Computer Communications

Peer to Peer networking

Ack: Many of the slides are adaptations of slides by authors in the bibliography section.

p2p

- Quickly grown in popularity
  - numerous sharing applications
  - many million people worldwide use P2P networks

- But what is P2P in the Internet?
  - Searching or location?
  - Computers "Peering"?
  - Take advantage of resources at the *edges* of the network
    - End-host resources have increased dramatically
    - Broadband connectivity now common

Lecture outline

- Evolution of p2p networking
  - seen through file-sharing applications
- Other applications
- Multimedia

P2P Networks: file sharing

- Common Primitives:
  - **Join**: how do I begin participating?
  - **Publish**: how do I advertise my file?
  - **Search**: how to I find a file/service?
  - **Fetch**: how to I retrieve a file/use service?

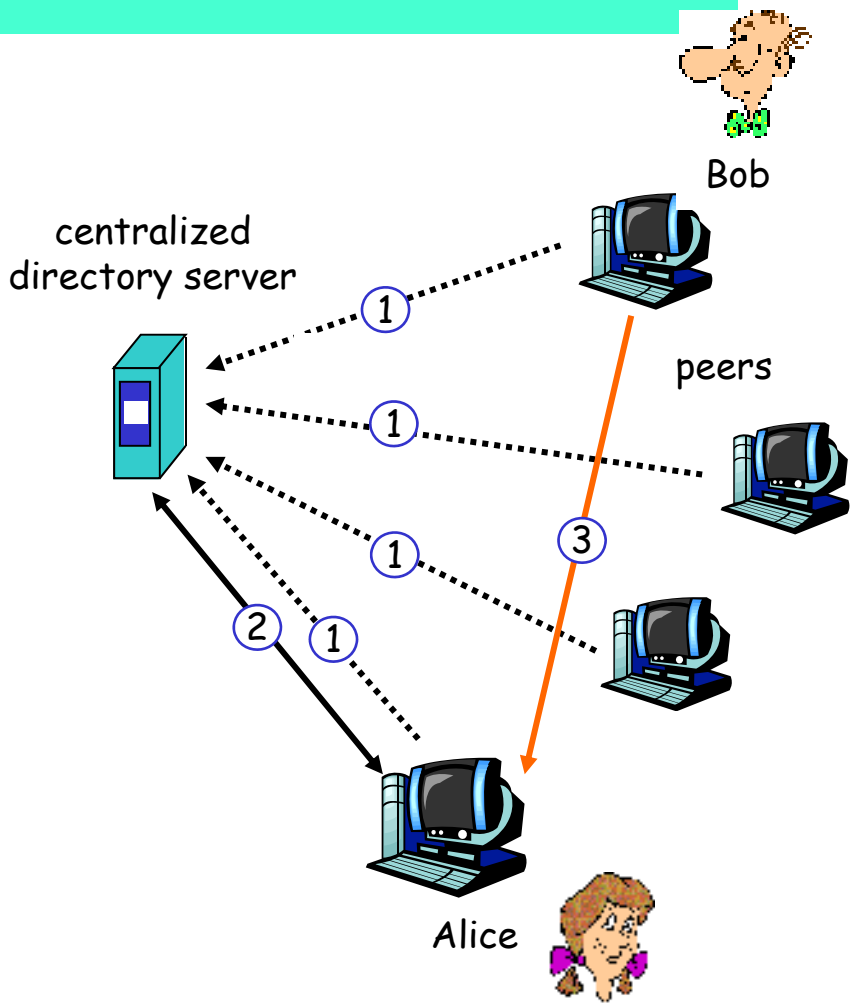
First generation in p2p file sharing/lookup

- **Centralized Database: single directory**
  - Napster
- **Query Flooding**
  - Gnutella
- **Hierarchical Query Flooding**
  - KaZaA
- **(Further unstructured Overlay Routing**
  - Freenet, …)
- **Structured Overlays**
  - …

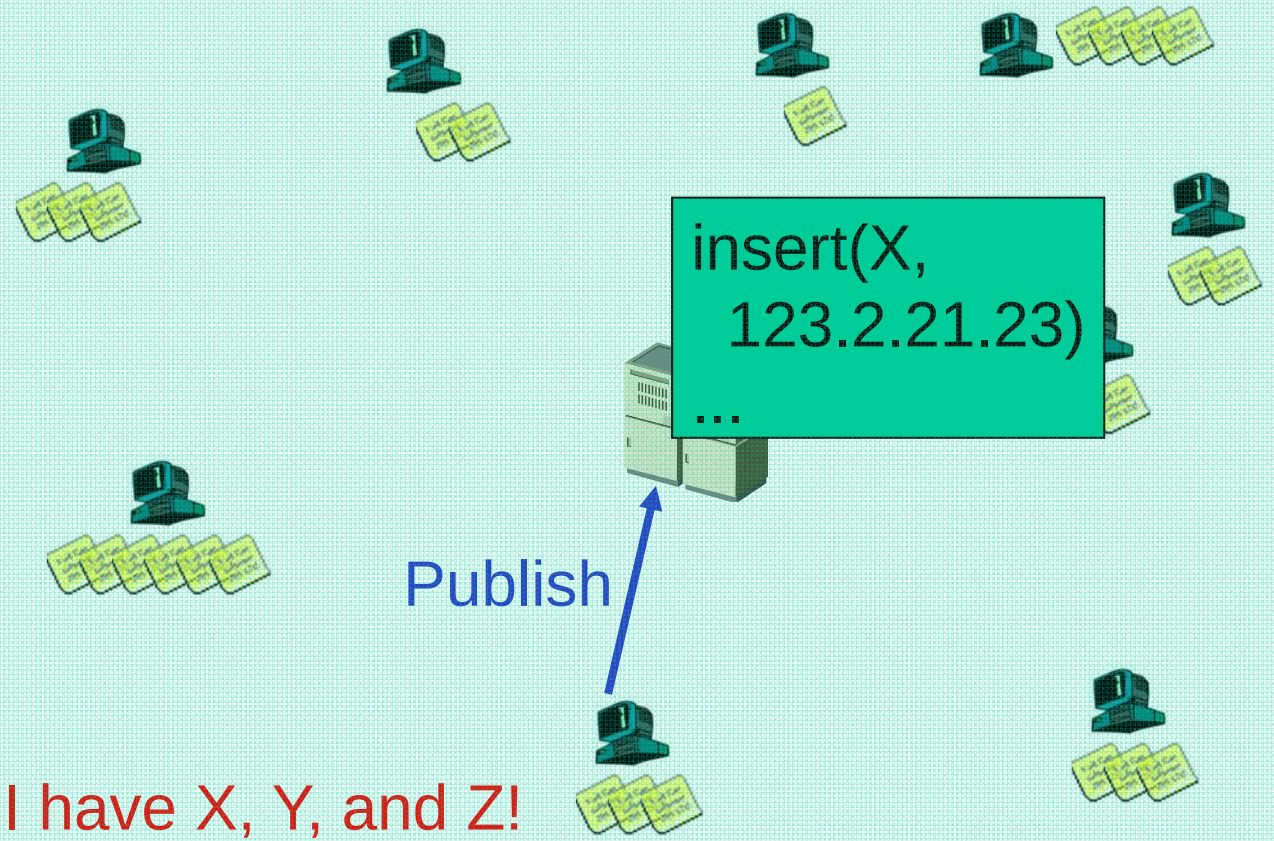
P2P: centralized directory

original "Napster" design (1999, S. Fanning)

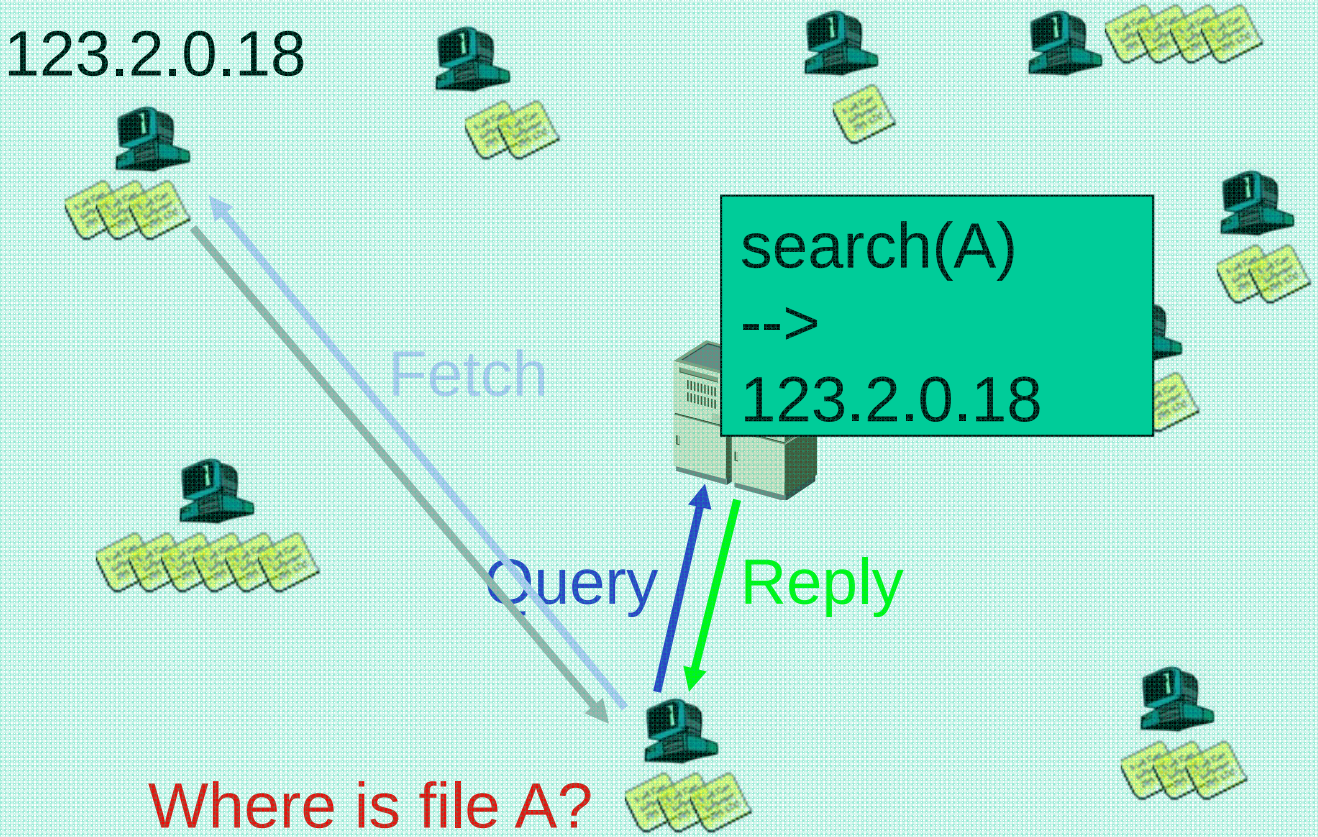
1) when peer connects, it informs central server:
   - IP address, content
2) Alice queries directory server for "Boulevard of Broken Dreams"
3) Alice requests file from Bob



