

P2P file sharing

Notes based on notes by
K.W. Ross, J. Kurose, D.
Rubenstein, and others

P2P: centralized directory

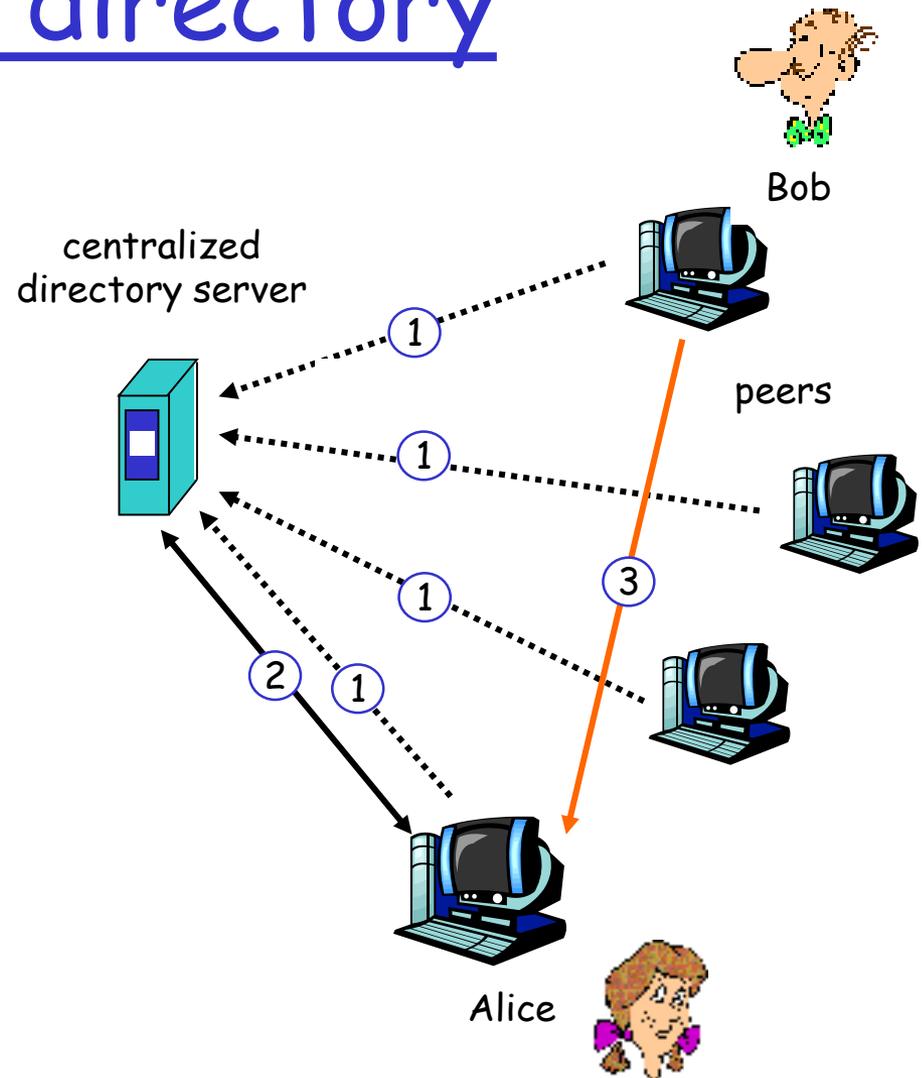
original "Napster" design

1) when peer connects, it informs central server:

- IP address
- content

2) Alice queries for "Hey Jude"

3) Alice requests file from Bob

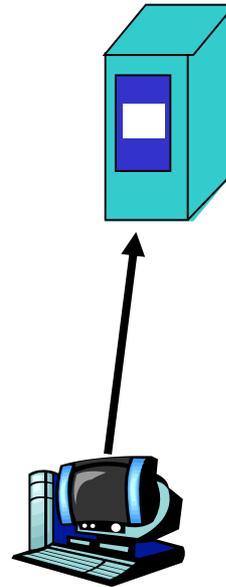


Napster

1. File list and IP address is uploaded



napster.com
centralized directory



Napster

2. User requests search at server.



napster.com
centralized directory

Query
and
results

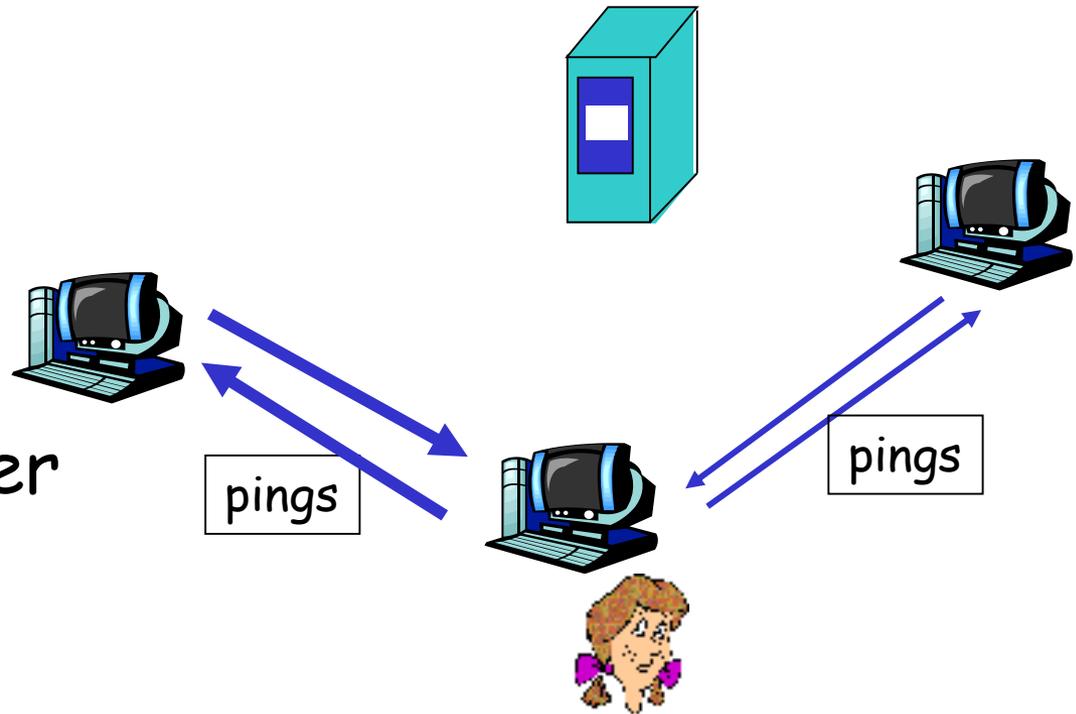


Napster

3. User pings hosts that apparently have data.

Looks for best transfer rate.

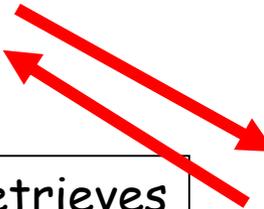
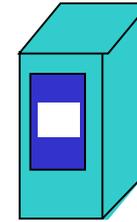
napster.com
centralized directory



Napster

4. User chooses server

napster.com
centralized directory



Retrieves
file



Napster's centralized server farm had difficult time keeping up with traffic

P2P: problems with centralized directory

- ❑ single point of failure
- ❑ performance bottleneck
- ❑ copyright infringement:
“target” of lawsuit is
obvious

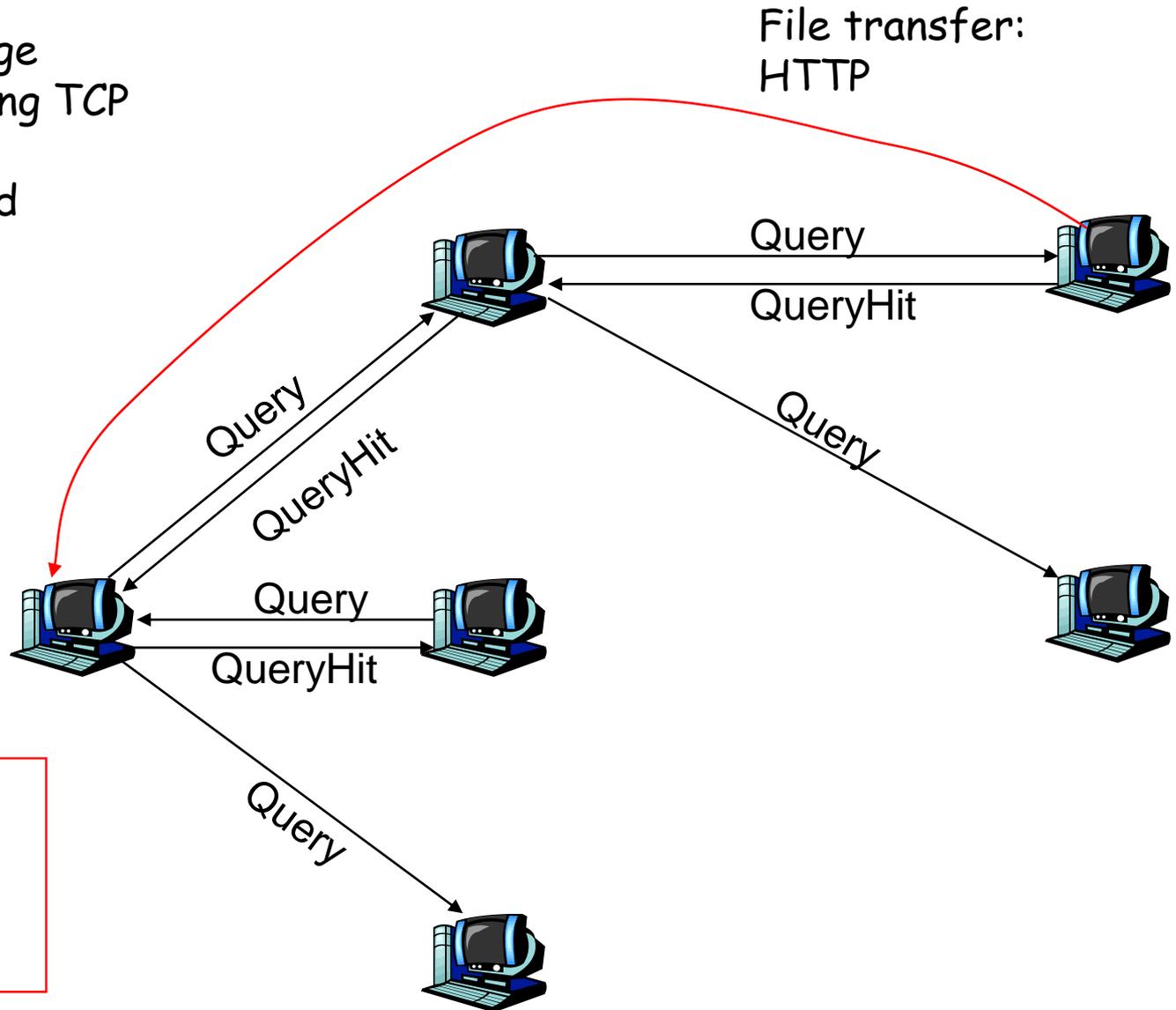
file transfer is
decentralized, but
locating content is
highly centralized

