Course on Computer Communication and Networks

Lecture 10
Chapter 2; peer-to-peer applications (and network overlays)

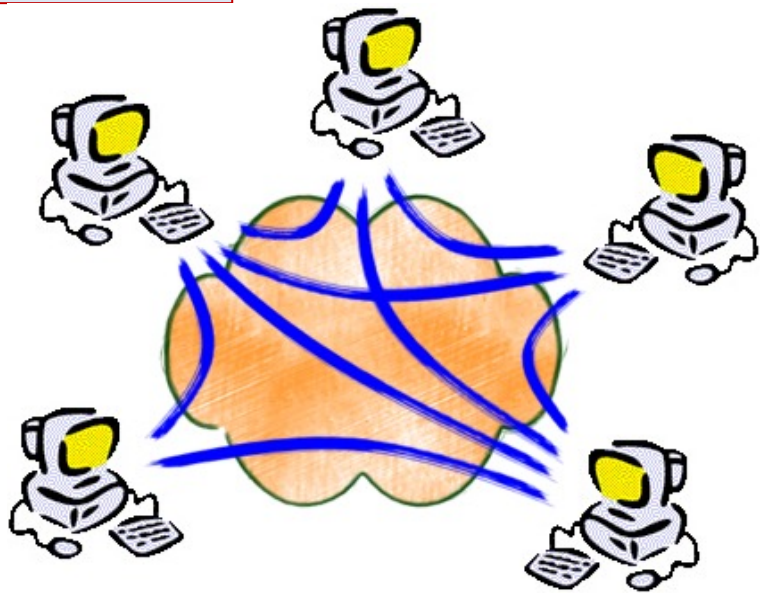
EDA344/DIT 420, CTH/GU

Based on the book Computer Networking: A Top Down Approach, Jim Kurose, Keith Ross, Addison-Wesley.

Network overlays

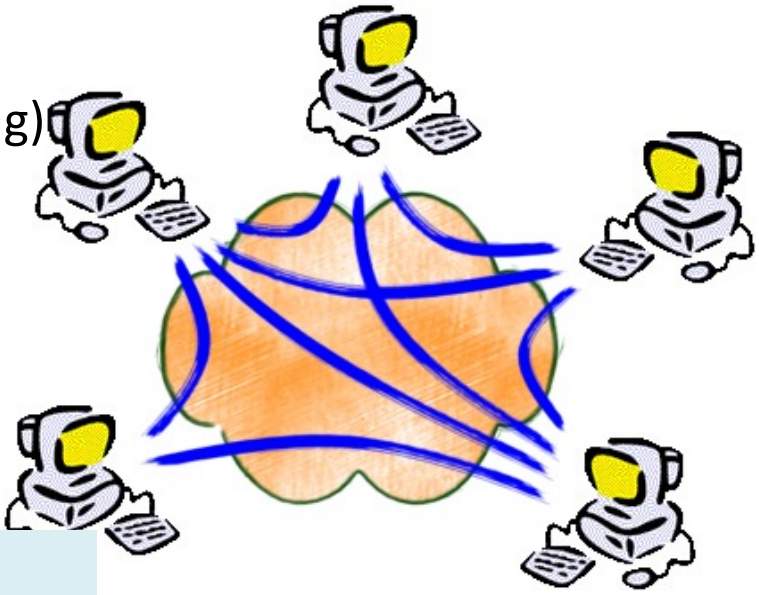
Overlay: a network implemented on top of a network

Why? What to do with this?



Overlay-based applications…

- Content delivery, software publication
- Streaming media applications
- Collaborative platforms
- Distributed computations (volunteer computing)
- Distributed search engines
- Social applications
- Emerging applications ….



Today's topic; overlay networking

– seen through file-sharing applications

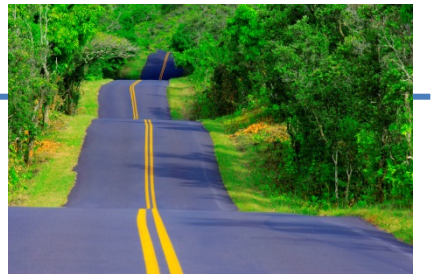
Other applications in next lecture(s)

Overlays in file-sharing peer-to-peer (p2p) applications: what for?

Background: Common Primitives in file-sharing p2p apps:

- **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?

Roadmap



First generation in p2p: file sharing/lookup

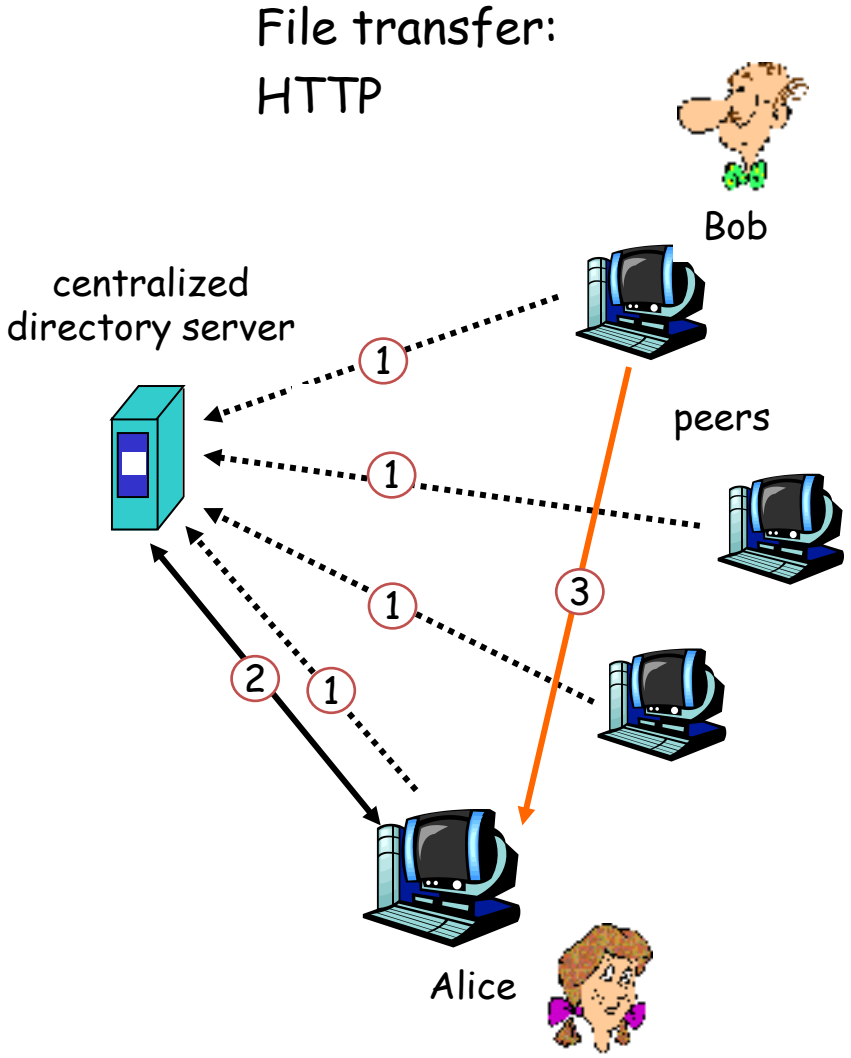
- Centralized Database: single directory
  - Napster
- Query Flooding
  - Gnutella
- Hierarchical Query Flooding
  - KaZaA
- Structured Overlays
  - DHT