Napster: Publish



Napster: Search



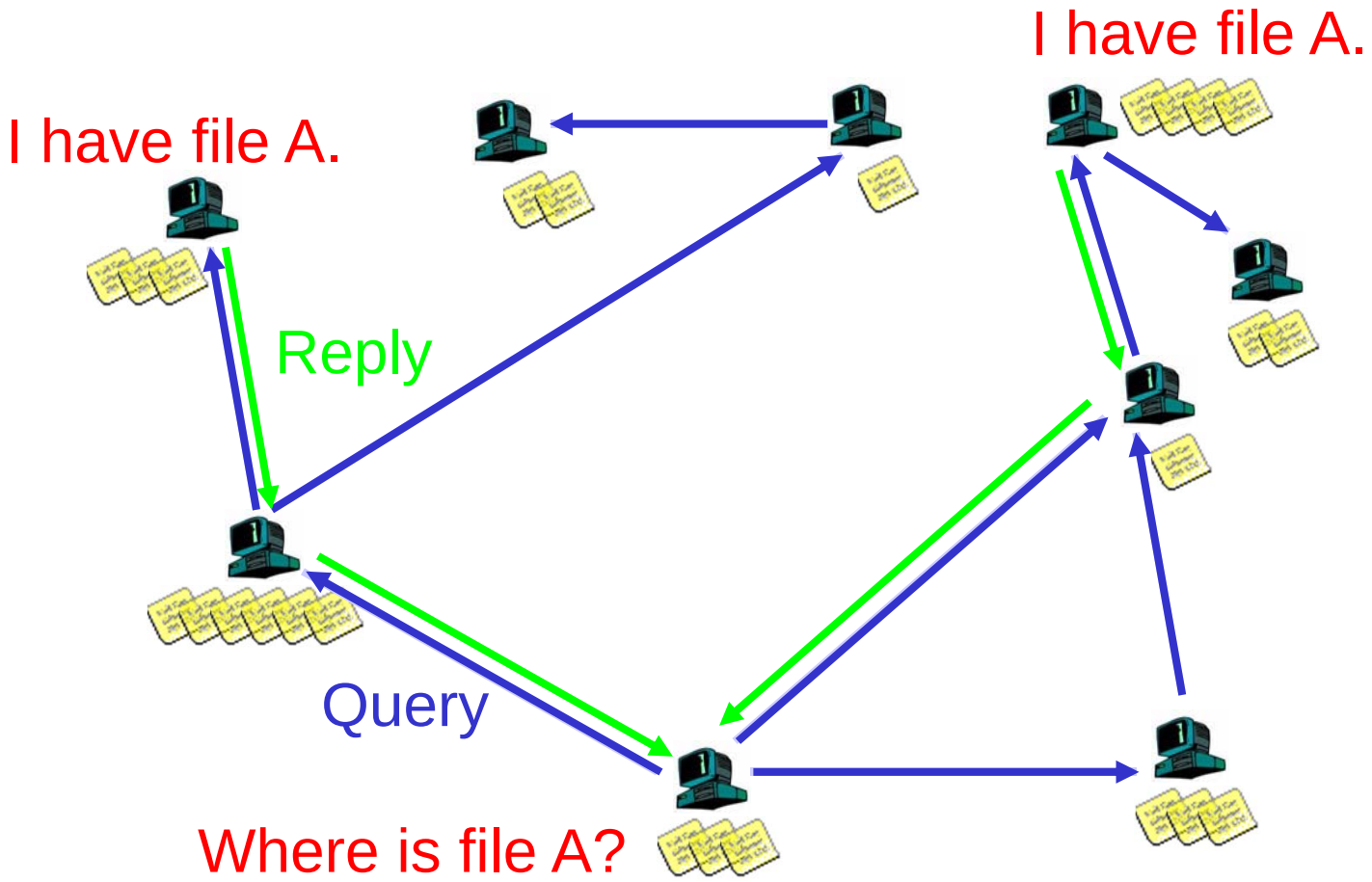
First generation in p2p file sharing/lookup

- Centralized Database
  - Napster
- Query Flooding: no directory
  - Gnutella
- Hierarchical Query Flooding
  - KaZaA
- (Further unstructured Overlay Routing
  - Freenet)
- Structured Overlays
  - ...

Gnutella: Overview

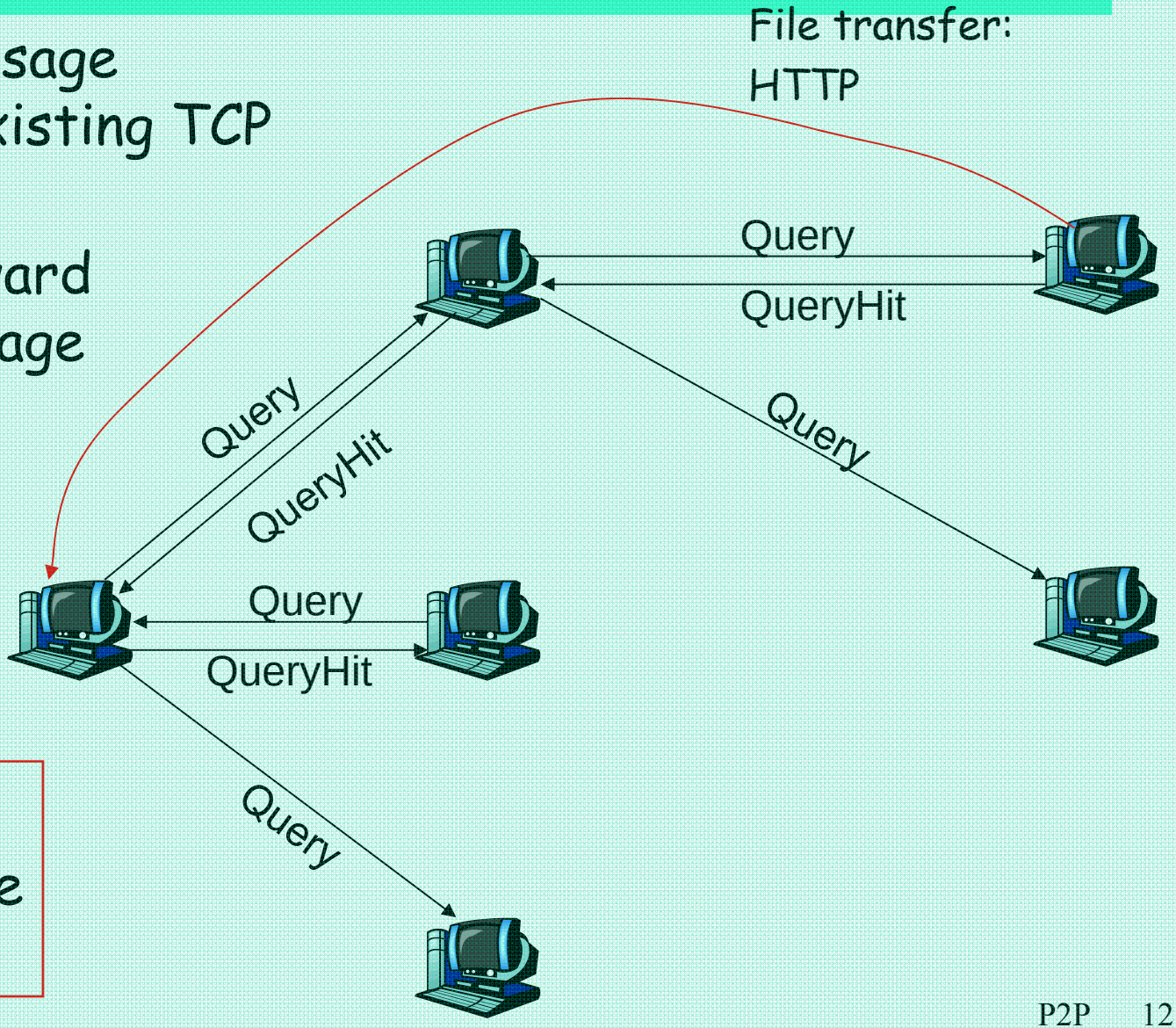
- Query Flooding:
  - **Join**: on startup, client contacts a few other nodes (learn from bootstrap-node); these become its "neighbors"
  - **Publish**: no need
  - **Search**: ask neighbors, who ask their neighbors, and so on... when/if found, reply to sender.
  - **Fetch**: get the file directly from peer

Gnutella: Search



Gnutella: protocol

- Query message sent over existing TCP connections
- peers forward Query message
- QueryHit sent over reverse path



Scalability:
limited scope flooding

Napsetr vs Gnutella: Discussion +, -?

- Pros:
  - Simple
  - Search scope is O(1)
- Cons:
  - Server maintains O(N) State
  - Server performance bottleneck
  - Single point of failure

- Pros:
  - Simple
  - Fully de-centralized
  - Search cost distributed
- Cons:
  - Search scope is O(N)
  - Search time is O(???)

Gnutella

Interesting concept in practice: overlay network:

active gnutella peers and edges form an overlay

- A network on top of another network:
  - Edge is not a physical link (what is it then?)

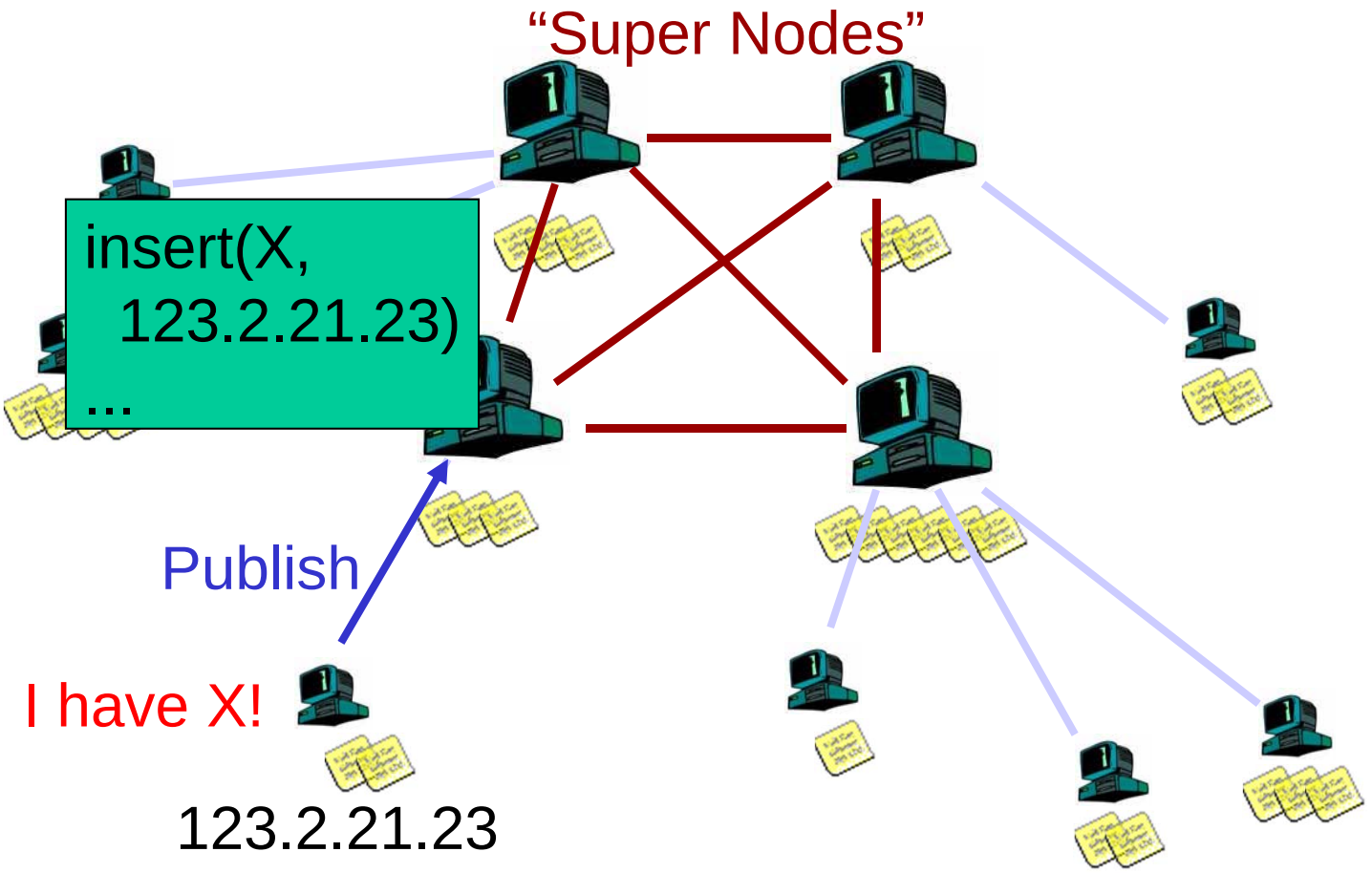
First generation in p2p file sharing/lookup

- **Centralized Database**
  - Napster
- **Query Flooding**
  - Gnutella
- **Hierarchical Query Flooding: some directories**
  - KaZaA
- **Further unstructured Overlay Routing**
  - Freenet
- ...

