Peer-peer and Application-level Networking

Presented by Richard Gold

Based ***strongly*** on material by

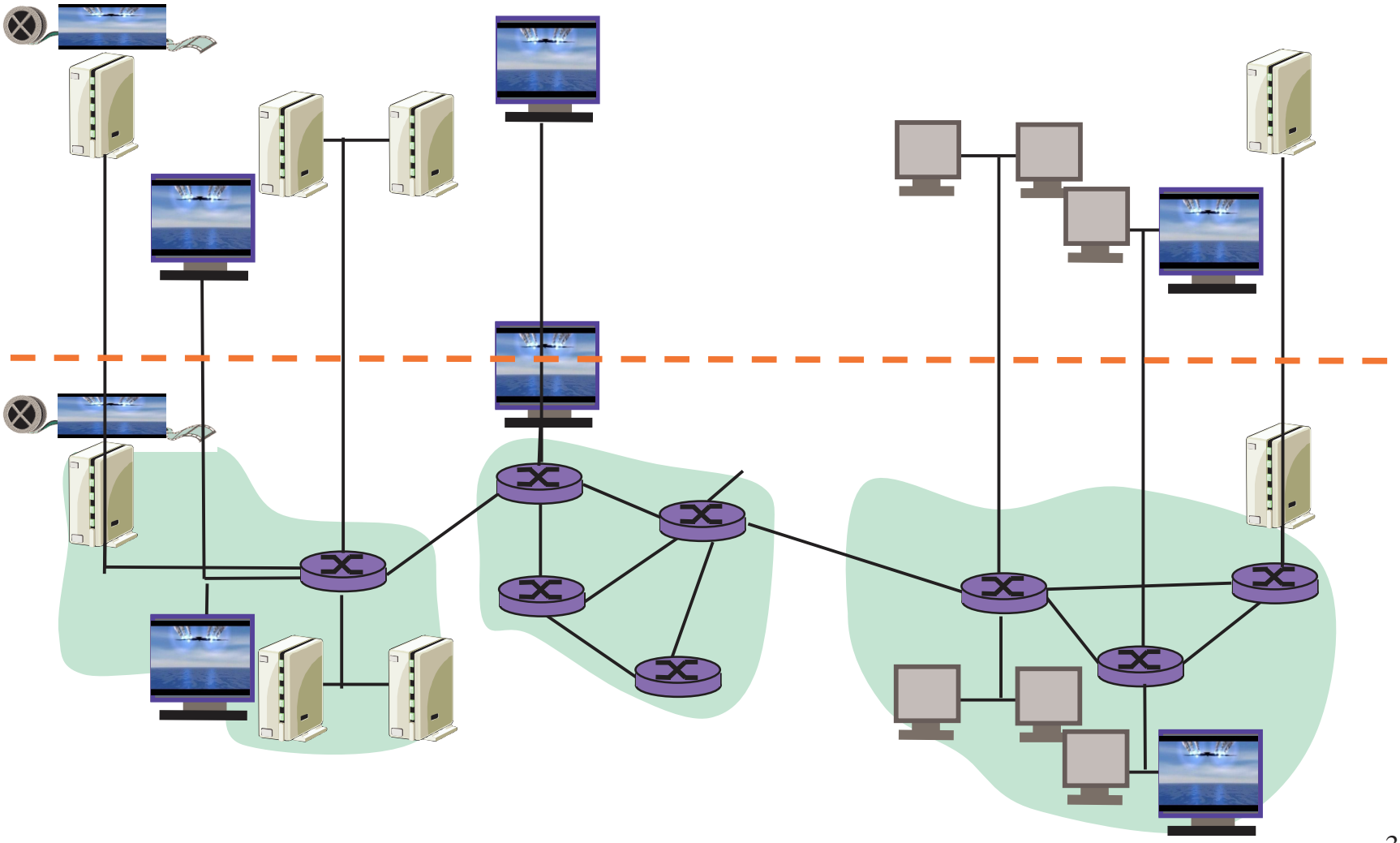
Jim Kurose,Brian Levine,Don Towsley, and the class of 2001 for the Umass Comp Sci 791N course..

Presented originally by Jon Crowcroft

0.Introduction

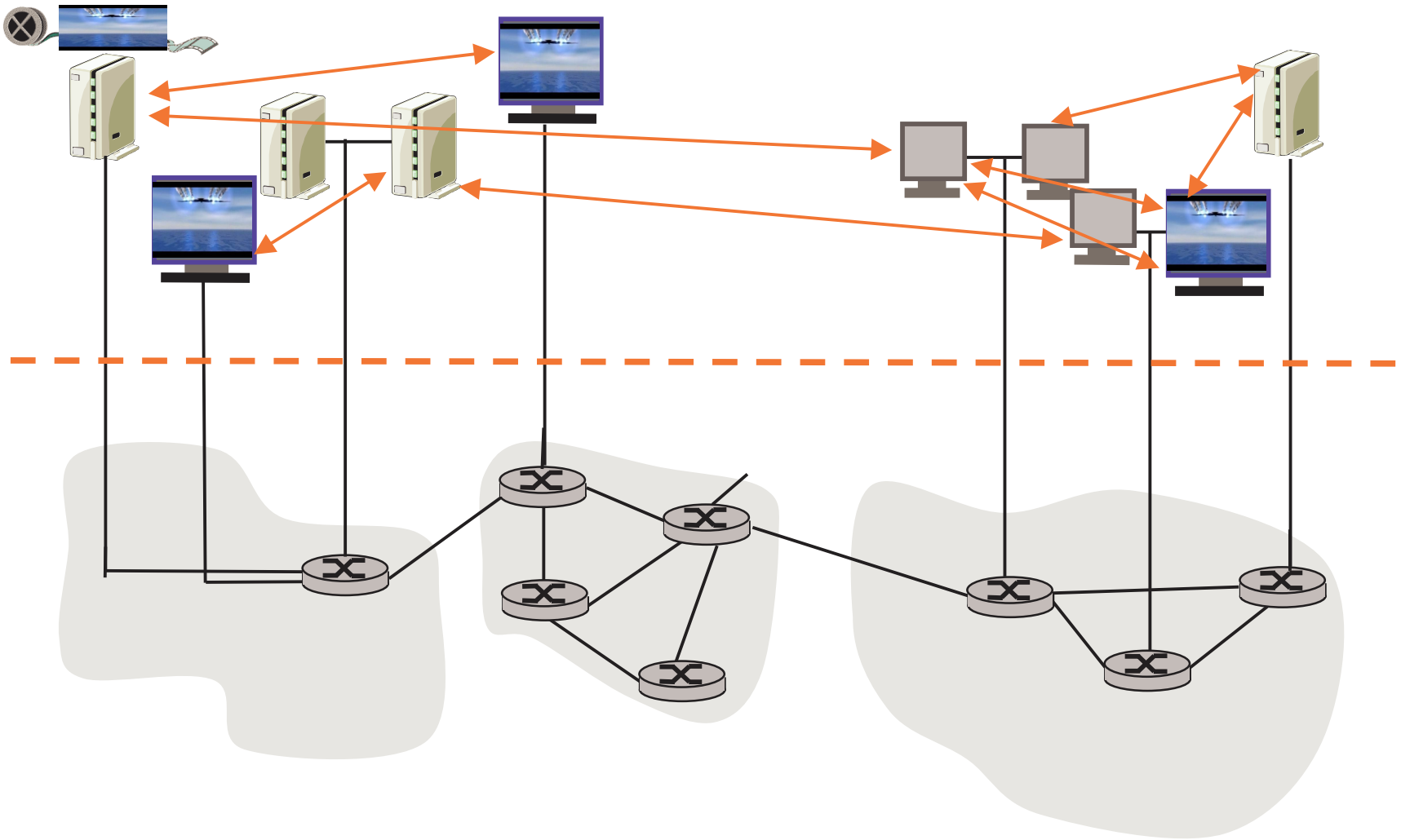
- Background
- Motivation
- outline of the tutorial