Unstructured P2P: Gnutella

- ❑ focus: decentralized method of searching for files
 - central directory server no longer the bottleneck
 - more difficult to “pull plug”
- ❑ each application instance serves to:
 - store selected files
 - route queries from and to its neighboring peers
 - respond to queries if file stored locally
 - serve files

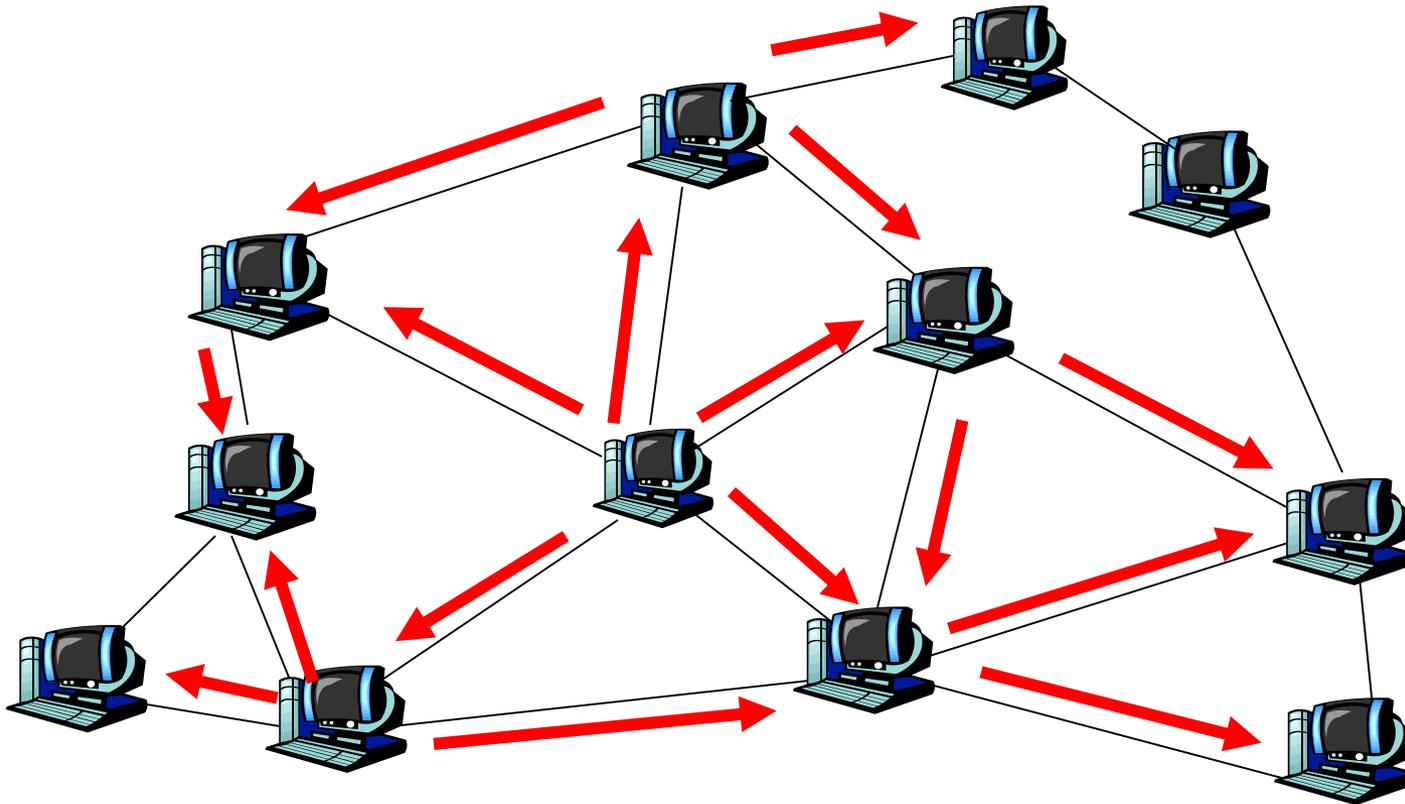
Gnutella: protocol

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

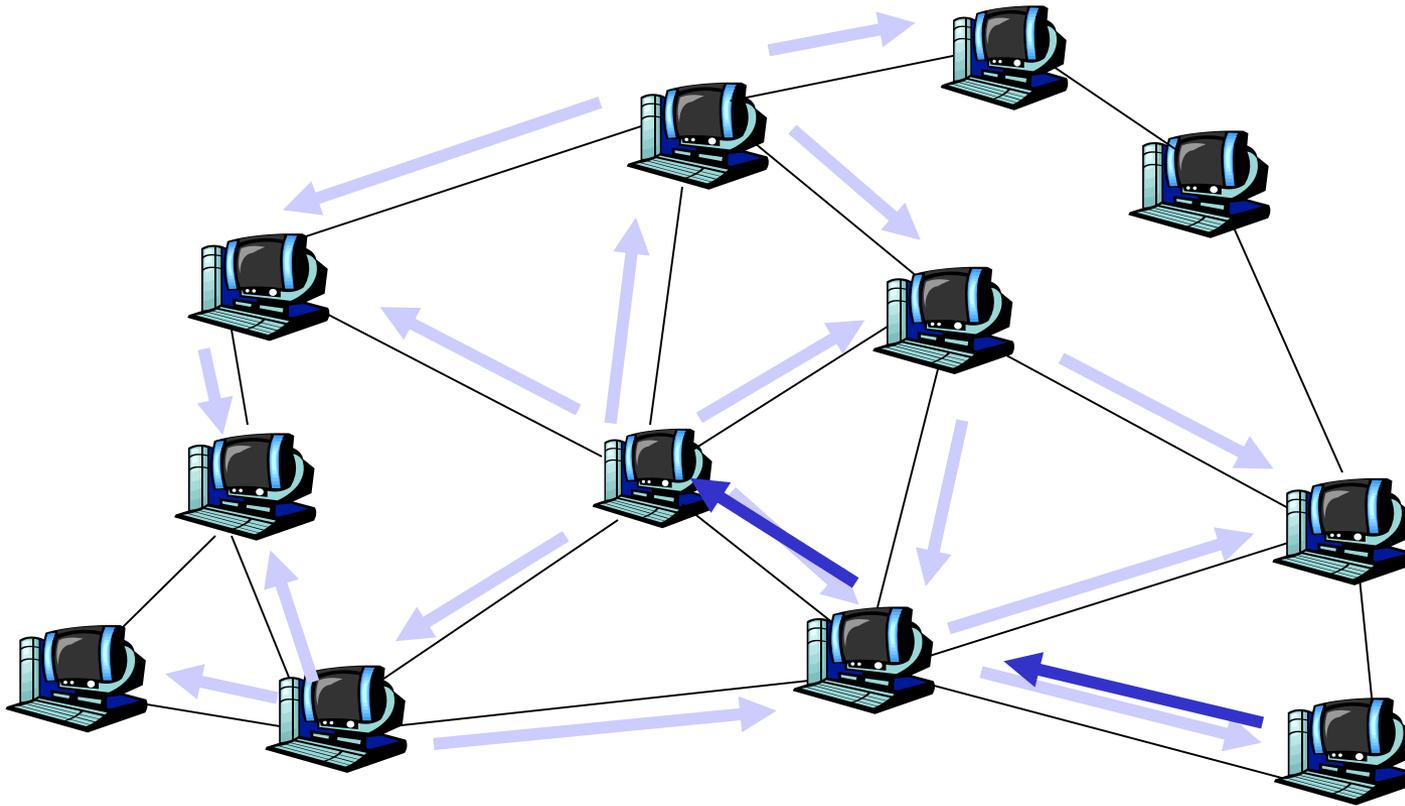


Scalability:
limited scope
flooding

Distributed Search/Flooding

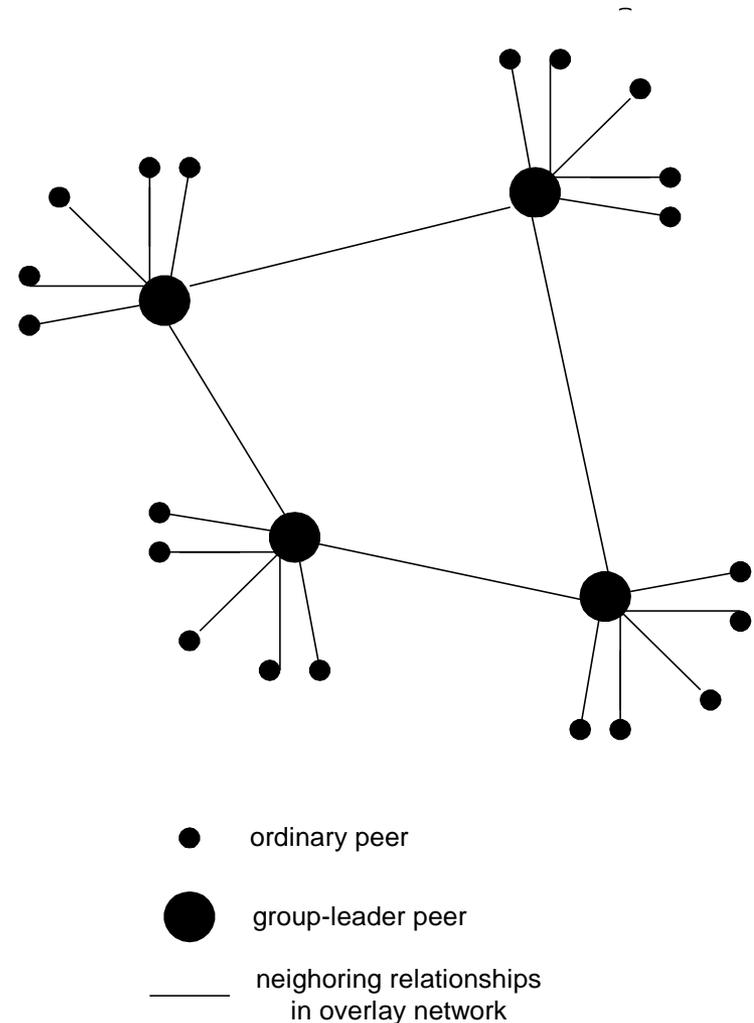


Distributed Search/Flooding



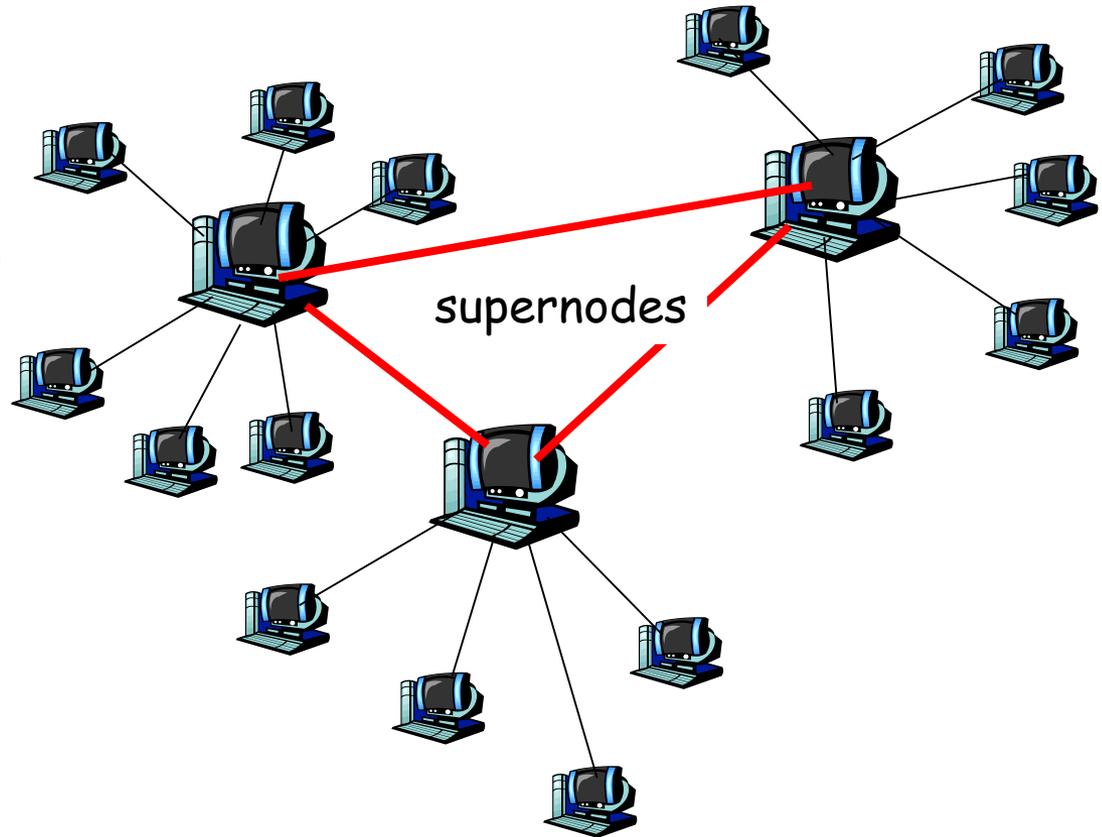
Hierarchical Overlay

- ❑ between centralized index, query flooding approaches
- ❑ each peer is either a *group leader* or assigned to a group leader.
 - TCP connection between peer and its group leader.
 - TCP connections between some pairs of group leaders.
- ❑ group leader tracks content in its children



KaZaA: Architecture

- Each peer is either a supernode or is assigned to a supernode
- Each supernode knows about many other supernodes (almost mesh overlay)



KaZaA: Architecture (2)

- ❑ Nodes that have more connection bandwidth and are more available are designated as supernodes
- ❑ Each supernode acts as a mini-Napster hub, tracking the content and IP addresses of its descendants