Second generation in p2p ….

P2P: centralized directory

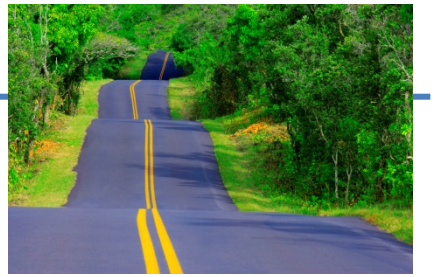
original "Napster" design (1999, S. Fanning)

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



Q: What is p2p in this?

Roadmap



First generation in p2p: file sharing/lookup

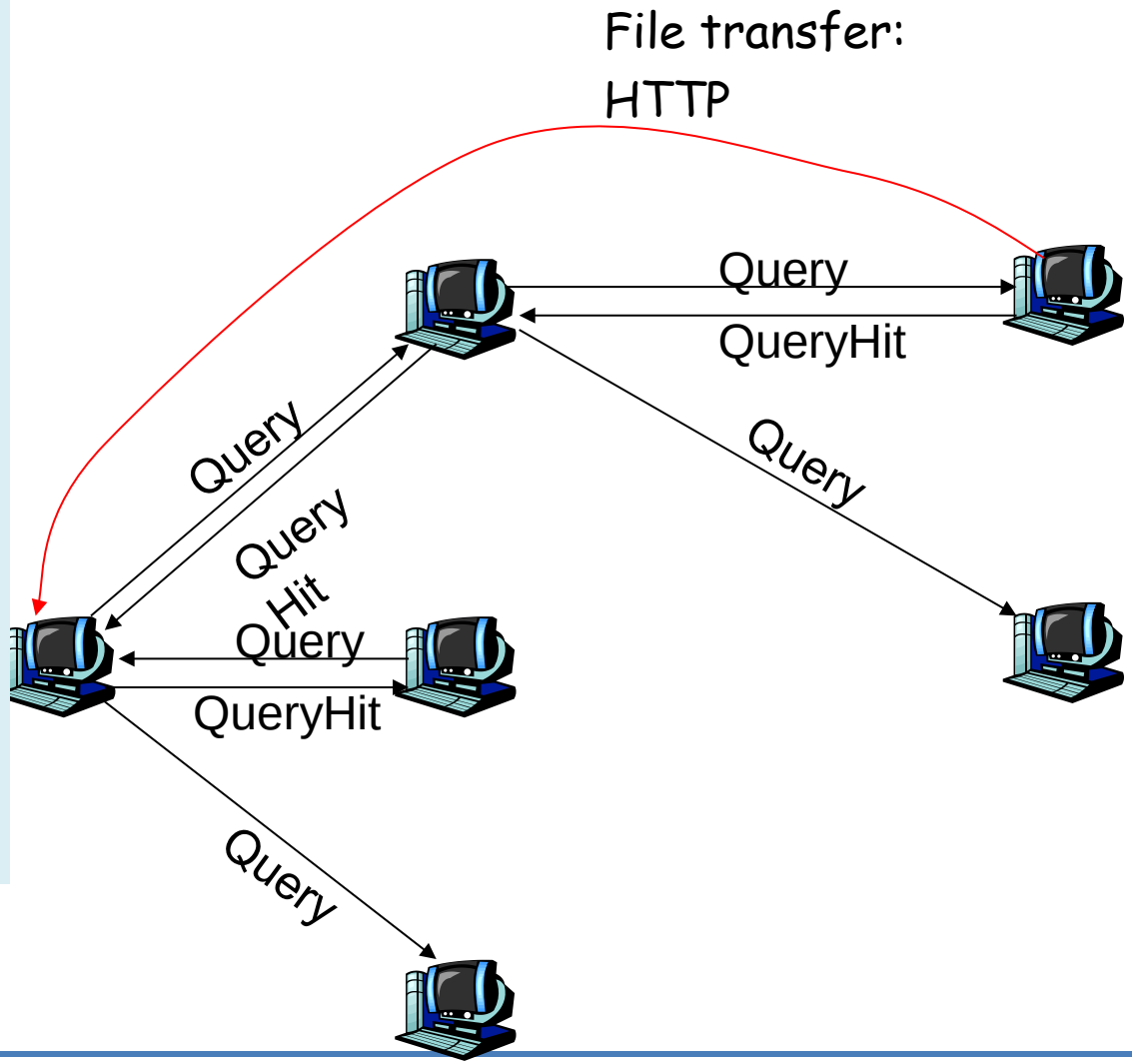
- Centralized Database: single directory
  - Napster
- Query Flooding
  - Gnutella
- Hierarchical Query Flooding
  - KaZaA
- Structured Overlays
  - DHT

Second generation in p2p ….