Peer-peer networking



Peer-peer networking

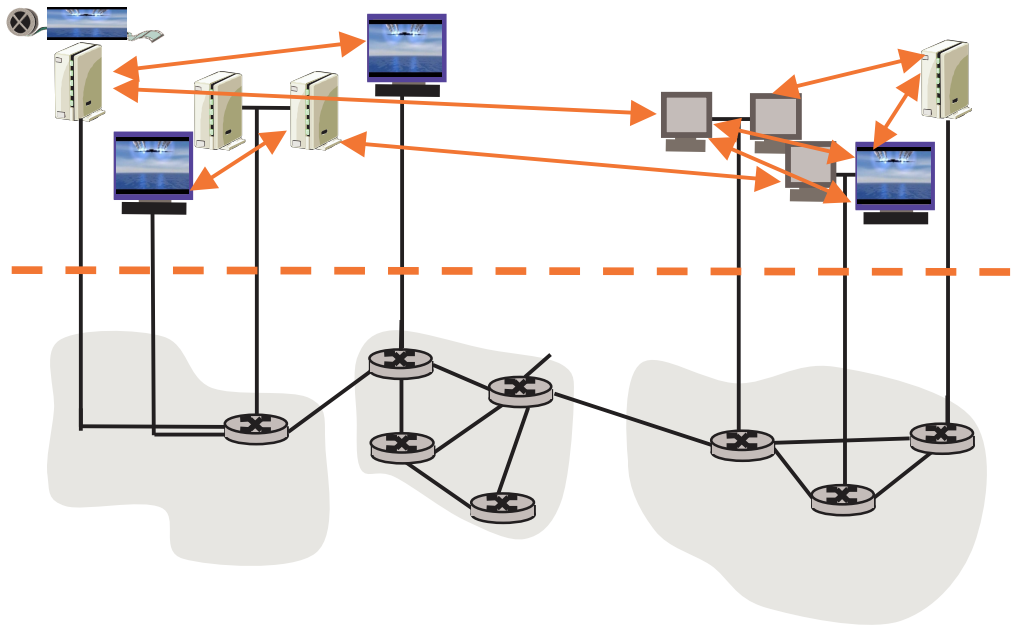
Focus at the application level



Peer-peer networking

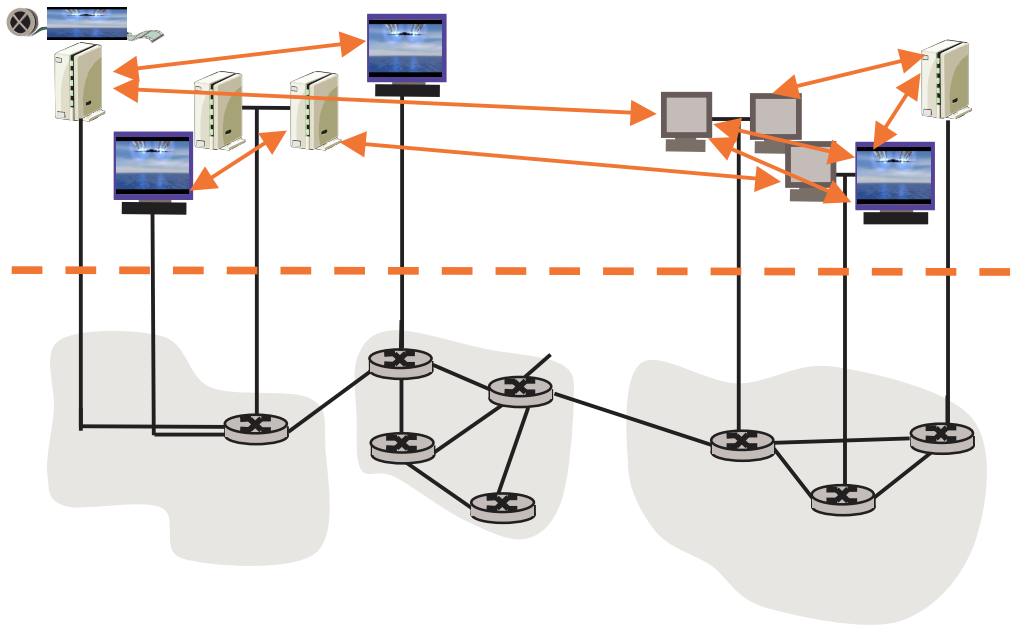
Peer-peer applications

- Napster, Gnutella, Freenet: file sharing



Peer-peer networking

- Q: What are the new technical challenges?
- Q: What new services/applications enabled?
- Q: Is it just "networking at the application-level"?
  - "There is nothing new under the Sun" (William Shakespeare)



Tutorial Contents

- Introduction
- Client-Server v. P2P
- Case Studies
  - Napster
  - Gnutella
  - Freenet
- More another time?