- ❑ Guess@peak: supernode had (on average) 200-500 descendants; roughly 10,000 supernodes
- ❑ There is also dedicated user authentication server and supernode list server

Parallel Downloading; Recovery

- ❑ If file is found in multiple nodes, user can select parallel downloading
- ❑ Most likely HTTP byte-range header used to request different portions of the file from different nodes
- ❑ Automatic recovery when server peer stops sending file

KaZaA Corporate Structure

- ❑ Software developed by FastTrack in Amsterdam
- ❑ FastTrack also deploys KaZaA service
- ❑ FastTrack licenses software to Music City (Morpheus) and Grokster
- ❑ Later, FastTrack terminates license, leaves only KaZaA with killer service
- ❑ Summer 2001, Sharman networks, founded in Vanuatu (small island in Pacific), acquires FastTrack
 - Board of directors, investors: secret
- ❑ Employees spread around, hard to locate
- ❑ Code in Estonia

Lessons learned from KaZaA

KaZaA provides powerful file search and transfer service without server infrastructure

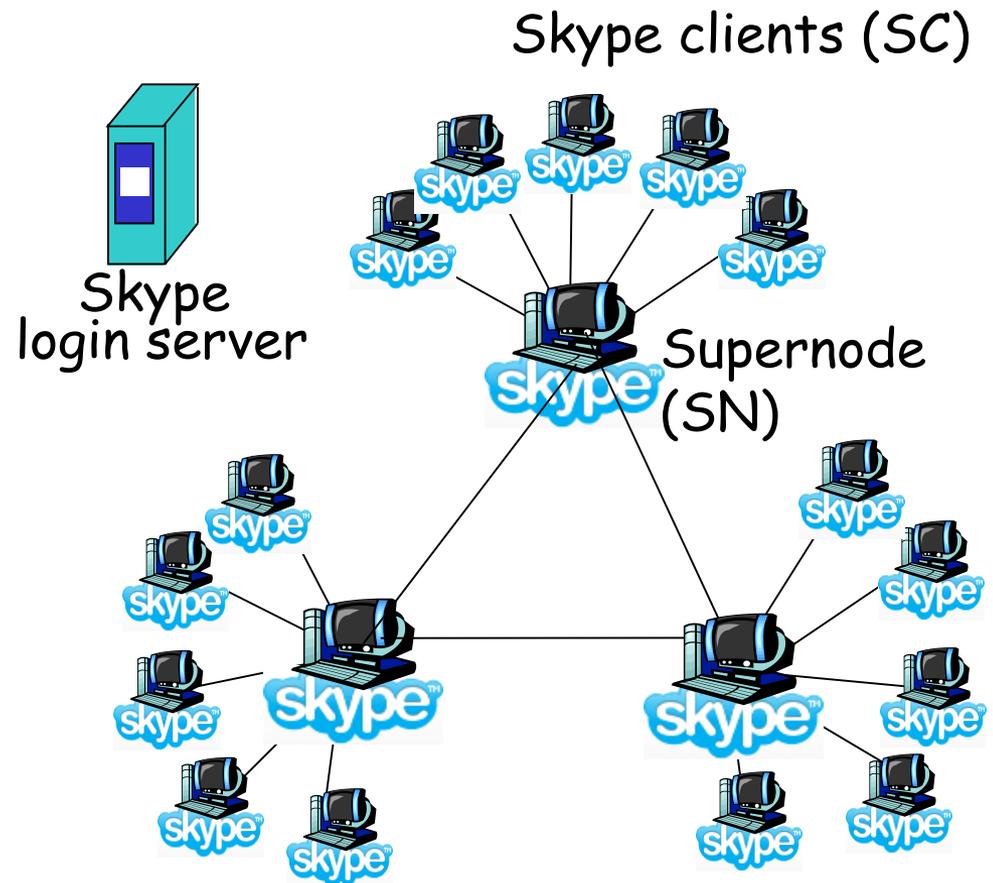
- ❑ Exploit heterogeneity
- ❑ Provide automatic recovery for interrupted downloads
- ❑ Powerful, intuitive user interface

Copyright infringement

- ❑ International cat-and-mouse game
- ❑ With distributed, serverless architecture, can the plug be pulled?
- ❑ Prosecute users?
- ❑ Launch DoS attack on supernodes?
- ❑ Pollute?