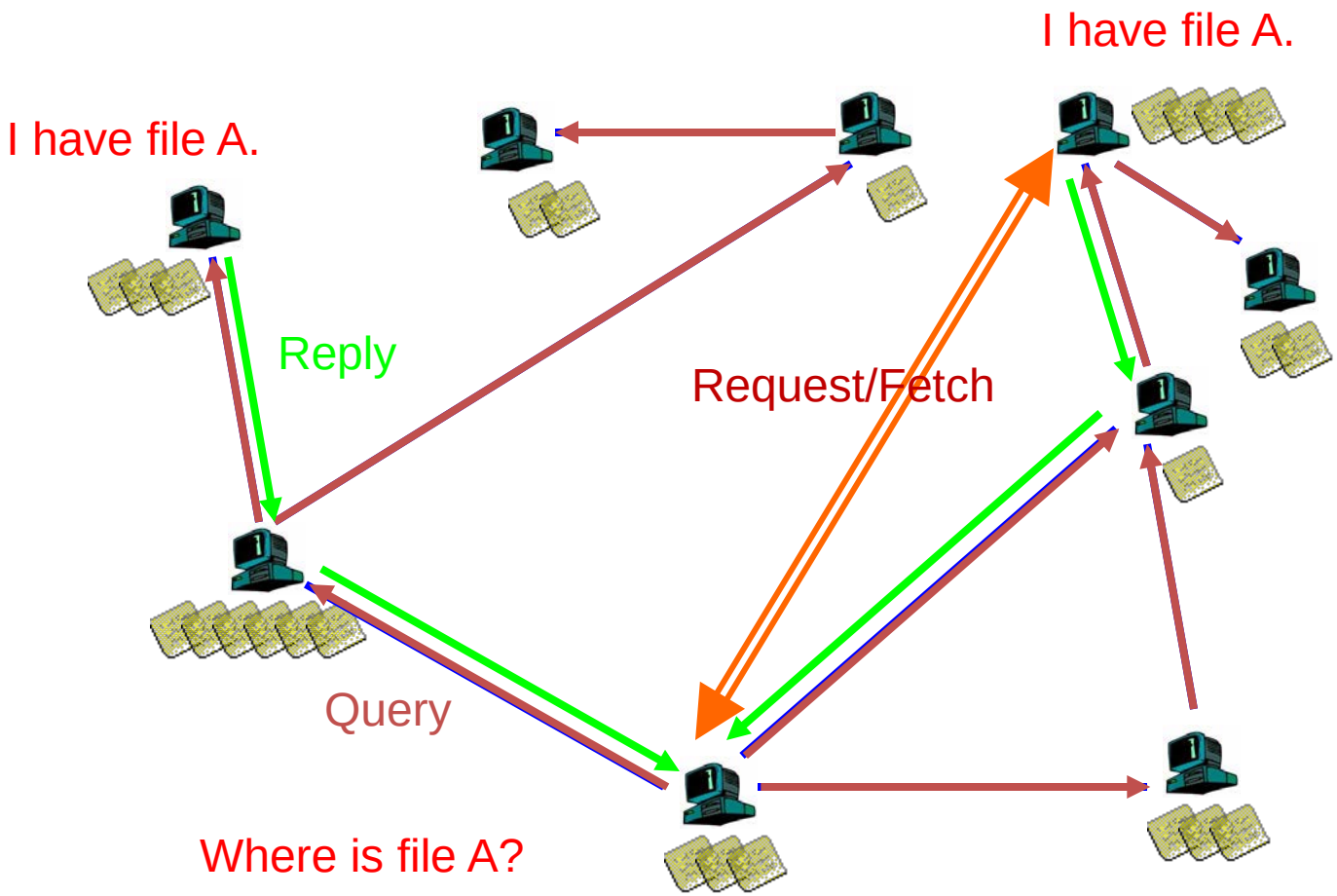
Gnutella: protocol

Query Flooding:

- **Join**: on startup, client contacts a few other nodes (learn from bootstrap-node); these become its "neighbors" (overlay!! ☺)
- **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



Q: Compare with Napster (publishing, searching, anything else)

Discussion +, -?

Napster

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

Gnutella:

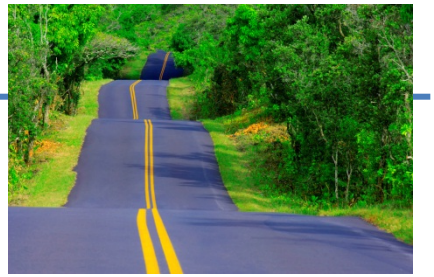
- Pros:
  - Simple
  - Fully de-centralized
  - Search cost distributed
- Cons:
  - Search scope is O(*N*)
  - Search time is O(???)

Synch questions:

how are the ”neighbors” connected?
what is the overlay here useful for?

- Edge is not a physical link E.g. edge between peer X and Y if there’s a TCP connection
- Used for supporting the search operation (**aka routing in p2p networks**)