Client Server v. Peer to Peer(1)

- RPC/RMI
- Synchronous
- Assymmetric
- Emphasis on language integration and binding models (stub *IDL/XDR* compilers etc)
- Kerberos style security – access control, crypto

- Messages
- Asynchronous
- Symmetric
- Emphasis on service location, content addressing, application layer routing.
- Anonymity, high availability, integrity.
- Harder to get right☺

Peer to peer systems actually old

- IP routers are peer to peer.
- Routers discover topology, and maintain it
- Routers are neither client nor server
- Routers continually chatter to each other
- Routers are fault tolerant, inherently
- Routers are autonomous

Peer to peer systems

- Have no distinguished role
- So no single point of bottleneck or failure.
- However, this means they need distributed algorithms for
  - Service discovery (name, address, route, metric, etc)
  - Neighbour status tracking
  - Application layer routing (based possibly on content, interest, etc)
  - Resilience, handing link and node failures
  - Etc etc etc

Next we look at some case studies

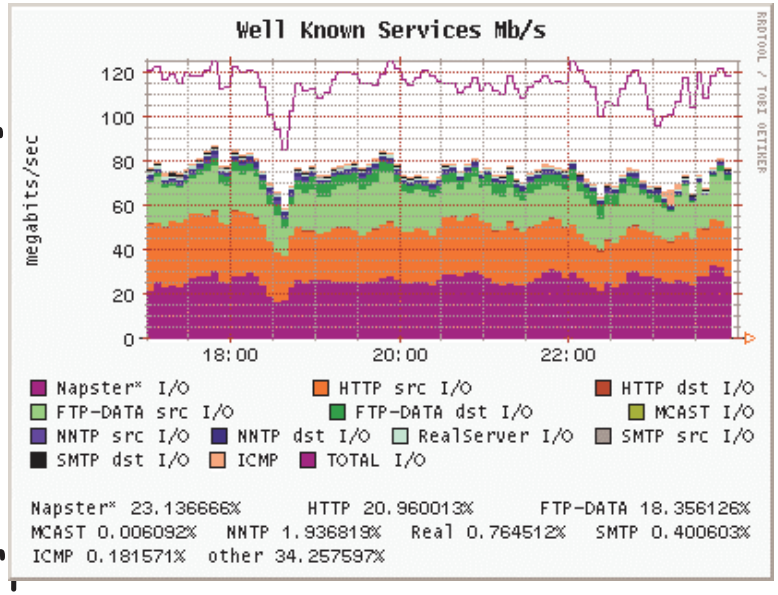
- Piracy^H^H^H^H^H^content sharing ☺
- Napster
- Gnutella
- Freenet
- Etc. (lessons here apply to other p2p systems too)

1. NAPSTER

- The most (in)famous
- Not the first (c.f. probably Eternity, from Ross Anderson in Cambridge)
- But instructive for what it gets right, and
- Also wrong…
- Also has a political message…and economic and legal…etc etc etc

Napster

- program for sharing files over the Internet
- a "disruptive" application/technology?
- history:
  - 5/99: Shawn Fanning (freshman, Northeasten U.) founds Napster Online music service
  - 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, Gnutella: 40K, Mor



Napster: how does it work

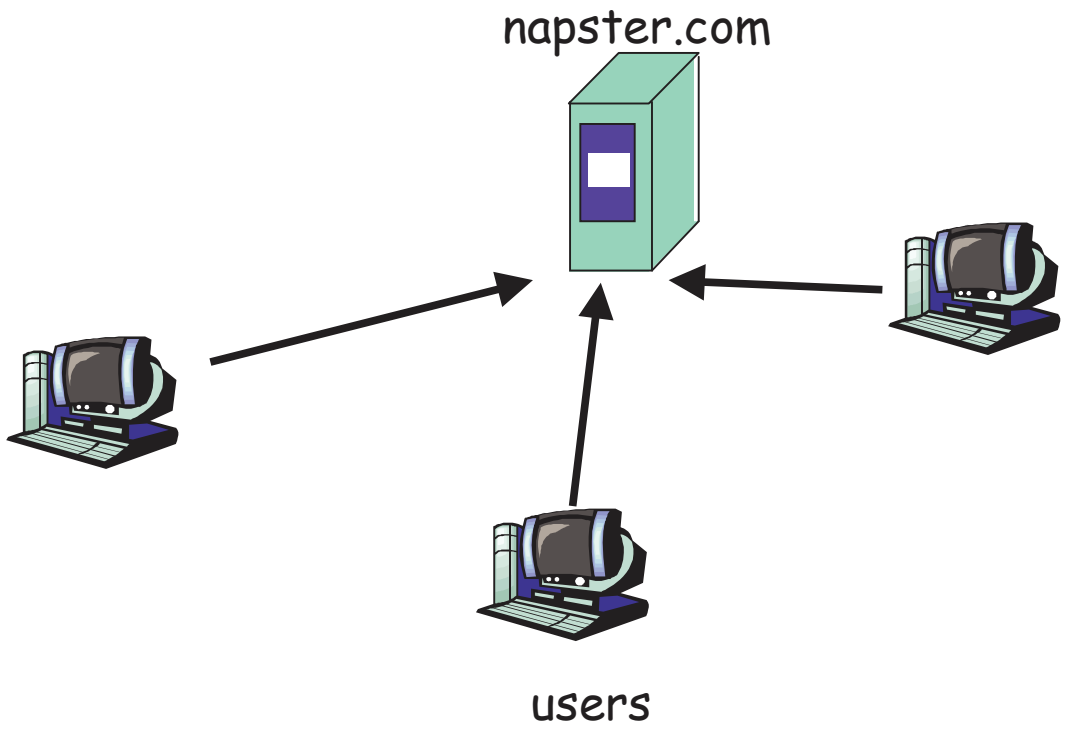
Application-level, client-server protocol over point-to-point TCP

Four steps:

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

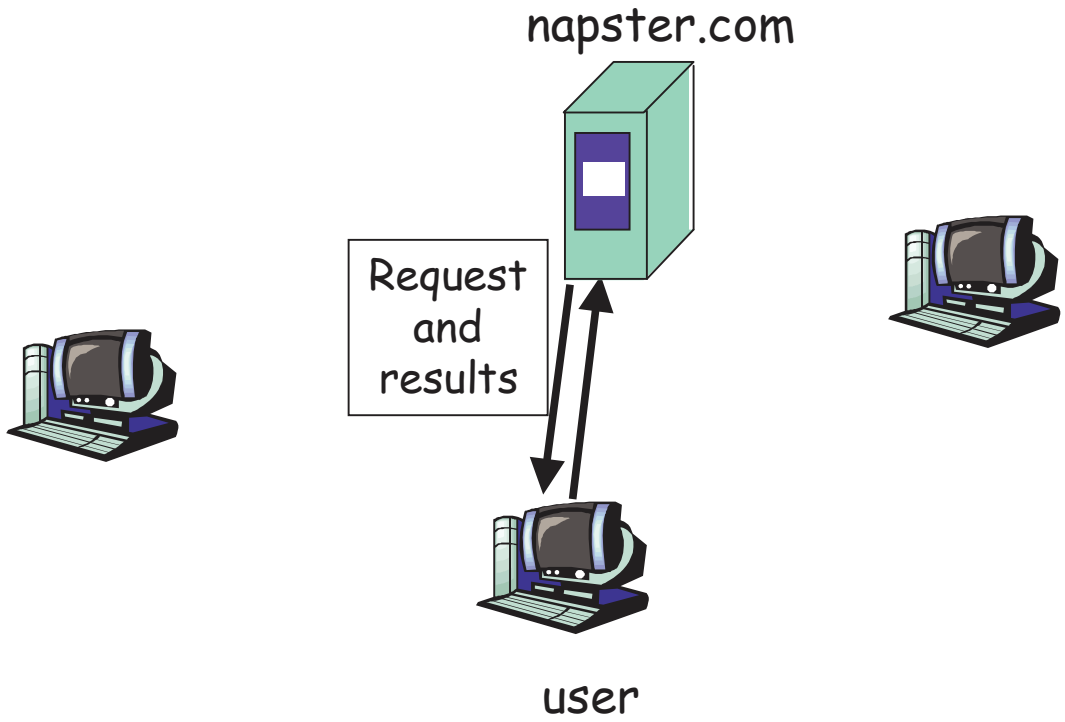
Napster

1. File list is uploaded



Napster

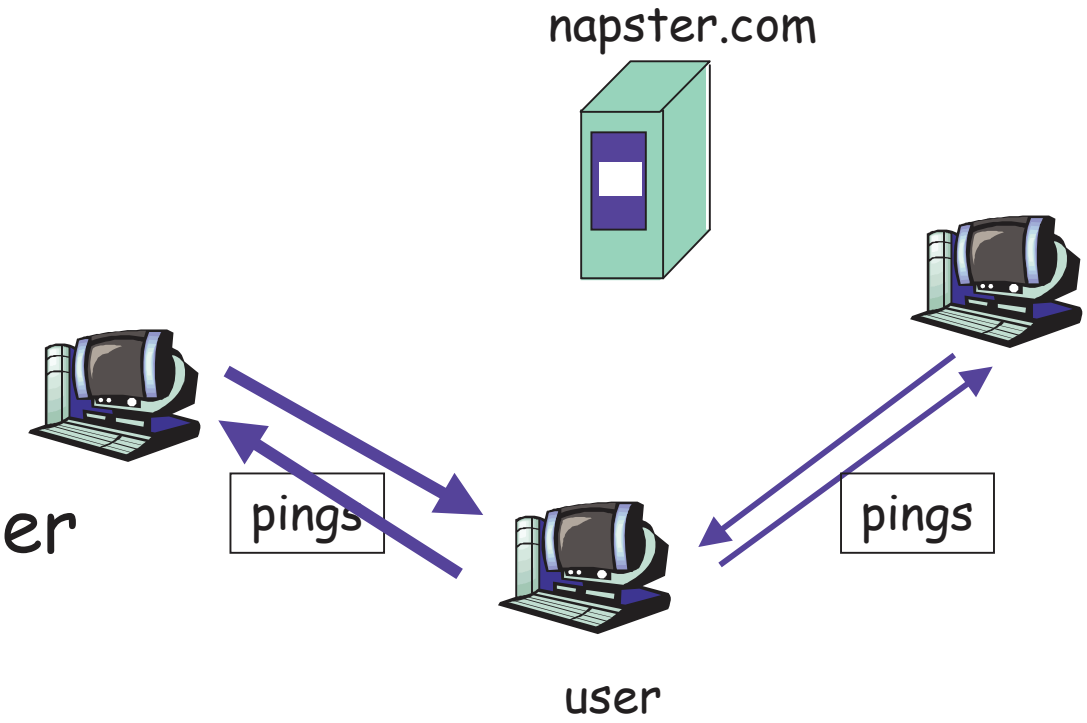
2. User requests search at server.



Napster

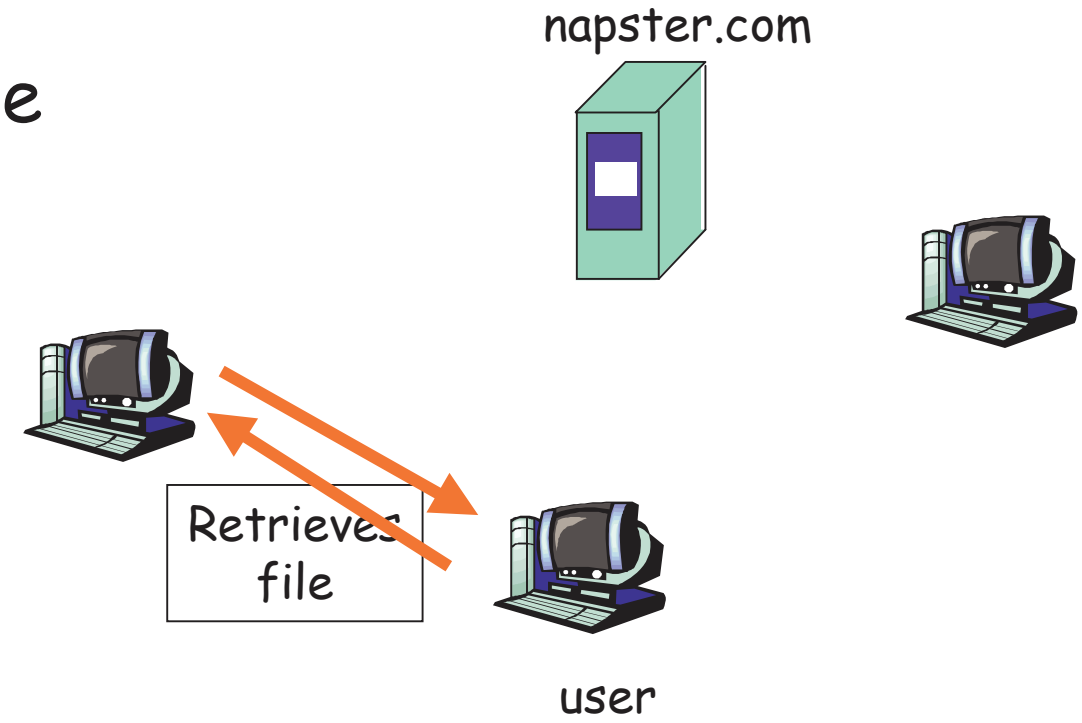
3. User pings hosts that apparently have data.

   Looks for ***best*** transfer rate.



