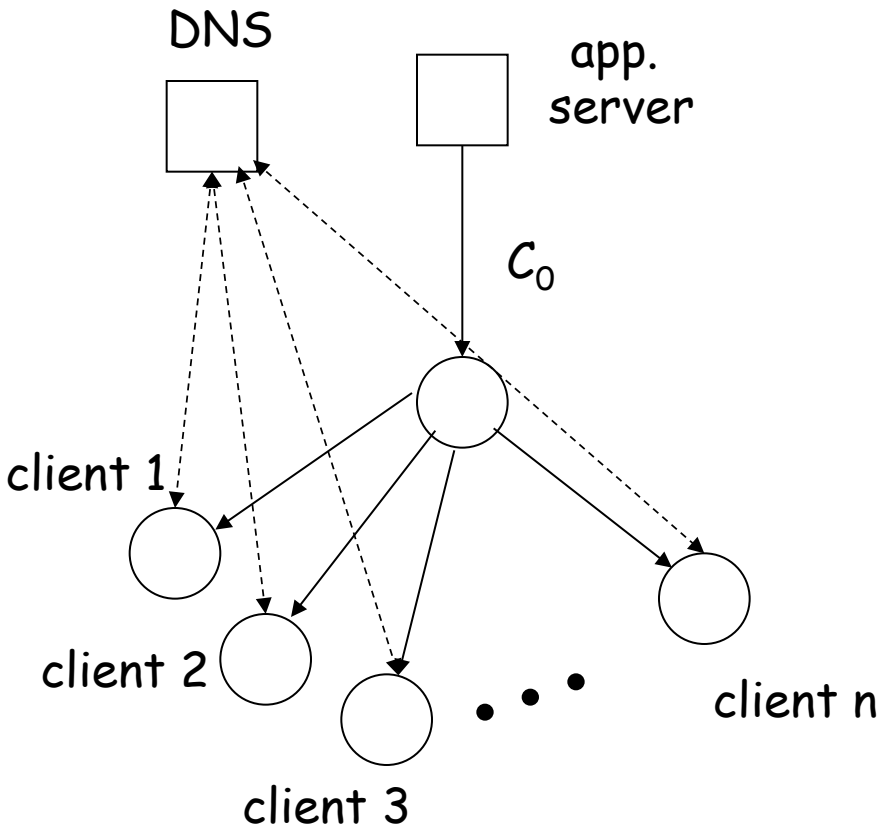
Network Applications: P2P Applications

Summary of Traditional C-S Network Applications

- How does a client locate a server?
- Is the application extensible, robust, scalable?

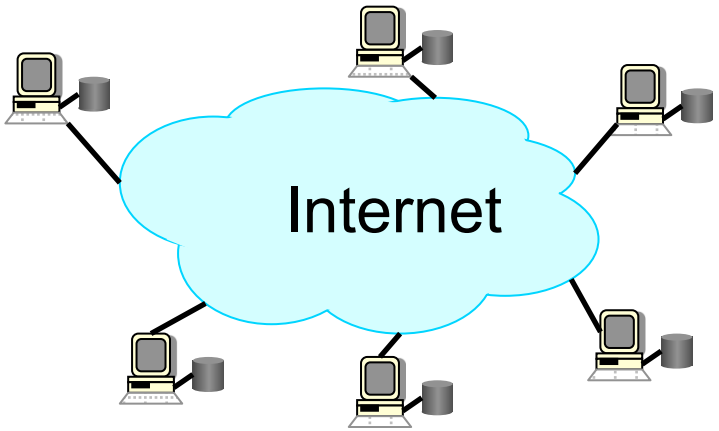


down speed to the clients?

slashdot effect, CNN on 9/11

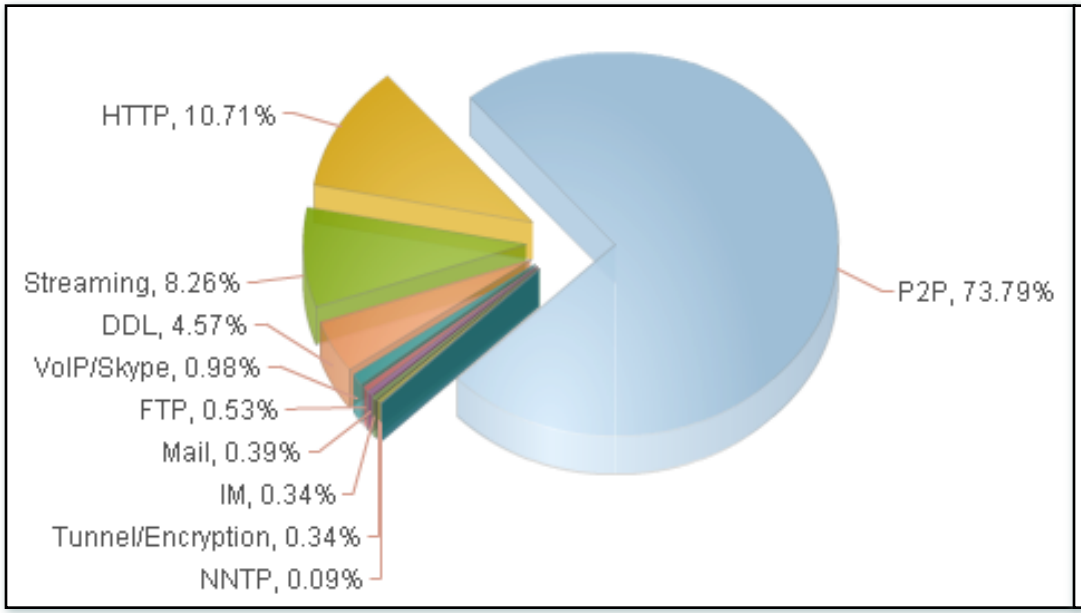
Objectives of P2P

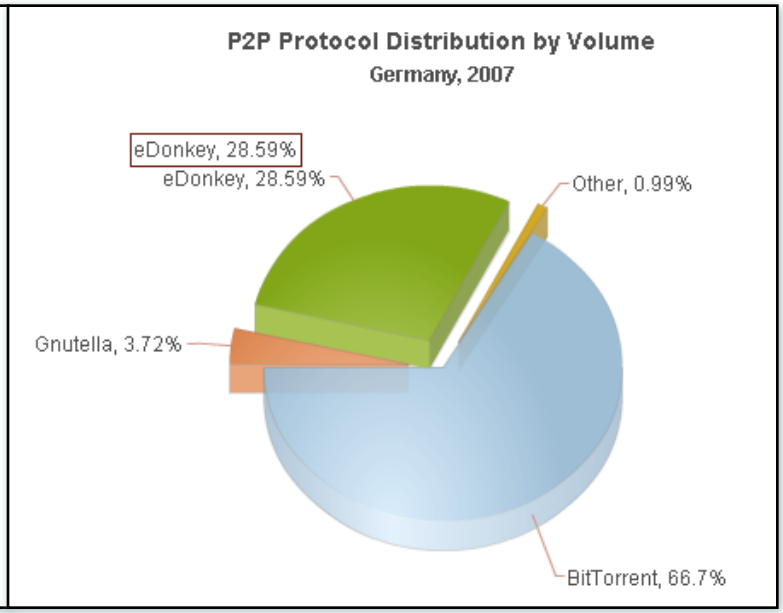
- Bypass DNS to access resources!
  - Examples: instant messaging, skype
- Share the storage *and* bandwidth of individual clients to improve scalability
  - Examples: file sharing and streaming



Peer-to-Peer Computing

- Quickly grown in popularity:
  - Dozens or hundreds of file sharing applications
  - 50-80% Internet traffic is P2P
  - Upset the music industry, drawn college students, web developers, recording artists and universities into court





From ipoque web site; Nov. 2007

What is P2P?

- But P2P is not new and is probably here to stay
- Original Internet was a P2P system:
  - The original ARPANET connected UCLA, Stanford Research Institute, UCSB, and Univ. of Utah
  - No DNS or routing infrastructure, just connected by phone lines
  - Computers also served as routers
- P2P is simply an iteration of scalable distributed systems

P2P Systems

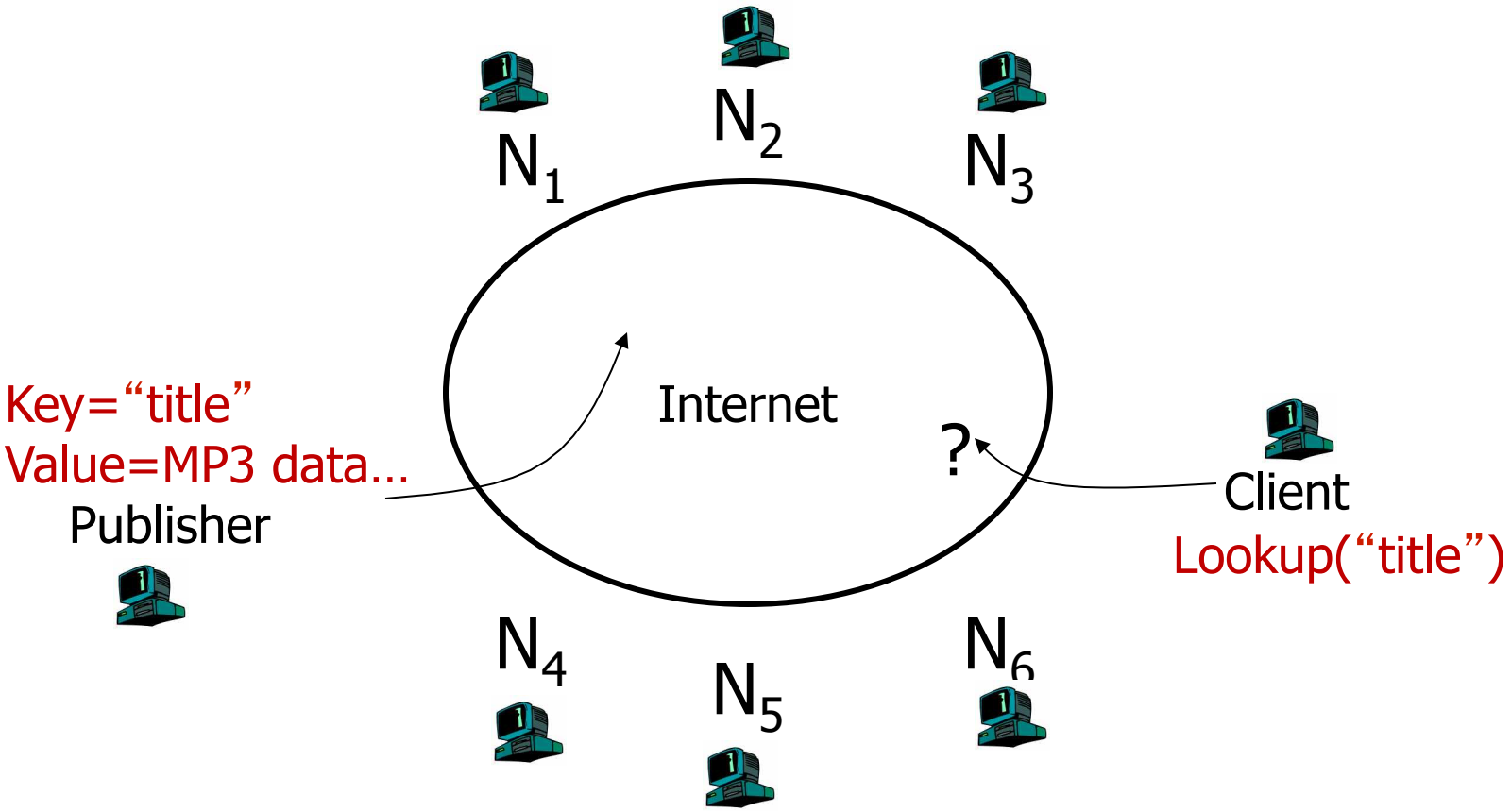
- File Sharing: BitTorren, LimeWire, eMule, eDonkey
- Streaming: PPLive, PPStream, …
- Research systems
  - Distributed Hash Tables
  - Content distribution networks

- Collaborative computing:
  - SETI@Home project (screen saver)
  - Human genome mapping
  - Intel NetBatch: 10,000 computers in 25 worldwide sites for simulations, saved about 500 million

Outline

- P2P
  - the lookup problem

The Lookup Problem



find where a particular file is stored

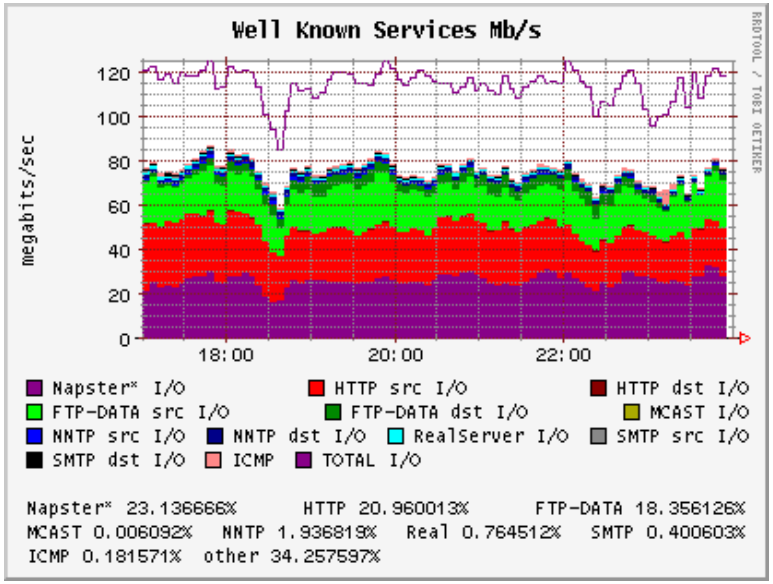
pay particular attention to see its equivalence of DNS

Outline

- P2P
  - The lookup problem
    - Napster

Centralized Database: Napster

- Program for sharing music over the Internet
- History:
  - 5/99: Shawn Fanning (freshman, Northeasten U.) founded Napster Online music service, wrote the program in 60 hours
  - 12/99: first lawsuit
  - 3/00: 25% UWisc traffic Napster
  - 2000: est. 60M users
  - 2/01: US Circuit Court of Appeals: Napster knew users violating copyright laws
  - 7/01: # simultaneous online users: Napster 160K
  - 9/02: bankruptcy



03/2000

We are referring to the Napster before closure.