Roadmap



First generation in p2p: file sharing/lookup

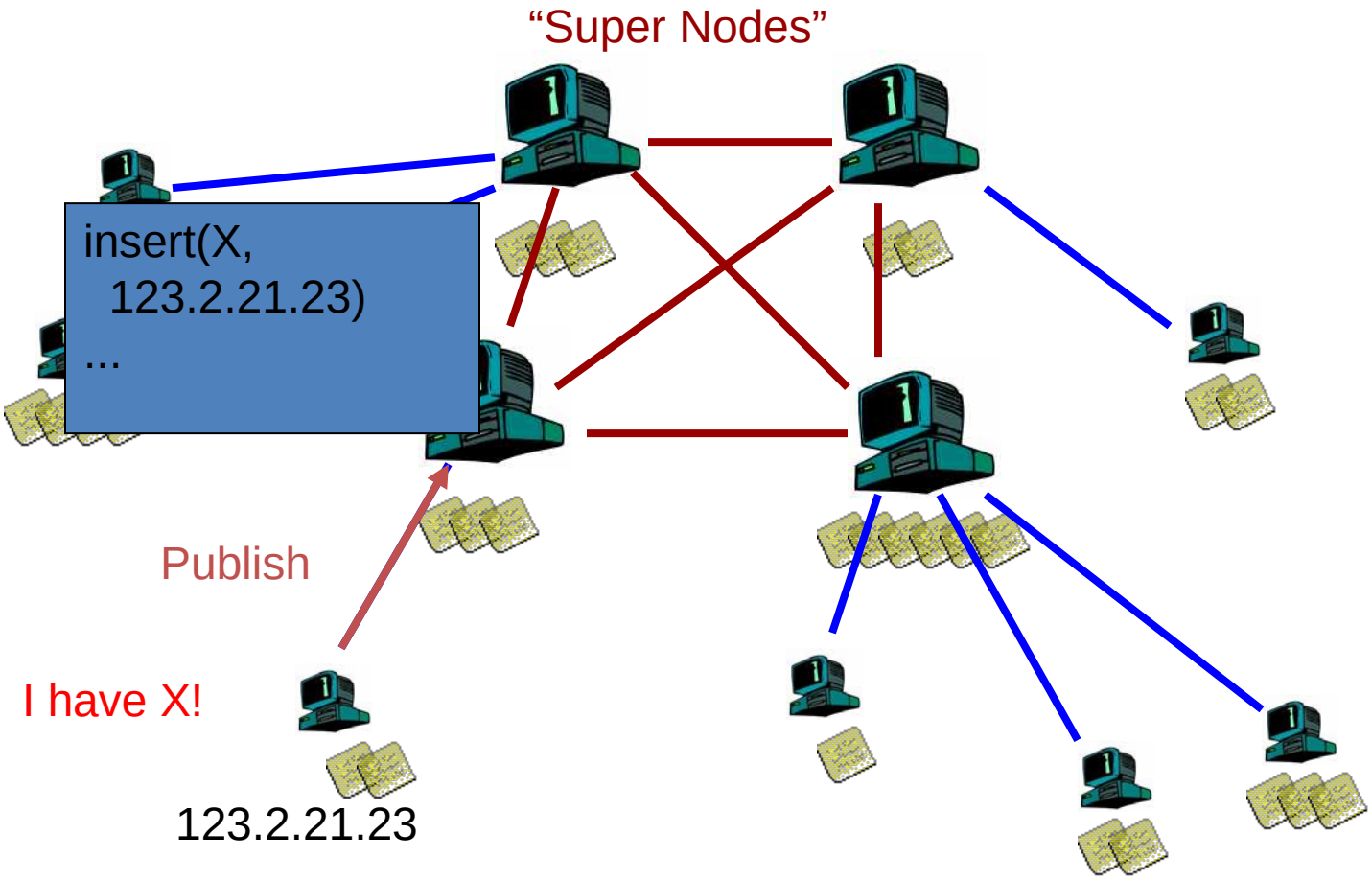
- Centralized Database: single directory
  - Napster
- Query Flooding
  - Gnutella
- Hierarchical Query Flooding : some directory structure
  - KaZaA
- Structured Overlays
  - DHT

Second generation in p2p ….

KaZaA: join, publish

"Smart" Query Flooding:

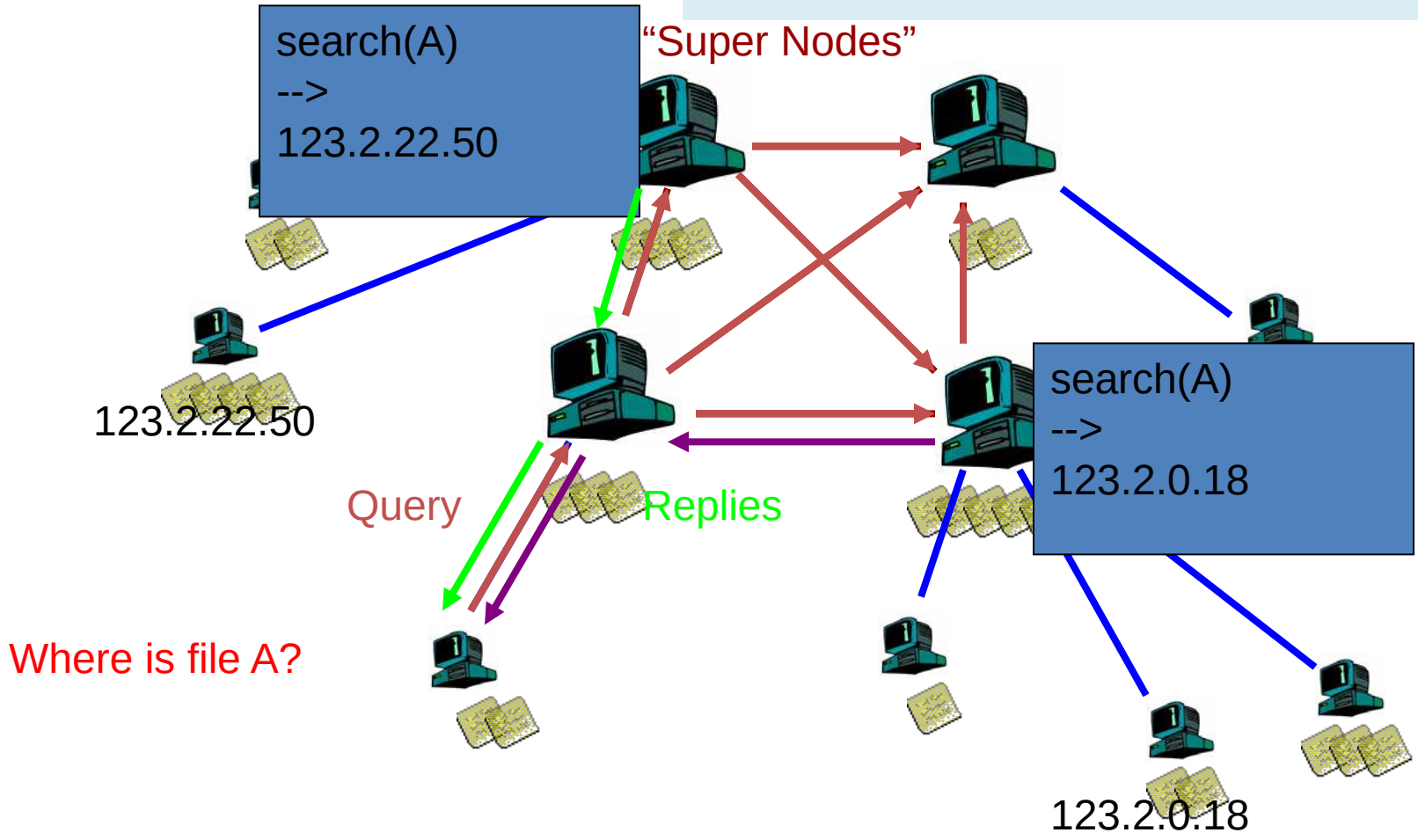
- **Join**: on startup, client contacts a "supernode" ... may at some point become one itself
- **Publish**: send list of files to supernode



KaZaA: Search

"Smart" Query Flooding:

- **Search**: send query to supernode, supernodes flood query amongst themselves.
- **Fetch**: get the file directly from peer(s); can fetch simultaneously from multiple peers



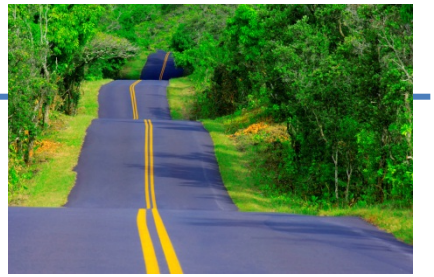
Q: Compare with Napster, Gnutella (publishing, searching, anything else)

KaZaA: Discussion

- Pros:
  - Tries to balance between search overhead and space needs
  - Tries to take into account node heterogeneity:
    - Bandwidth
    - Host Computational Resources
- Cons:
  - No real guarantees on search scope or search time
  - Super-peers may "serve" a lot!

