

TDT506: Computer Networks

Instructor: Niklas Carlsson

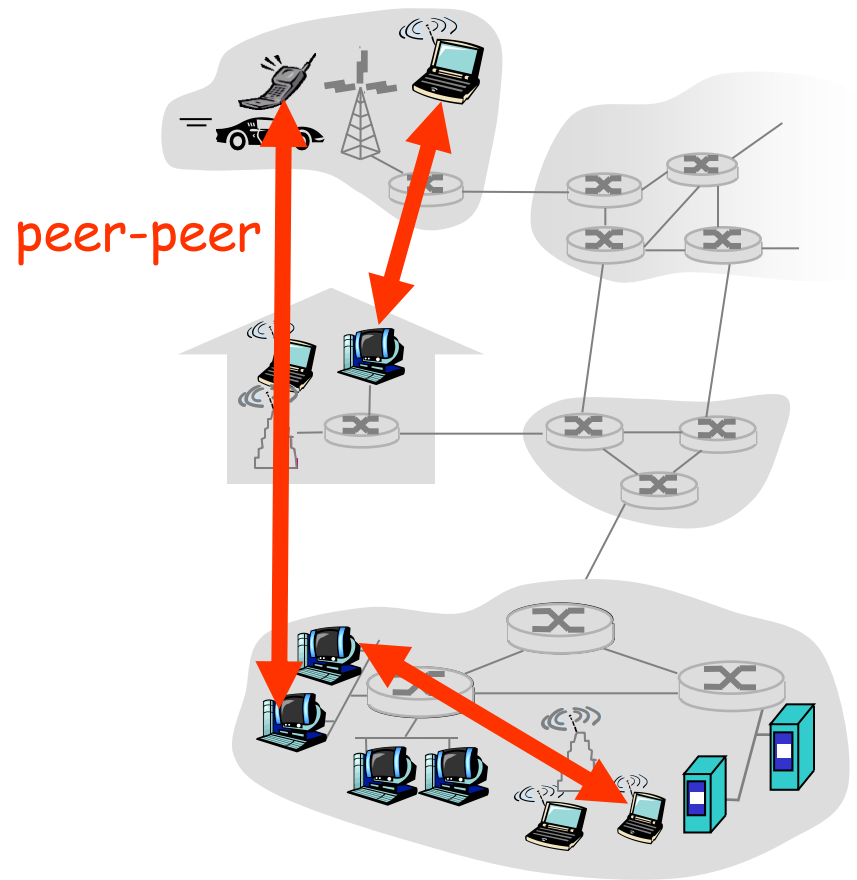
Email: niklas.carlsson@liu.se

Notes derived from "*Computer Networking: A Top Down Approach*", by Jim Kurose and Keith Ross, Addison-Wesley.

The slides are adapted and modified based on slides from the book's companion Web site, as well as modified slides by Anirban Mahanti and Carey Williamson.

Pure P2P architecture

- ❑ *no* always-on server
- ❑ arbitrary end systems directly communicate
- ❑ peers are intermittently connected and change IP addresses
- ❑ Three topics (in slides):
 - File sharing
 - File distribution
 - Searching for information
 - Case Studies: Bittorrent and Skype



P2P: centralized directory

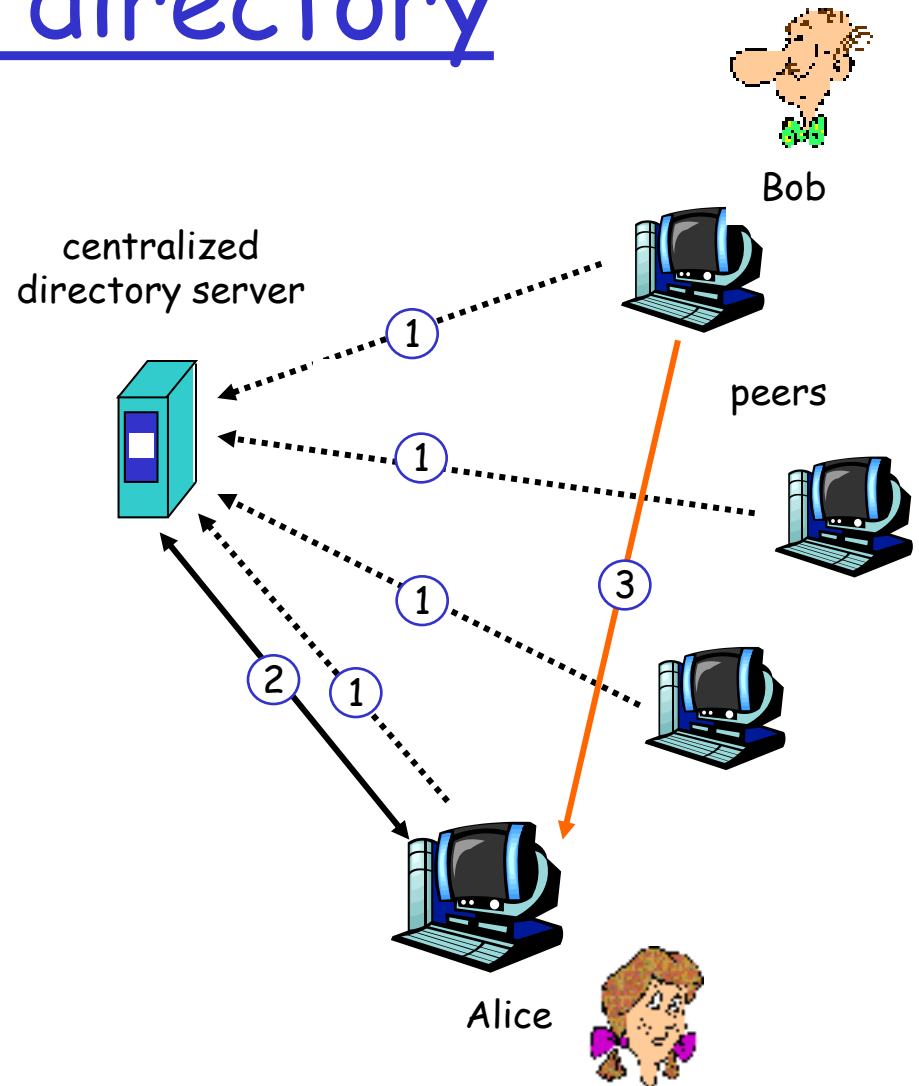
original "Napster" design

1) when peer connects, it informs central server:

- IP address
- content

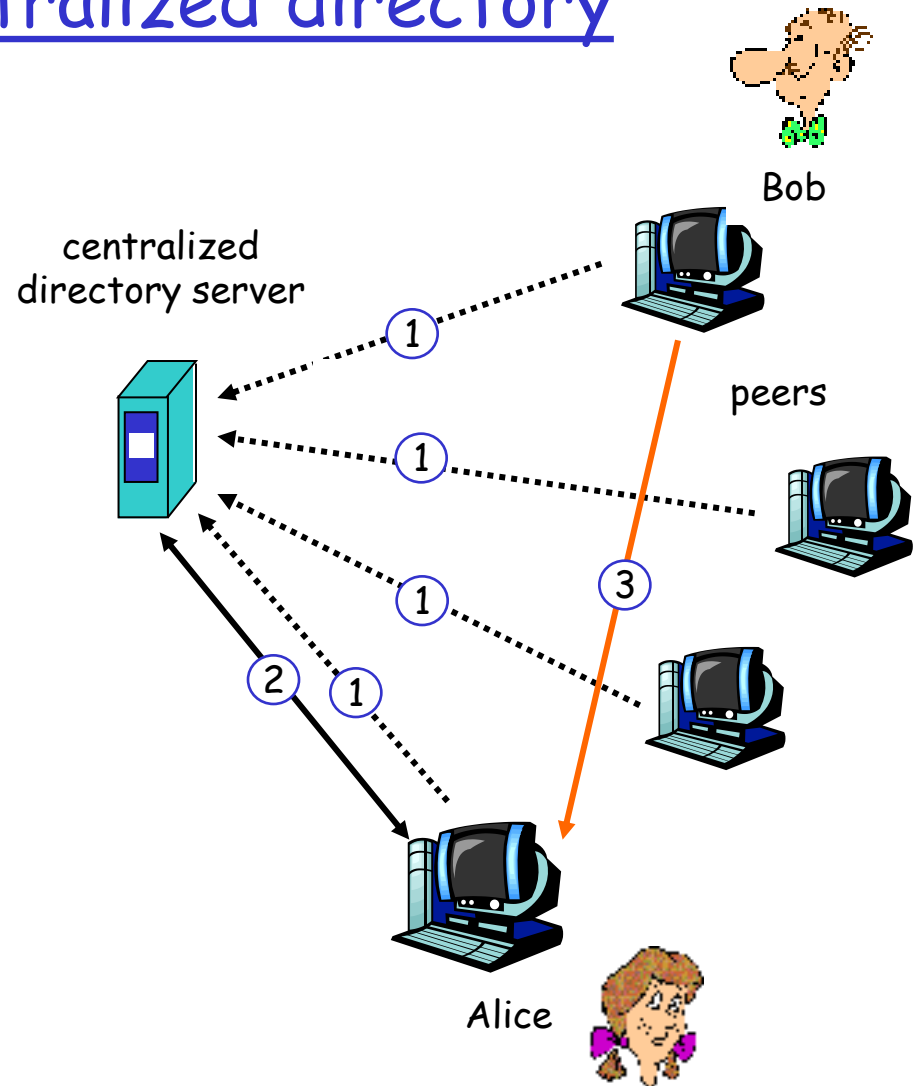
2) Alice queries for "Hey Jude"

3) Alice requests file from Bob



P2P: problems with centralized directory

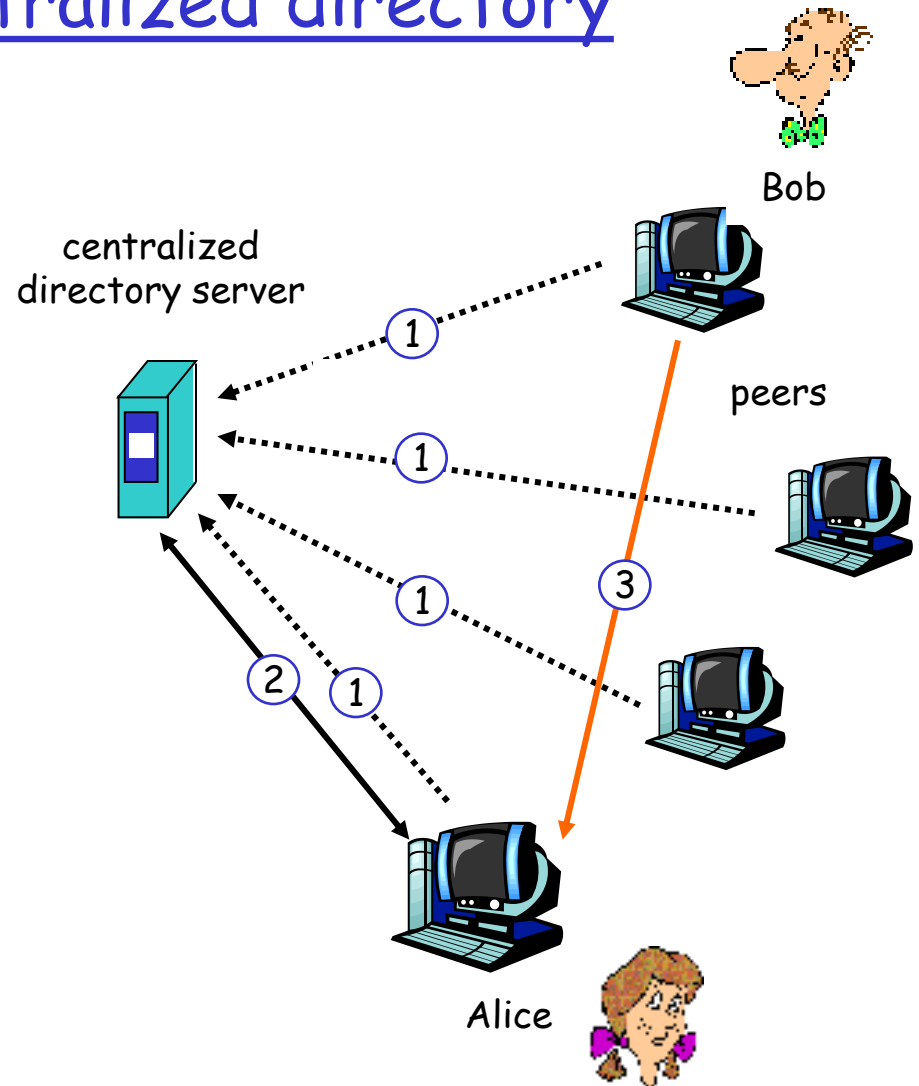
file transfer is
decentralized, but
locating content is
highly centralized