Napster: How Does it Work?

Application-level, client-server protocol over TCP

A centralized index system that maps files (songs) to machines that are alive and with files

Steps:

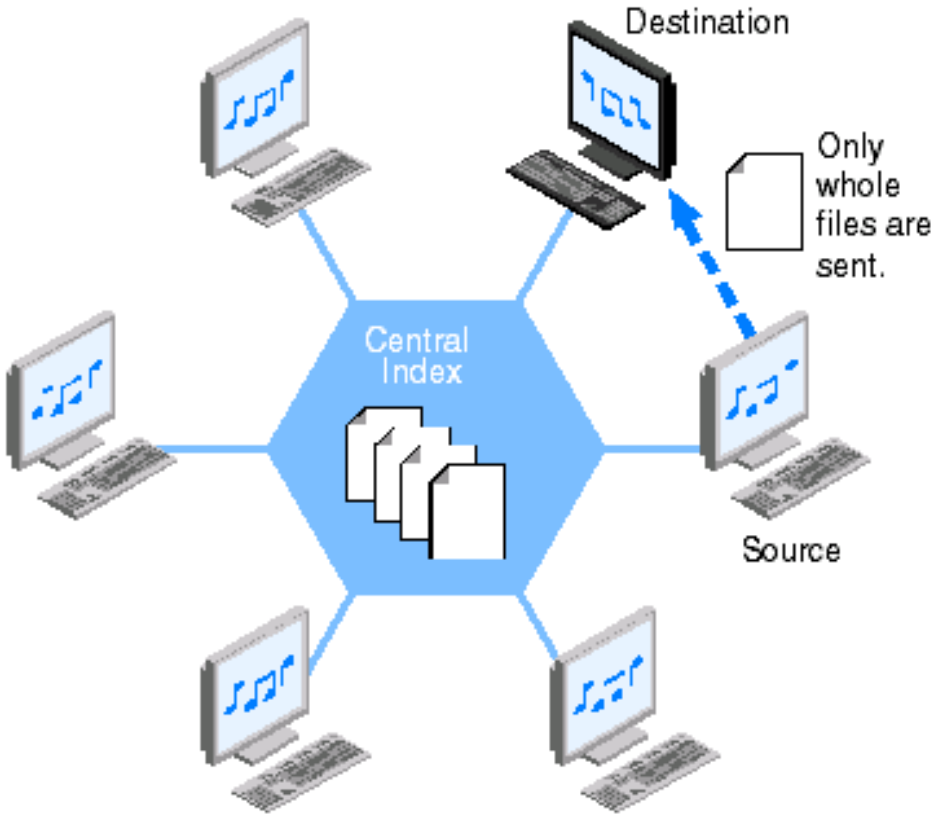
- Connect to Napster server
- Upload your list of files (push) to server
- Give server keywords to search the full list
- Select "best" of hosts with answers

Napster Architecture

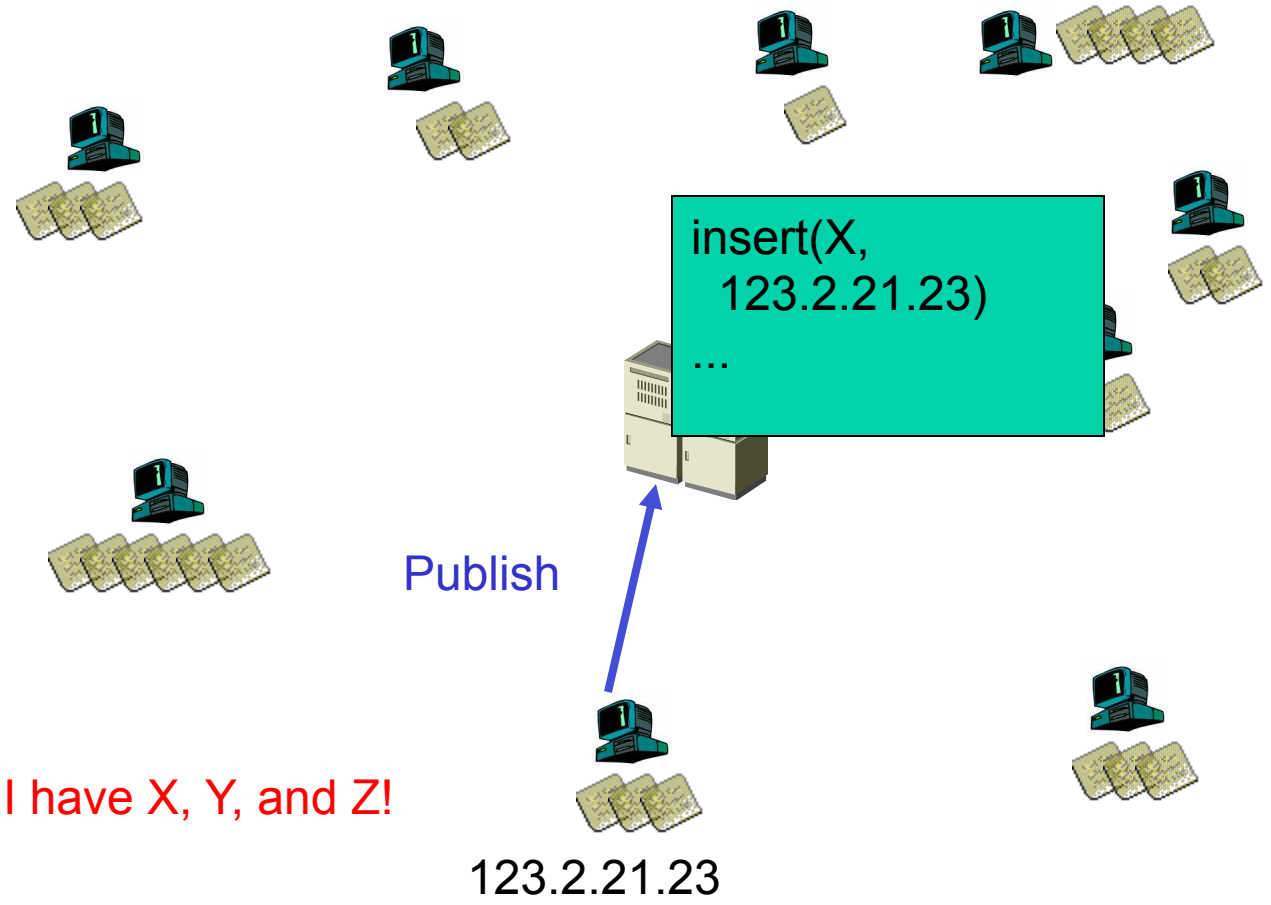
From Computer Desktop Encyclopedia
© 2004 The Computer Language Co. Inc.

THE ORIGINAL NAPSTER

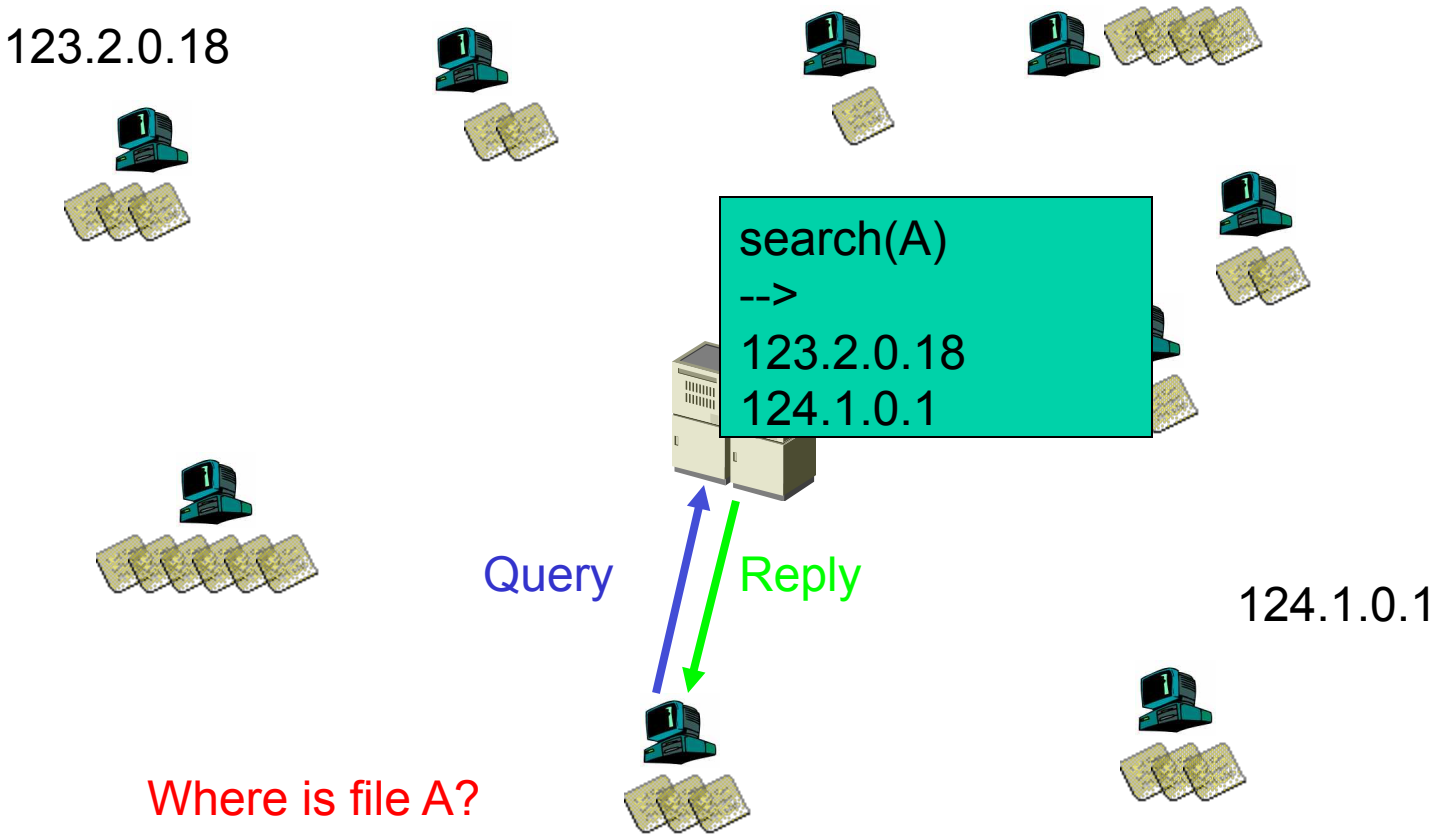
Napster provided a central directory of users who had files to share.