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

Roadmap



First generation in p2p: file sharing/lookup

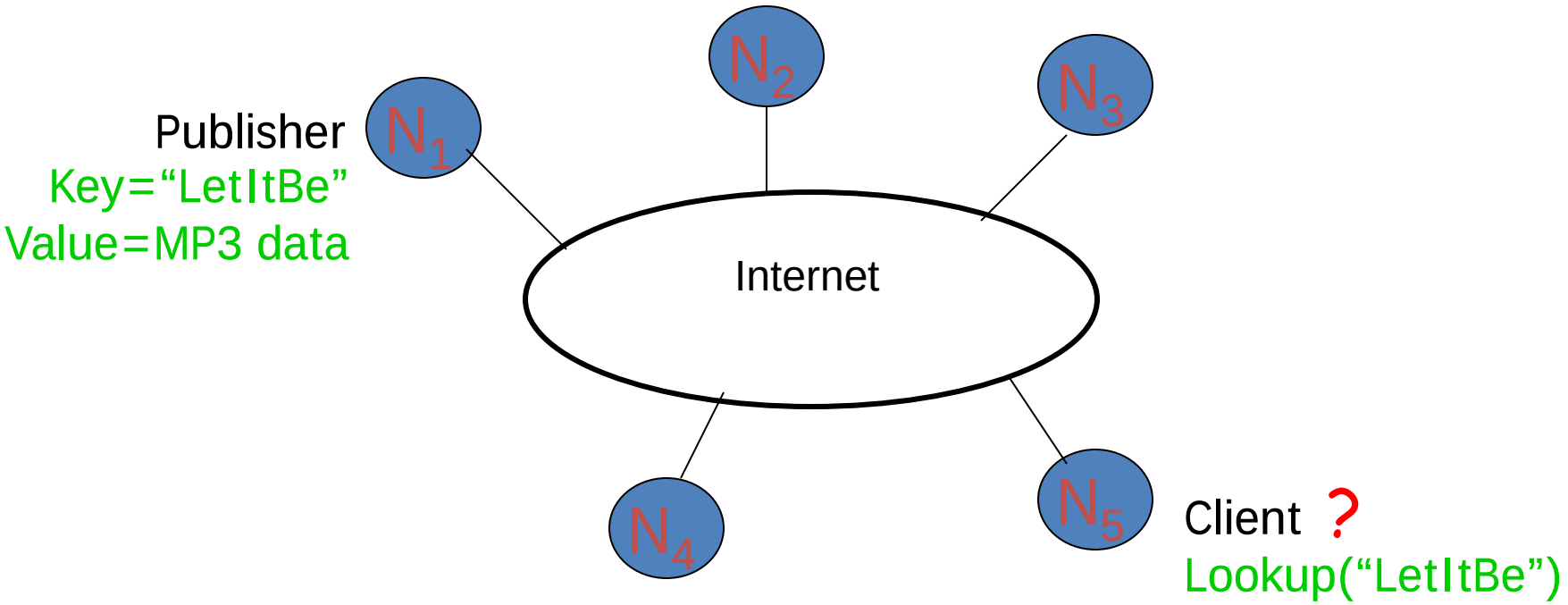
- Centralized Database: single directory
  - Napster
- Query Flooding
  - Gnutella
- Hierarchical Query Flooding
  - KaZaA
- Structured Overlays
  - Combine database+distributed system know-how

Second generation in p2p ….

Problem from this perspective

How to find data in a distributed file sharing system?

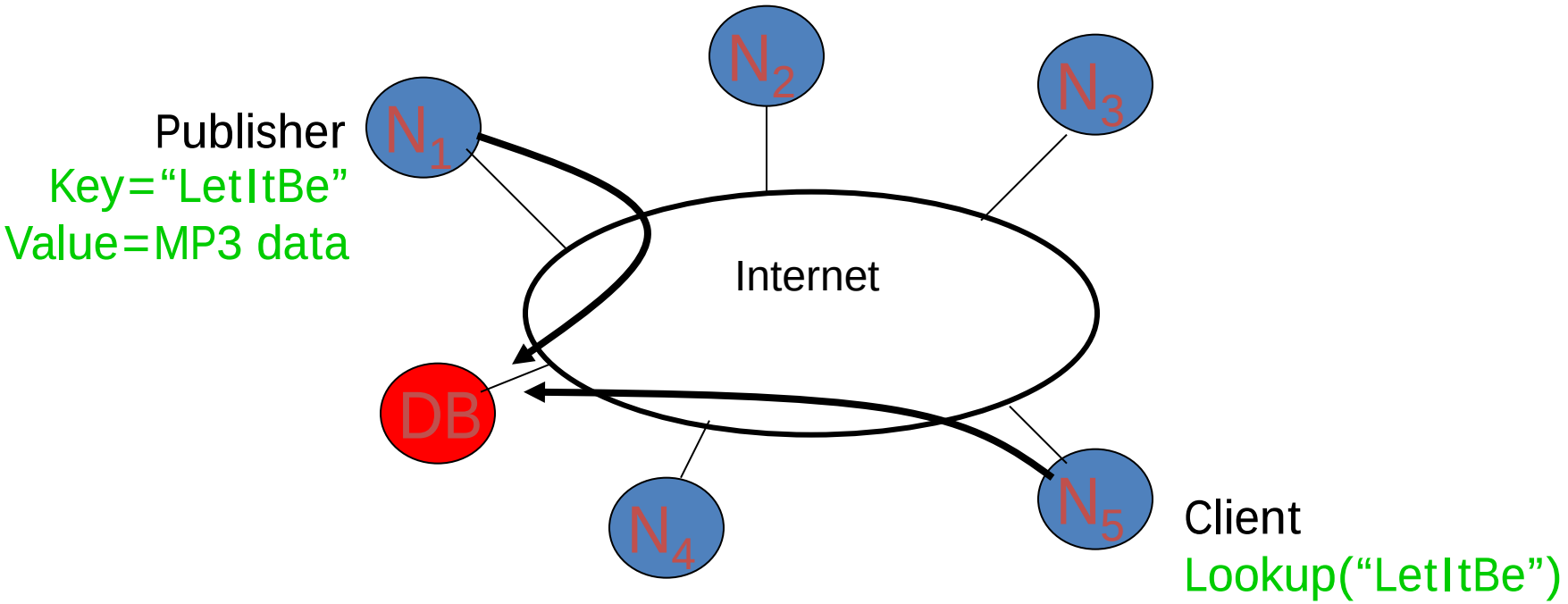
(Routing to the data)