P2P: problems with centralized directory

file transfer is
decentralized, but
locating content is
highly centralized

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

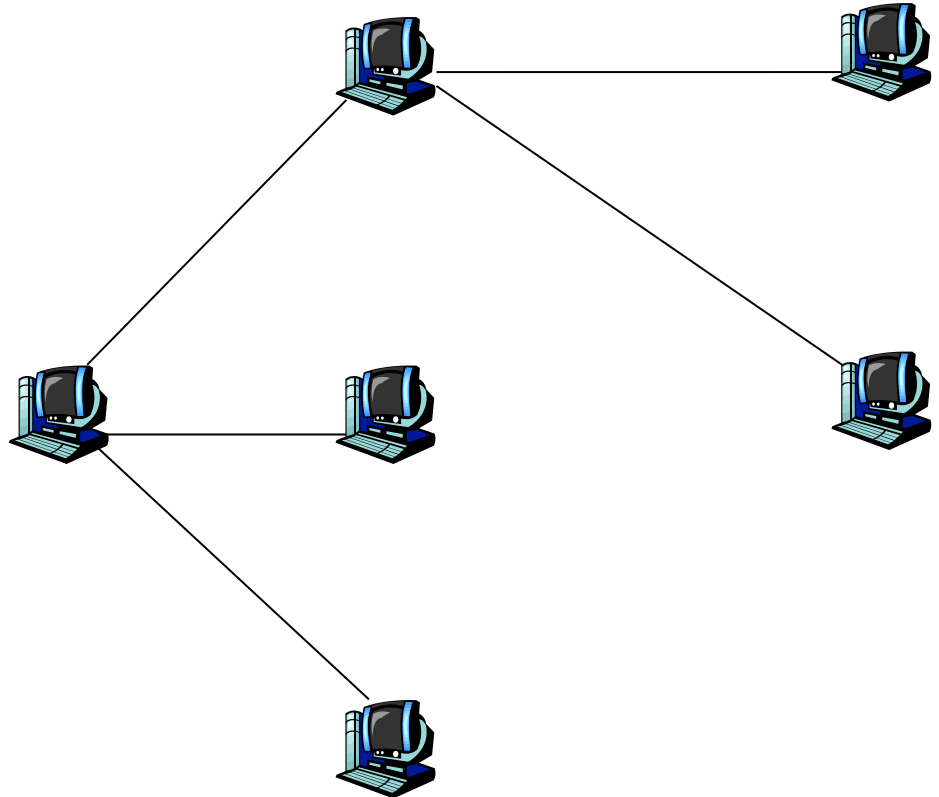


Query flooding: Gnutella

- ❑ fully distributed
 - no central server
- ❑ public domain protocol
- ❑ many Gnutella clients implementing protocol

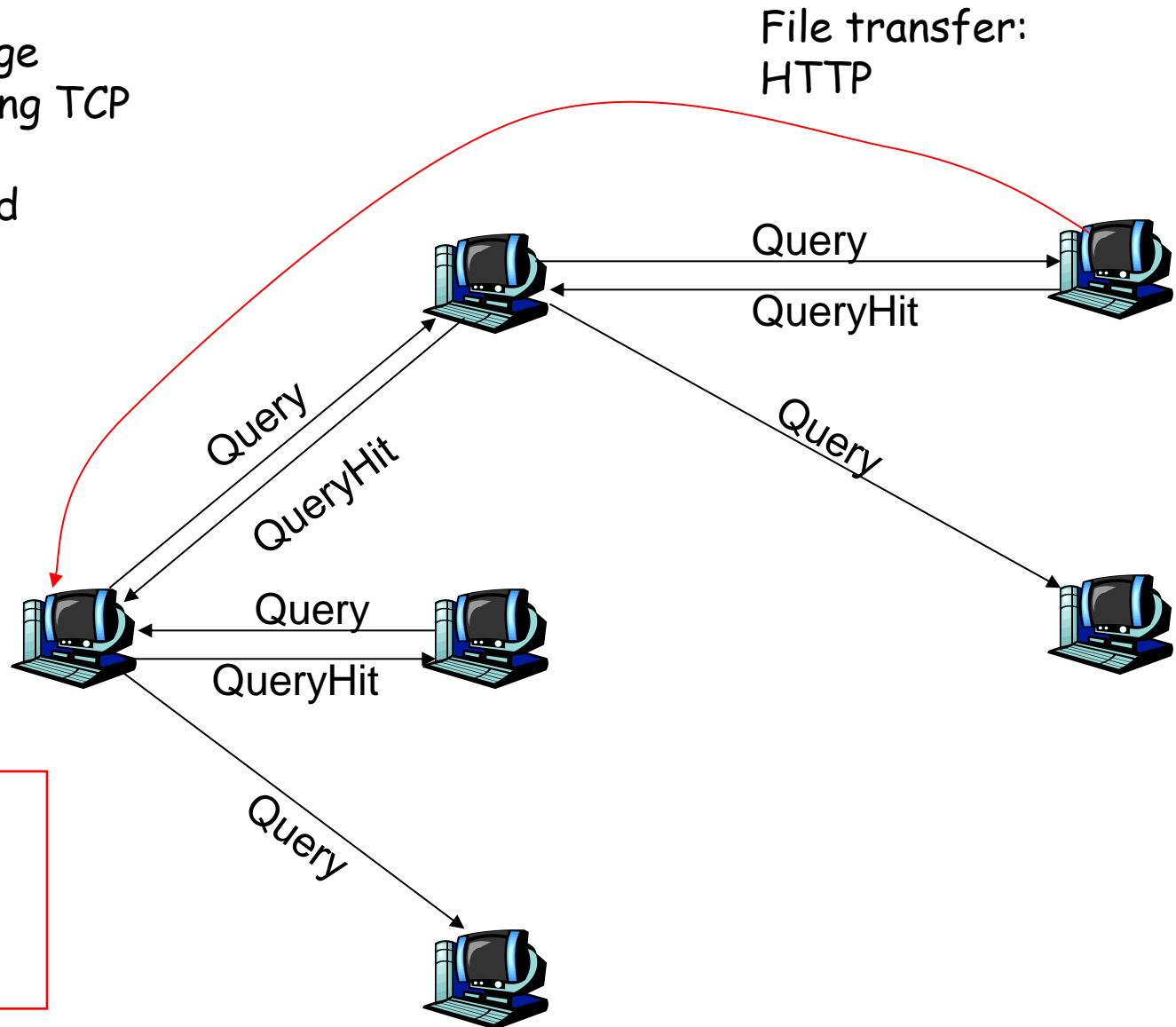
overlay network: graph

- ❑ edge between peer X and Y if there's a TCP connection
- ❑ all active peers and edges form overlay net
- ❑ edge: virtual (*not* physical) link
- ❑ given peer typically connected with < 10 overlay neighbors



Gnutella: protocol

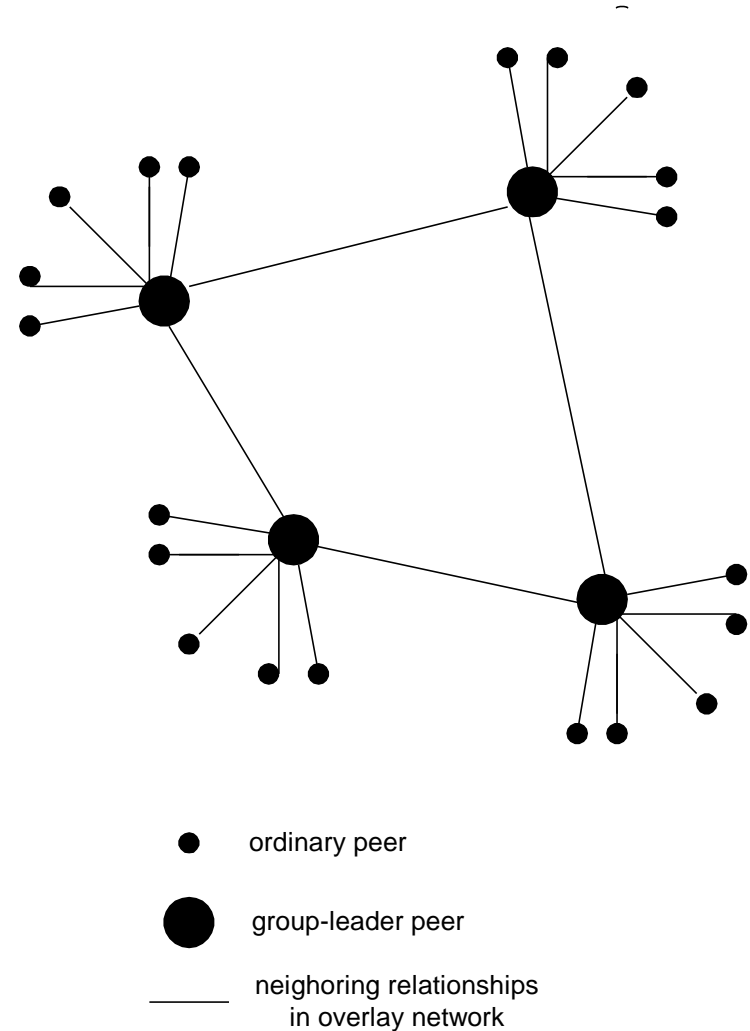
- ❑ Query message sent over existing TCP connections
- ❑ peers forward Query message
- ❑ QueryHit sent over reverse path