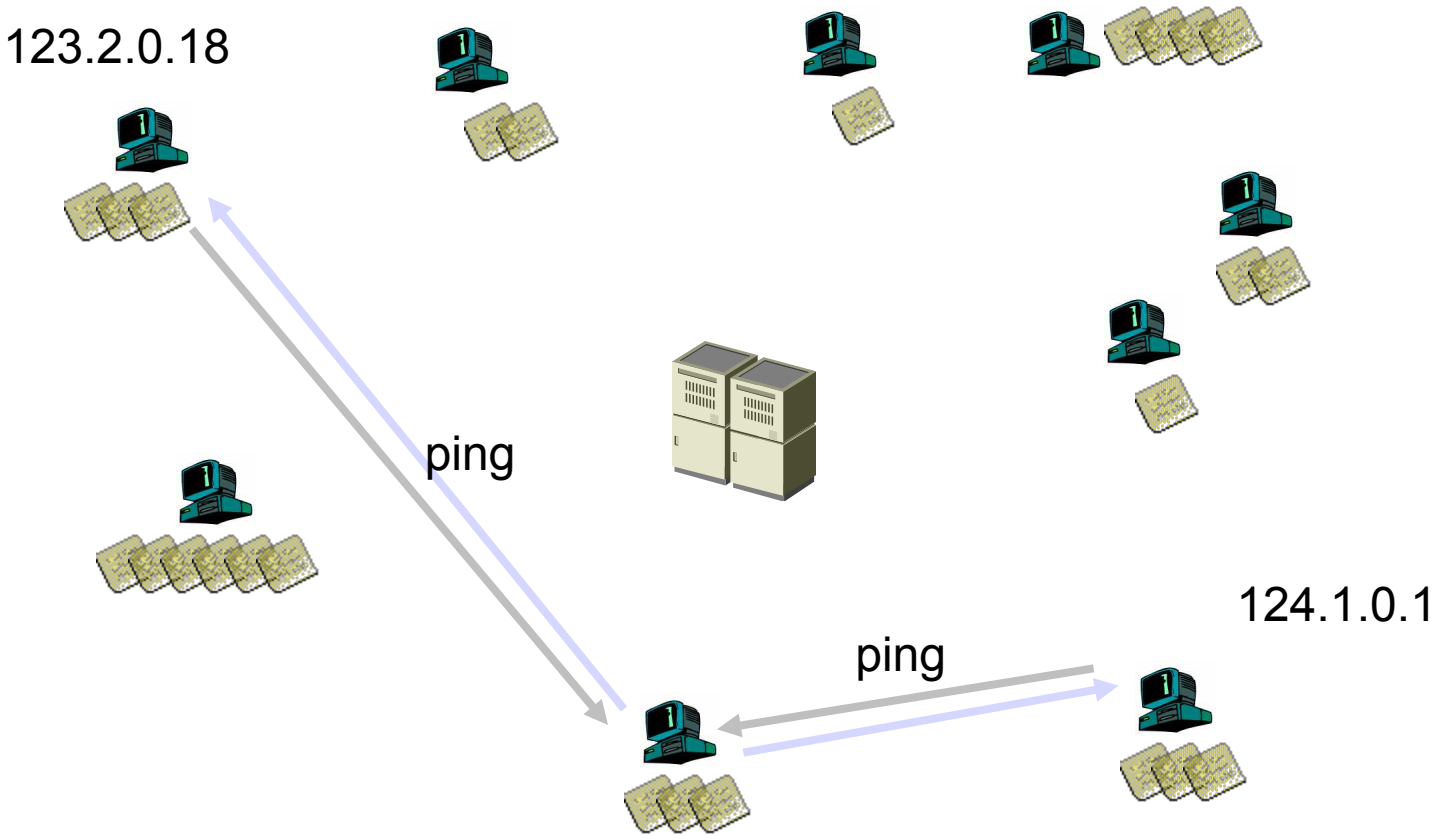
Napster: Publish



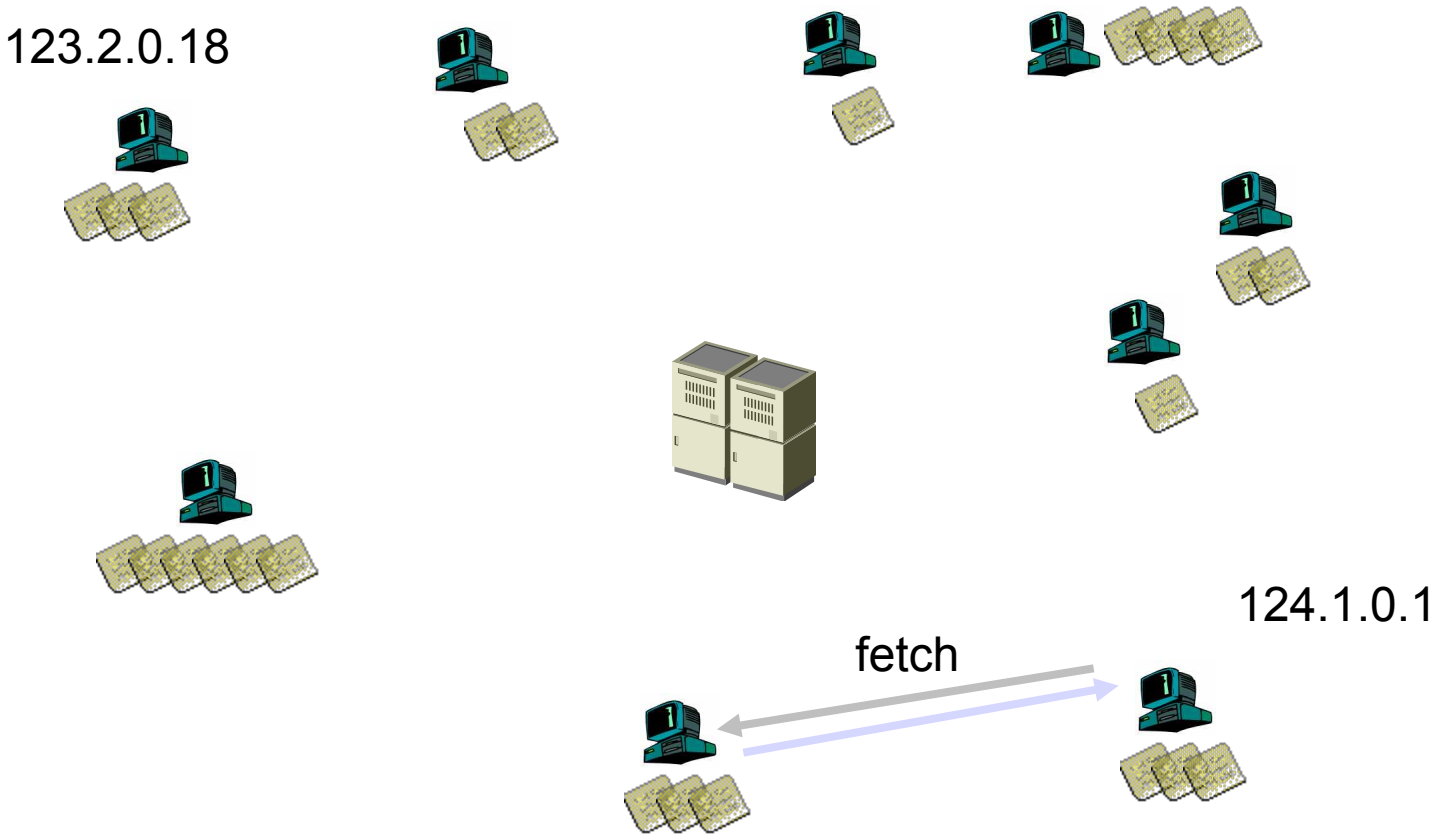
Napster: Search



Napster: Ping



Napster: Fetch



Napster Messages

General Packet Format

[chunksize] [chunkinfo] [data...]

CHUNKSIZE:
Intel-endian 16-bit integer
size of [data...] in bytes

CHUNKINFO: (hex)
Intel-endian 16-bit integer.

00 - login rejected
02 - login requested
03 - login accepted
0D - challenge? (nuprin1715)
2D - added to hotlist
2E - browse error (user isn't online!)
2F - user offline

5B - whois query
5C - whois result
5D - whois: user is offline!
69 - list all channels
6A - channel info
90 - join channel
91 - leave channel
.....

Centralized Database: Napster

- Summary of features: a hybrid design
  - Control: client-server (aka special DNS) for files
  - Data: peer to peer
- Advantages
  - Simplicity, easy to implement sophisticated search engines on top of the index system
- Disadvantages
  - Application specific (compared with DNS)
  - Lack of robustness, scalability: central search server single point of bottleneck/failure
  - Easy to sue !

Variation: BitTorrent

- A global central index server is replaced by one tracker per file (called a swarm)
  - Reduces centralization; but needs other means to locate trackers
- The bandwidth scalability management technique is more interesting
  - More later

Outline

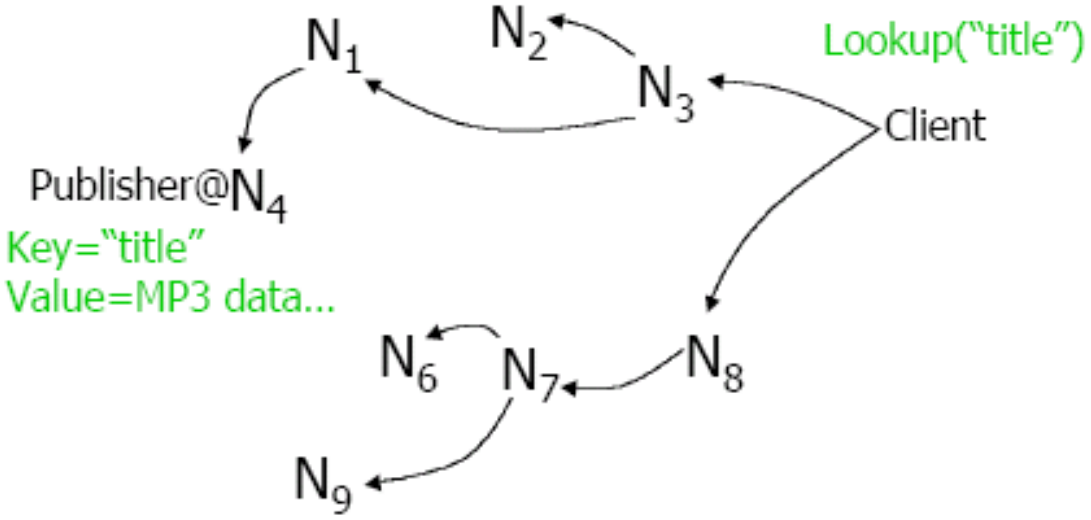
- *P2P*
  - *The lookup problem*
    - Napster (central query server; distributed data servers)
    - *Gnutella*

Gnutella

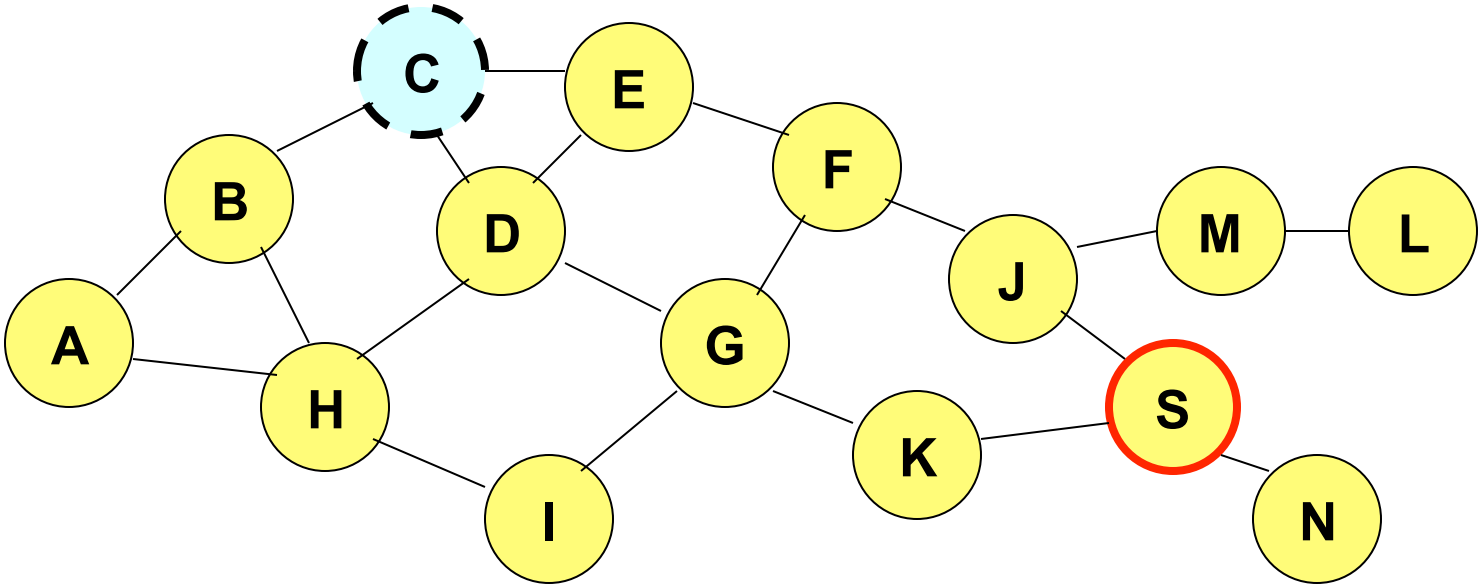
- On March 14th 2000, J. Frankel and T. Pepper from AOL's Nullsoft division (also the developers of the popular Winamp mp3 player) released Gnutella
- Within hours, AOL pulled the plug on it
- Quickly reverse-engineered and soon many other clients became available: Bearshare, Morpheus, LimeWire, etc.