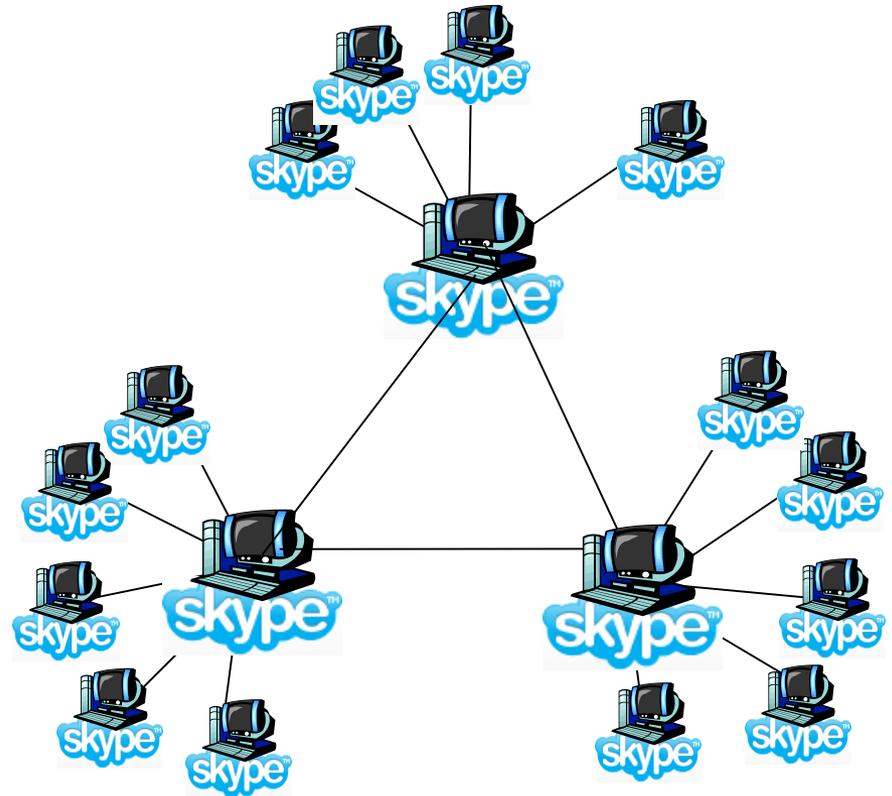
P2P Case study: Skype

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



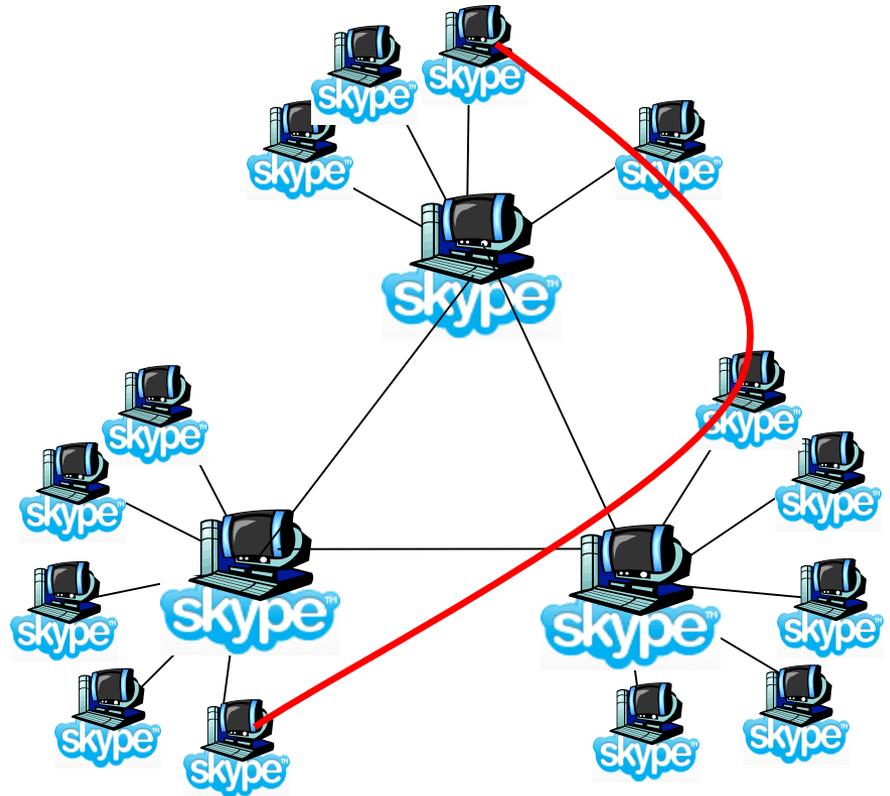
Peers as relays

- ❑ Problem when both Alice and Bob are behind "NATs".
 - NAT prevents an outside peer from initiating a call to insider peer



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



Structured p2p systems

Distributed Hash Table (DHT)

- ❑ DHT = distributed P2P database
- ❑ Database has **(key, value)** pairs;
 - key: ss number; value: human name
 - key: content type; value: IP address
- ❑ Peers **query** DB with key
 - DB returns values that match the key
- ❑ Peers can also **insert** (key, value) peers

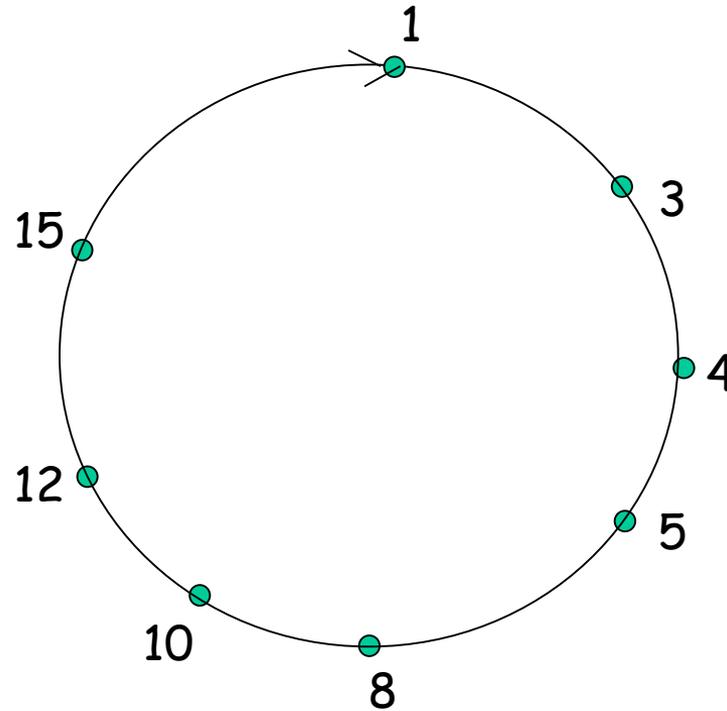
DHT Identifiers

- ❑ Assign integer identifier to each peer in range $[0, 2^n - 1]$.
 - Each identifier can be represented by n bits.
- ❑ Require each key to be an integer in **same range**.
- ❑ To get integer keys, hash original key.
 - eg, key = $h(\text{"Led Zeppelin IV"})$
 - This is why they call it a distributed "hash" table

How to assign keys to peers?

- Central issue:
 - Assigning (key, value) pairs to peers.
- Rule: assign key to the peer that has the **closest** ID.
- Convention in lecture: closest is the **closest successor** of the key.
- Ex: $n=4$; peers: 1,3,4,5,8,10,12,14;
 - key = 13, then successor peer = 14
 - key = 15, then successor peer = 1

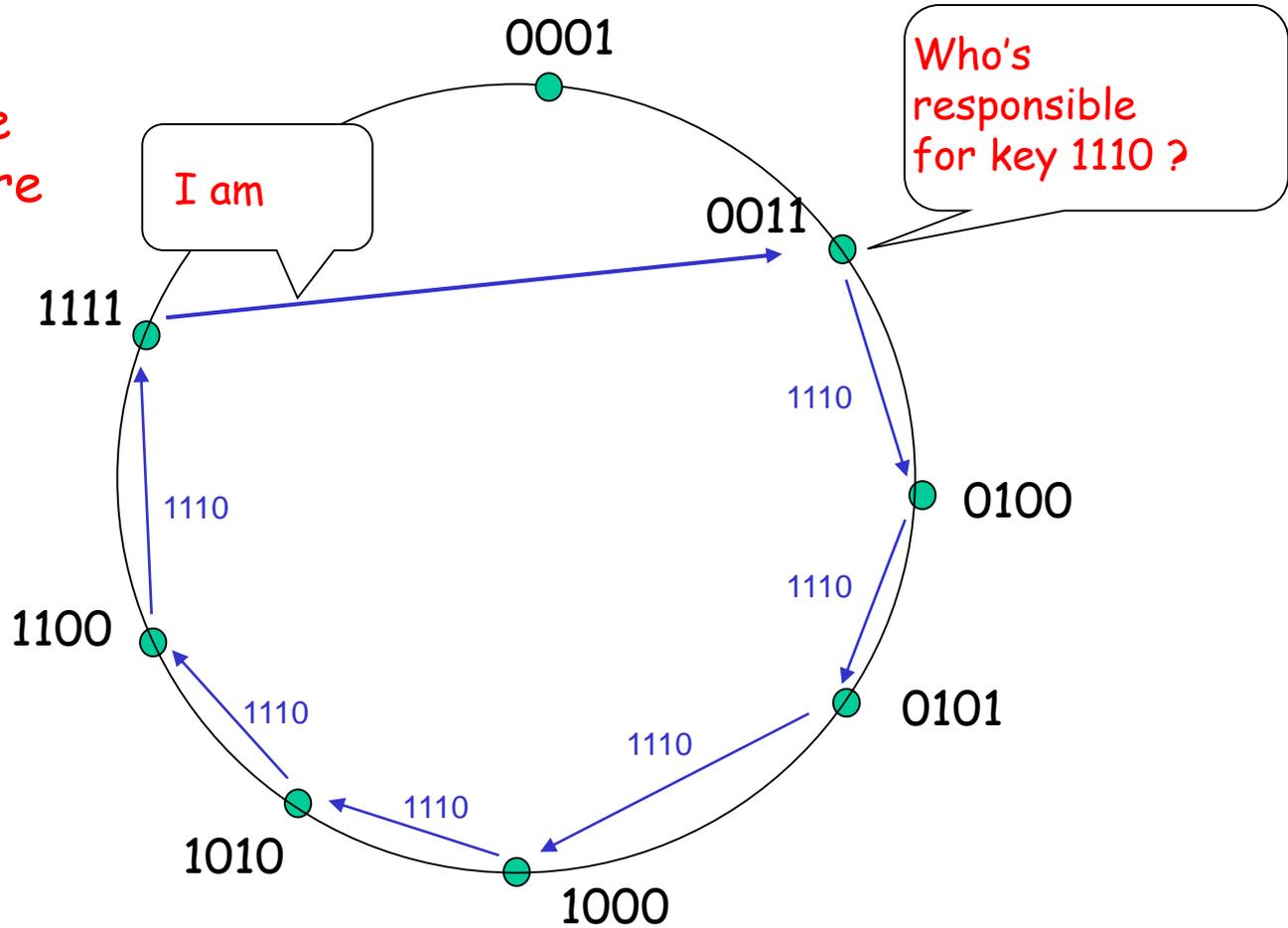
Circular DHT (1)



