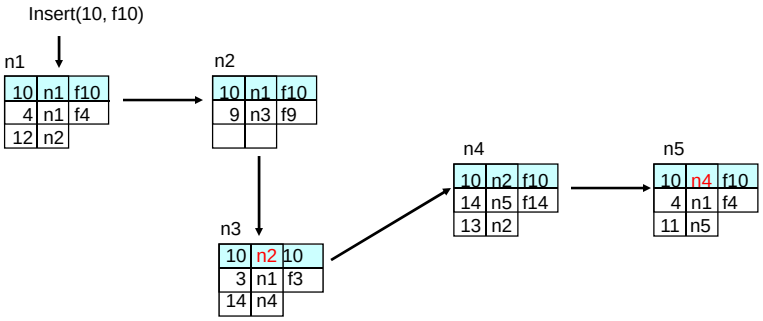
15

Insert Example



16

Insert Example



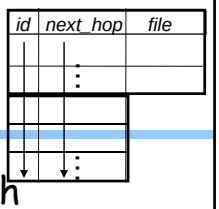
17

Freenet Analysis

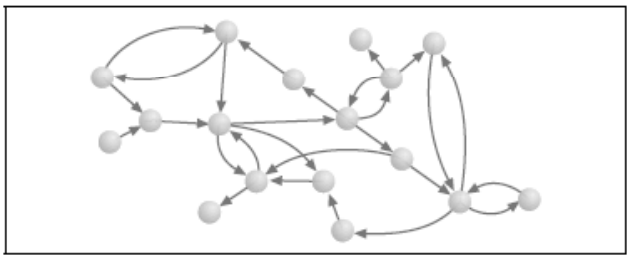
- Authors claim the following effects:
  - nodes eventually specialize in locating similar keys
    - if a node is listed in a routing table, it will get queries for related keys
    - thus will gain "experience" answering those queries
  - popular data will be transparently replicated and will exist closer to requestors
  - as nodes process queries, connectivity increases
    - nodes will discover other nodes in the network
- Caveat: lexigraphic closeness of file names/keys may not imply content similarity

18

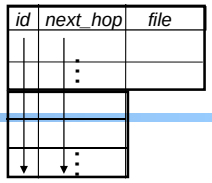
Understanding Freenet Self-Organization: Freenet Graph

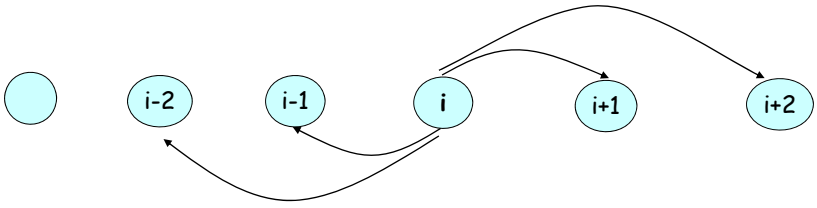


- We create a Freenet reference graph
  - creating a vertex for each Freenet node
  - adding a directed link from A to B if A refers to an item stored at B



Experiment: Freenet Graph: Init

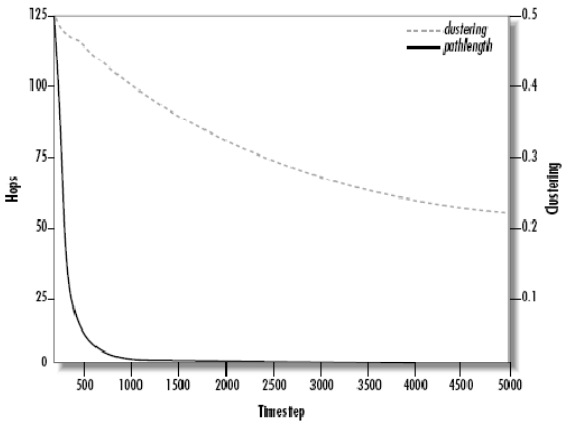




- Assume a network of 1000 nodes, with node id 0 to 999
- Each node can store 50 data items, and 200 references
- Assume initially each node i has i, and knows the storage of i – 2, -1, i + 1, i + 2 (all mod 1000)
- thus a regular, locally-clustered graph with avg path length ~ 1000 / 8 = 125

Experiment: Evolution of Freenet Graph

- At each step
  - pick a node randomly
  - flip a coin to determine search or insert
    - if search, randomly pick a key in the network
    - if insert, pick a random key