Decentralized Flooding: Gnutella

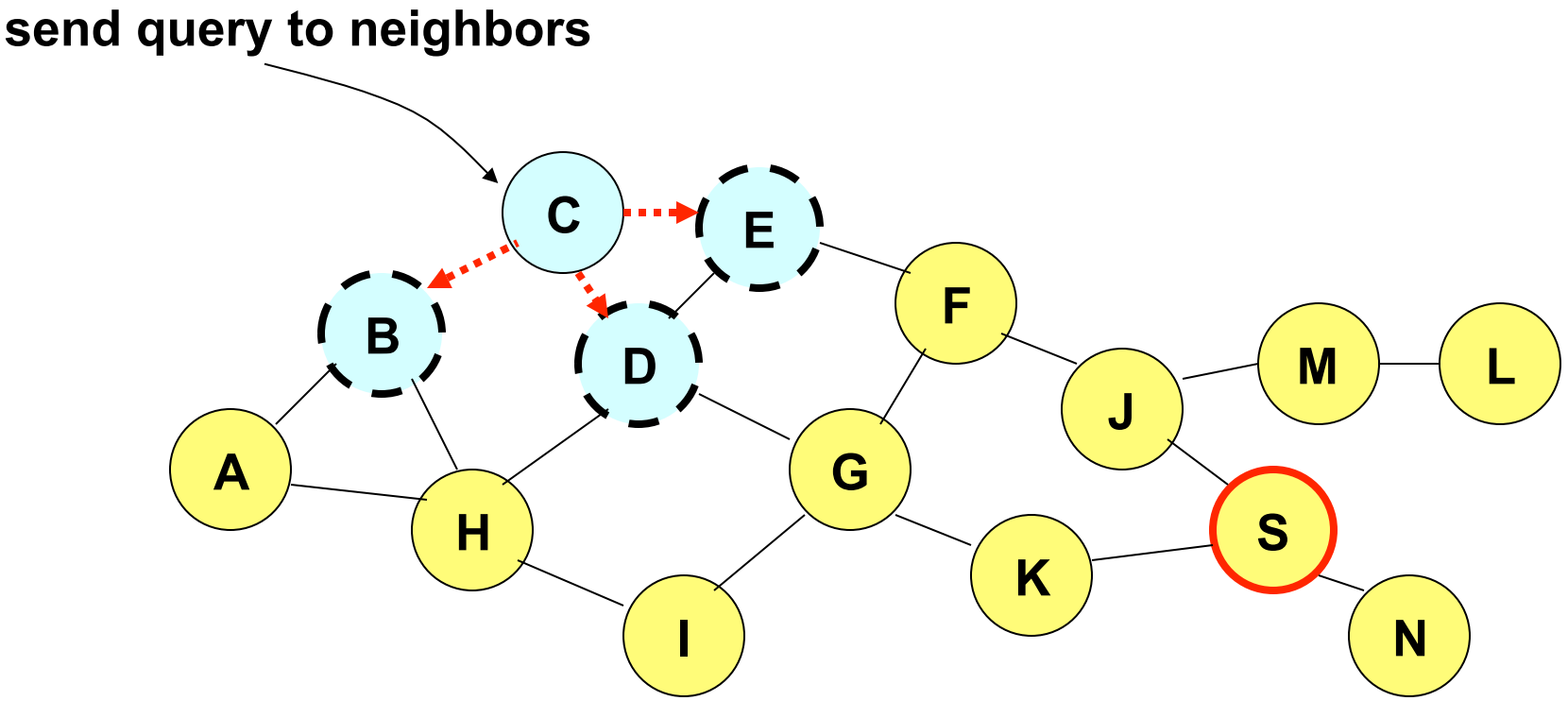
- On startup, client contacts other servents (**serv**er + cli**ent**) in network to form interconnection/peering relationships
  - Servent interconnection used to forward control (queries, hits, etc)
- How to find a resource record: decentralized flooding
  - Send requests to neighbors
  - Neighbors recursively forward the requests



Decentralized Flooding

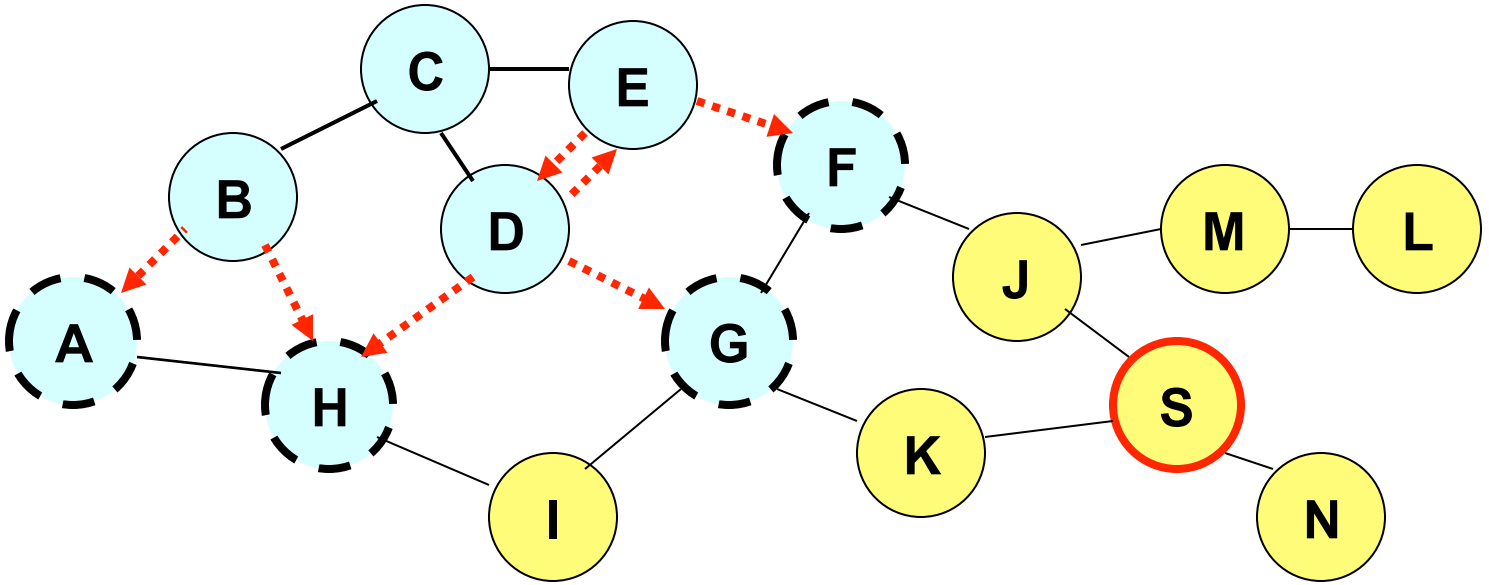


Decentralized Flooding



- Each node forwards the query to its neighbors other than the one who forwards it the query

Background: Decentralized Flooding



- Each node should keep track of forwarded queries to avoid loop !
  - Nodes keep state (which will time out---soft state)
  - Carry the state in the query, i.e. carry a list of visited nodes

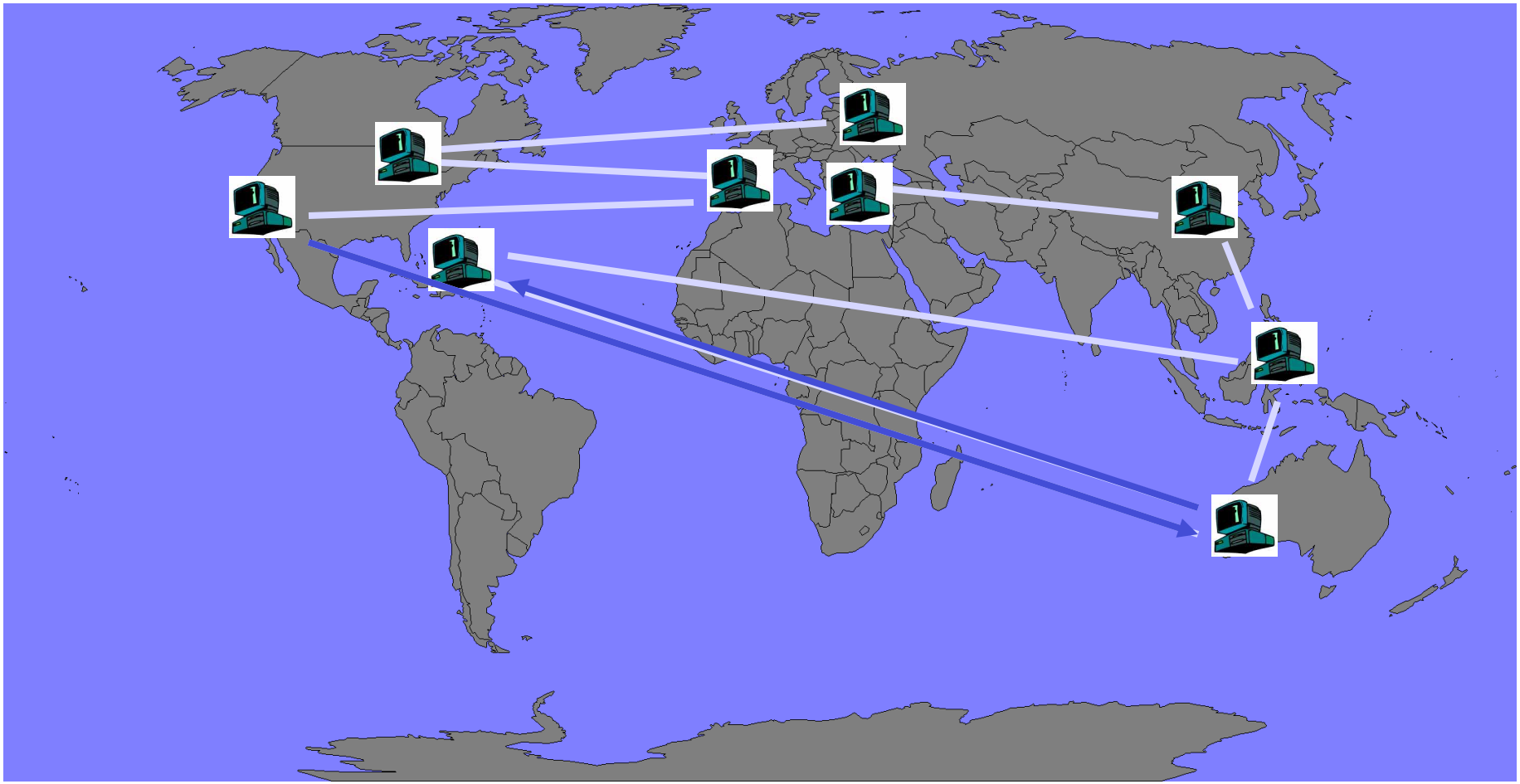
Decentralized Flooding: Gnutella

- Basic message header
  - Unique ID, TTL, Hops
- Message types
  - Ping – probes network for other servents
  - Pong – response to ping, contains IP addr, # of files, etc.
  - Query – search criteria + speed requirement of servent
  - QueryHit – successful response to Query, contains addr + port to transfer from, speed of servent, etc.
  - Ping, Queries are flooded
  - QueryHit, Pong: reverse path of previous message