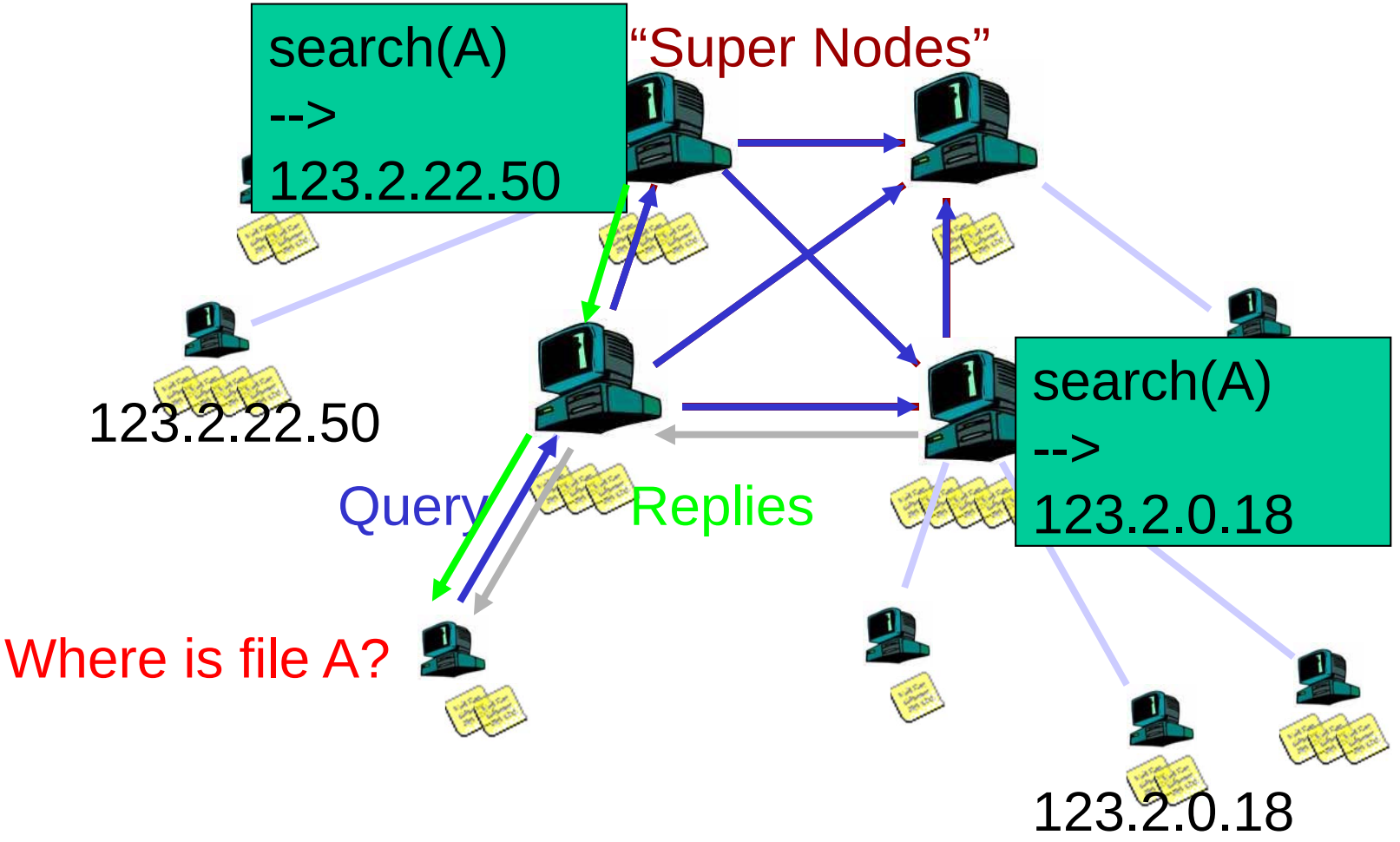
KaZaA: Overview

- "Smart" Query Flooding:
  - **Join**: on startup, client contacts a "supernode" ... may at some point become one itself
  - **Publish**: send list of files to supernode
  - **Search**: send query to supernode, supernodes flood query amongst themselves.
  - **Fetch**: get the file directly from peer(s); can fetch simultaneously from multiple peers

KaZaA: File Insert



KaZaA: File Search



KaZaA: Discussion

- Pros:
  - Tries to balance between search overhead and space needs
  - Tries to take into account node heterogeneity:
    - Bandwidth
    - Host Computational Resources
  - Rumored to take into account network locality
- Cons:
  - Still no real guarantees on search scope or search time

- P2P architecture used by Skype, Joost (communication, video distribution p2p systems)

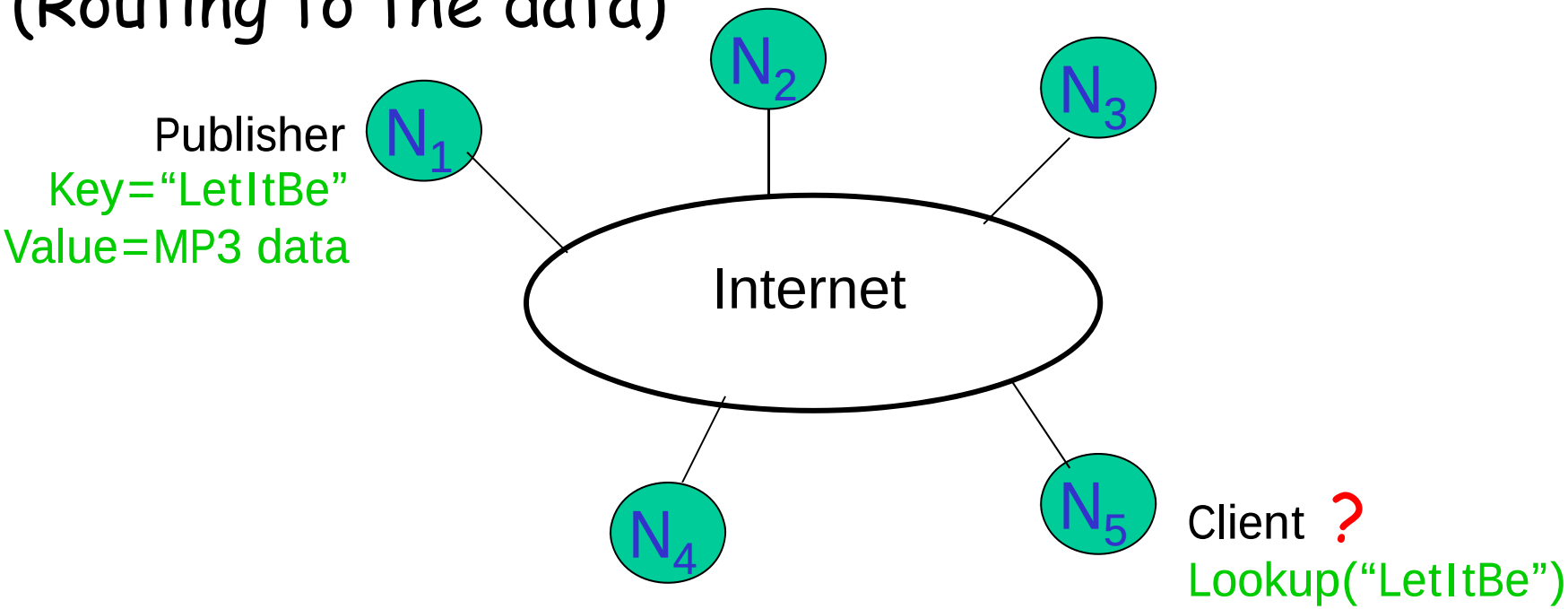
First steps in p2p file sharing/lookup

- **Centralized Database**
  - Napster
- **Query Flooding**
  - Gnutella
- **Hierarchical Query Flooding**
  - KaZaA
- **Further unstructured Overlay Routing**
  - Freenet: some directory, cache-like, based on recently seen targets; see literature pointers for more
- **Structured Overlay Organization and Routing**
  - Distributed Hash Tables
  - Combine database+distributed system expertise

Problem from this perspective

How to find data in a distributed file sharing system?

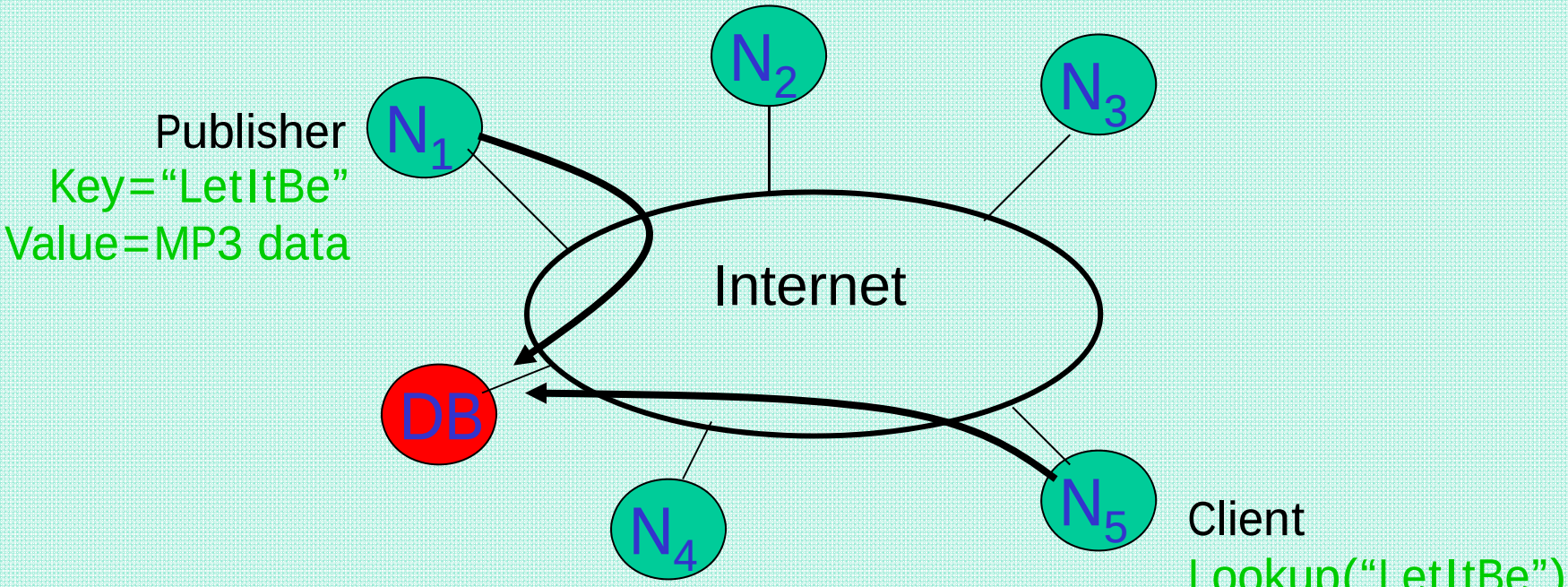
(Routing to the data)



Lookup is a key problem

Centralized Solution

Central server (Napster)