How to do Lookup?

Centralized Solution

Central server (Napster)



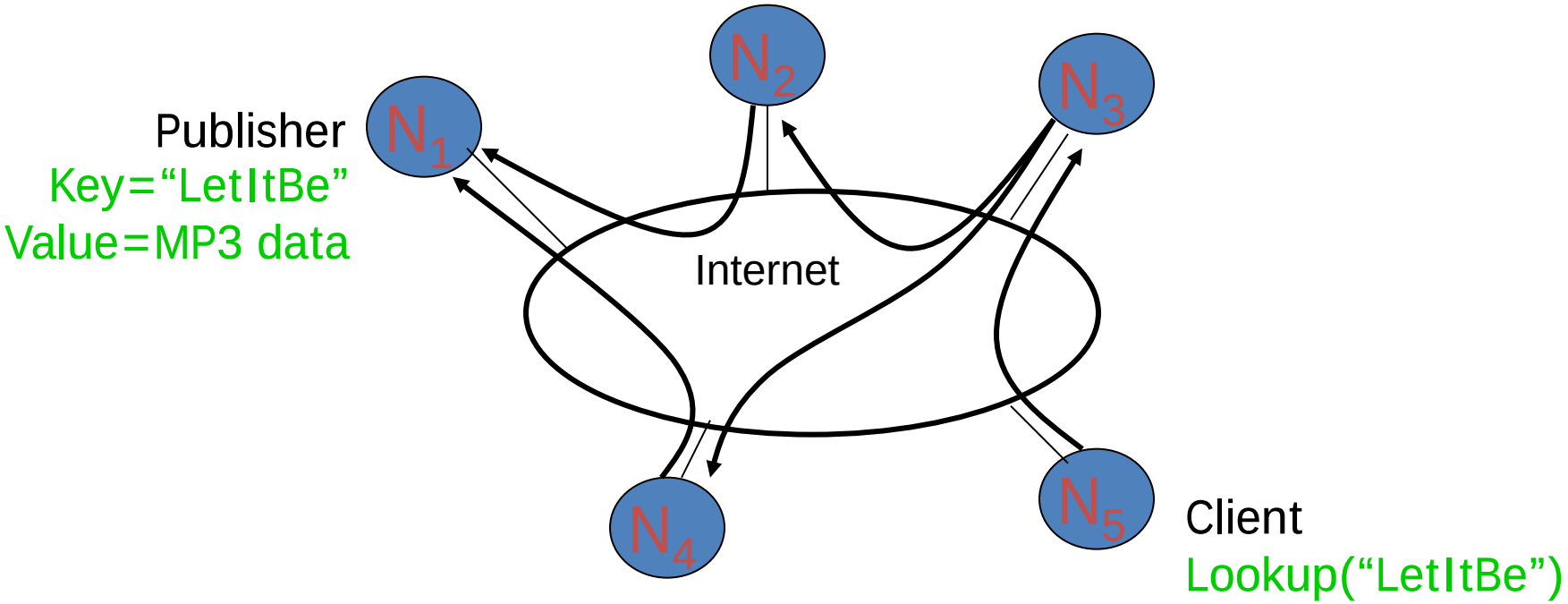
O(M) state at server, O(1) at client

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

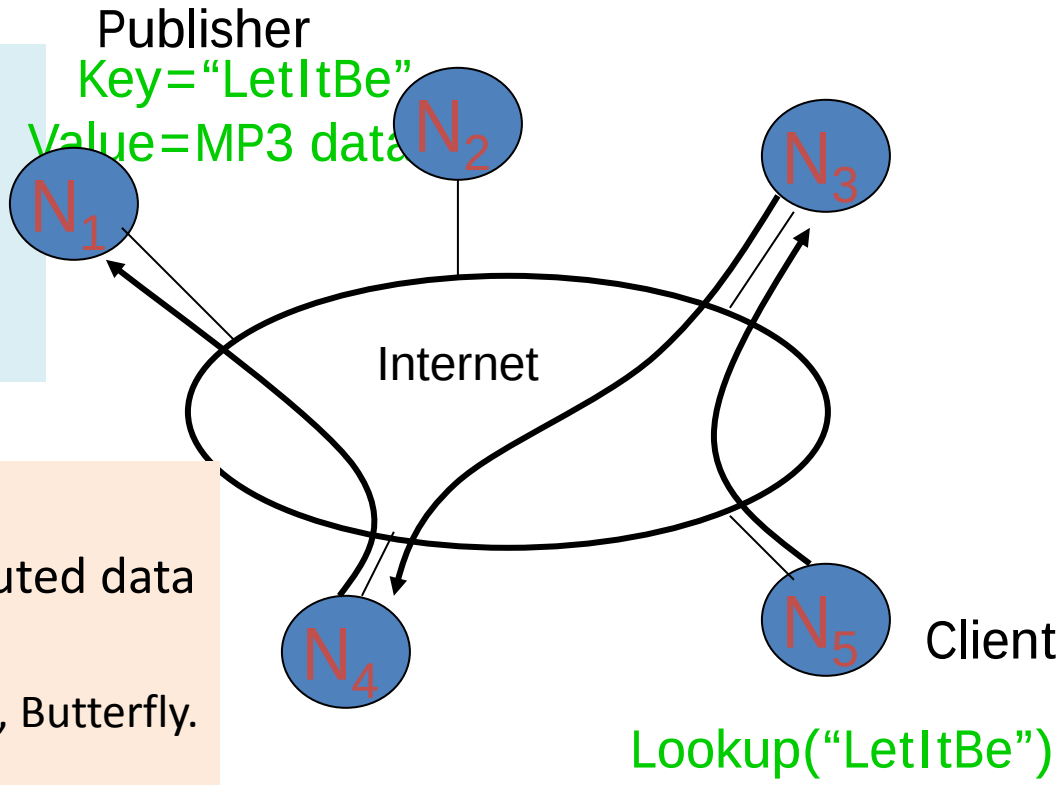
Distributed Solution

(´with some more structure? In-between the two?)

balance the update/lookup complexity..