Scalability:
limited scope
flooding

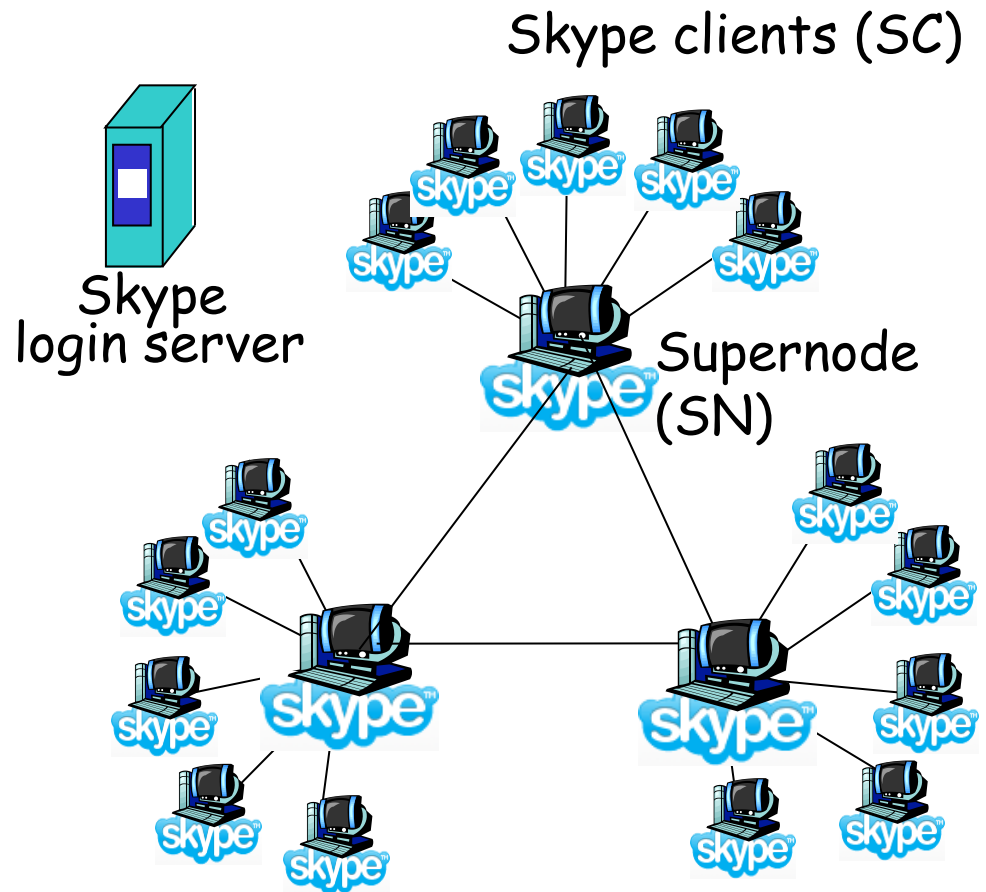
Hierarchical Overlay

- ❑ between centralized index, query flooding approaches
- ❑ each peer is either a *group leader* or assigned to a group leader.
- ❑ group leader tracks content in its children