Napster

4. User retrieves file



Napster: architecture notes

- centralized server:
  - single logical point of failure
  - can load balance among servers using DNS rotation
  - potential for congestion
  - Napster "in control" (freedom is an illusion)
- no security:
  - passwords in plain text
  - no authentication
  - no anonymity

2 Gnutella

- Napster fixed
- Open Source
- Distributed
- Still very political…

Gnutella

- peer-to-peer networking: applications connect to peer applications
- focus: decentralized method of searching for files
- each application instance serves to:
  - store selected files
  - route queries (file searches) from and to its neighboring peers
  - respond to queries (serve file) if file stored locally
- Gnutella history:
  - 3/14/00: release by AOL, almost immediately withdrawn
  - many iterations to fix poor initial design (poor design turned many people off)

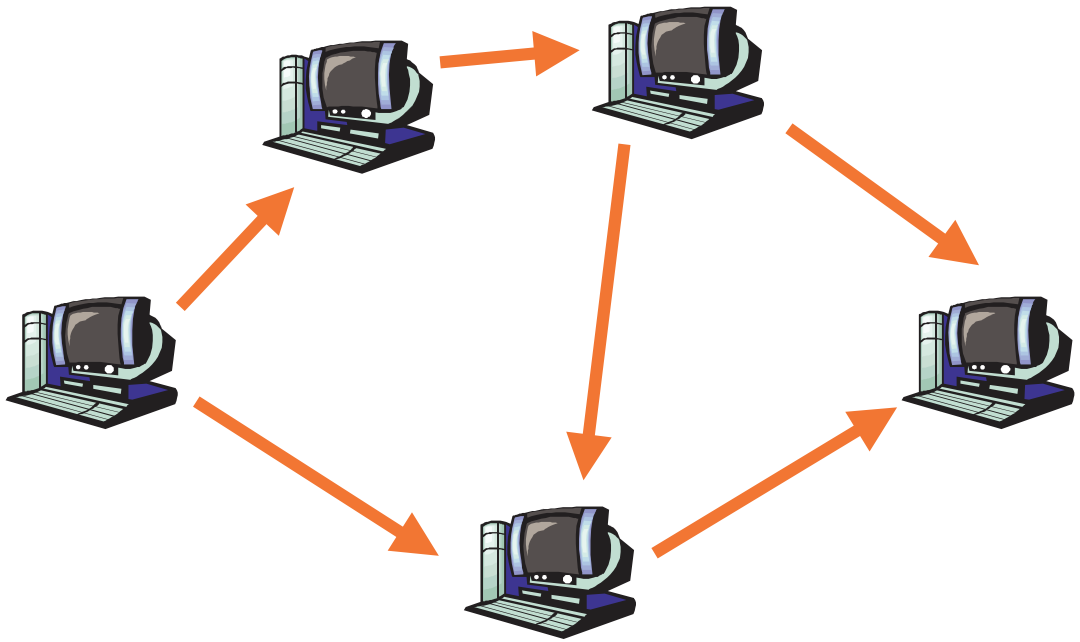
Gnutella: how it works

Searching by flooding:

- If you don't have the file you want, query 7 of your partners.
- If they don't have it, they contact 7 of their partners, for a maximum hop count of 10.
- Requests are flooded, but there is no tree structure.
- No looping but packets may be received twice.
- Reverse path forwarding(?)

Note: Play gnutella animation at:
http://www.limewire.com/index.jsp/p2p

Flooding in Gnutella: loop prevention



Seen already list: "A"

Gnutella message format

- Message ID: 16 bytes (yes bytes)
- FunctionID: 1 byte indicating
  - 00 ping: used to probe gnutella network for hosts
  - 01 pong: used to reply to ping, return # files shared
  - 80 query: search string, and desired minimum bandwidth
  - 81: query hit: indicating matches to 80:query, my IP address/port, available bandwidth
- RemainingTTL: decremented at each peer to prevent TTL-scoped flooding
- HopsTaken: number of peer visited so far by this message
- DataLength: length of data field

Gnutella: initial problems and fixes

- Freeloading: WWW sites offering search/retrieval from Gnutella network without providing file sharing or query routing.
  - Block file-serving to browser-based non-file-sharing users
- Prematurely terminated downloads:
  - long download times over modems
  - modem users run gnutella peer only briefly (Napster problem also!) or any users becomes overloaded
  - fix: peer can reply "I have it, but I am busy. Try again later"
  - late 2000: only 10% of downloads succeed
  - 2001: more than 25% downloads successful (is this success or failure?)

Gnutella: initial problems and fixes (more)

- 2000: avg size of reachable network ony 400-800 hosts. Why so smalll?
  - modem users: not enough bandwidth to provide search routing capabilities: routing black holes
- Fix: create peer hierarchy based on capabilities
  - previously: all peers identical, most modem blackholes
  - connection preferencing:
    - favors routing to well-connected peers
    - favors reply to clients that themselves serve large number of files: prevent freeloading
  - Limewire gateway functions as Napster-like central server on behalf of other peers (for searching purposes)

Anonymous?

- Not anymore than it's scalable.
- The person you are getting the file from knows who you are. That's not anonymous.

- Other protocols exist where the owner of the files doesn't know the requester.

- Peer-to-peer anonymity exists.
- See Eternity and, next, Freenet!

Gnutella Discussion:

- Architectural lessons learned?
- anonymity and security?
- Other?
- Good source for technical info/open questions:

  http://www.limewire.com/index.jsp/tech_papers
- Also this year's SIGCOMM:

http://www.acm.org/sigcomm/sigcomm2002/

Freenet history

- Final Year project Ian Clarke , Edinburgh University, Scotland, June, 1999
- Sourceforge Project, most active
- V.0.1 (released March 2000)
- Latest version(Sept, 2001): 0.4

What is Freenet and Why?

- Distributed, Peer to Peer, file sharing system
- Completely anonymous, for producers or consumers of information
- Resistance to attempts by third parties to deny access to information

Freenet: How it works

- Data structure
- Key Management
- Problems
  - How can one node know about others
  - How can it get data from remote nodes
  - How to add new nodes to Freenet
  - How does freenet manage its data

Data structure

- Routing Table
  - Pair: node address: ip, port; corresponding key value
- Data Store
  - Requirement:
    - rapidly find the document given a certain key
    - rapidly find the closest key to a given key
    - keep track the popularity of documents and know which document to delete when under pressure

Key Management(1)

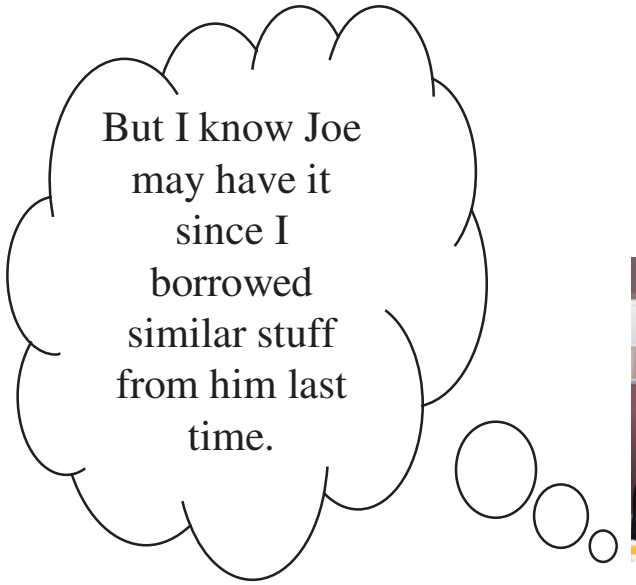
- A way to locate a document anywhere
- Keys are used to form a URI
- Two similar keys don't mean the subjects of the file are similar!
- Keyword-signed Key*(KSK)*
  - *Based on a short descriptive string, usually a set of keywords that can describe the document*
  - *Example: Bush/Oil/Iraq/Excuse*
  - *Uniquely identify a document*
  - *Potential problem – global namespace*