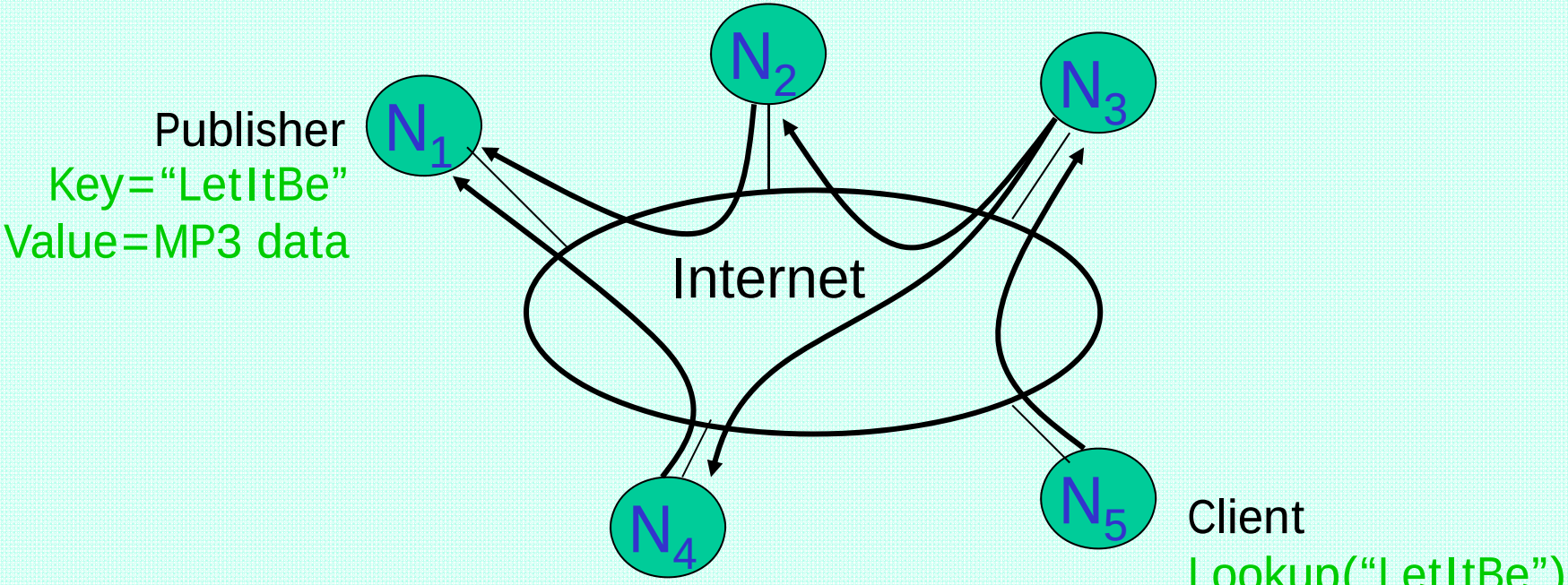
O(M) state at server, O(1) at client

O(1) search communication overhead

Single point of failure

Distributed Solution

Flooding (Gnutella, etc.)



O(1) state per node

Worst case O(E) messages per lookup

Distributed Solution
(some more structure? In-between the two?)

balance the update/lookup complexity..

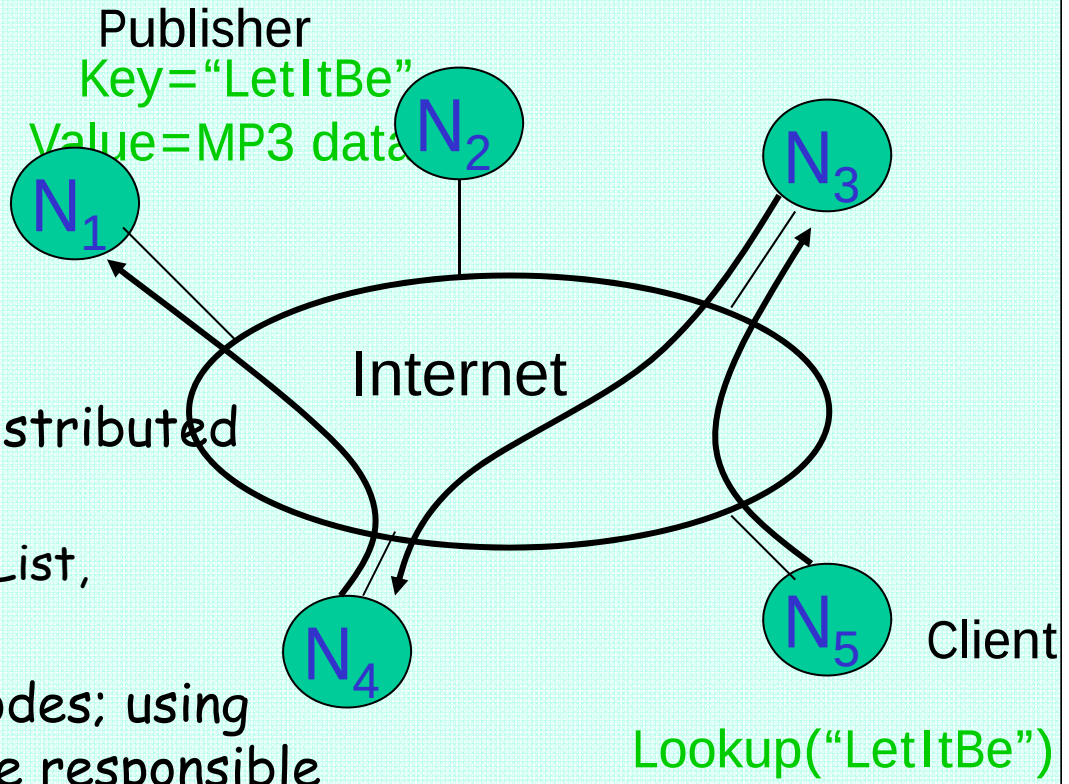
Abstraction: a distributed "hash-table" (DHT) data structure:

put(id, item);item = get(id);

Implementation: nodes form a distributed data structure

eg. Ring, Tree, Hypercube, SkipList, Butterfly.

Hash function maps entries to nodes; using the node structure, find the node responsible for item; that one knows where the item is



- >

Hash function maps entries to nodes; using the node structure, find the node responsible for item; that one knows where the item is

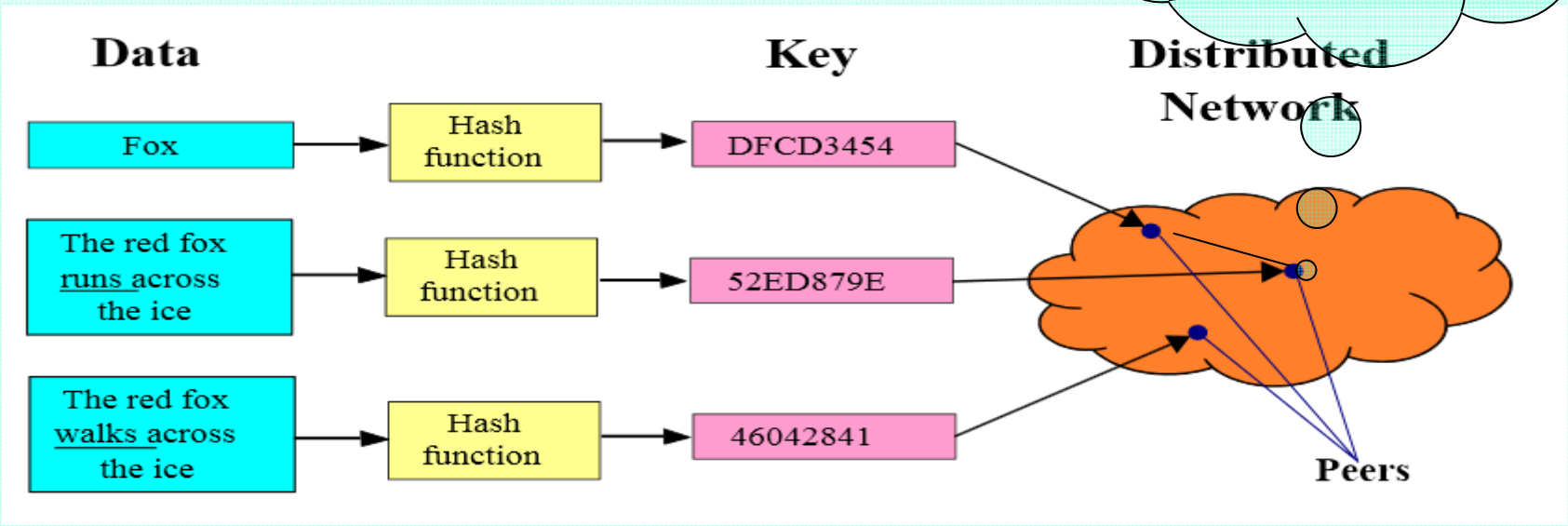


figure source: wikipedia

Challenges:

- Keep the hop count small
- Keep the routing tables "right size"
- Stay robust despite rapid changes in membership

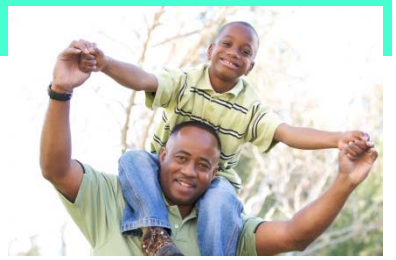
DHT: Overview

- Structured Overlay Routing:
  - **Join**: On startup, contact a "bootstrap" node and integrate yourself into the distributed data structure; get a *node id*
  - **Publish**: Route publication for *file id* toward an appropriate *node id* along the data structure
    - Need to think of updates when a node leaves
  - **Search**: Route a query for file id toward a close node id. Data structure guarantees that query will meet the publication.
  - **Fetch**: Two options:
    - Publication contains actual file => fetch from where query stops
    - Publication says "I have file X" => query tells you 128.2.1.3 has X, use http or similar (i.e. rely on IP routing) to get X from 128.2.1.3

DHT: Comments/observations?

- think about structure maintenance/benefits

Next generation in p2p netwoking



- **Swarming**
  - BitTorrent, Avalanche, …
- …

BitTorrent: Next generation fetching

- In 2002, B. Cohen debuted BitTorrent
- Key Motivation:
  - Popularity exhibits temporal locality (Flash Crowds)
- **Focused on Efficient *Fetching*, not *Searching*:**
  - Distribute the *same* file to groups of peers
  - Single publisher, multiple downloaders
- Used by publishers to distribute software, other large files