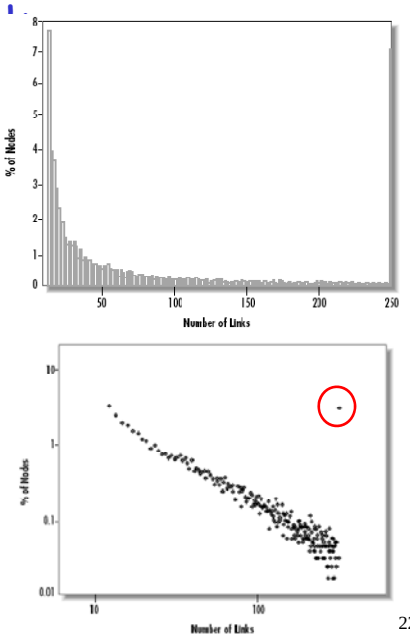


Evolution of path length and clustering; Clustering is defined as percentage of local links

21

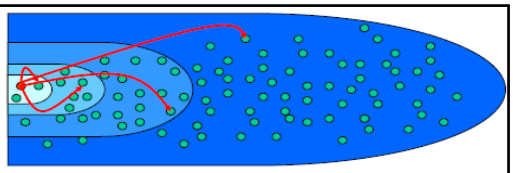
Freenet Evolves to Small-World Networ

- With usage, the regular, highly localized Freenet network evolved into one irregular graph
- High percentage of highly connected nodes provide shortcuts/bridges
  - make the world a "small world"
  - most queries only traverse a small number of hops to find the file



22

Small-World

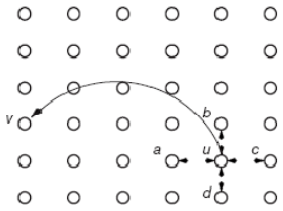


- First discovered by Milgrom
  - in 1967, Milgram mailed 160 letters to a set of randomly chosen people in Omaha, Nebraska
  - goal: pass the letters to a given person in Boston
    - each person can only pass the letter to an intermediary known on a first-name basis
    - pick the person who may make the best progress
  - result: 42 letters made it through !
  - median intermediaries was 5.5---thus six degree of separation
  - a potential explanation: highly connected people with non-local links in mostly locally connected communities improve search performance !

23

Kleinberg's Result on Distributed Search

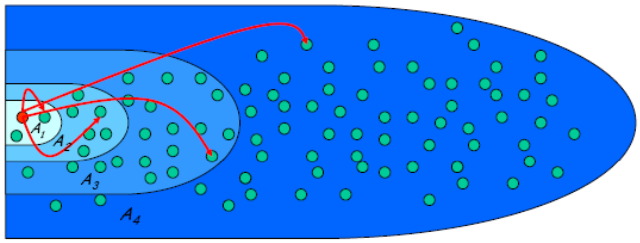
- Question: how many long distance links to maintain so that distributed (greedy) search is effective?
- Assume that the probability of a long link is some (α) inverse-power of the number of lattice steps
- Kleinberg's Law: Distributed algorithm exists only when probability is proportional to (lattice steps)$^{-d}$, where d is the dimension of the space



24

Distributed Search

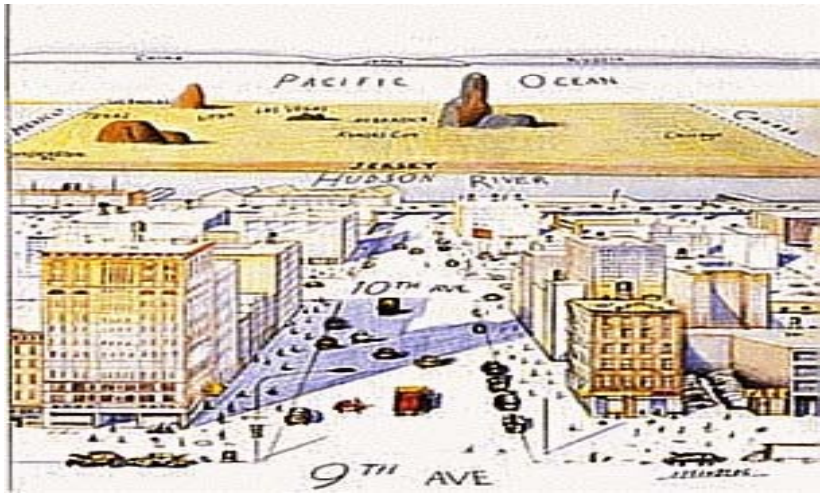
- In other words, if double distance, increase number of neighbors by a constant
  -> see Chord



probability is proportional to (lattice steps)$^{-d}$

25

Small World



Saul Steinberg; View of World from 9th Ave

26

Freenet: Properties

- Query using intelligent routing
  - decentralized architecture → robust
  - avoid flooding → low overhead
  - DFS search guided by closeness to target
- Integration of query and caching makes it
  - adaptive to usage patterns: reorganize network reference structure
  - free speech: attempts to discover/supplant existing files will just spread the files !
- Provide publisher anonymity, security
  - each node probabilistically replaces originator with itself