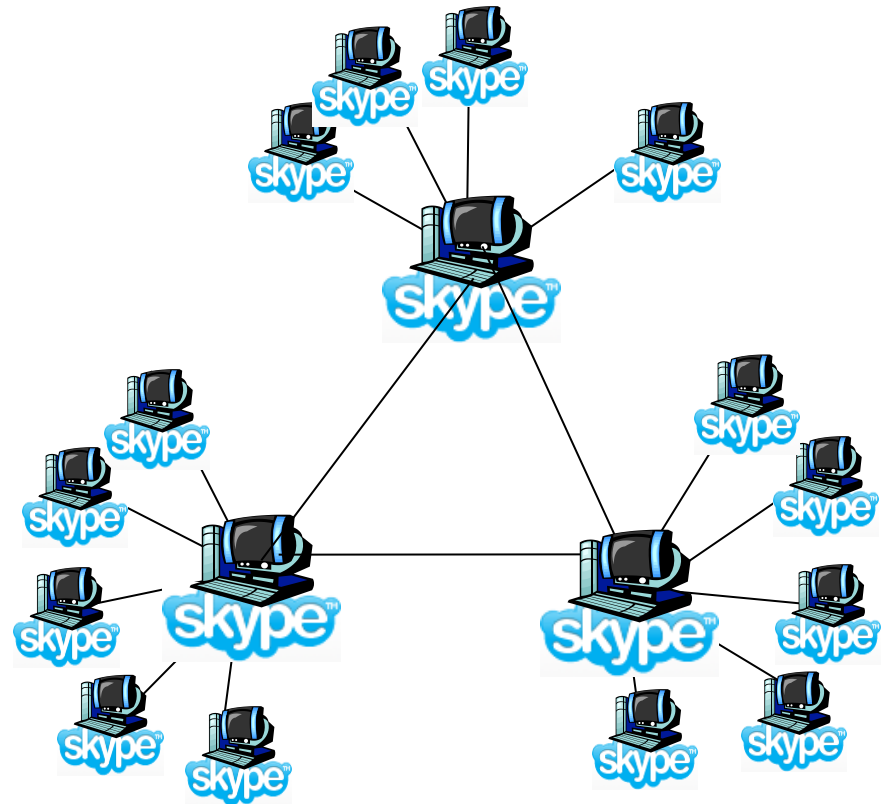
P2P Case study: Skype

- ❑ 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



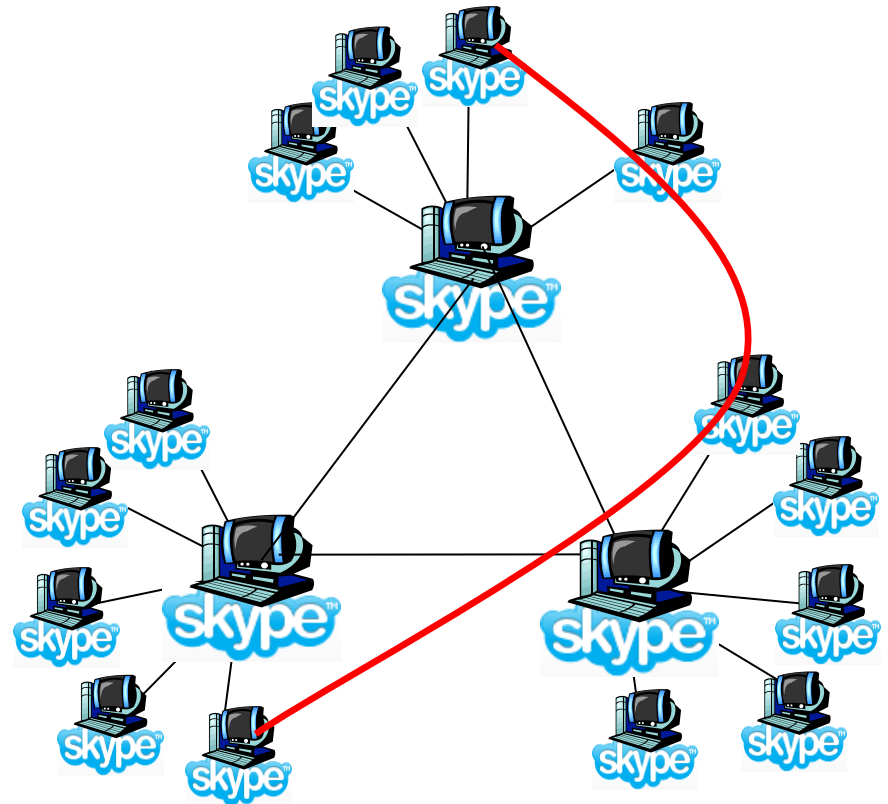
NAT/firewall problems ...

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



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

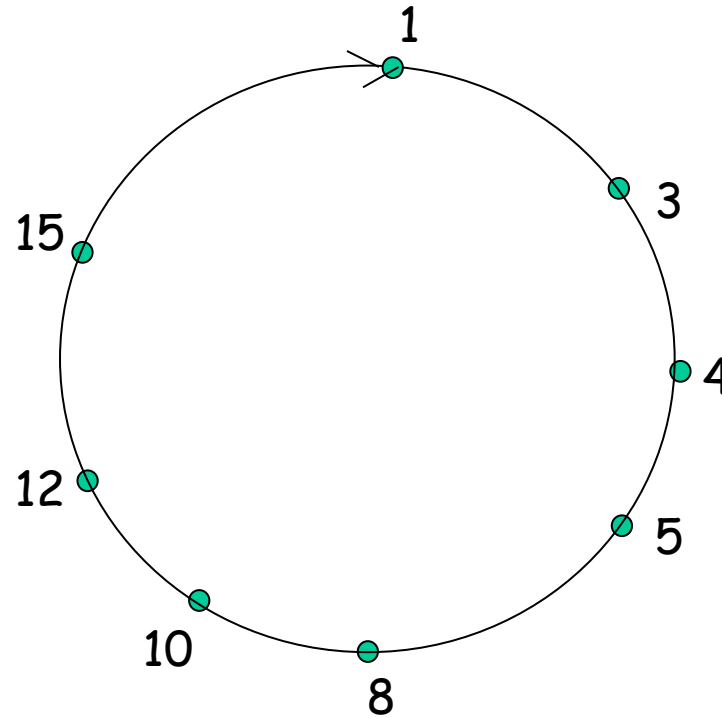
DHT Identifiers

- ❑ 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, $\text{key} = h(\text{"Led Zeppelin IV"})$
 - 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

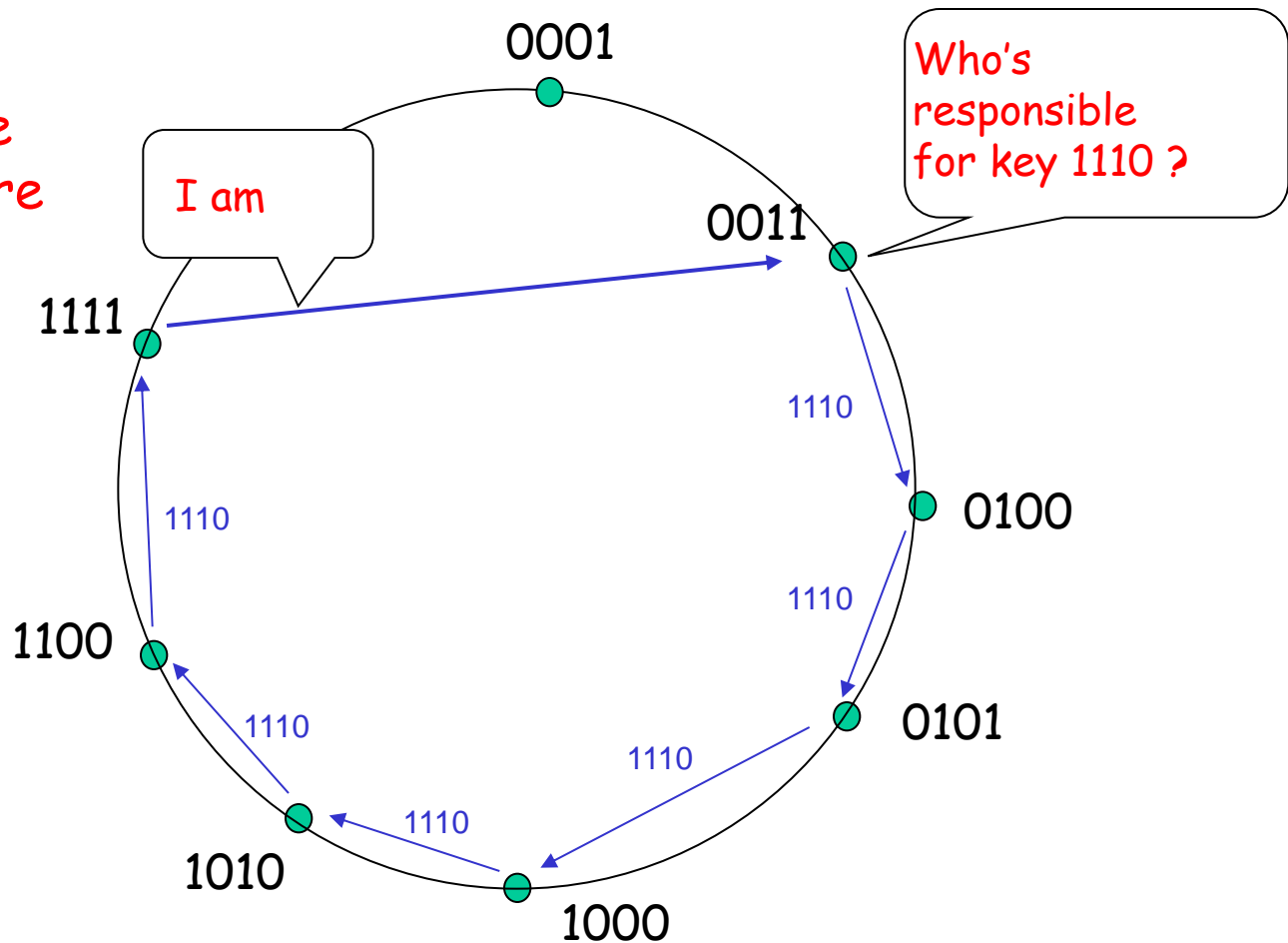
Circular DHT (1)