- ❑ Each peer *only* aware of immediate successor and predecessor.
- ❑ "Overlay network"

Circle DHT (2)

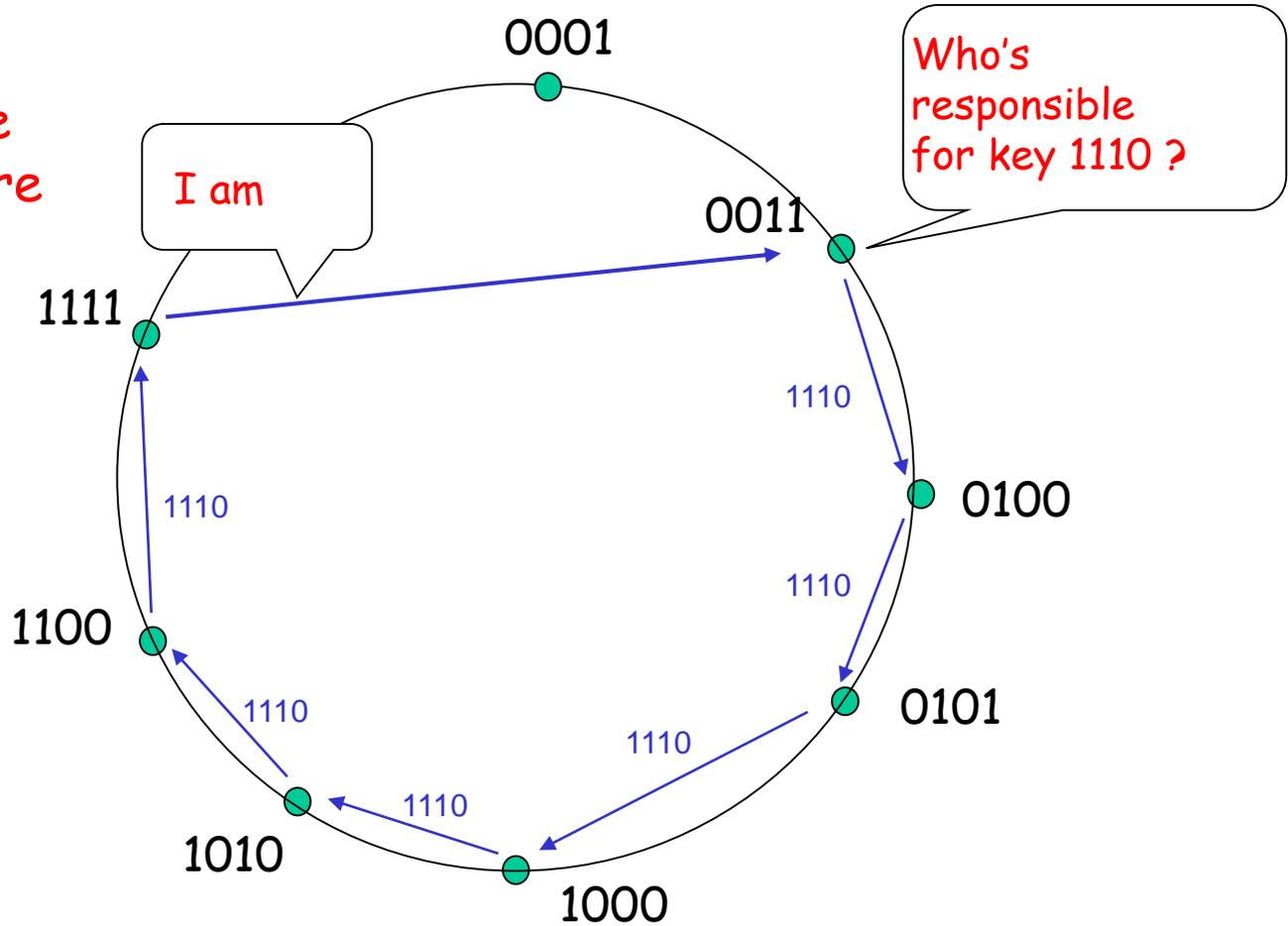
$O(N)$ messages on avg to resolve query, when there are N peers



Define closest as closest successor

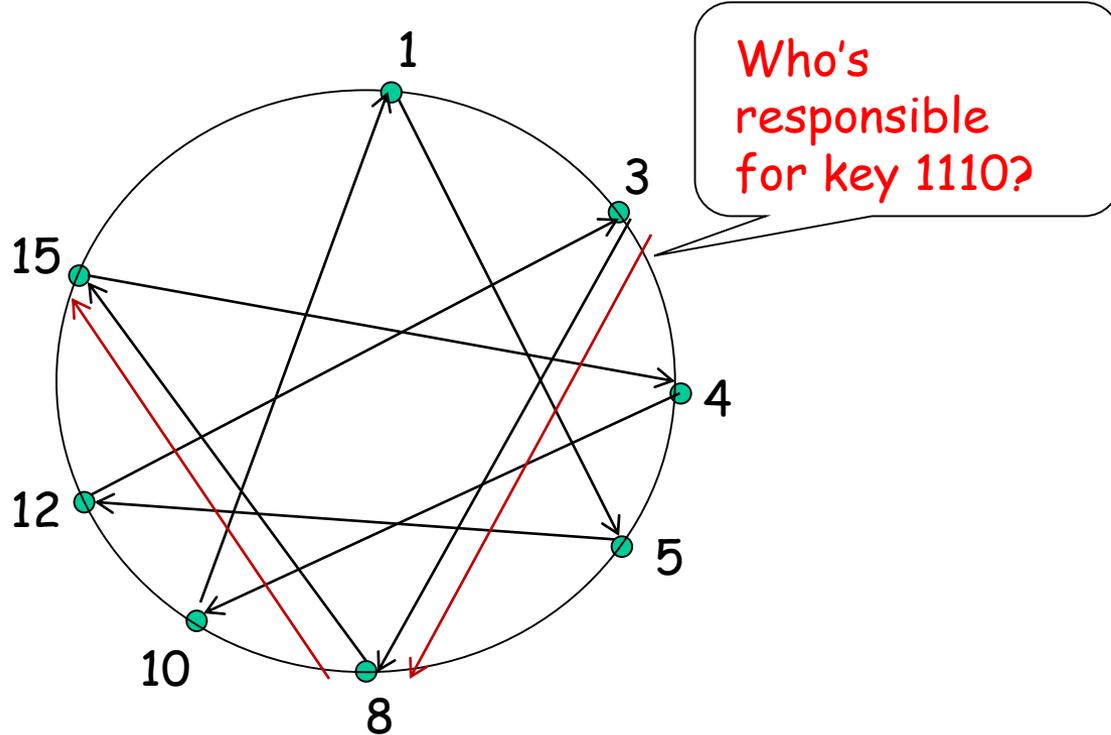
Circle DHT (2)

$O(N)$ messages
on avg to resolve
query, when there
are N peers



Define closest
as closest
successor

Circular DHT with Shortcuts



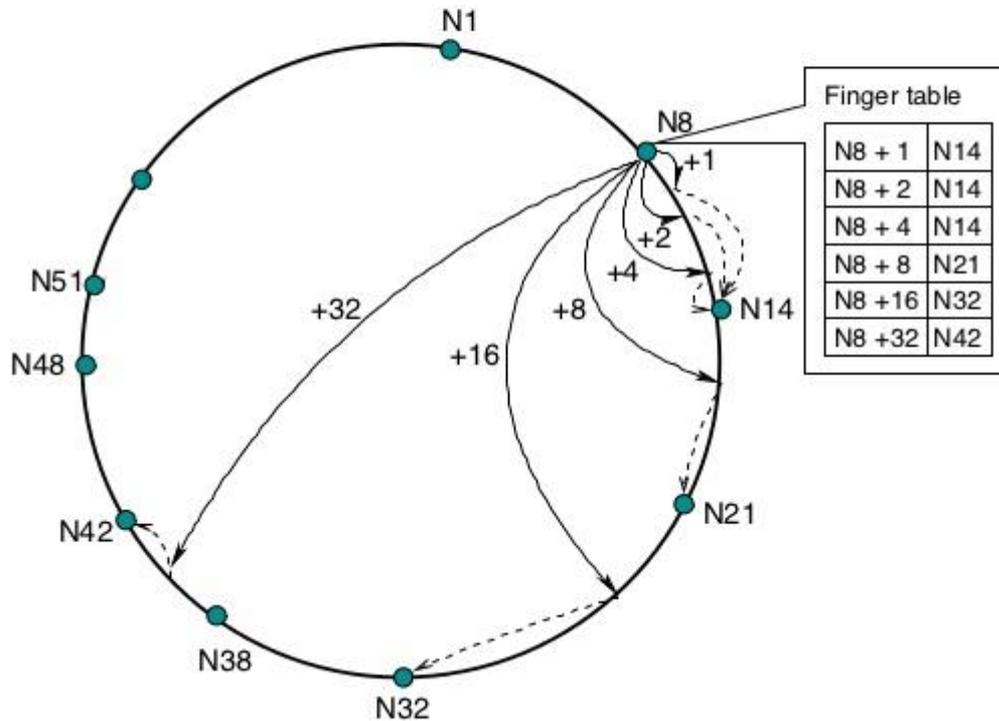
- ❑ Each peer keeps track of IP addresses of predecessor, successor, short cuts.
- ❑ Reduced from 6 to 2 messages.
- ❑ Possible to design shortcuts so $O(\log N)$ neighbors, $O(\log N)$ messages in query