27

Freenet: Issues

- Does not always guarantee that a file is found, even if the file is in the network
- Good average-case performance, but a potentially long search path in a large network
  - approaching small-world…

28

Summary

- All of the previous p2p systems are called unstructured p2p systems
- Advantages of unstructured p2p
  - algorithms tend to be simple
  - can optimize for properties such as locality
- Disadvantages
  - hard to make performance guarantee
  - failure even when files exist

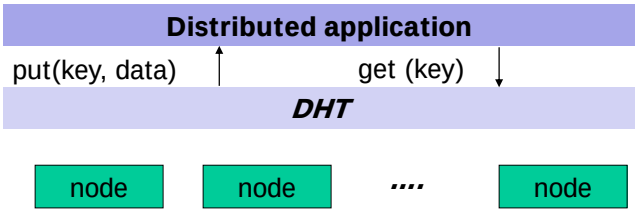
29

Distributed Hash Tables (DHT): History

- In 2000-2001, academic researchers jumped on to the P2P bandwagon
- Motivation:
  - frustrated by popularity of all these "half-baked" P2P apps. We can do better! (so they said)
  - guaranteed lookup success for data in system
  - provable bounds on search time
  - provable scalability to millions of node

DHT: Overview

- Abstraction: a distributed "hash-table" (DHT) data structure
  - put(key, value) and get(key) → value
  - DHT imposes no structure/meaning on keys
  - one can build complex data structures using DHT
- Implementation:
  - nodes in system form an interconnection network: ring, zone, tree, hypercube, butterfly network, ...



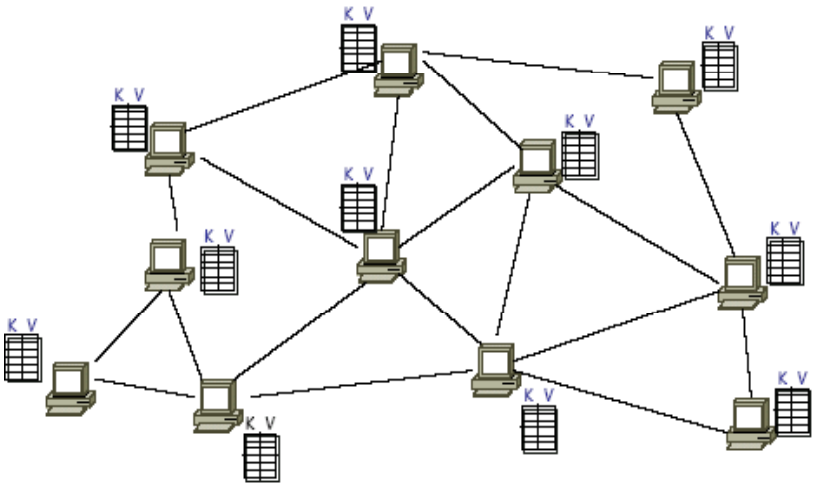
31

DHT Applications

- File sharing and backup [CFS, Ivy, OceanStore, PAST, Pastiche ...]
- Web cache and replica [Squirrel, Croquet Media Player]
- Censor-resistant stores [Eternity]
- DB query and indexing [PIER, Place Lab, VPN Index]
- Event notification [Scribe]
- Naming systems [ChordDNS, Twine, INS, HIP]
- Communication primitives [I3, ...]
- Host mobility [DTN Tetherless Architecture]