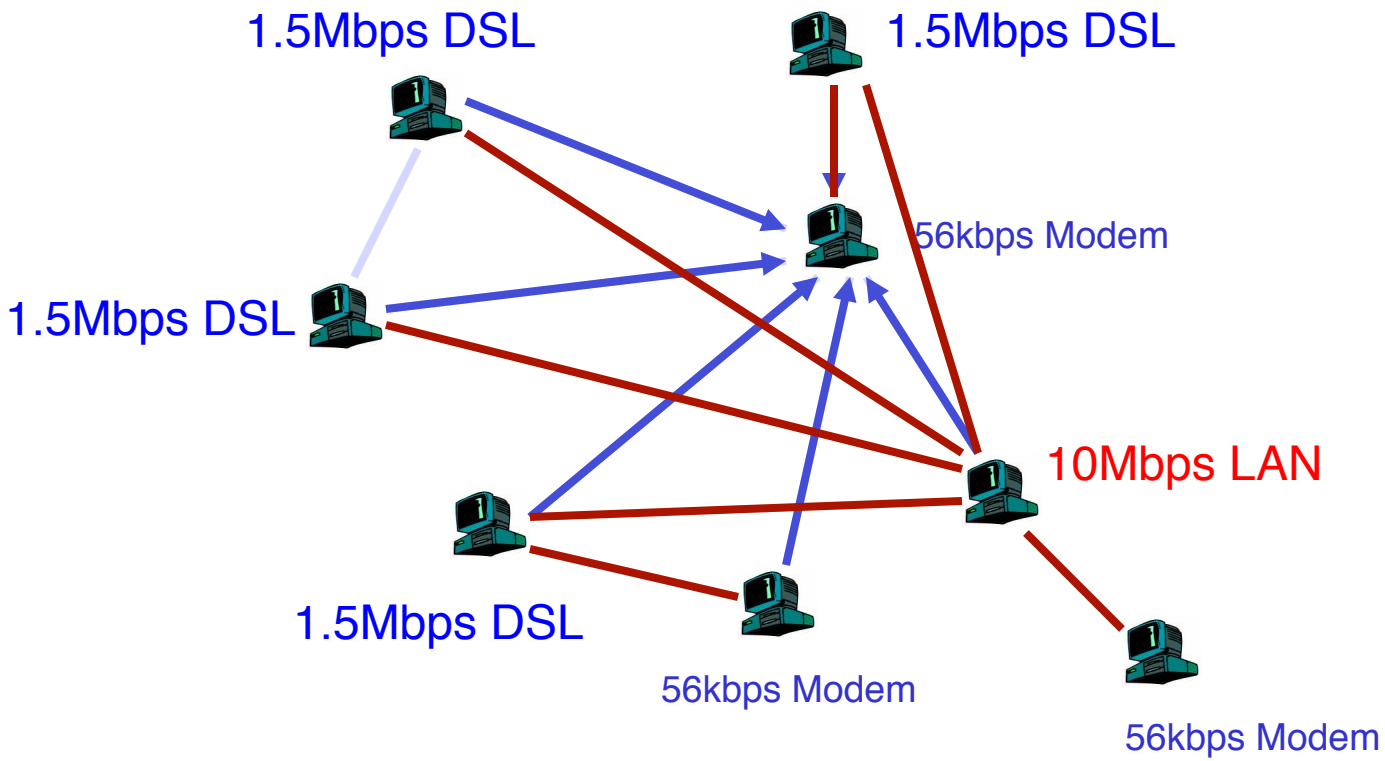
Advantages and Disadvantages of Gnutella

- Advantages:
  - Totally decentralized, highly robust
- Disadvantages:
  - Not scalable; the entire network can be swamped with flood requests
    - Especially hard on slow clients; at some point broadcast traffic on Gnutella exceeded 56 kbps
  - To alleviate this problem, each request has a TTL to limit the scope
    - Each query has an initial TTL, and each node forwarding it reduces it by one; if TTL reaches 0, the query is dropped (consequence?)

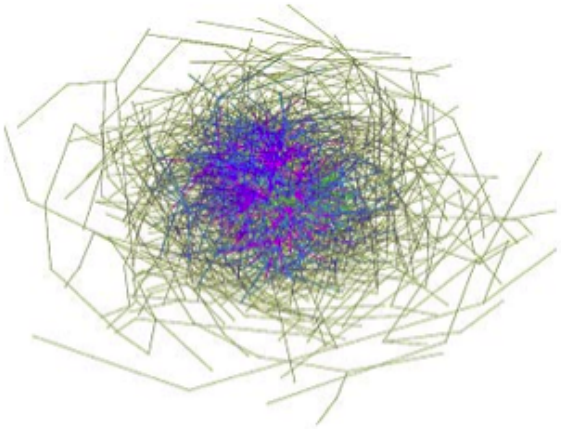
Aside: Search Time?



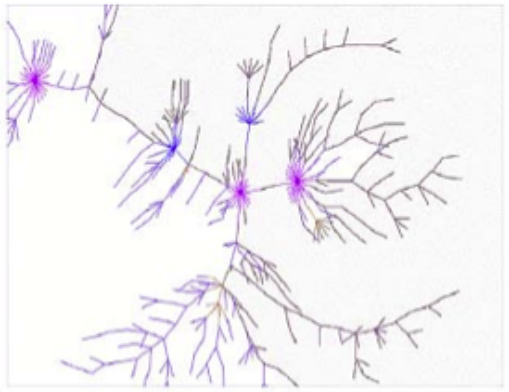
Aside: All Peers Equal?



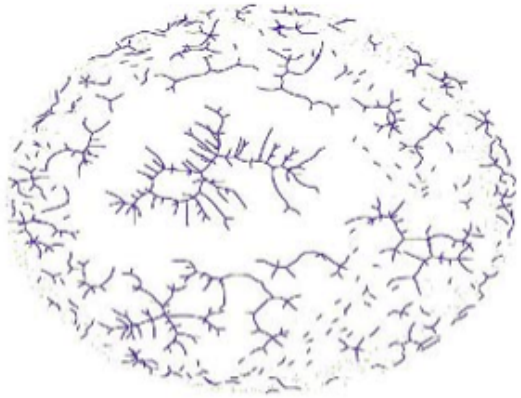
Aside: Network Resilience



Partial Topology



Random 30% die



Targeted 4% die

from Saroiu *et al.*, *MMCN* 2002

Flooding: FastTrack (aka Kazaa)

- 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
- Disadvantages
  - Kept a centralized registration → prone to law suits

Outline

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

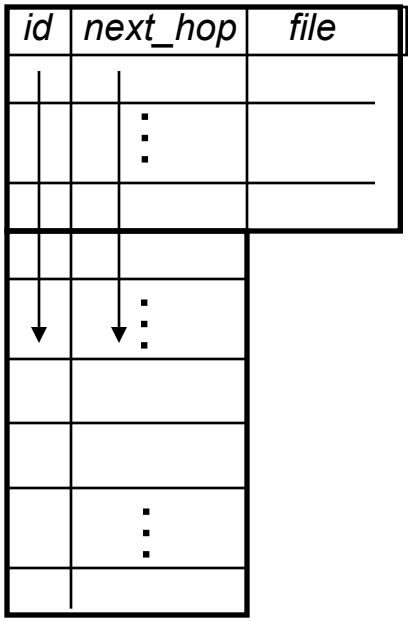
Freenet

- History
  - Final year project Ian Clarke , Edinburgh University, Scotland, June, 1999
- Goals:
  - Totally distributed system without using centralized index or broadcast (flooding)
  - Respond 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)

Basic Structure of Freenet

- Each machine stores a set of files; each file is identified by a unique identifier (called key or id)

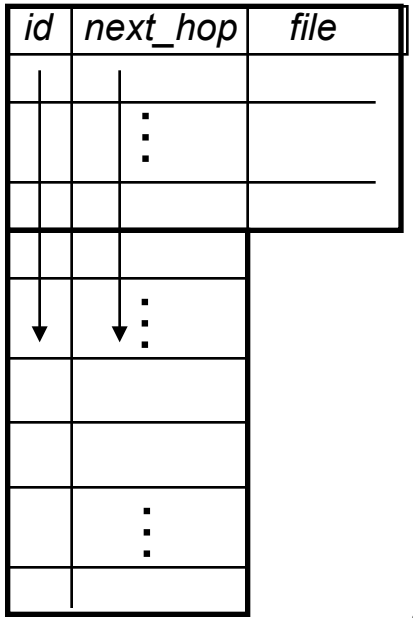
- Each node maintains a "routing table"
  - *id* – file id, key
  - *next_hop node* – where a file corresponding to the id might be available
  - *file* – local copy if one exists



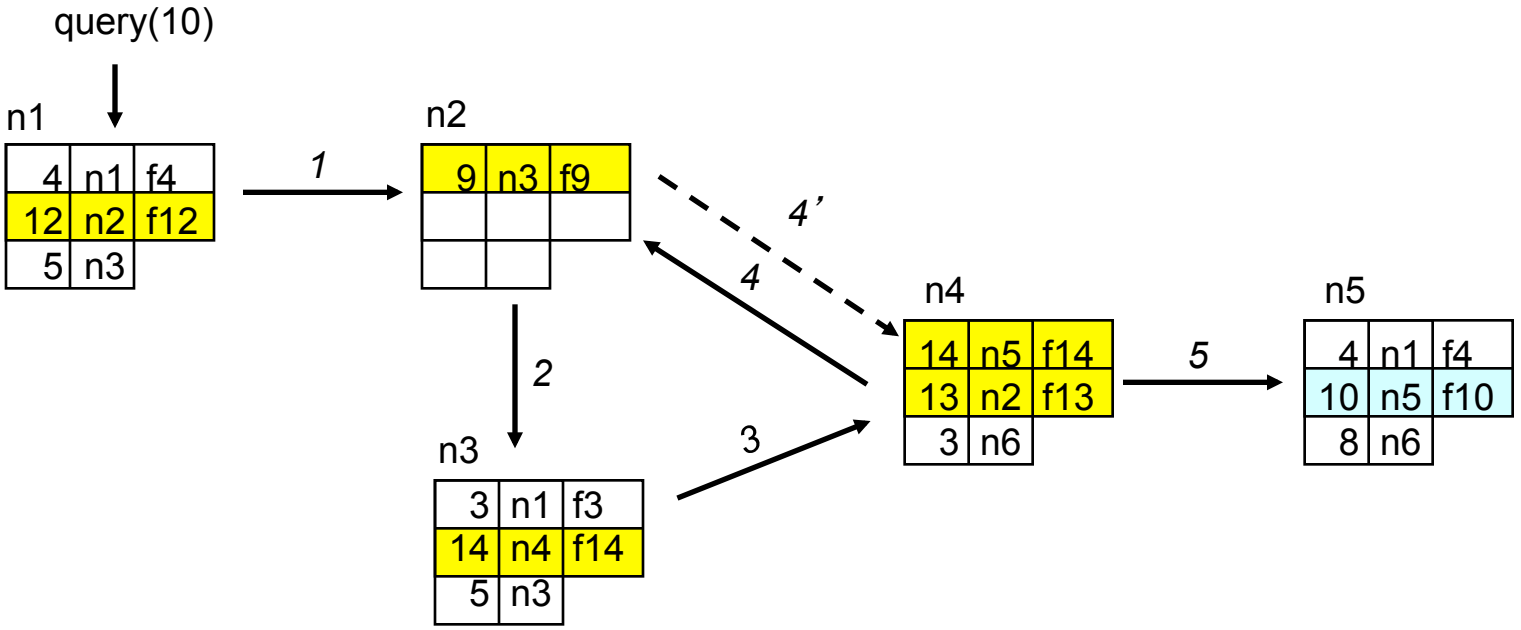
Query

- API: *file* = query(*id*);

- Upon receiving a query for file *id*
  - Check whether the queried file is stored locally
    - If yes, return it
    - If not, forward the query message
      - Key step: search for the "closest" *id* in the table, and forward the message to the corresponding *next_hop*



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

Query: the Complete Process

- Search by routing
- API: *file* = query(*id*);

- Upon receiving a query for file *id*
  - Check whether the queried file is stored locally
    - If yes, return it; otherwise
  - Check TTL to limit the search scope
    - Each query is associated a TTL that is decremented each time the message is forwarded
    - When TTL=1, the query is forwarded with a probability
    - TTL can be initiated to a random value within some bounds to obscure distance to originator
  - Look for the "**closest**" *id* in the table with an unvisited next_hope 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