Example: Chord Routing [see paper for details]

- ❑ A node s 's i^{th} neighbor has the ID that is equal to $s+2^i$ or is the next largest ID (mod ID space), $i \geq 0$
- ❑ To reach the node handling ID t , send the message to neighbor $\# \log_2(t-s)$
- ❑ Requirement: each node s must know about the next node that exists clockwise on the Chord (0^{th} neighbor)
- ❑ Set of known neighbors called a **finger table**

Chord Routing (cont'd)

Finger table

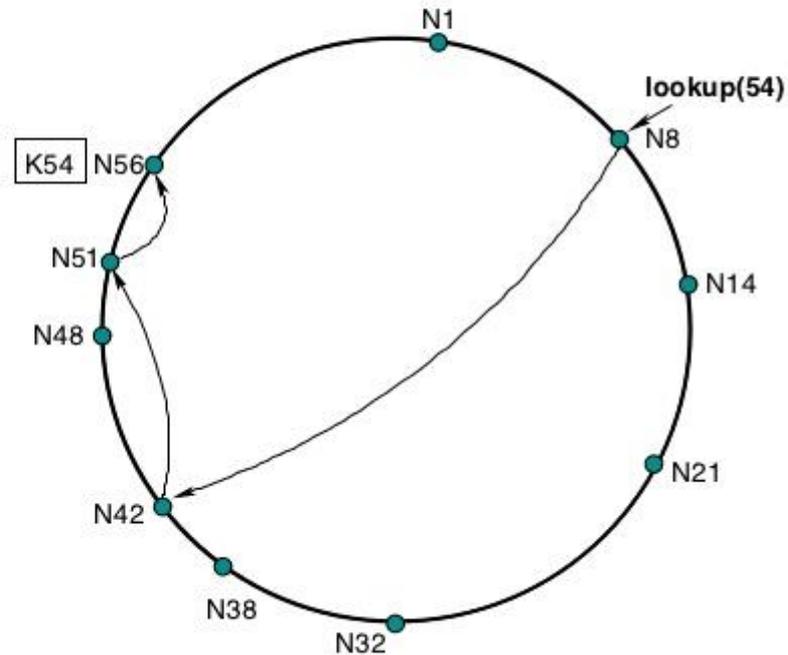


Chord Routing (cont'd)

Chord protocol

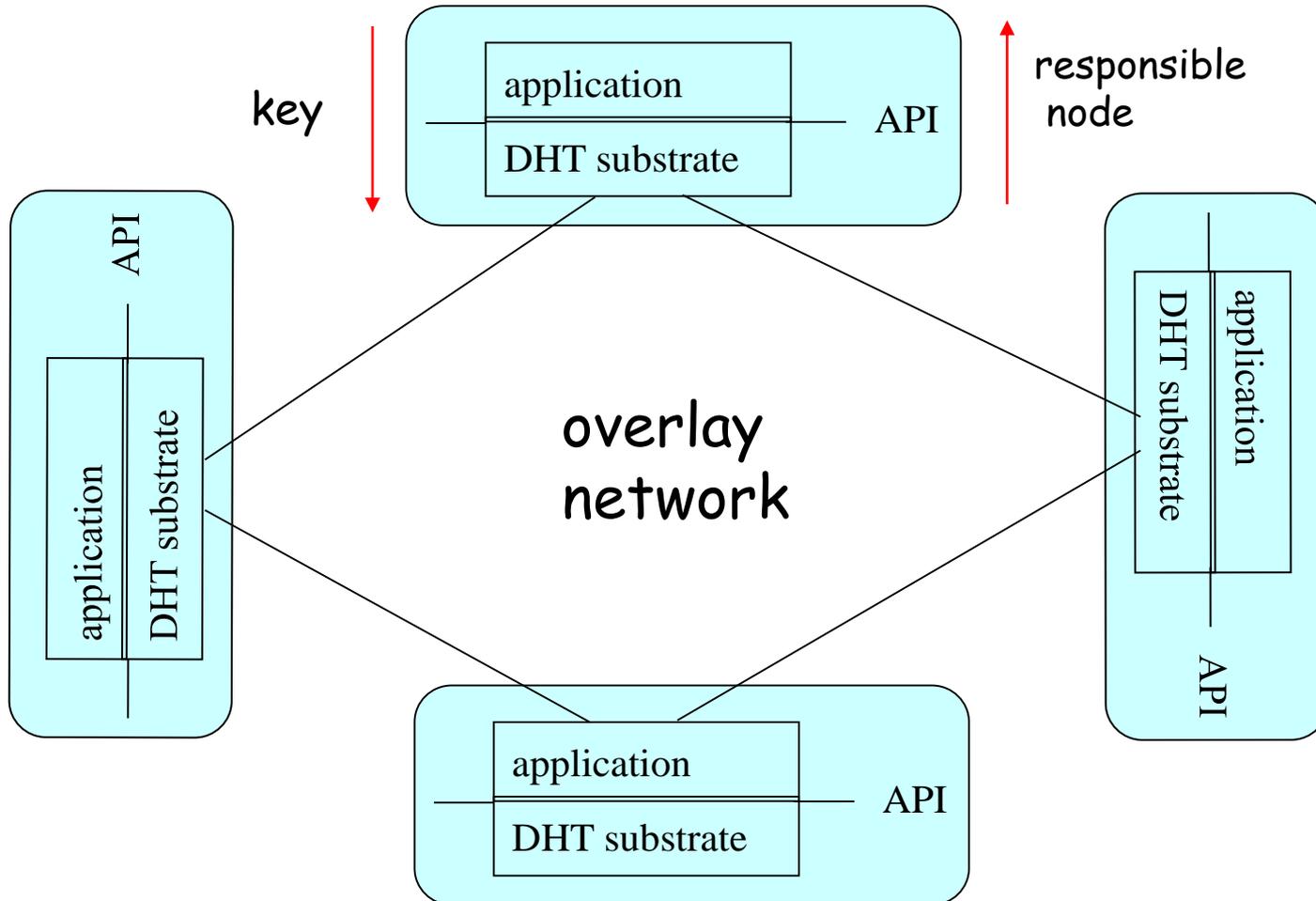
```
// ask node n to find the successor
// of id
n.find_successor(id)
  if (id ∈ (n, successor))
    return successor;
  else
    n' = closest_preceding_node(id);
    return n'.find_successor(id);

// search the local table for the
// highest predecessor of id
n.closest_preceding_node(id)
  for i = m downto 1
    if (finger[i] ∈ (n, id))
      return finger[i];
  return n;
```