Key Management (2)

- Signed-subspace Key*(SSK)*
  - *Add sender information (not personal information!) to avoid namespace conflict*
  - *Private key to sign/ public key to verify*
- Content-hash Key*(CHK)*
  - *Message digest algorithm, Basically a hash of the document*

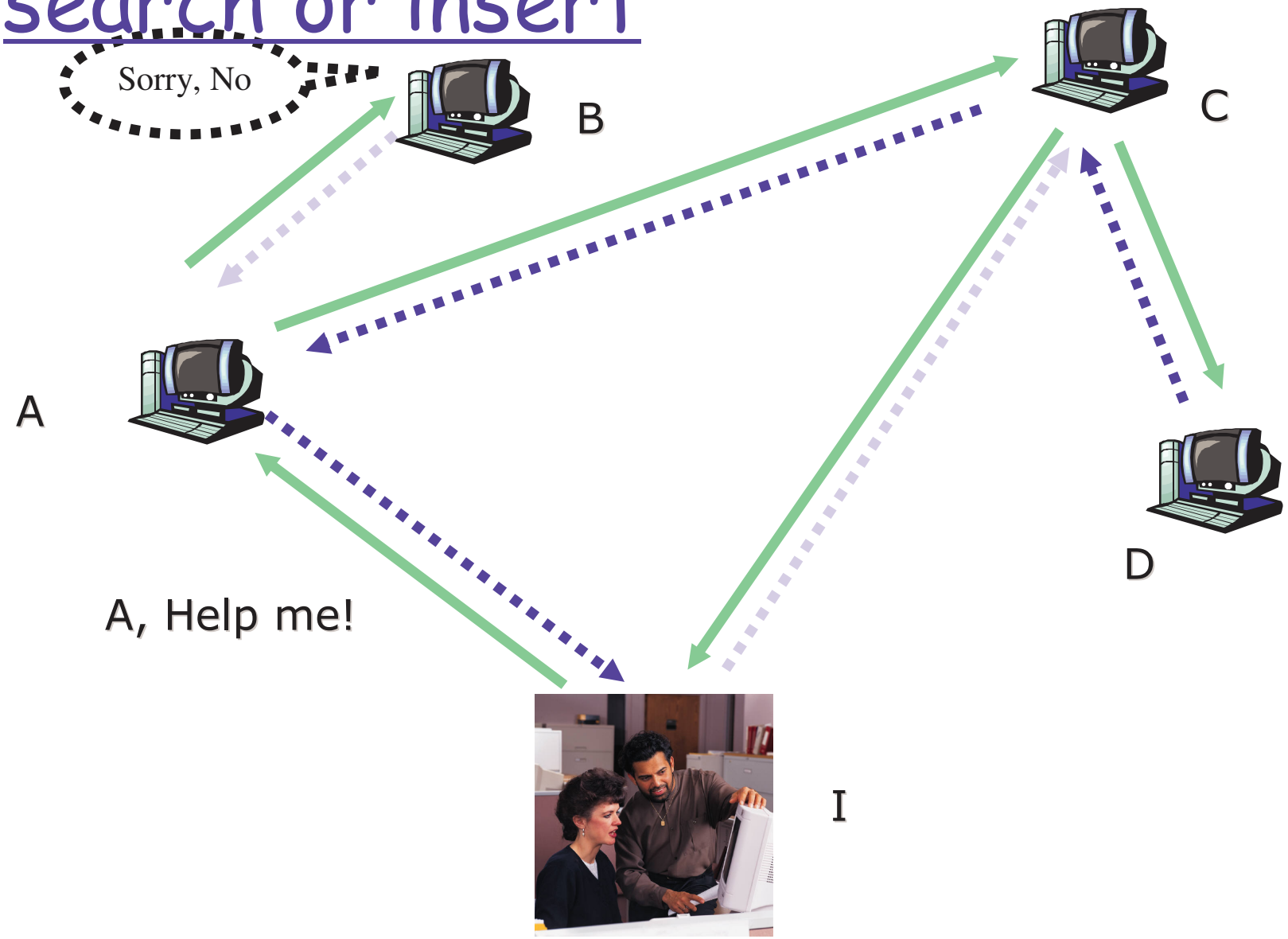
Freenet: Routing Algorithm: search or insert

Freenet: Routing Algorithm: search or insert



I

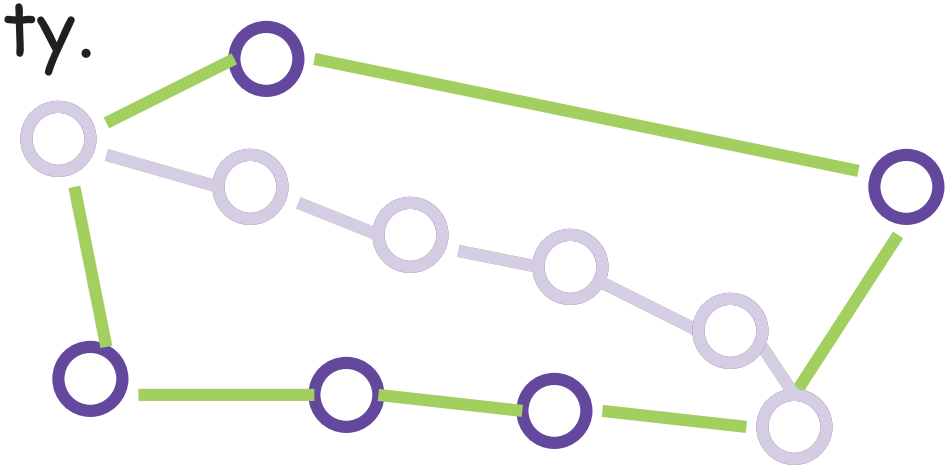
Freenet: Routing Algorithm: search or insert



Strength of routing algorithm(1)

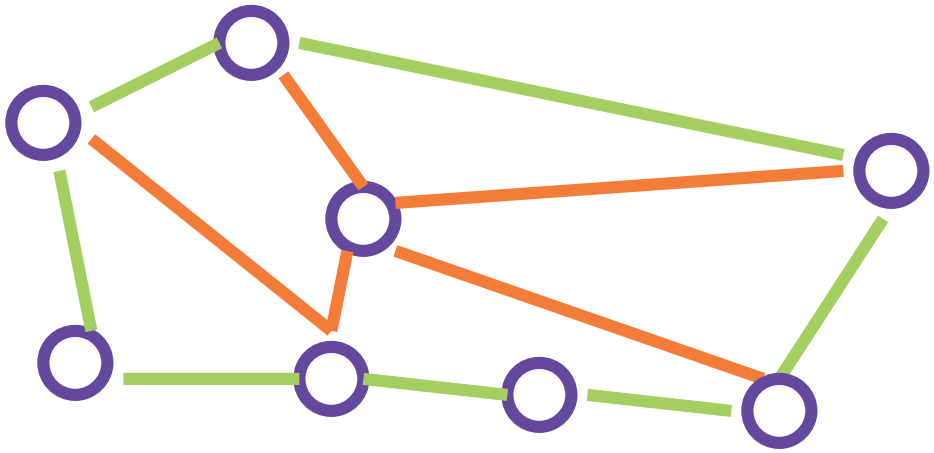
- Replication of Data Clustering (1)

  (Note: Not subject-clustering but key-clustering!)
- Reasonable Redundancy: improve data availability.