- http://vimeo.com/15228767

BitTorrent: Overview

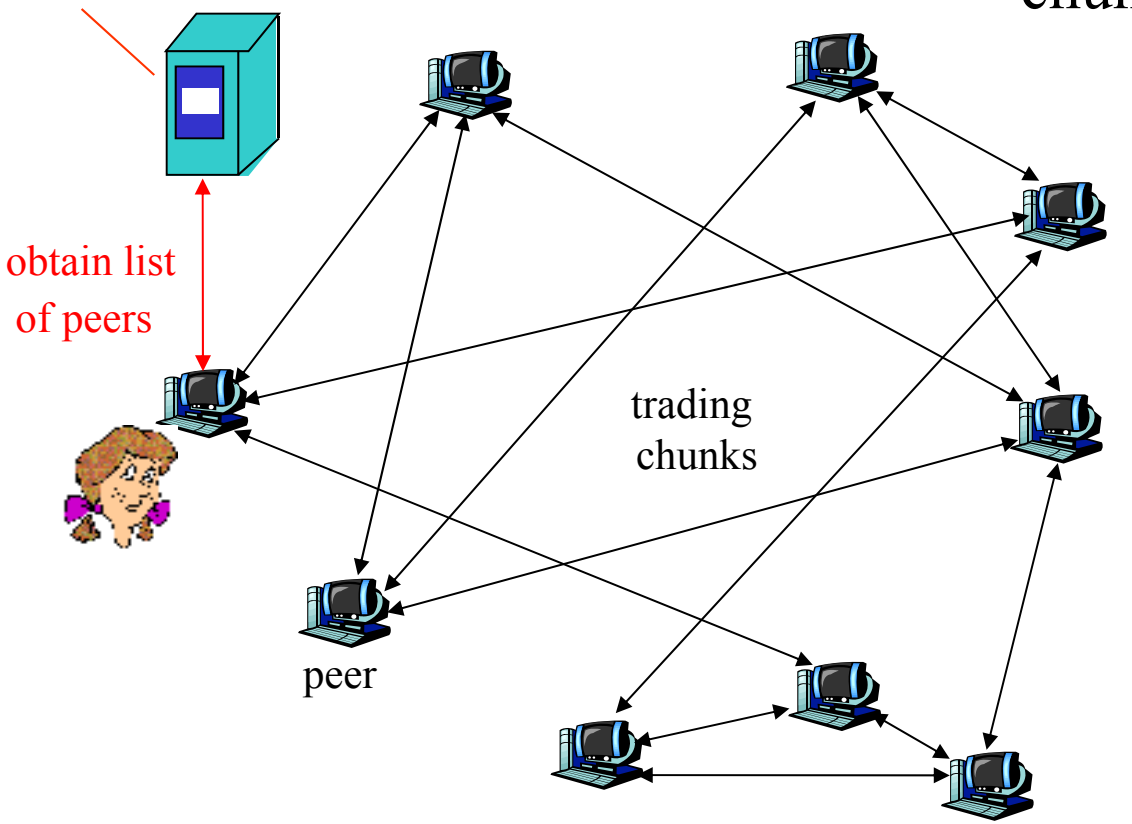
- Swarming:
  - **Join**: contact centralized "tracker" server, get a list of peers.
  - **Publish**: can run a tracker server.
  - **Search**: Out-of-band. E.g., use Google, some DHT, etc to find a tracker for the file you want. Get list of peers to contact for assembling the file in chunks
  - **Fetch**: Download chunks of the file from your peers. Upload chunks you have to them.

File distribution: BitTorrent

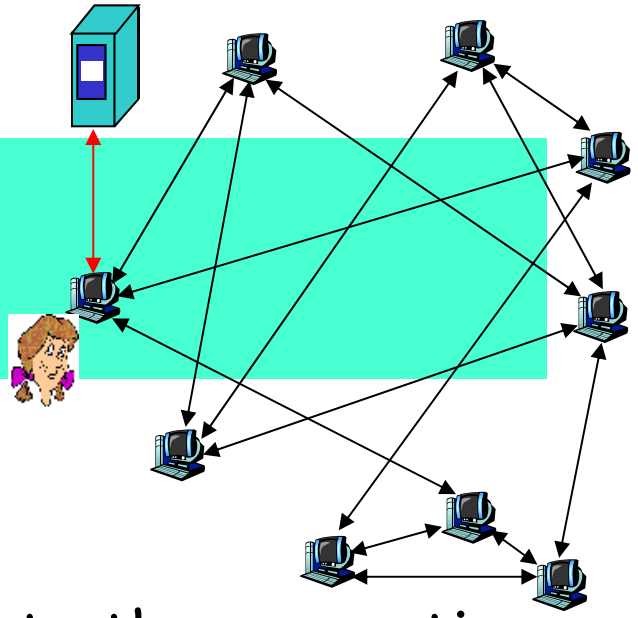
P2P file distribution

tracker: tracks peers participating in torrent

torrent: group of peers exchanging chunks of a file



BitTorrent (1)



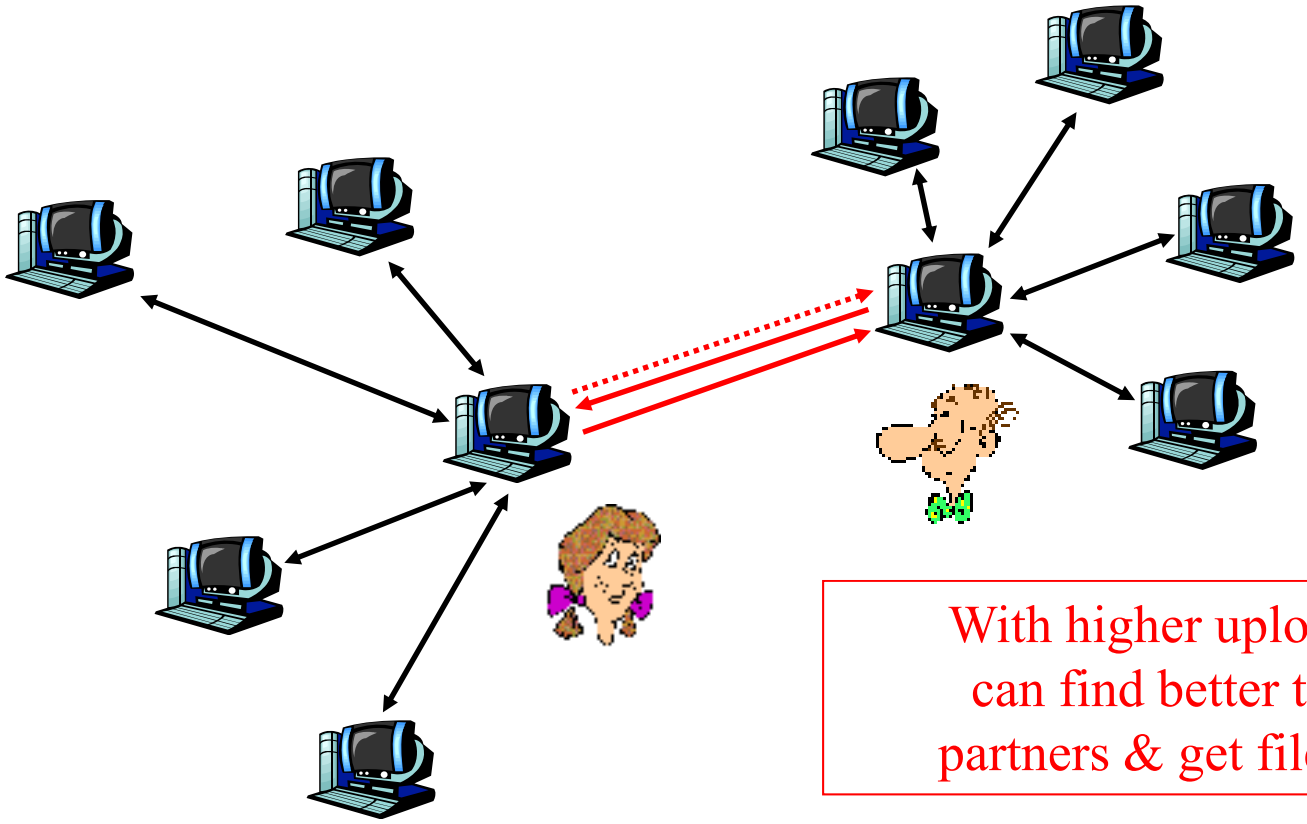
- file divided into *chunks*.
- peer joining torrent:
  - has no chunks, but will accumulate them over time
  - registers with tracker to get list of peers, connects to subset of peers ("neighbors")
- while downloading, peer uploads chunks to other peers.
- peers may come and go
- once peer has entire file, it may (selfishly) leave or (altruistically) remain

BitTorrent: Tit-for-tat

(1) Alice "optimistically unchokes" Bob

(2) Alice becomes one of Bob's top-four providers; Bob reciprocates

(3) Bob becomes one of Alice's top-four providers



With higher upload rate, can find better trading partners & get file faster!

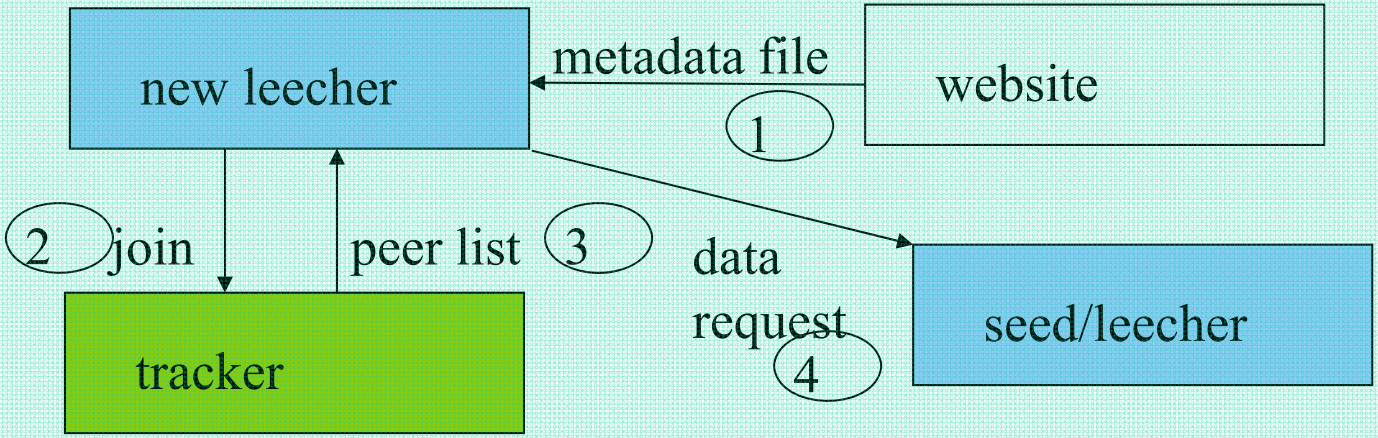
BitTorrent (2)

Pulling Chunks

- at any given time, different peers have different subsets of file chunks
- periodically, a peer (Alice) asks each neighbor for list of chunks that they have.
- Alice sends requests for her missing chunks
  - rarest first

- Sending Chunks: tit-for-tat
- Alice sends chunks to (4) neighbors currently sending her chunks *at the highest rate*
  - re-evaluate top 4 every 10 secs
- every 30 secs: randomly select another peer, starts sending chunks
  - newly chosen peer may join top 4
  - "optimistically unchoke"

BitTorrent – joining a torrent

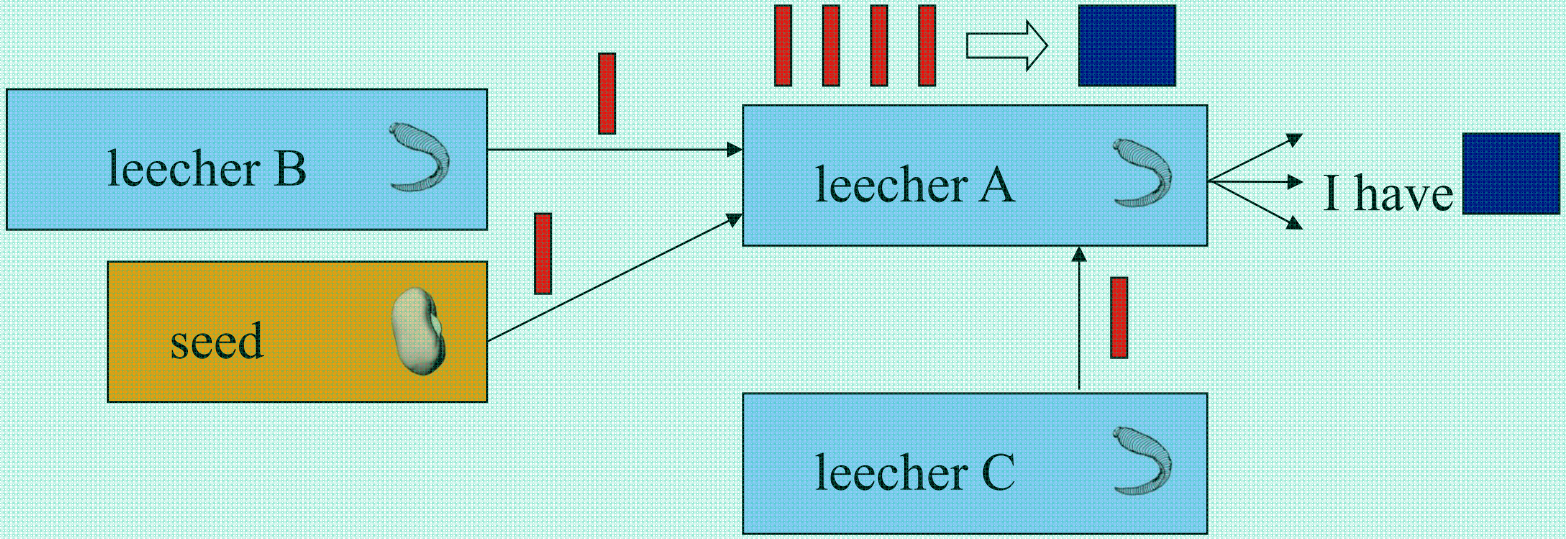


Peers divided into:

- *seeds*: have the entire file
- *leechers*: still downloading

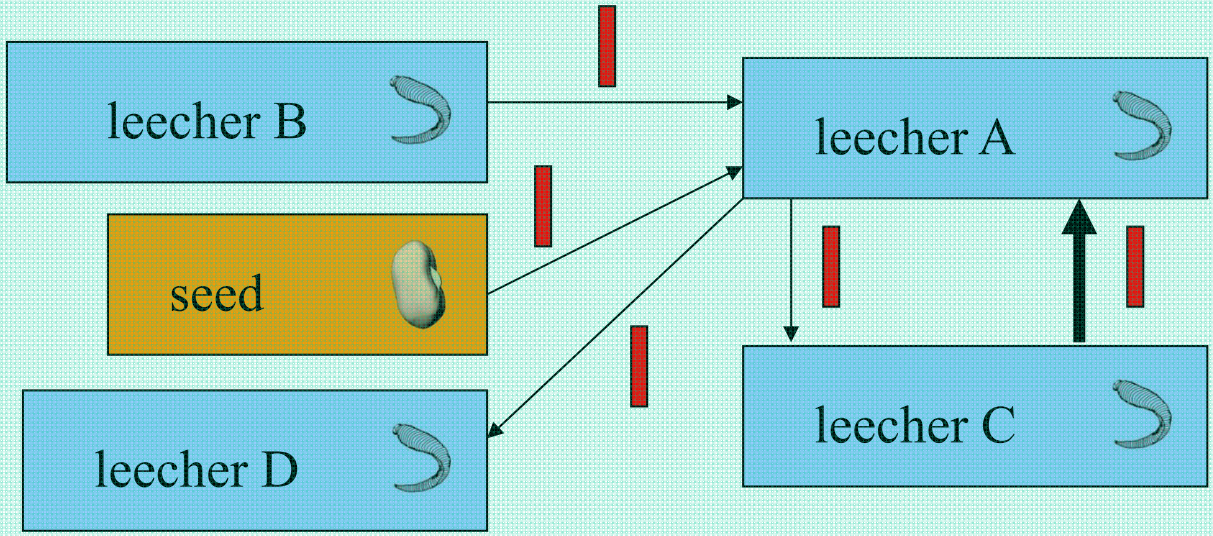
1. obtain the *metadata file*
2. contact the *tracker*
3. obtain a *peer list* (contains seeds & leechers)
4. contact peers from that list for data

BitTorrent - exchanging data



- Verify *pieces* using hashes
- Download sub-pieces *in parallel*
- *Advertise* received pieces to the entire peer list
- Look for the *rarest* pieces

BitTorrent - unchoking



- Periodically calculate data-receiving rates
- Upload to (*unchoke*) the fastest downloaders
- *Optimistic unchoking*
  - periodically select a peer at random and upload to it
  - continuously look for the fastest partners

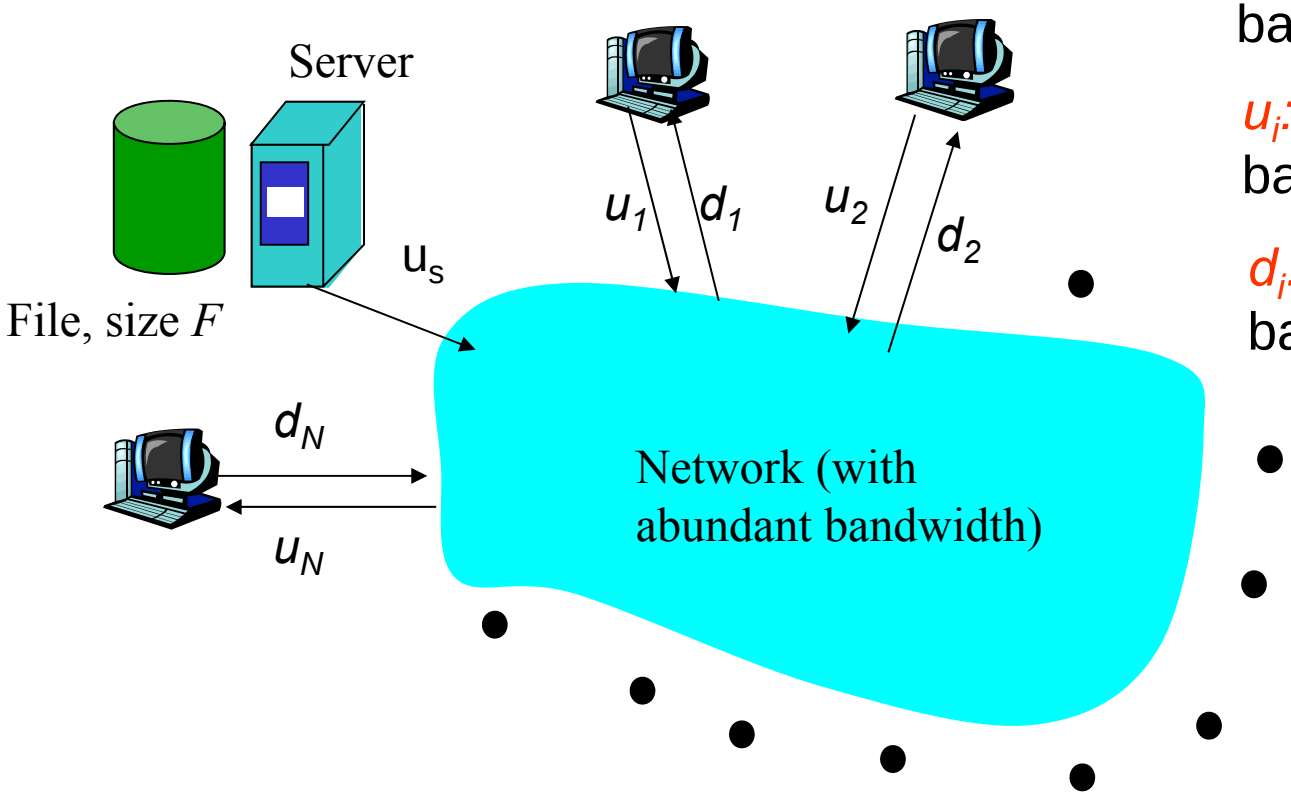
BitTorrent: Discussion

- Works reasonably well in practice
- Gives peers incentive to share resources; tries to avoids freeloaders

Efficiency/scalability through peering (collaboration)

File Distribution: Server-Client vs P2P

Question : How much time to distribute file from one server to *N peers*?



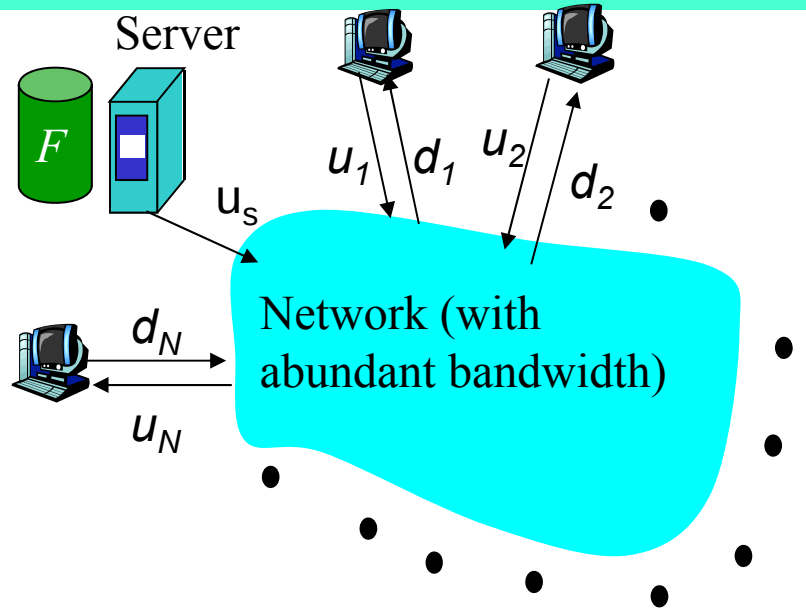
u_s: server upload bandwidth

u_i: peer i upload bandwidth

d_i: peer i download bandwidth

File distribution time: server-client

- server sequentially sends N copies:
  - NF/u_s time
- client i takes F/d_i time to download



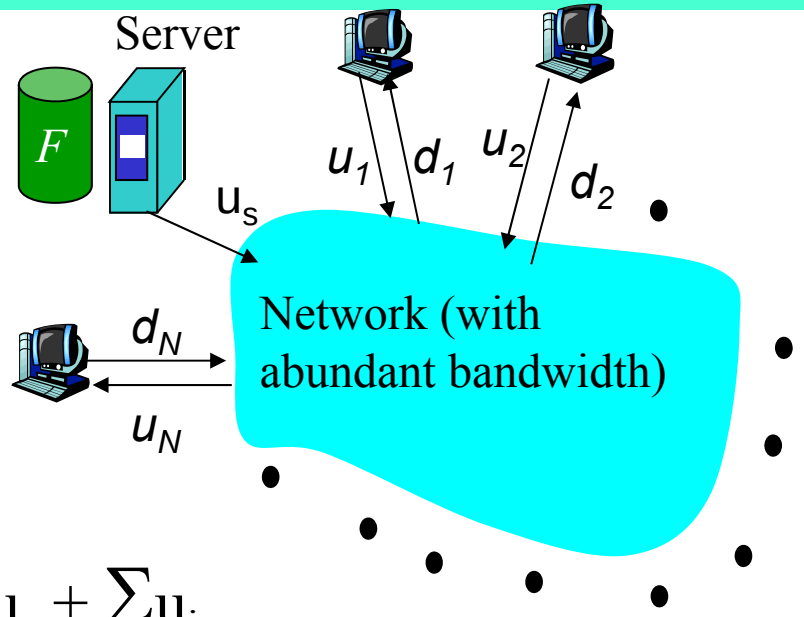
Time to distribute F to N clients using client/server approach = d_{cs} depends on

$$\max \{ NF/u_s, F/min(d_i)_i \}$$

increases linearly in N (for large N)