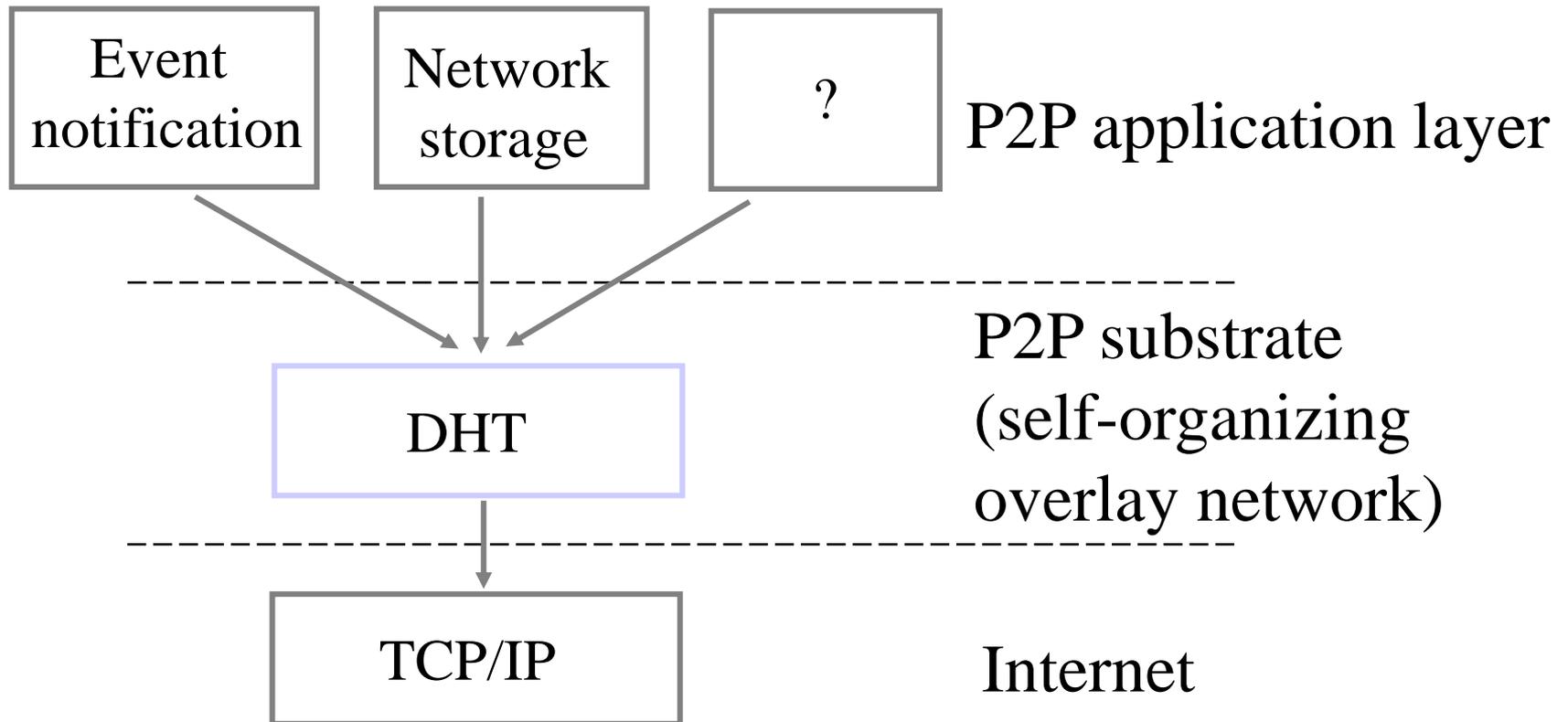


DHT API

each data item (e.g., file or metadata pointing to file copies) has a key

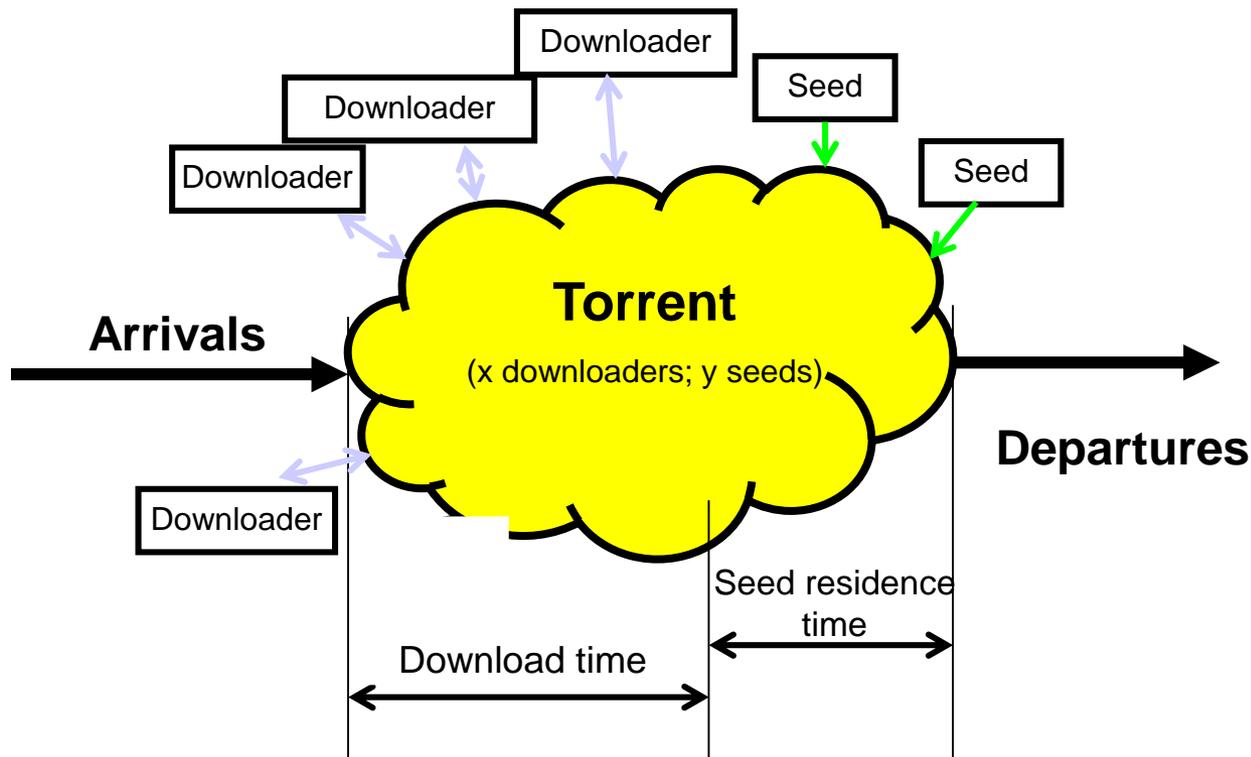


DHT Layered Architecture



BitTorrent-like systems

- ❑ File split into many smaller pieces
- ❑ Pieces are downloaded from both seeds and downloaders
- ❑ Distribution paths are dynamically determined
 - Based on data availability

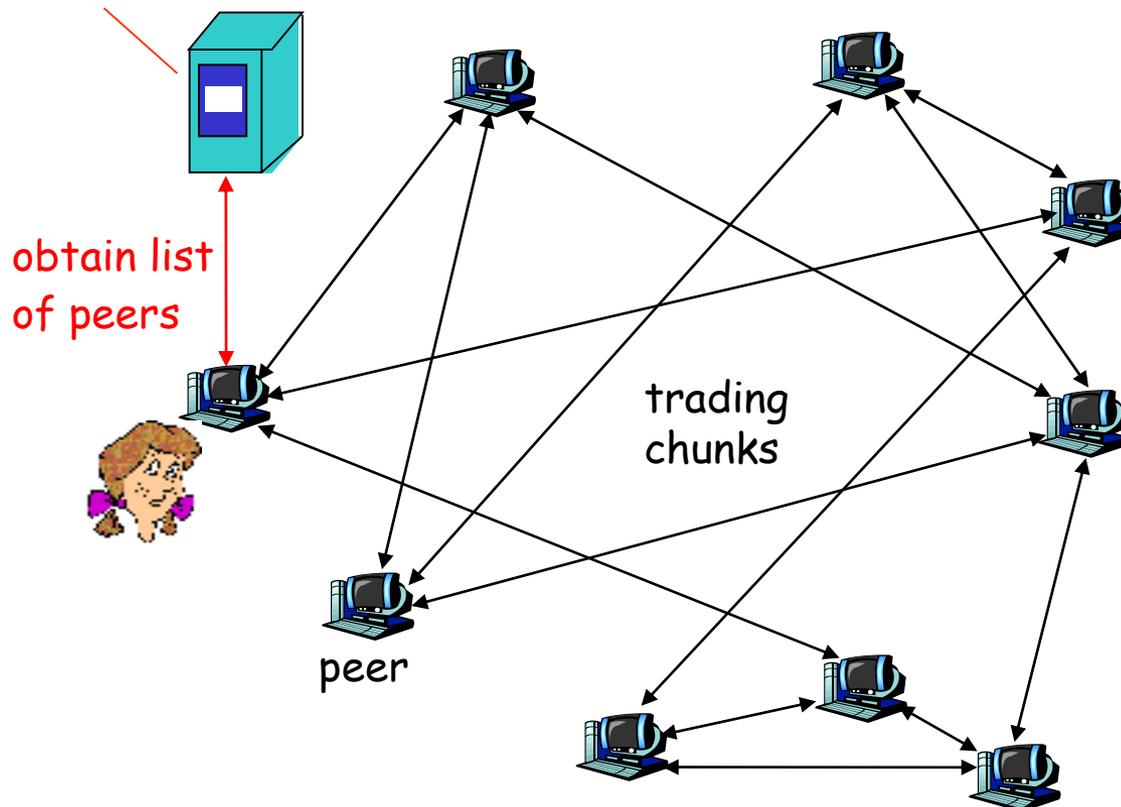


File distribution: BitTorrent

- P2P file distribution

tracker: tracks peers participating in torrent

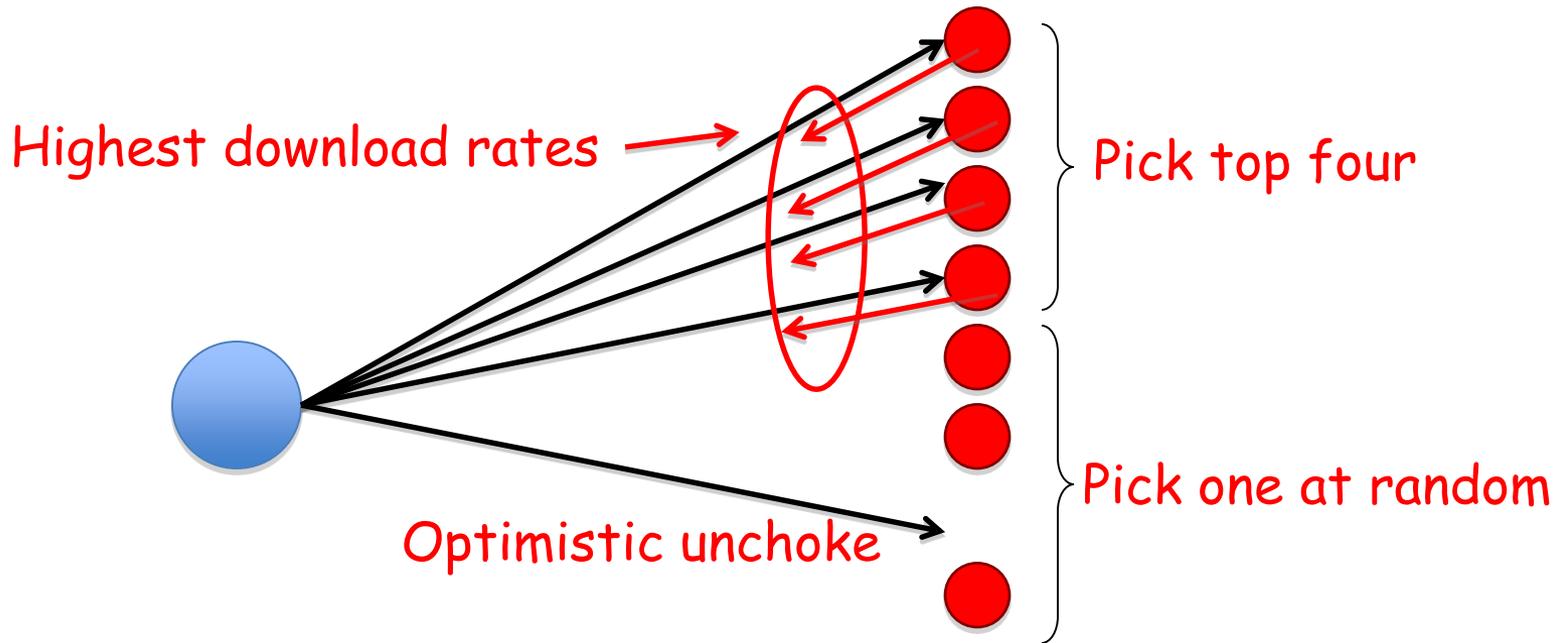
torrent: group of peers exchanging chunks of a file