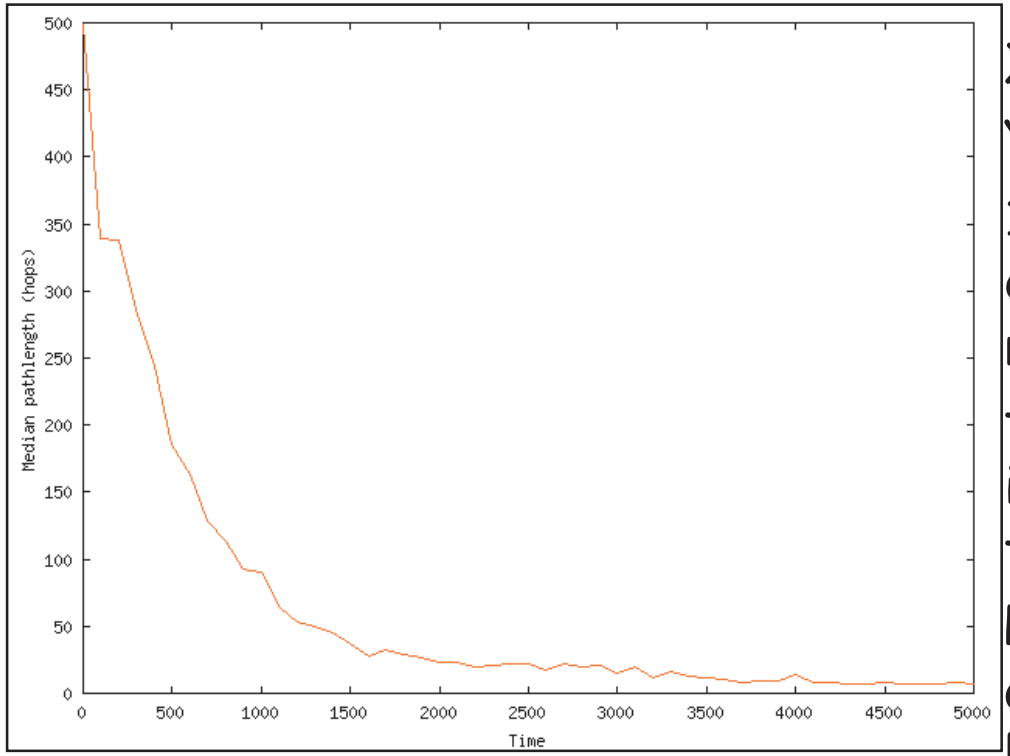


Strength of routing algorithm(2)

- New Entry in the Routing Table: the graph will be more and more connected. --- Node discovery