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

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



Implementation:

- nodes form an overlay (a distributed data structure)

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

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

- >

- Hash function maps entries to nodes
- Nodes-overlay has a structure
- *Using the node structure, can:*
  - Lookup: find the node responsible for item; that one knows where the item is

I do not know DFCD3454 but can ask a neighbour in the DHT

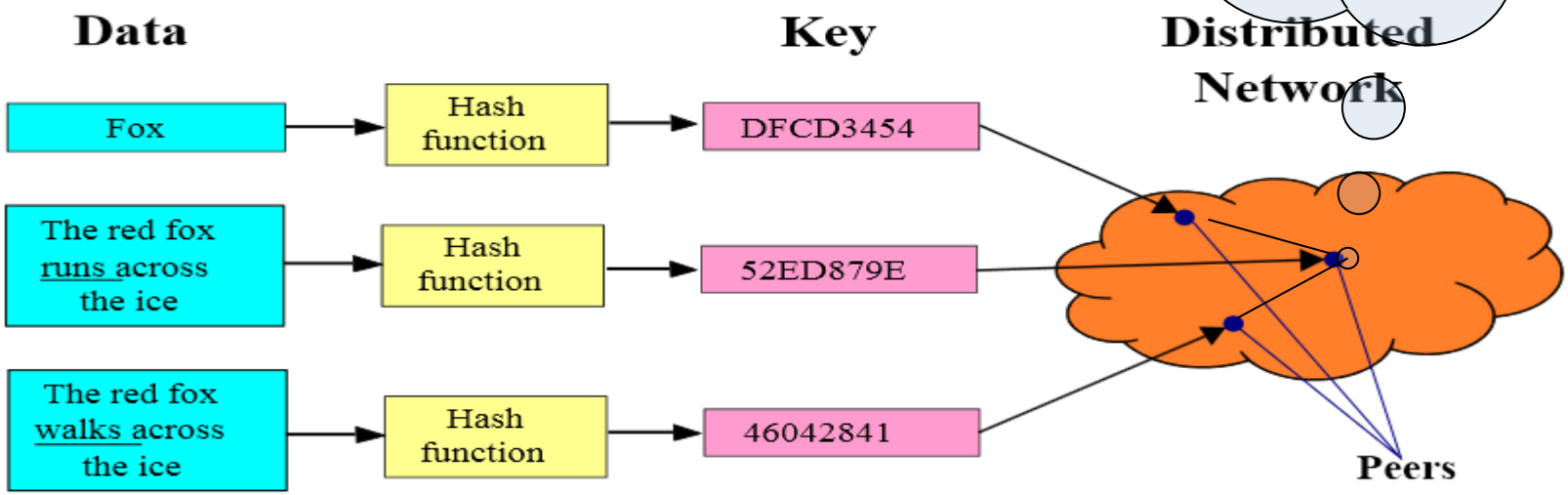
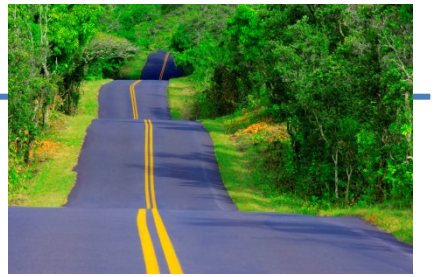


figure source: wikipedia

Challenges:
- Keep the hop count (asking chain) small
- Keep the routing tables (#neighbours) "right size"
- Stay robust despite rapid changes in membership

Roadmap



First generation in p2p: file sharing/lookup

- Centralized Database: single directory
  - Napster
- Query Flooding
  - Gnutella
- Hierarchical Query Flooding
  - KaZaA
- Structured Overlays
  - DHT

Second generation in p2p ….

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

BitTorrent: Next generation fetching

- Key Motivation:
  - Popularity exhibits temporal locality (Flash Crowds)
  - Can bring file "provider" to "its knees"

- **Idea: Focused on Efficient *Fetching*, not *Searching*:**
  - Files are "chopped" in chunks, fetching is done from many sources
  - Overlay: nodes "hold hands" with those who share (send chunks) at similar rates