Download using BitTorrent

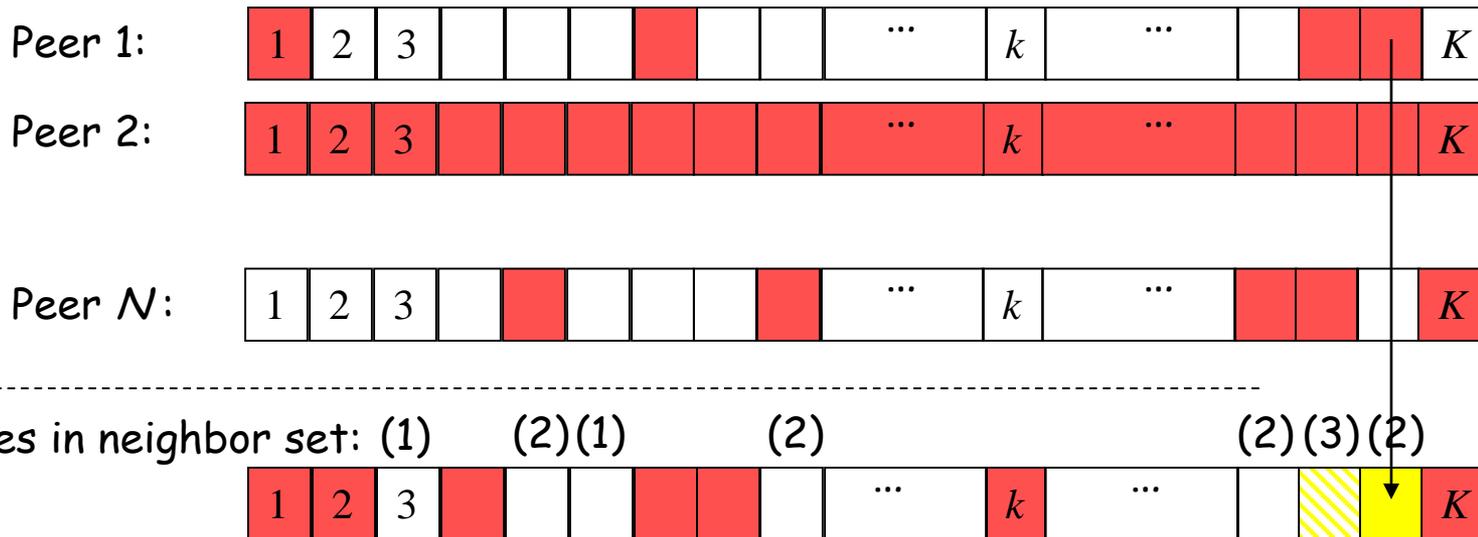
Background: Incentive mechanism

- ❑ Establish connections to large set of peers
 - At each time, only upload to a small (changing) set of peers
- ❑ Rate-based tit-for-tat policy
 - Downloaders give upload preference to the downloaders that provide the highest download rates



Download using BitTorrent

Background: Piece selection

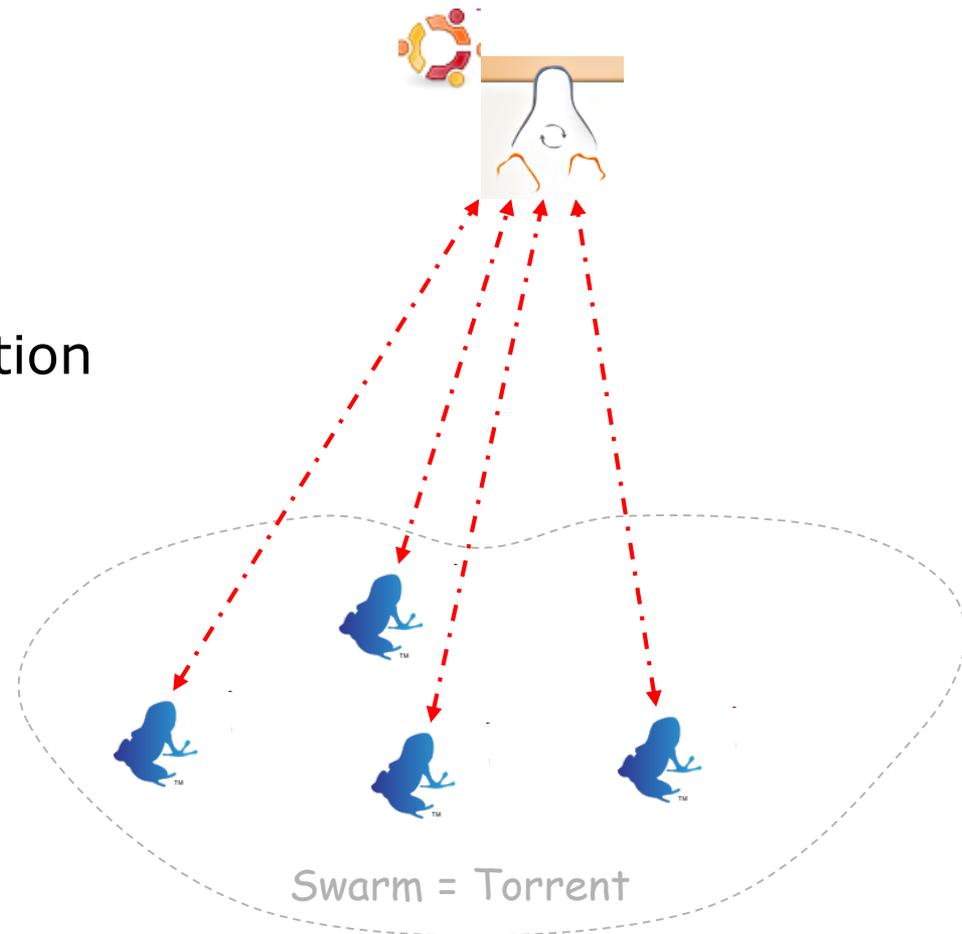


- ❑ Rarest first piece selection policy
 - Achieves high piece diversity
- ❑ Request pieces that
 - the uploader has;
 - the downloader is interested (wants); and
 - is the rarest among this set of pieces

Background

Peer discovery in BitTorrent

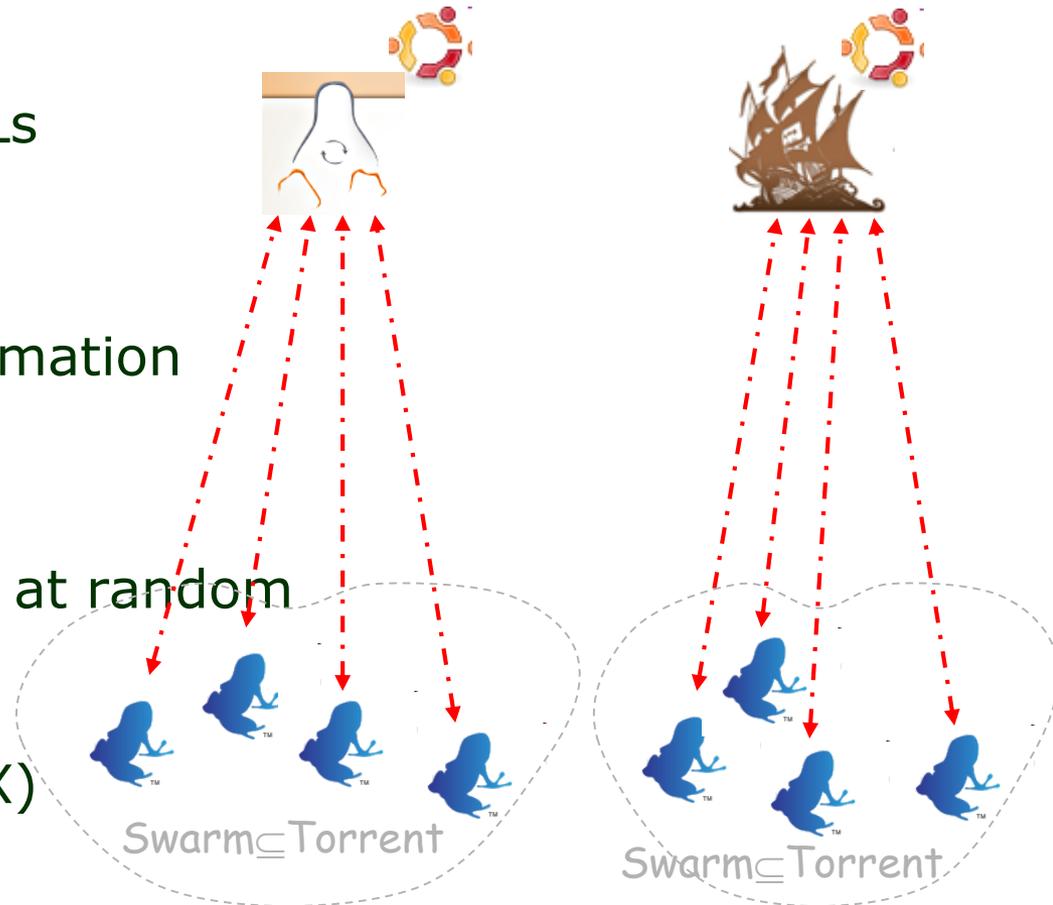
- ❑ Torrent file 
 - "announce" URL
- ❑ Tracker 
 - Register torrent file
 - Maintain state information
- ❑ Peers
 - Obtain torrent file
 - Announce
 - Report status
 - Peer exchange (PEX)
- ❑ Issues
 - Central point of failure
 - Tracker load



Background

Multi-tracked torrents

- ❑ Torrent file 
 - "announce-list" URLs
- ❑ Trackers 
 - Register torrent file
 - Maintain state information
- ❑ Peers 
 - Obtain torrent file
 - Choose **one** tracker at random
 - Announce
 - Report status
 - Peer exchange (PEX)
- ❑ Issue 
 - Multiple smaller swarms



Tracker-less torrents

- Combine DHTs and BT ... [notes on board]