Network convergence



X-axis: time

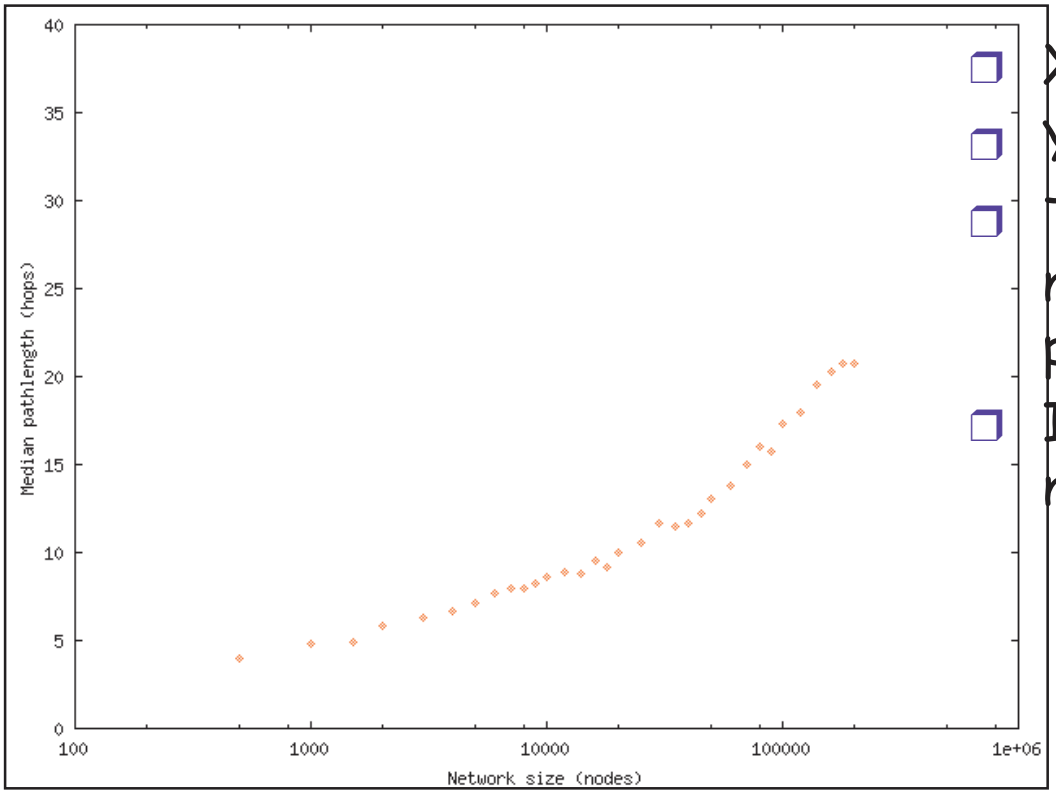
Y-axis: # of pathlength

1000 Nodes, 50 items datastore, 250 entries routing table

the routing tables were initialized to ring-lattice topology

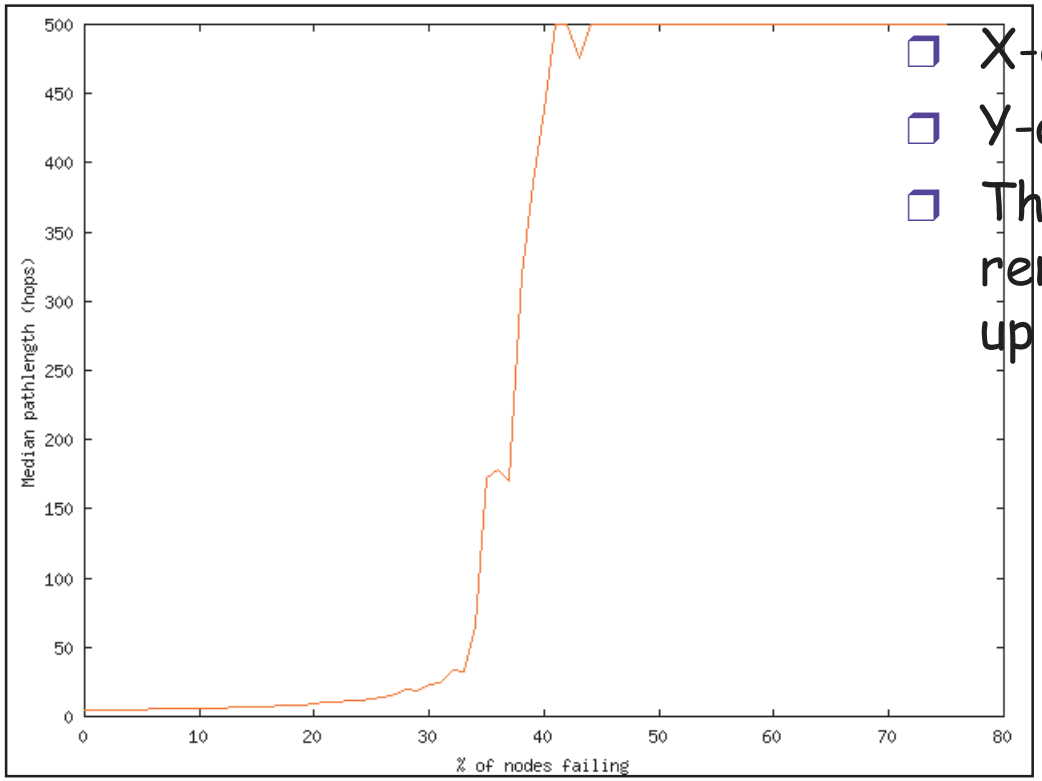
Pathlength: the number of hops actually taken before finding the data.

Scalability



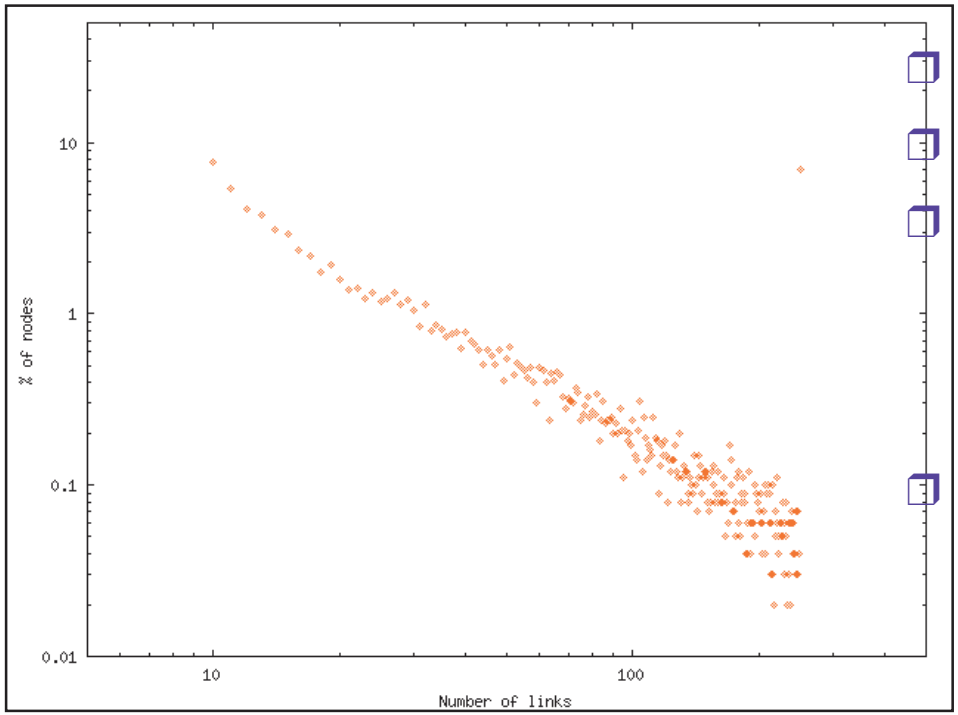
- X-axis: # of nodes
- Y-axis: # of pathlength
- The relation between network size and average pathlenth.
- Initially, 20 nodes. Add nodes regularly.

Fault Tolerance



- X-axis: # of nodes failing
- Y-axis: # of pathlength
- The median pathlength remains below 20 even when up to 30% nodes fails.

Small world Model



- X-axis: # of nodes failing
- Y-axis: # of pathlength
- Most of nodes have only few connections while a small number of news have large set of connections.
- The authors claim it follows power law.

So far, it's really a good model

- Keep anonymity
- Distributed model; data available
- Converge fast
- Adaptive

Is it Perfect?

- How long will it take to search or insert?
  - Trade off between anonymity and searching efforts:

  How do I know what key to use??
  - Can we come up a better algorithm? A good try: "Search in Power-Law Networks"
- Have no idea about if search fails due to no such document or just didn't find it.
- File lifetime. Freenet doesn't guarantee a document you submit today will exist tomorrow!!

Question??

- Anonymity? Security?
- Better search algorithm? Power law?
- …

P2P Characteristics

- Centralized model
  - e.g. Napster
  - global index held by central authority
  - direct contact between requestors and providers
- Decentralized model
  - e.g. Freenet, Gnutella
  - no global index – local knowledge only (approximate answers)
  - contact mediated by chain of intermediaries

Wrapup discussion questions (1):

- What is a peer-peer network (what is not a peer-to-peer network?). Necessary:
  - *every* node is designed to (but may not by user choice) *provide some service* that helps other nodes in the network get service
  - no 1-N service providing
  - each node potentially has the same responsibility, functionality (maybe nodes can be polymorphic)
    - corollary: by design, nothing (functionally) prevents two nodes from communicating directly
  - some applications (e.g., Napster) are a mix of peer-peer and centralized (lookup is centralized, file service is peer-peer) [recursive def. of peer-peer]
  - (logical connectivity rather than physical connectivity) routing will depend on service and data

Wrapup discussion questions (2):

- Why peer-peer over client-server?
  - A well-deigned p2p provides better "scaability"
- Why is client-server different from peer-peer?
  - peer-peer is harder to make reliable
  - availability different from client-server (p2p is more often at least partially "up")
  - more trust is required
- If all music were free in the future (and organized), would we have peer-peer.
  - Is there another app: ad hoc networking, any copyrighted data, peer-peer sensor data gathering and retrieval, simulation
- Evolution #101 – what can we learn about systems?