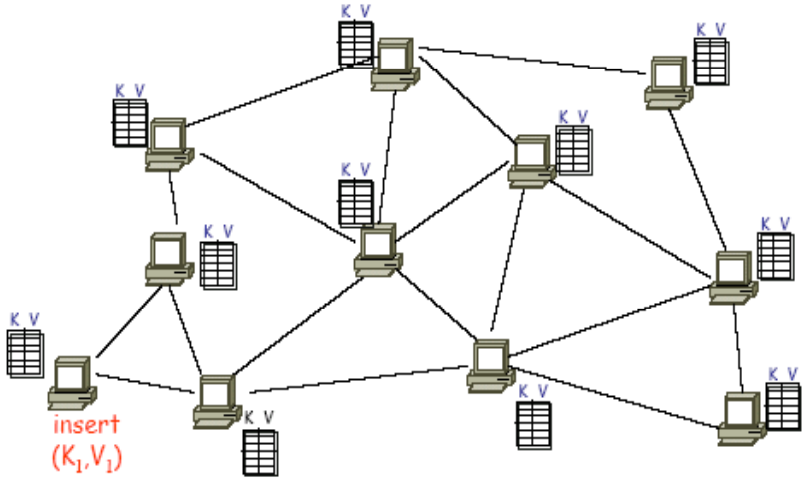
32

DHT: Basic Idea



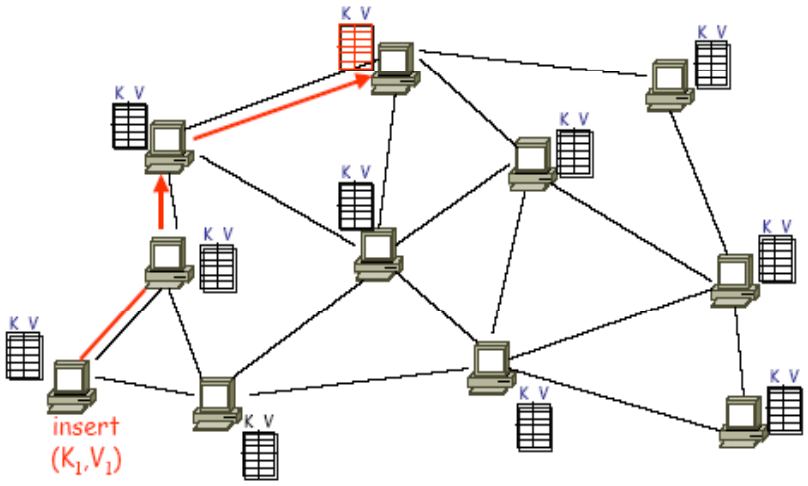
33

DHT: Basic Idea (2)



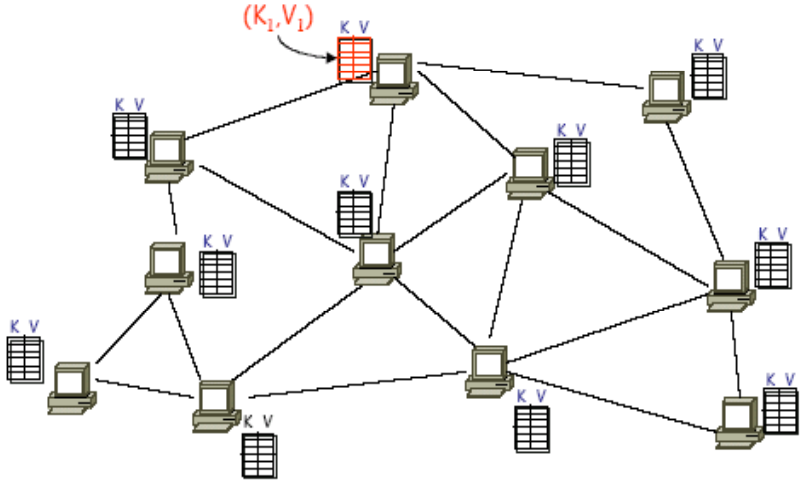
34

DHT: Basic Idea (3)



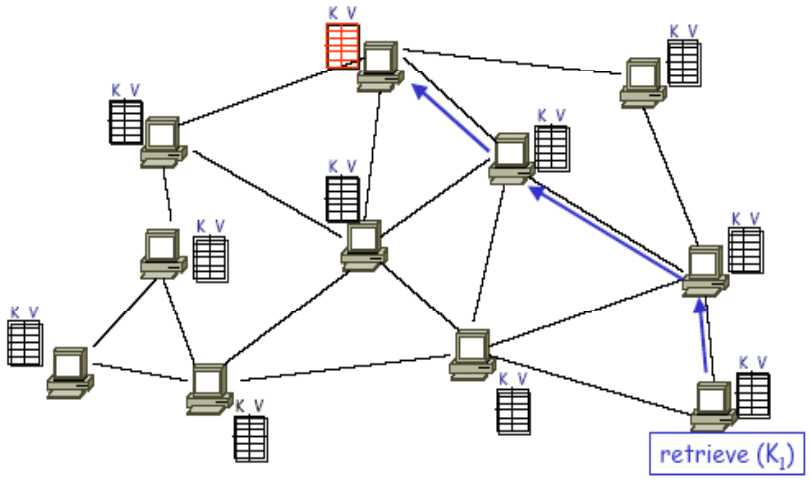
35

DHT: Basic Idea (4)



36

DHT: Basic Idea (5)



37