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

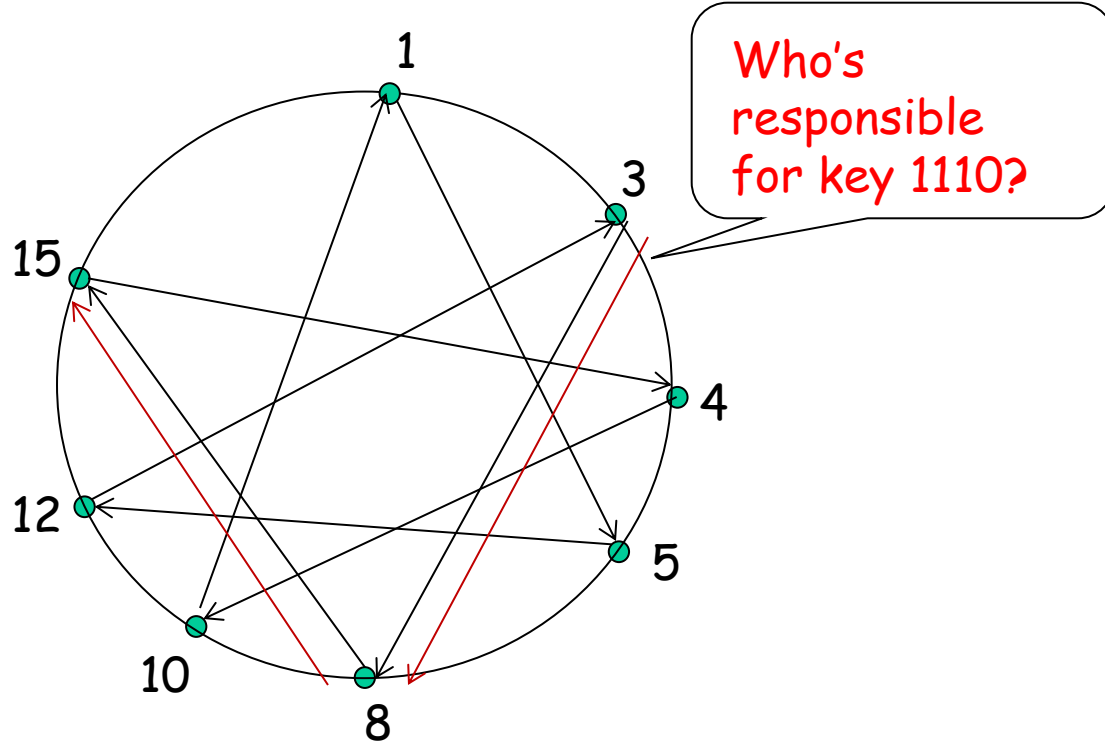
Circle DHT (2)

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



Define closest
as closest
successor

Circular DHT with Shortcuts

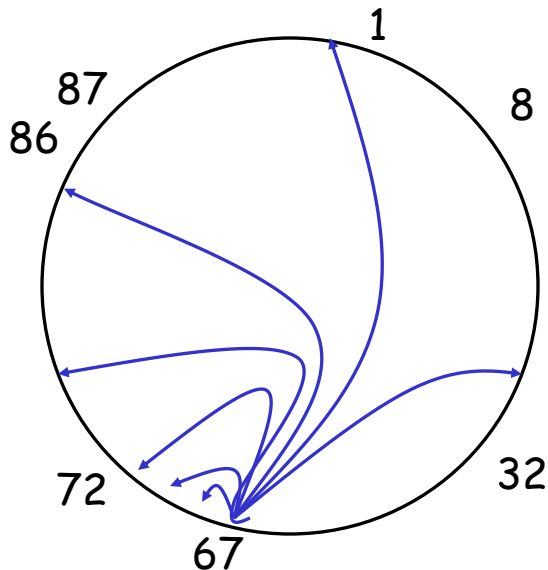


- ❑ Each peer keeps track of IP addresses of predecessor, successor, short cuts.
- ❑ Reduced from 6 to 2 messages.
- ❑ Possible to design shortcuts so $O(\log N)$ neighbors, $O(\log N)$ messages in query

Example: Chord Routing [see paper for details instead]

- ❑ 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)



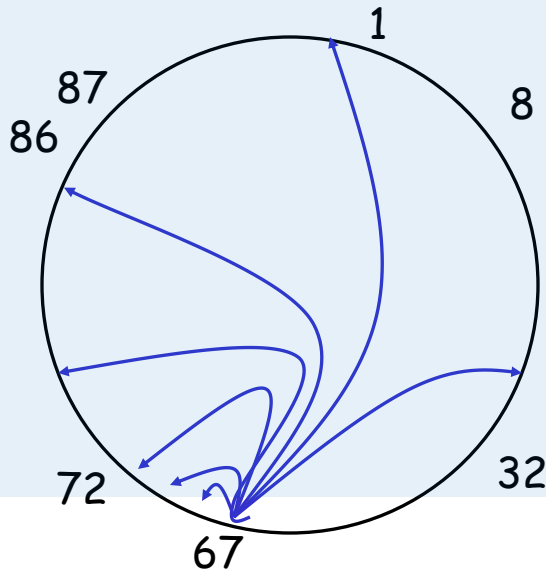
i	Finger table for node 67
0	72
1	72
2	72
3	86
4	86
5	1
6	32

Closest node clockwise to

$67 + 2^i \bmod 100$

Chord Routing (cont'd)

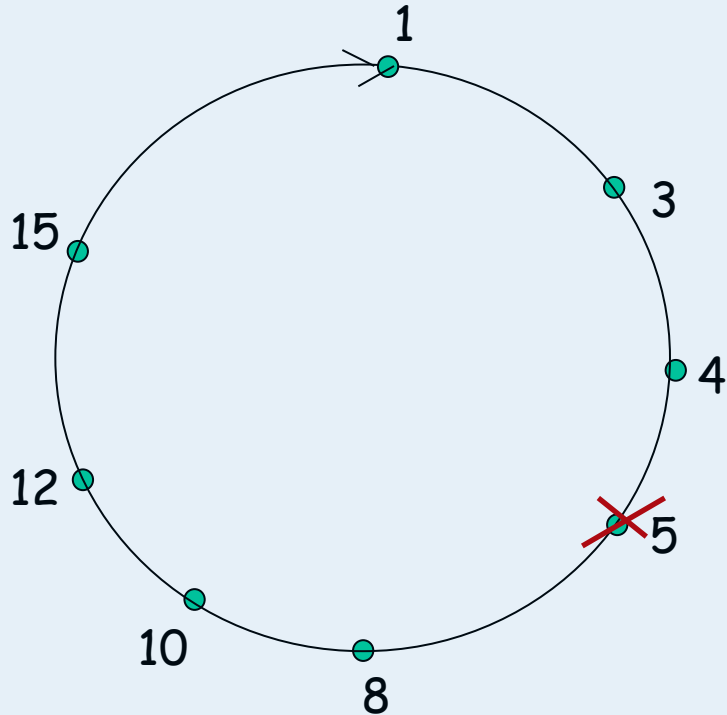
- A node s is node t 's neighbor if s is the closest node to $t+2^i \bmod H$ for some i . Thus,
 - each node has at most $\log_2 N$ neighbors
 - for any object, the node whose range contains the object is reachable from any node in no more than $\log_2 N$ overlay hops
(each step can always traverse at least half the distance to the ID)
- Given K objects, with high probability each node has at most $(1 + \log_2 N) K / N$ in its range
- When a new node joins or leaves the overlay, $O(K / N)$ objects move between nodes