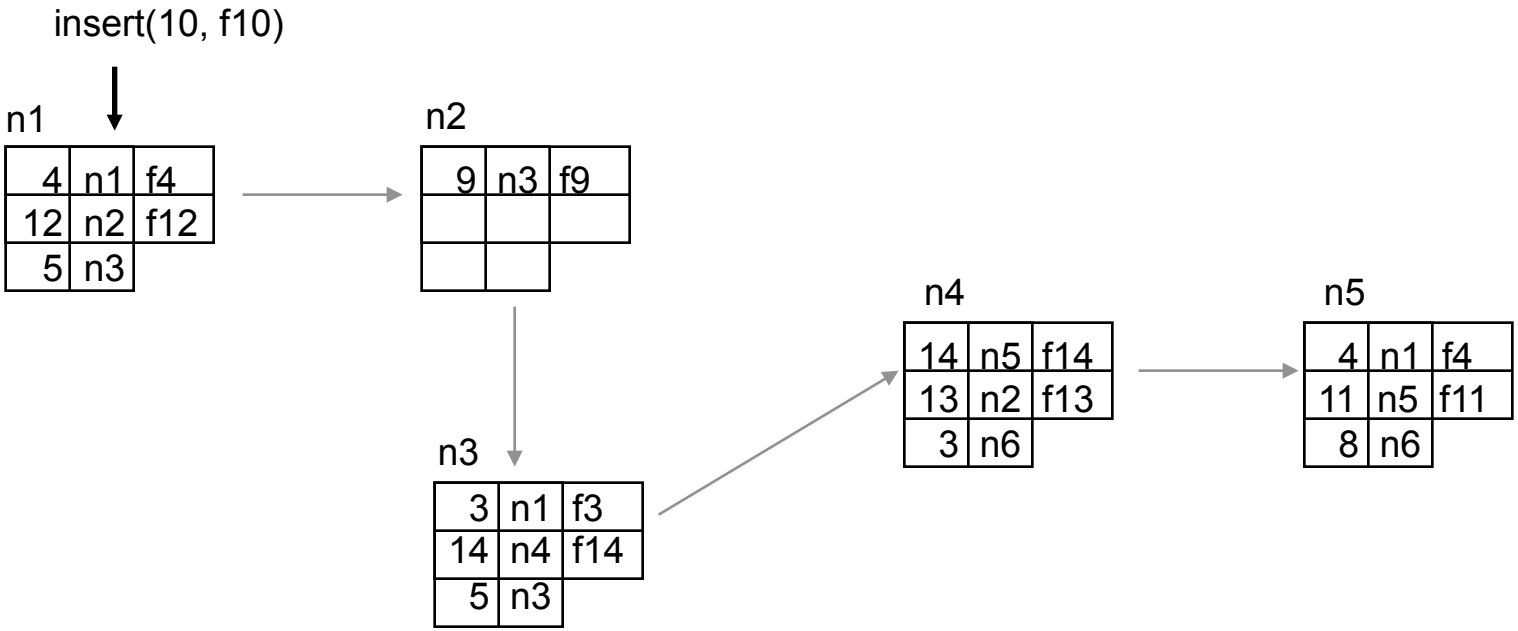
- When file is returned it is cached along the reverse path (any advantage?)

Insert

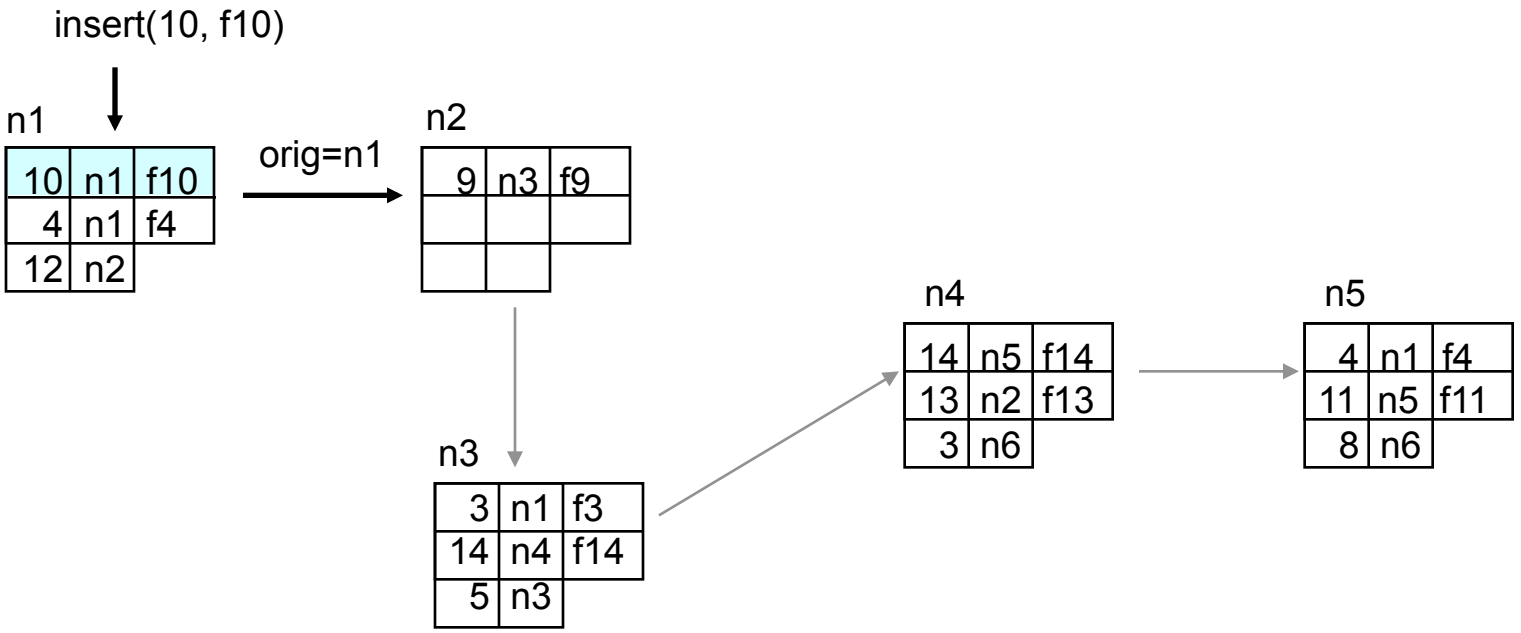
- API: insert(*id*, *file*);
- Two steps
  - First attempt a "search" for the file to be inserted
  - If found, report collision
  - If not found, insert the file by sending it along the query path
    - Inserted files are placed on nodes already possessing files with similar keys
    - A node probabilistically replaces the originator with itself (why?)

Insert Example

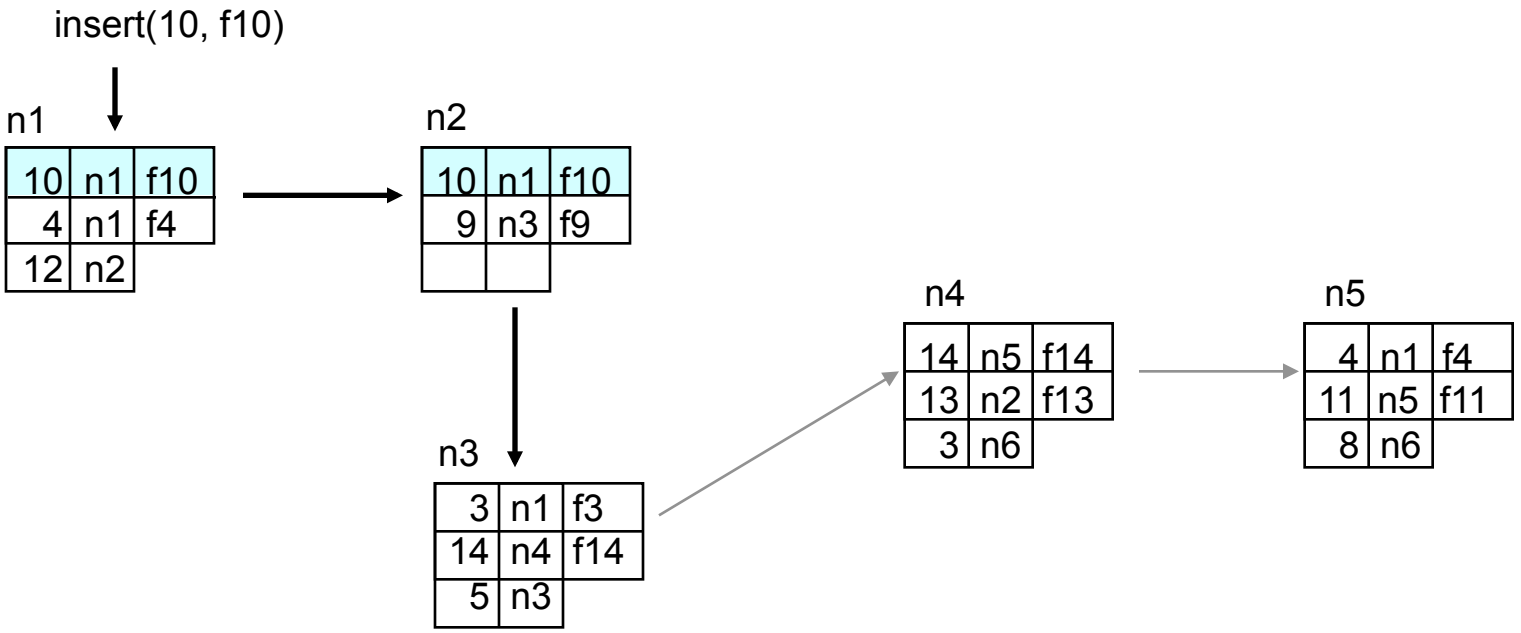
- Assume query returned failure along the shown path (backtrack slightly complicate things); insert f10