File distribution time: P2P

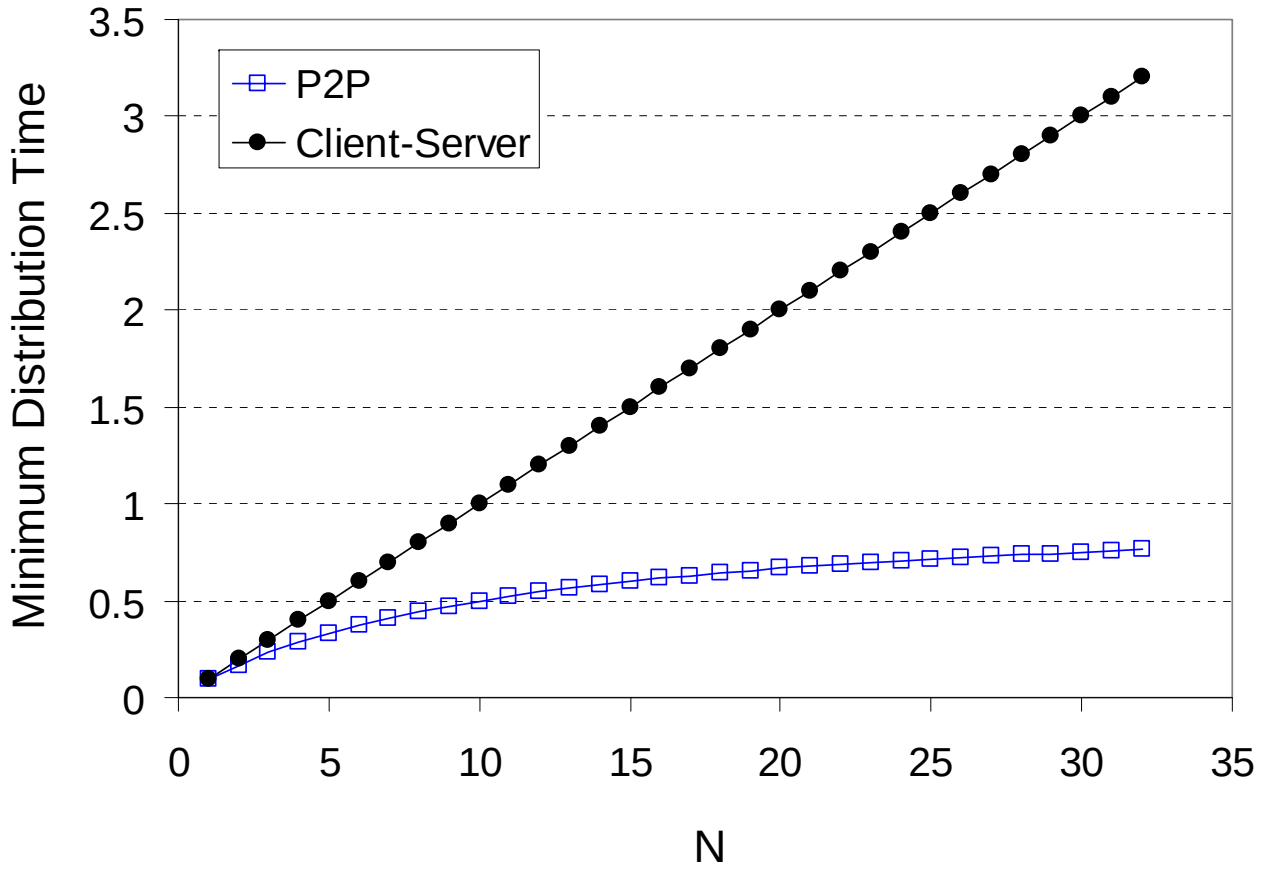
- server must send one copy: F/u_s time
- client i takes F/d_i time to download
- NF bits must be downloaded (aggregate)
  - fastest possible upload rate: $u_s + \Sigma u_i$



d_{P2P} depends on max $\{ F/u_s, F/min(d_i) , NF/(u_s + \sum_i u_i) \}$

Server-client vs. P2P: example

Client upload rate = u, F/u = 1 hour, $u_s = 10u$, $d_{min} \geq u_s$



Lecture outline

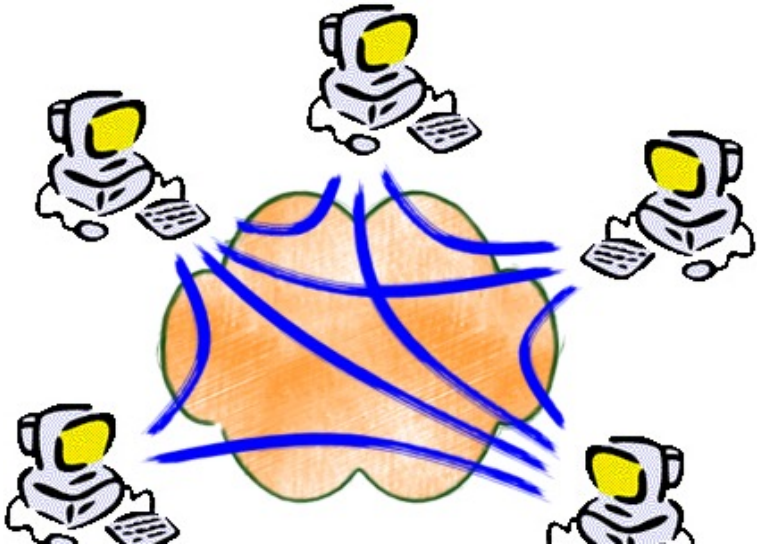
- Evolution of p2p networking
  - seen through file-sharing applications
- Other applications

P2P – not only sharing files…

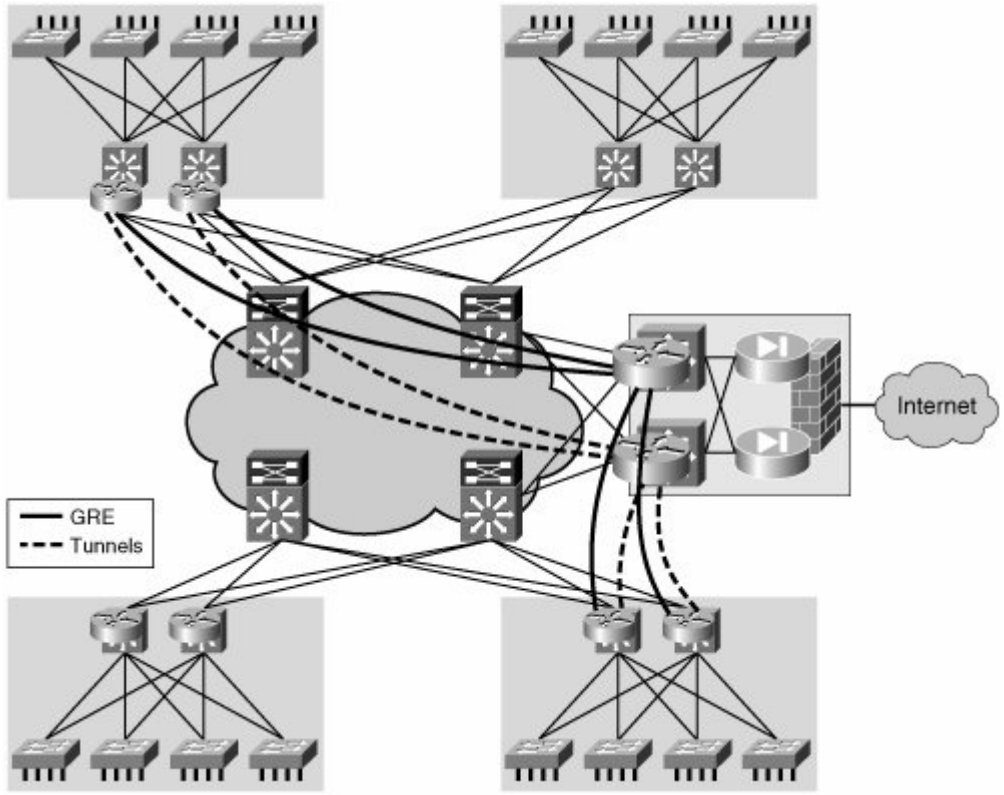
- Content delivery, software publication
- Streaming media applications
- Distributed computations (volunteer computing)
- Portal systems
- Distributed search engines
- Collaborative platforms
- Communication networks
- Social applications
- Other overlay-related applications….

Overlay: a network implemented on top of a network

- E.g. Peer-to-peer networks, "backbones" in adhoc networks, transportaiton network overlays, electricity grid overlays …

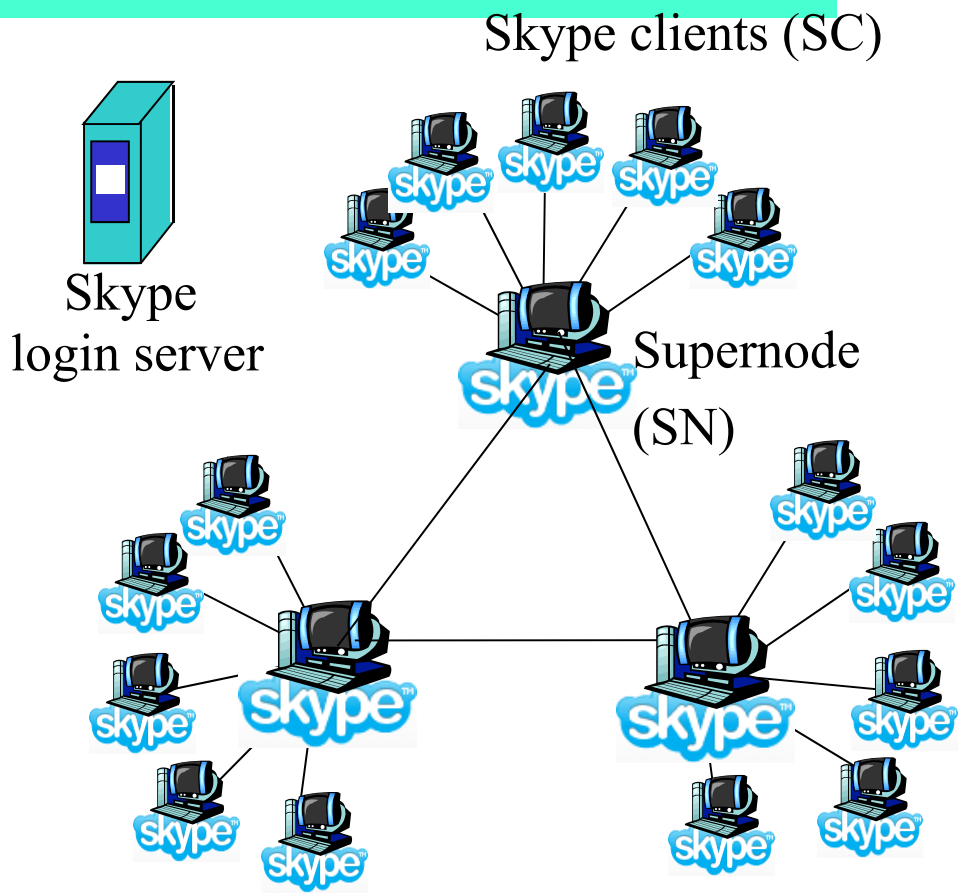


Router Overlays for e.g. protection/mitigation of flooding attacks



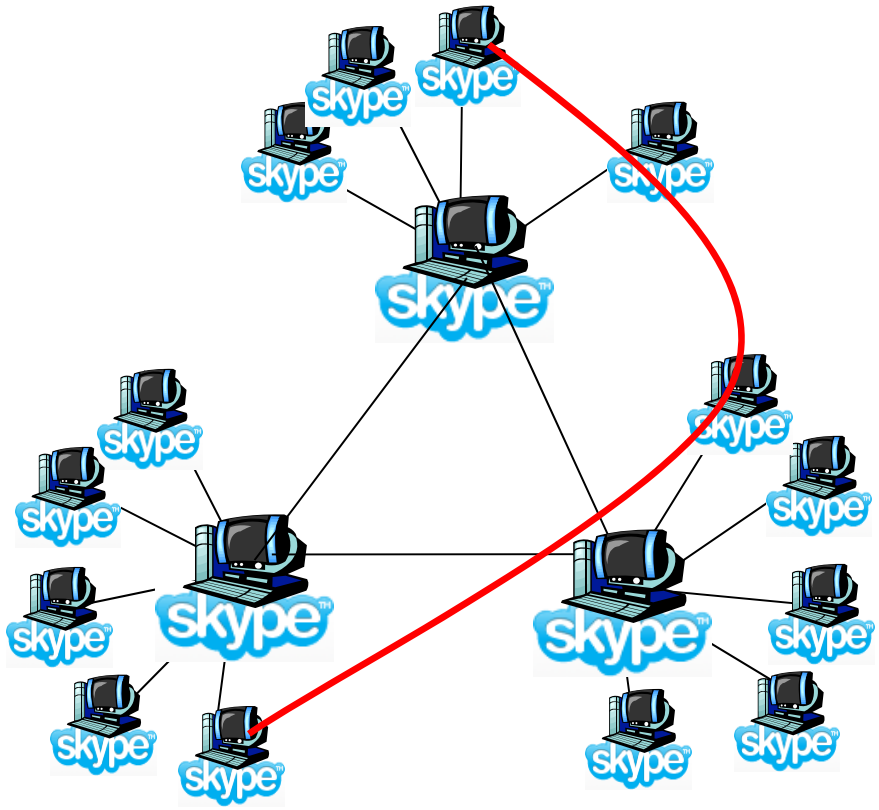
P2P Case study: Skype

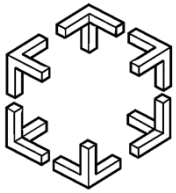
- inherently P2P: pairs of users communicate.
- proprietary application-layer protocol (inferred via reverse engineering)
- hierarchical overlay with SNs
- Index maps usernames to IP addresses; distributed over SNs