i	Finger table for node 67
0	72
1	72
2	72
3	86
4	86
5	1
6	32

Closest node clockwise to $67+2^i \bmod 100$

Peer Churn

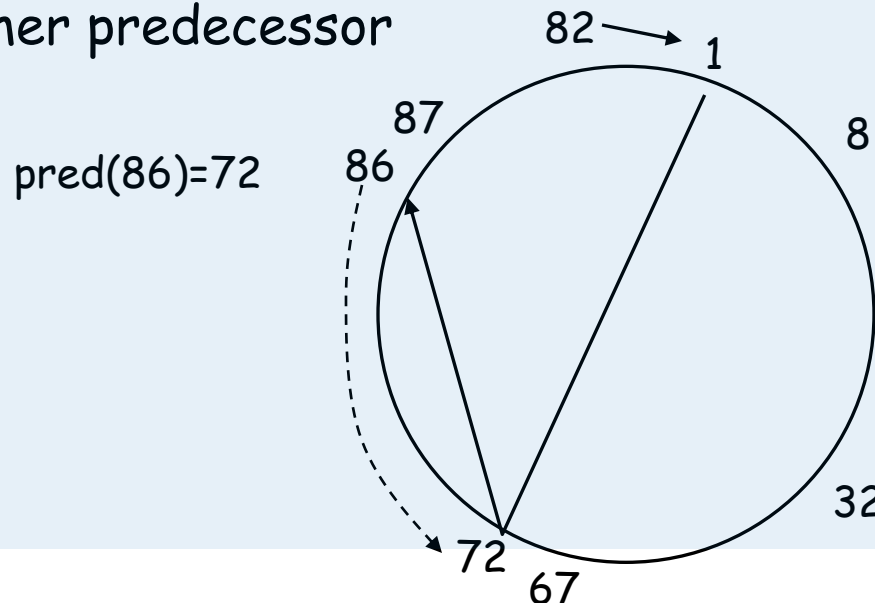


- To handle peer churn, require each peer to know the IP address of its two successors.
- Each peer periodically pings its two successors to see if they are still alive.

- ❑ Peer 5 abruptly leaves
- ❑ Peer 4 detects; makes 8 its immediate successor; asks 8 who its immediate successor is; makes 8's immediate successor its second successor.
- ❑ What if peer 13 wants to join?

Chord Node Insertion

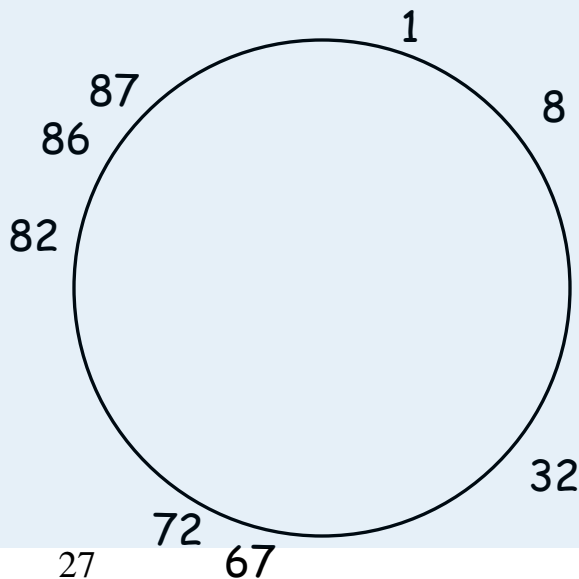
- ❑ One protocol addition: each node knows its closest counter-clockwise neighbor
- ❑ A node selects its unique (pseudo-random) ID and uses a bootstrapping process to find some node in the Chord
- ❑ Using Chord, the node identifies its successor in the clockwise direction
- ❑ An newly inserted node's predecessor is its successor's former predecessor



Example: Insert 82

Chord Node Insertion (cont'd)

- First: set added node s 's fingers correctly
 - s 's predecessor t does the lookup for each distance of 2^i from s



Lookups from node 72

Lookup(83) = 86 →

Lookup(84) = 86 →

Lookup(86) = 86 →

Lookup(90) = 1 →

Lookup(98) = 1 →

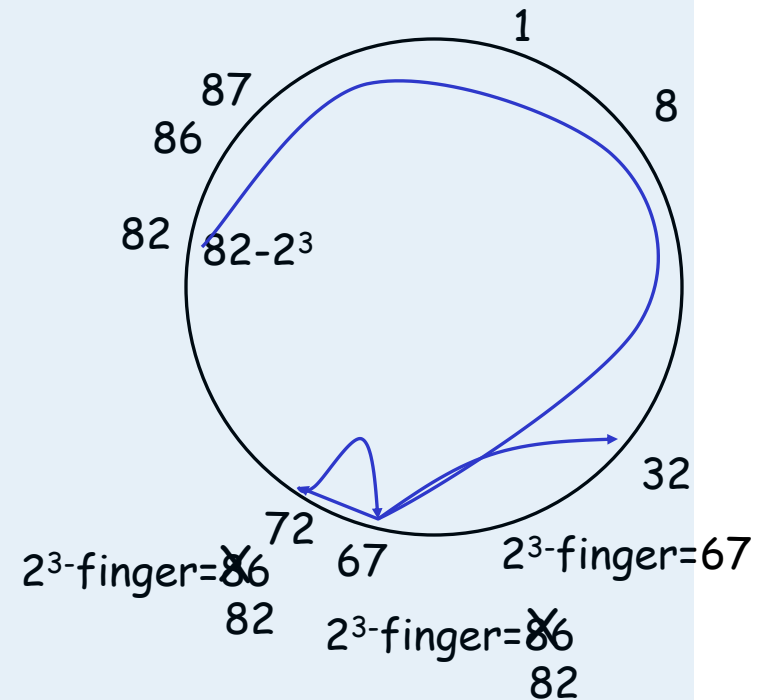
Lookup(14) = 32 →

Lookup(46) = 67 →

i	Finger table for node 82
0	86
1	86
2	86
3	1
4	1
5	32
6	67

Chord Node Insertion (cont'd)