Insert Example

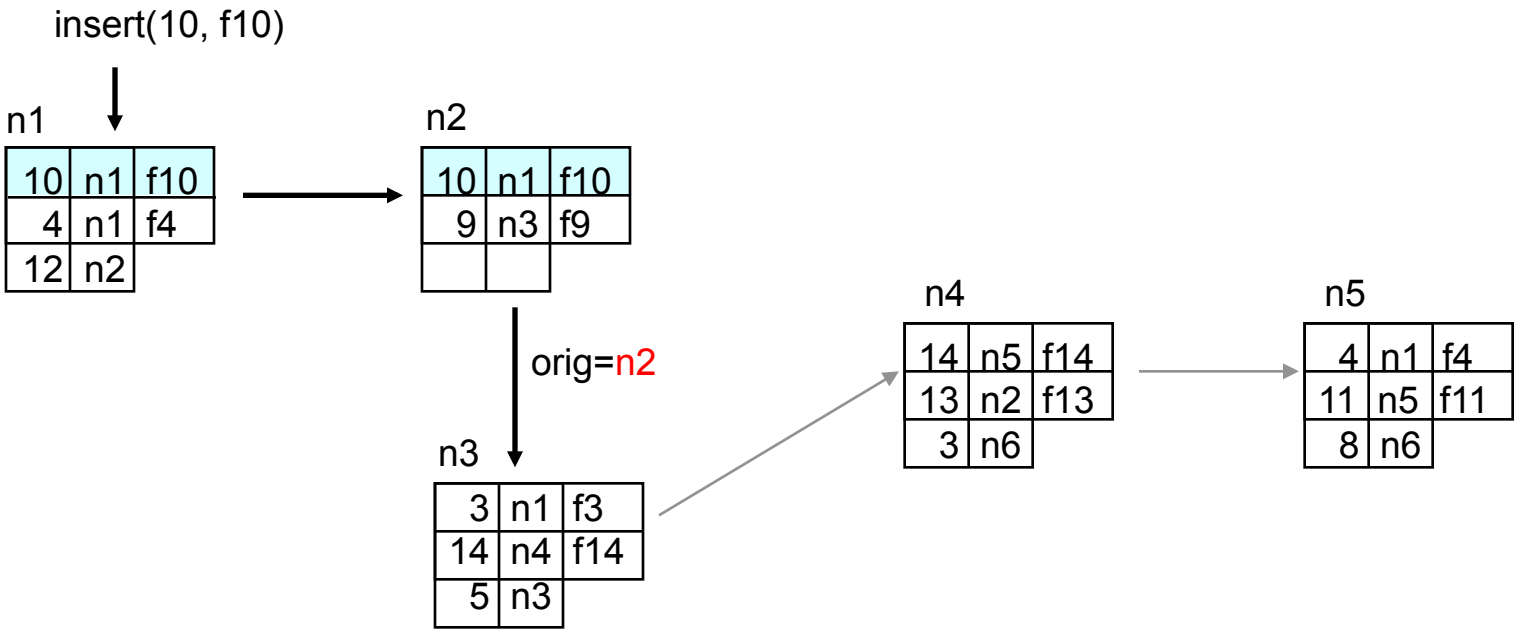


Insert Example

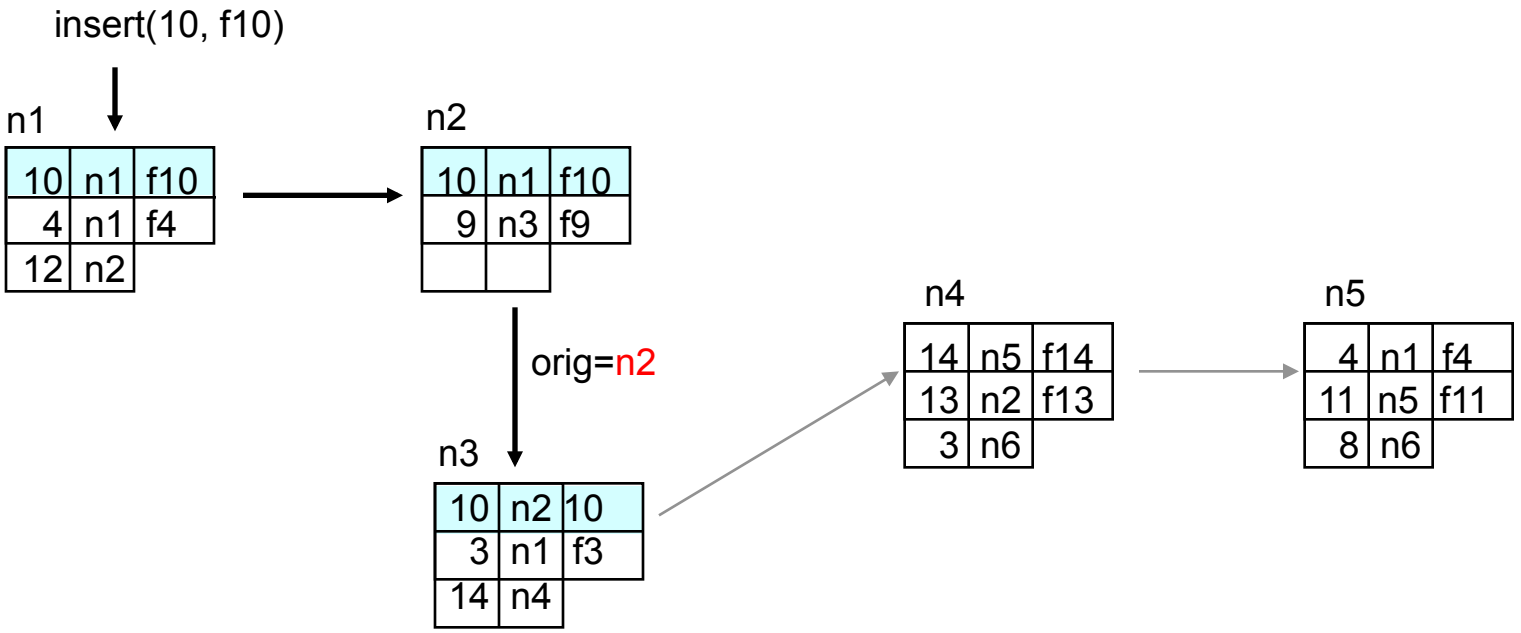


Insert Example

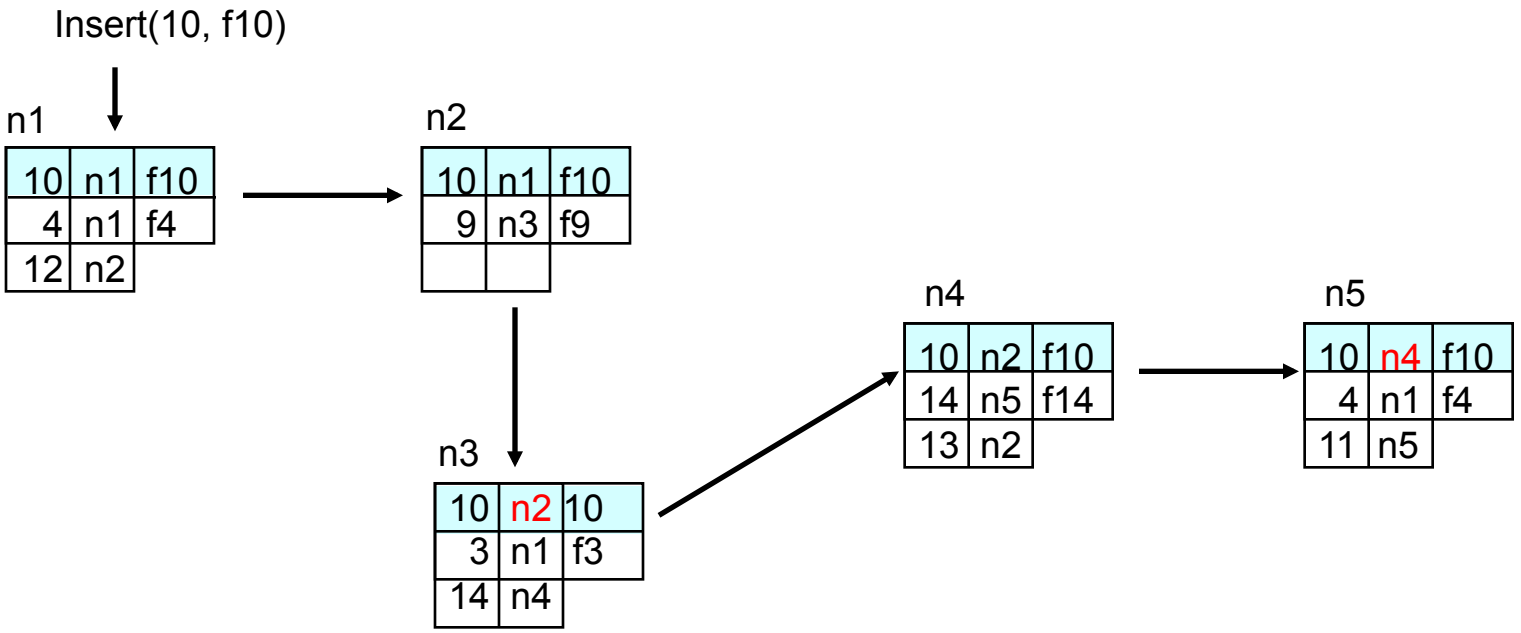
- n2 replaces the originator (n1) with itself



Insert Example



Insert Example

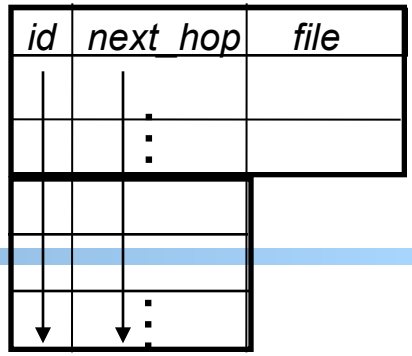


Freenet Analysis

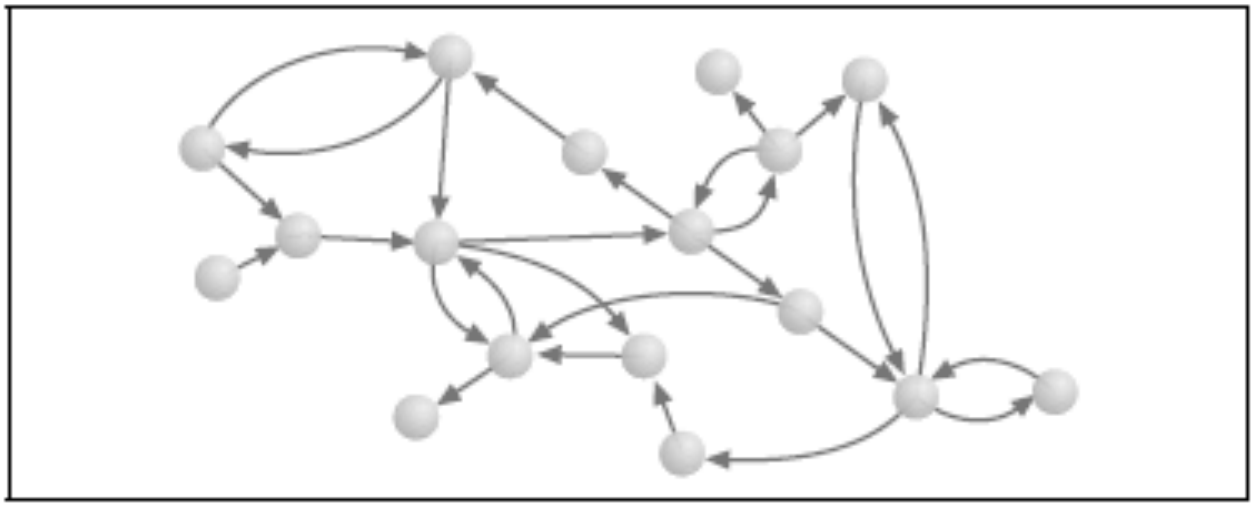
- Authors claim the following effects:
  - Nodes eventually specialize in locating similar keys
    - If a node is listed in a routing table, it will get queries for related keys
    - Thus will gain "experience" answering those queries
  - Popular data will be transparently replicated and will exist 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

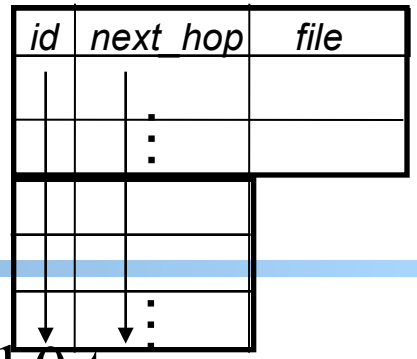
Understanding Freenet Self-Organization: Freenet Graph



- 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

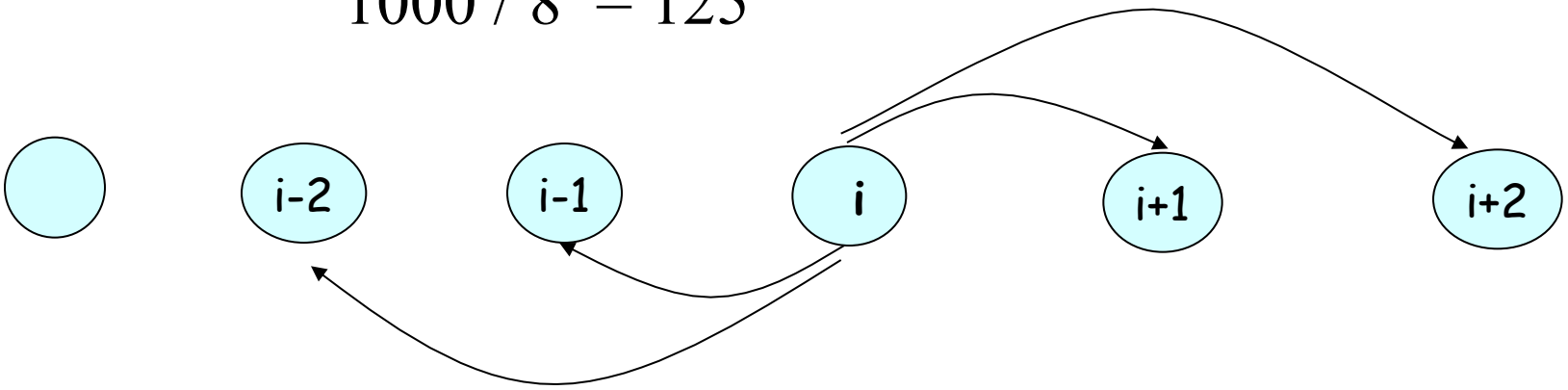


Experiment: Freenet Graph: Init



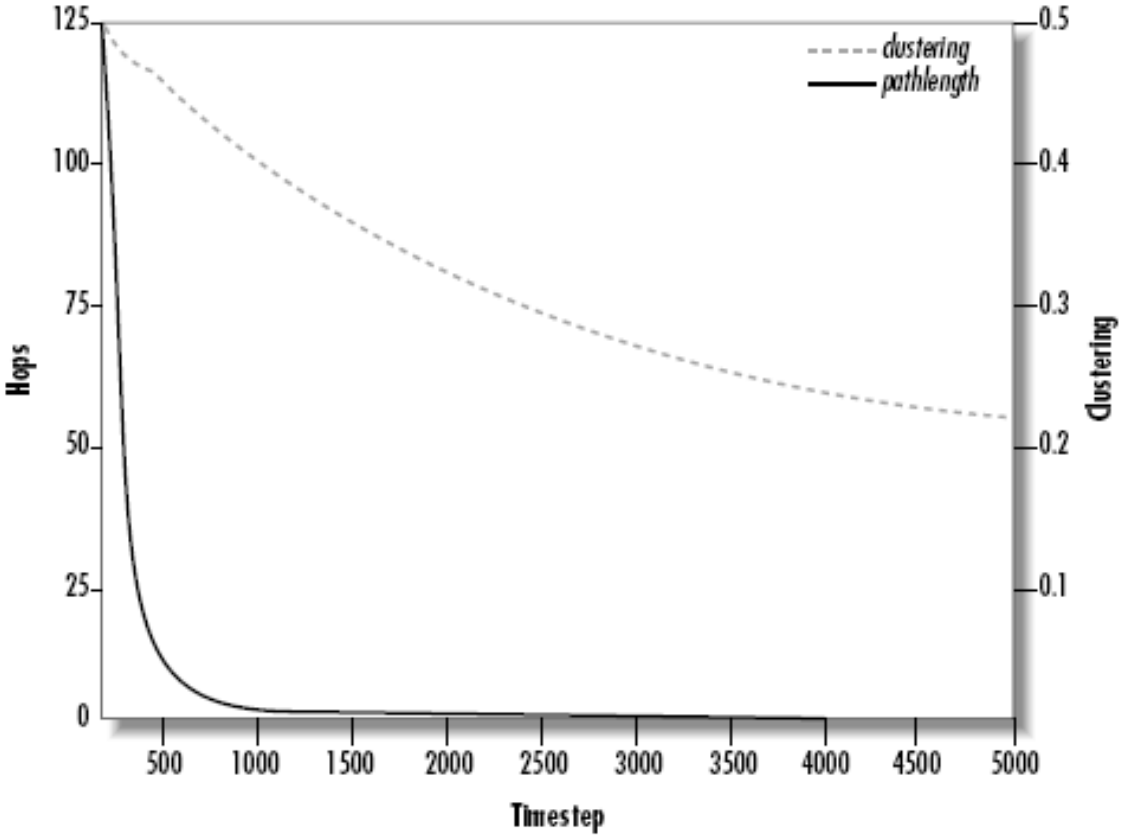
- Assume a network of 1000 nodes, with node id 0 to 999
- Each node can store 50 data items, and 200 references
- Assume initially each node i has i, and knows the storage of i – 2, -1, i + 1, i + 2 (all mod 1000)
- Thus a regular, locally-clustered graph with avg path length ~