Peers as relays

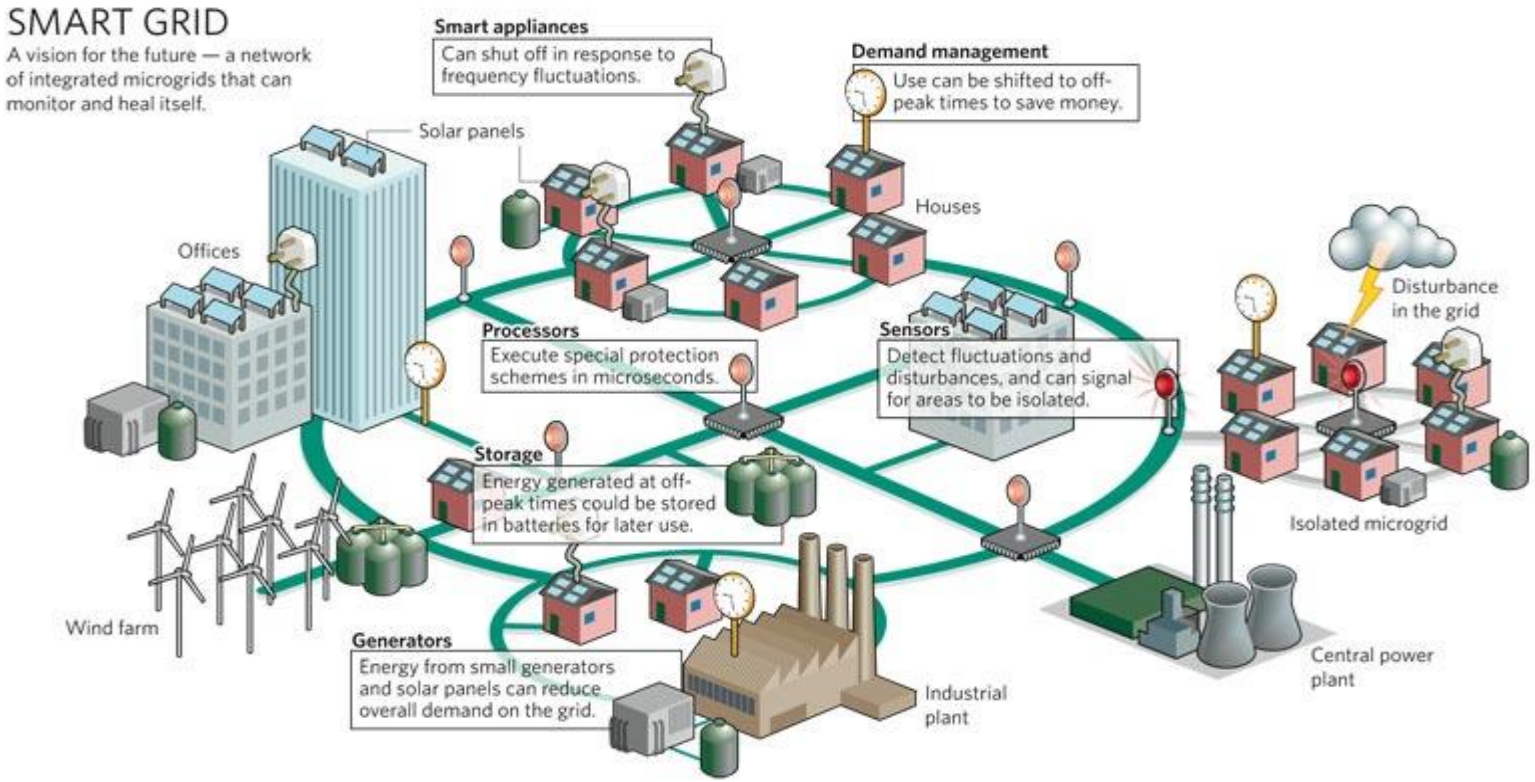
- Problem when both Alice and Bob are behind "NATs".
  - NAT prevents an outside peer from initiating a call to insider peer
- Solution:
  - Using Alice's and Bob's SNs, Relay is chosen
  - Each peer initiates session with relay.
  - Peers can now communicate through NATs via relay





More examples: a story in progress...

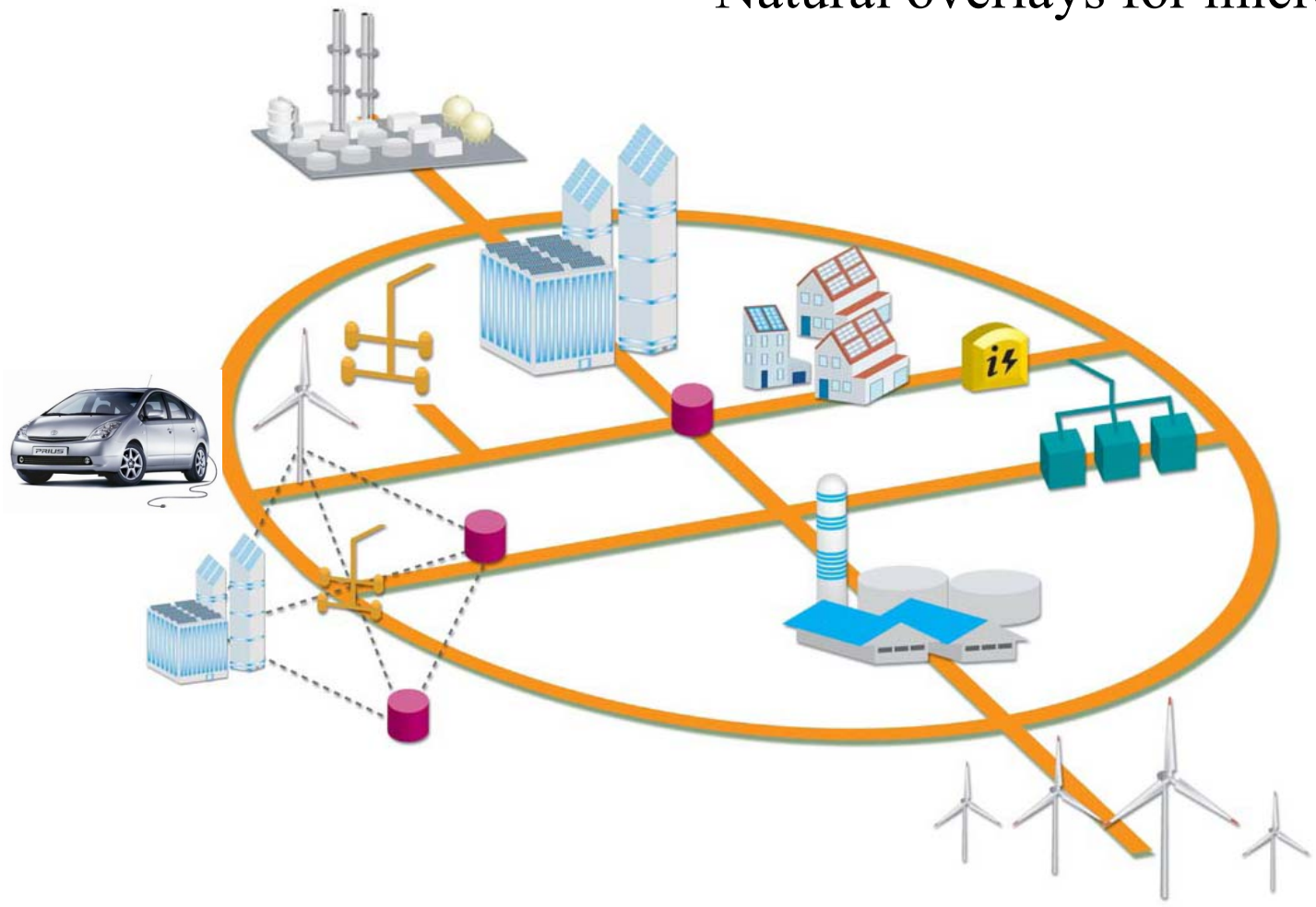
New power grids: be adaptive!



- Bidirectional power and information flow
  - Micro-producers or "prosumers", can share resources
  - Distributed energy resources
- Communication + resource-administration (distributed system) layer
  - aka "smart" grid

New power grids

Natural overlays for microgrids



Interested in project on the topic?

- To express interest, send email
  - <ptrianta>@ chalmers.se
  - Incl. Motivation and info on you (courses, interests, some own work)
  - Research course in preparation, ac. year 2012-2013

Bibliography

Collective sources

- Kurose, Ross: Computer Networking, a top-down approach, p2p in applications chapter AdisonWesley 2009
- Aberer's coursenotes
  - http://lsirwww.epfl.ch/courses/dis/2007ws/lecture/week%208%20P2P%20systems-general.pdf
  - http://lsirwww.epfl.ch/courses/dis/2007ws/lecture/week%209%20Structured%20Overlay%20Networks.pdf

Papers for Further Study

- Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications, Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. ACM SIGCOMM 2001, San Deigo, CA, August 2001, pp. 149-160.
- Kademlia: A Peer to peer information system Based on the XOR Metric. Petar Maymounkov and David Mazières , 1st International Workshop on Peer-to-peer Systems, 2002.
- Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems, A. Rowstron and P. Druschel, IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), November 2001.

Bibliography (cont)

- A Scalable Content-Addressable Network, S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, Sigcomm 2001, San Diego, CA, USA, August, 2001.
- Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. Int'l Workshop on Design Issues in Anonymity and Unobservability. LNCS 2009. Springer Verlag 2001.
- Viceroy: A Scalable and Dynamic Emulation of the Butterfly. By D. Malkhi, M. Naor and D. Ratajczak. In Proceedings of the 21st ACM Symposium on Principles of Distributed Computing (PODC '02), August 2002. Postscript.
- Incentives build Robustness in BitTorrent, Bram Cohen. Workshop on Economics of Peer-to-Peer Systems, 2003.
- Do incentives build robustness in BitTorrent? Michael Piatek, Tomas Isdal, Thomas Anderson, Arvind Krishnamurthy and Arun Venkataramani, NSDI 2007
- Exploiting *BitTorrent* For Fun (But Not Profit)
  *iptps06.cs.ucsb.edu/talks/Liogkas_**BitTorrent**.ppt*
- J. Mundinger, R. R. Weber and G. Weiss. Optimal Scheduling of Peer-to-Peer File Dissemination. Journal of Scheduling, Volume 11, Issue 2, 2008. [arXiv] [JoS]
- Christos Gkantsidis and Pablo Rodriguez, Network Coding for Large Scale Content Distribution, in *IEEE INFOCOM*, March 2005 (avalanche swarming)