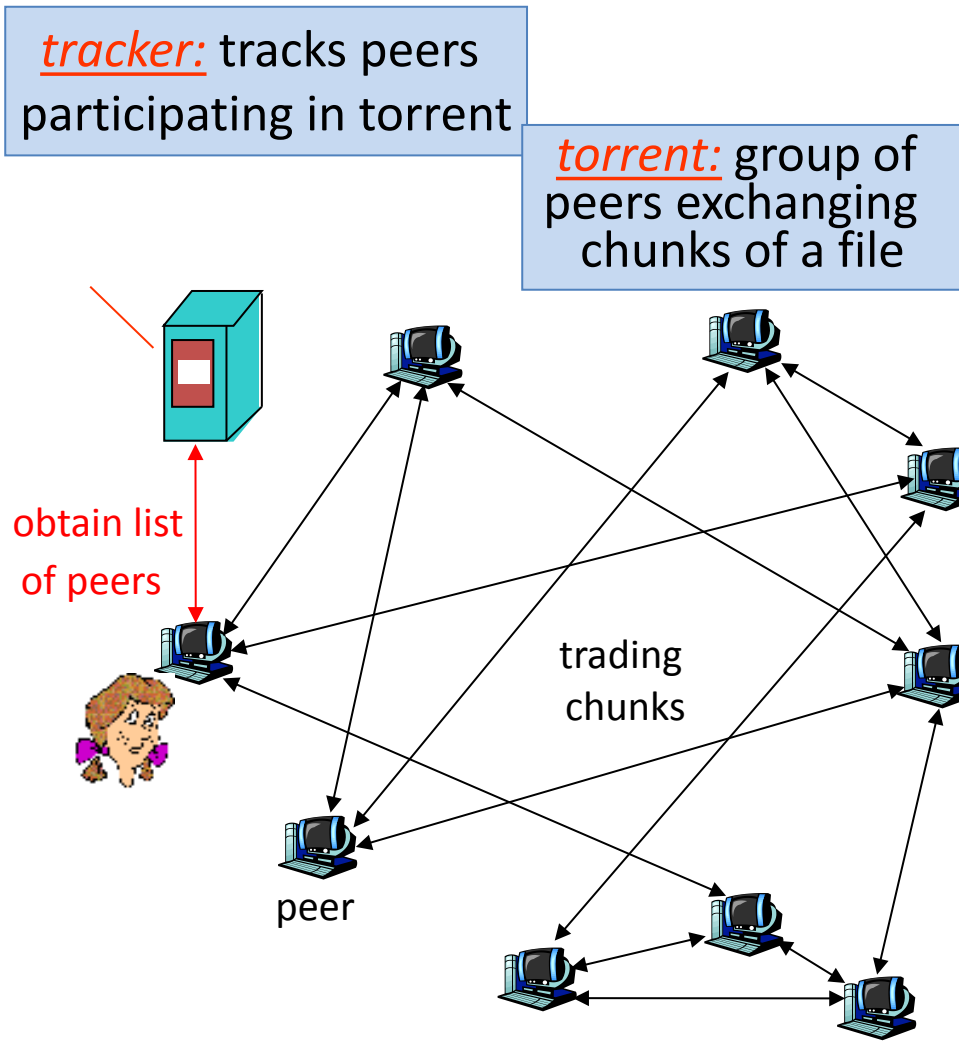
- Used by publishers to distribute software, other large files

- http://vimeo.com/15228767

BitTorrent: Overview

Swarming:

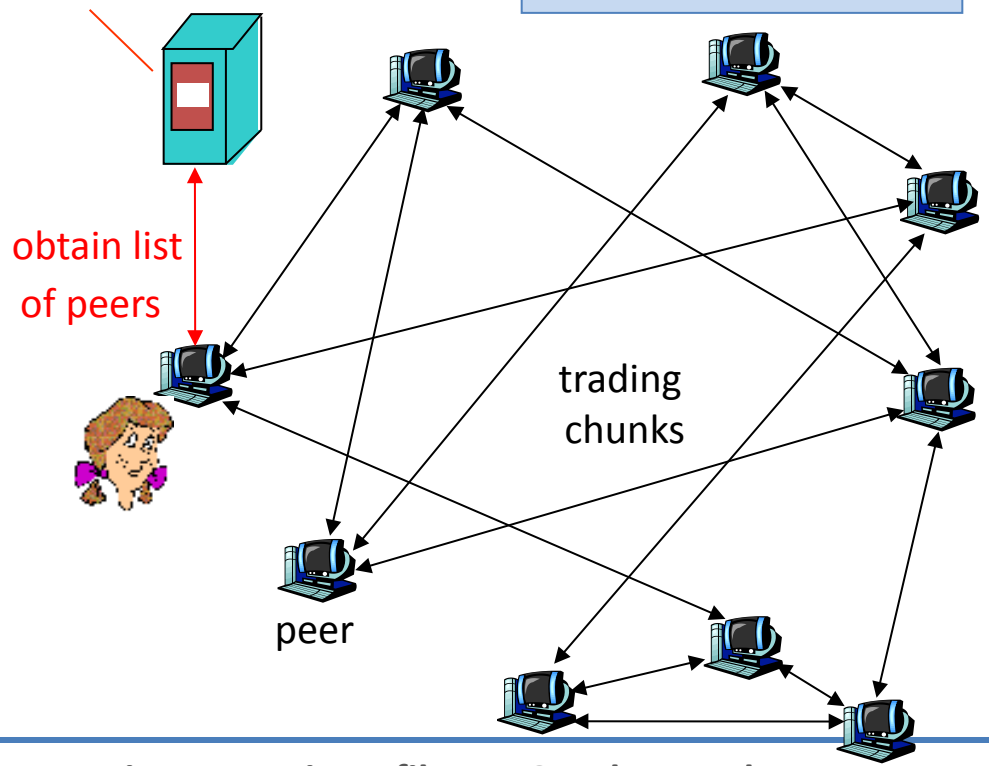
- **Join**: contact some server, aka "tracker" get a list of peers.
- **Publish**: can run a tracker server.
- **Search**: Out-of-band. E.g., use Google, some DHT, etc to find a tracker for the file you want. Get list of peers to contact for assembling the file in chunks
- **Fetch**: Download chunks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



File distribution: BitTorrent

tracker: tracks peers participating in torrent

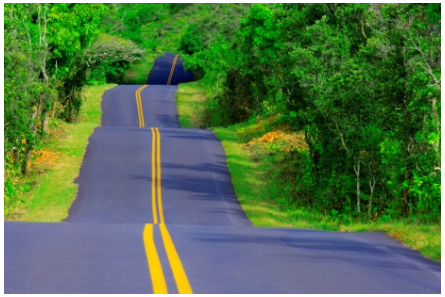
torrent: group of peers exchanging chunks of a file



- Peer joining torrent:
  - has no chunks, but will accumulate over time
  - gets list of peers from tracker, connects to subset of peers ("neighbors") who share at similar rates (tit-for-tat)
- while downloading, peer uploads chunks to other peers.
- once peer has entire file, it may (selfishly) leave or (altruistically) remain

Roadmap

- First generation in p2p: file sharing/lookup
  - gle directory
  - ng



Next: guest lecture Monday

”SDN: Software-Defined Networks”
Zhang Fu, Ericsson research

Reading instructions

- **KuroseRoss book: chapter 2.6**

f

•

cture/week%208%20P2P%20systems-general.pdf
cture/week%209%20Structured%20Overlay%20Networks.p

• m Cohen. Workshop on Economics of Peer-to-Peer

• Michael Piatek, Tomas Isdal, Thomas Anderson,
ni, NSDI 2007

• vork Coding for Large Scale Content Distribution, in
ning: combining p2p + streaming)

I

• with preferences: Approximation algorithms for
10

• framework for studying unstructured overlays,