$$1000 / 8 = 125$$



Experiment: Evolution of Freenet Graph

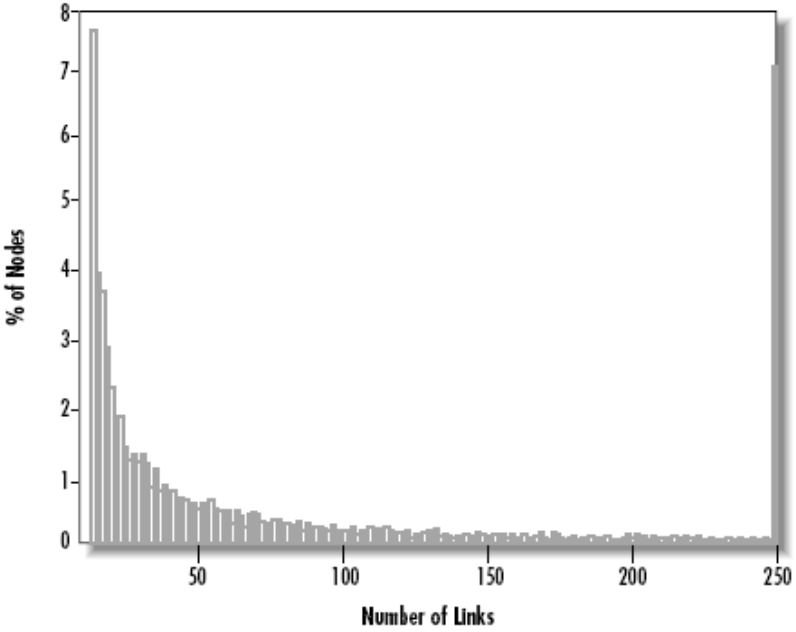
- At each step
  - Pick a node randomly
  - Flip a coin to determine search or insert
    - If search, randomly pick a key in the network
    - If insert, pick a random key

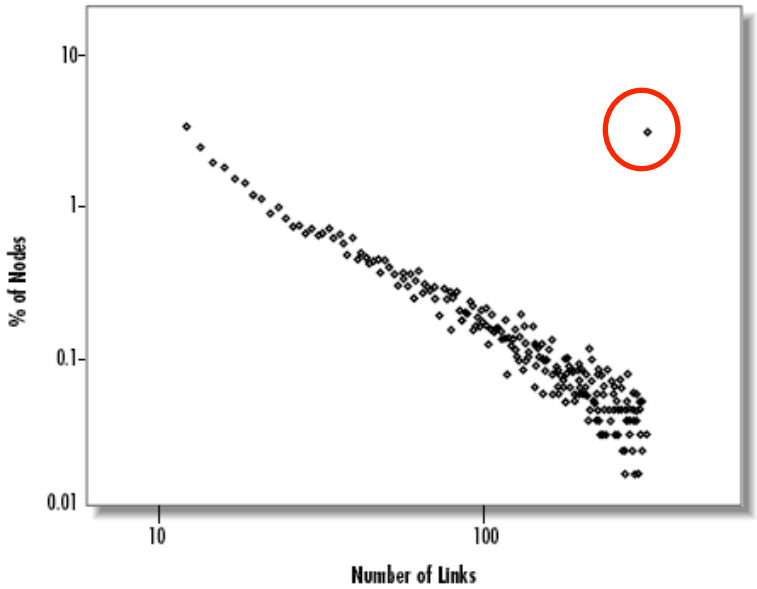


Evolution of path length and clustering; Clustering is defined as percentage of local links

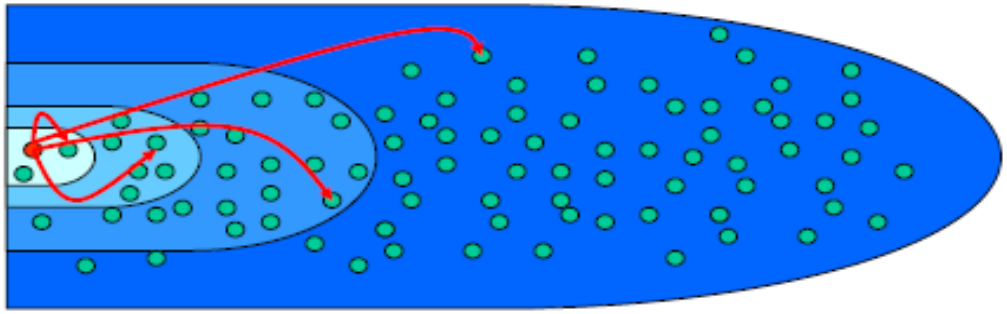
Freenet Evolves to Small-World Network

- 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





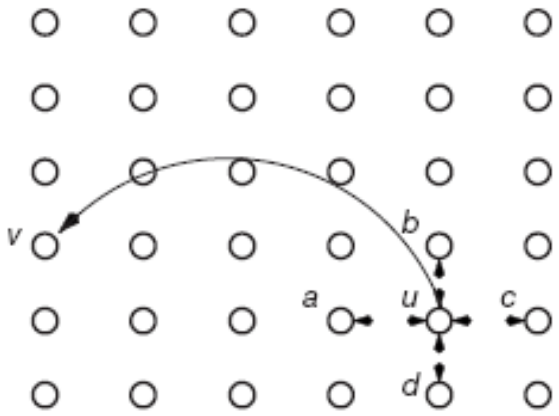
Small-World



- 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!
    - 1/4 of all the chains reaching the target person passed through a single person
    - Half the chains were mediated by just three people

Kleinberg's Result on Distributed Search

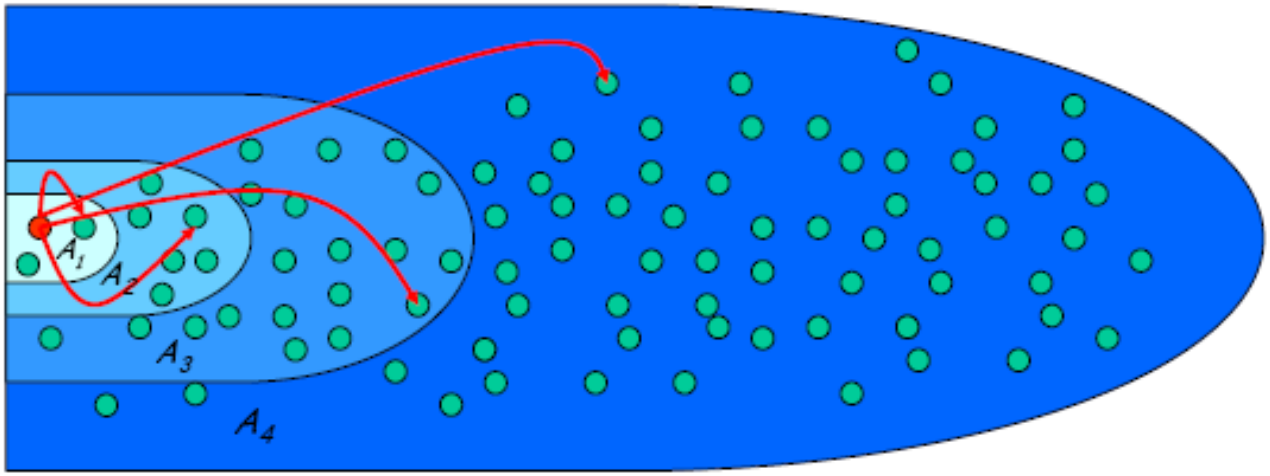
- Question: how many long distance links to maintain so that distributed (greedy) search is effective?
- Assume that the probability of a long link is some (α) inverse-power of the number of lattice steps
- Kleinberg's Law: Distributed algorithm exists only when probability is proportional to (lattice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



Distributed Search

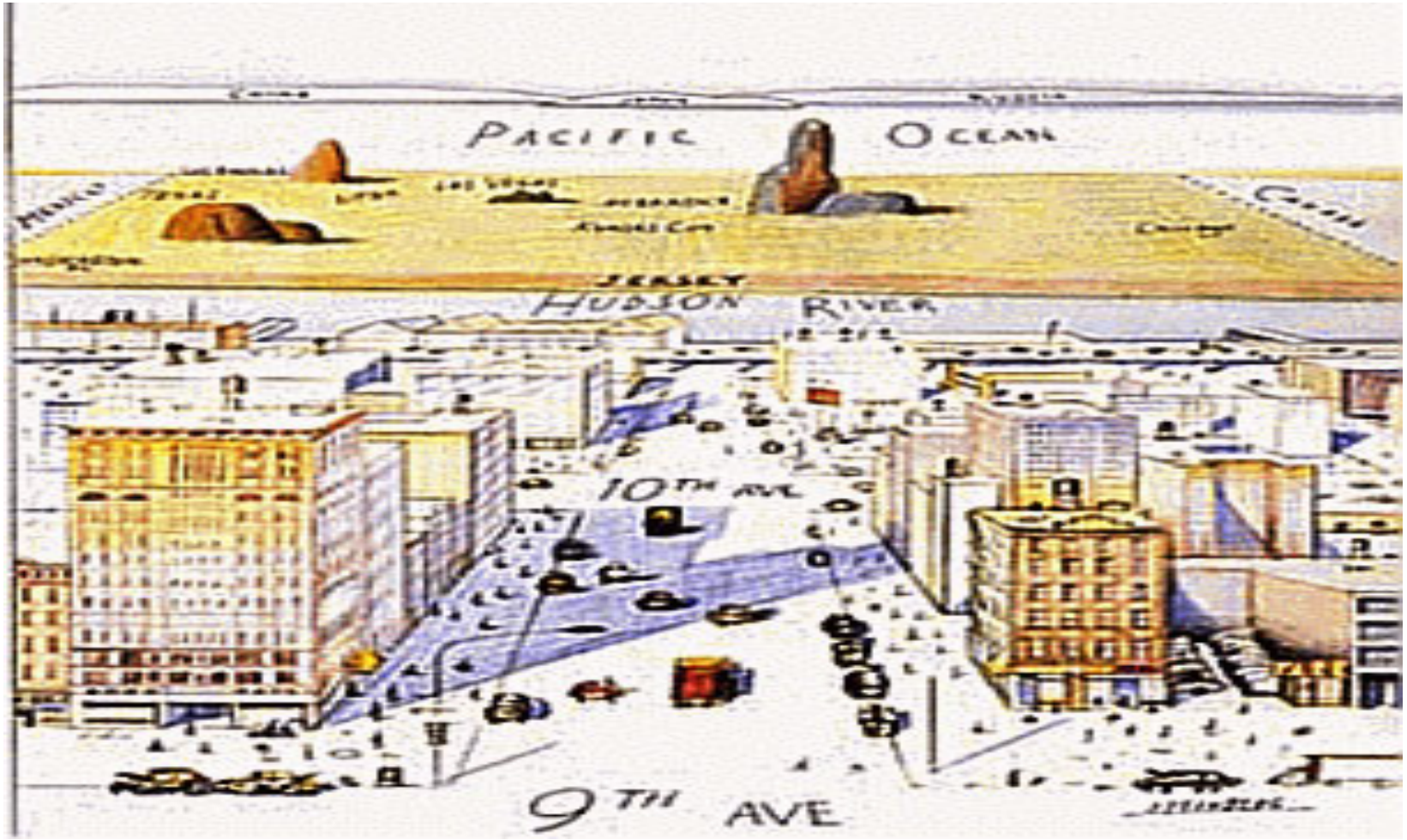
- In other words, if double distance, increase number of neighbors by a constant

  -> see Chord



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

Small World



Saul Steinberg; View of World from 9th Ave

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

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…

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