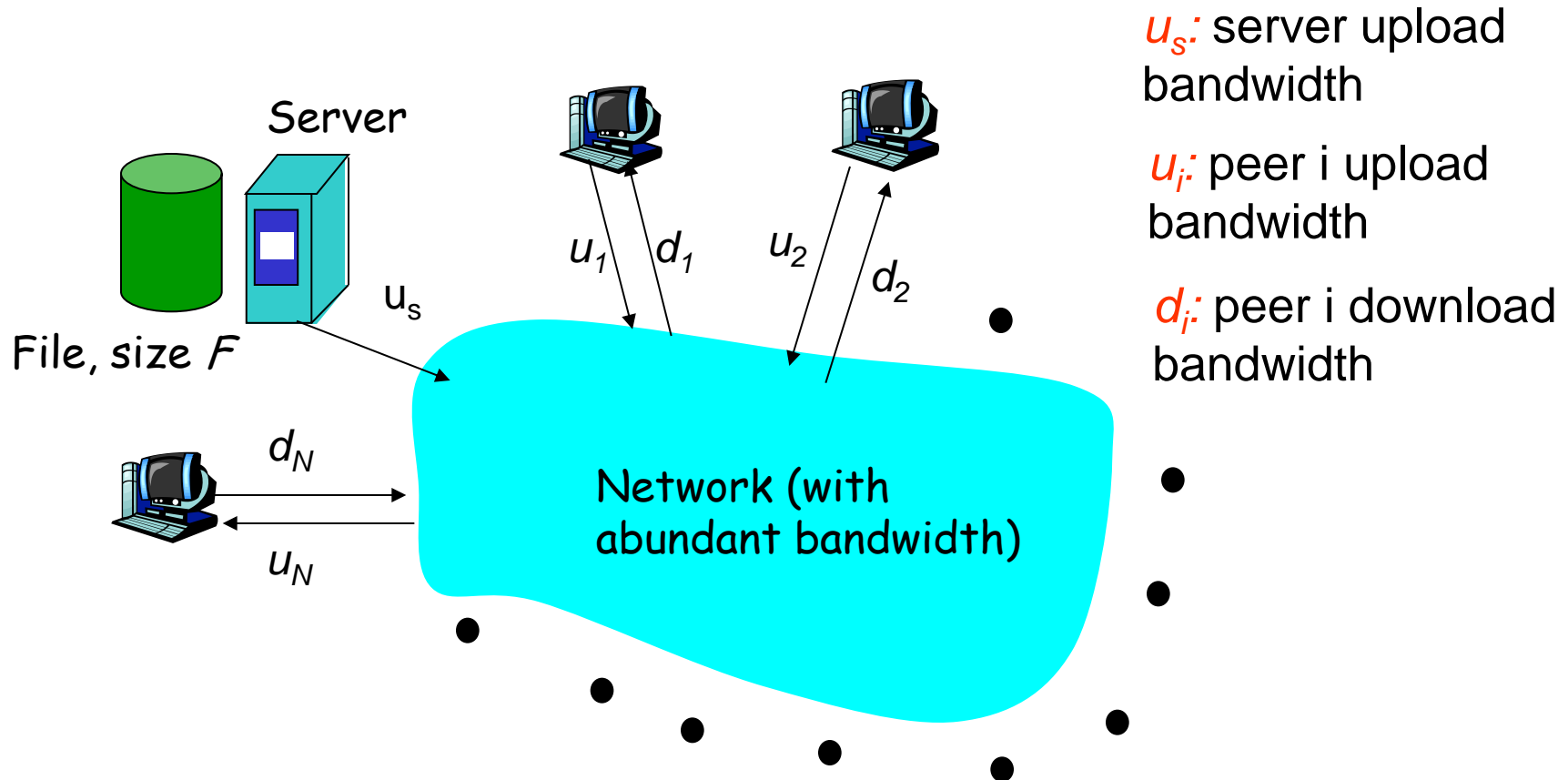
- Next, update other nodes' fingers about the entrance of s (when relevant). For each i :
 - Locate the closest node to s (counter-clockwise) whose 2^i -finger can point to s : largest possible is $s - 2^i$
 - Use Chord to go (clockwise) to largest node t before or at $s - 2^i$
 - route to $s - 2^i$, if arrived at a larger node, select its predecessor as t
 - If t 's 2^i -finger routes to a node larger than s
 - change t 's 2^i -finger to s
 - set $t = \text{predecessor of } t$ and repeat
 - Else $i++$, repeat from top
- $O(\log^2 N)$ time to find and update nodes



e.g., for $i=3$

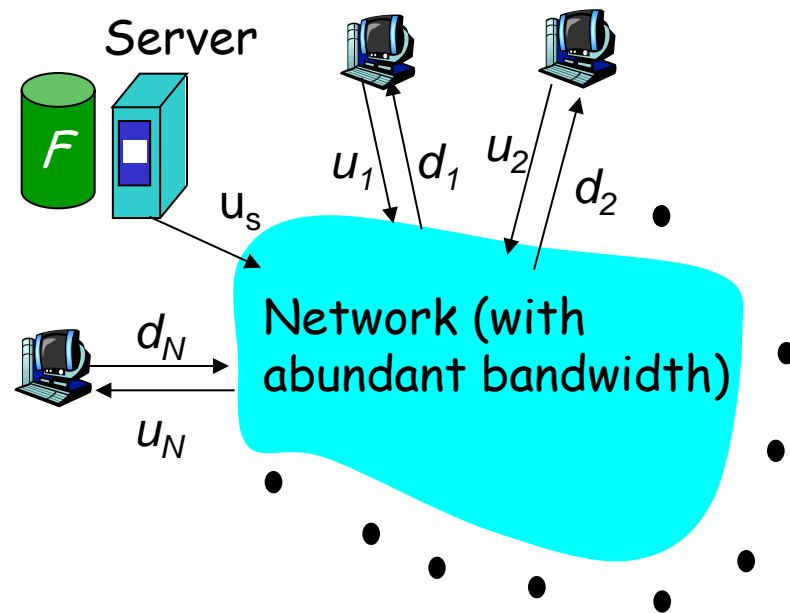
File Distribution: Server-Client vs P2P

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



File distribution time: server-client

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

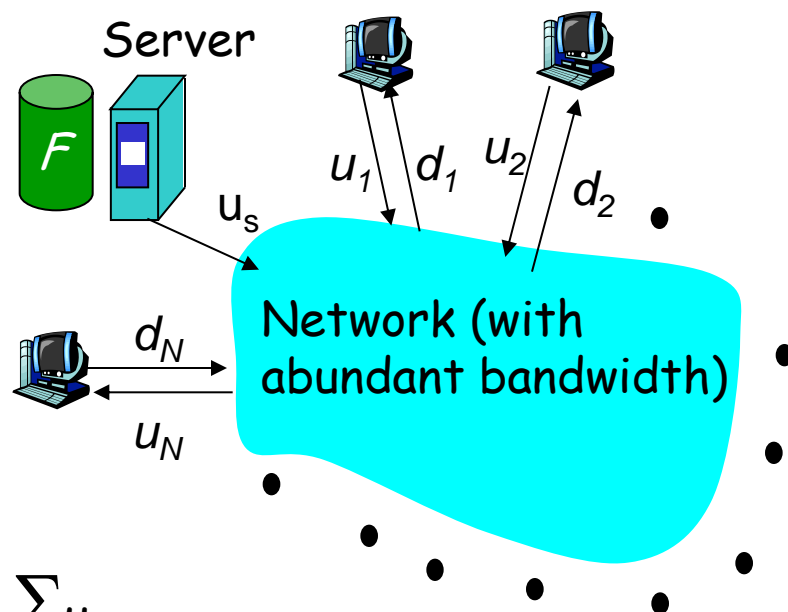


Time to distribute F to N clients using client/server approach = $d_{cs} = \max \left\{ NF/u_s, F/\min(d_i) \right\}_i$

increases linearly in N (for large N)

File distribution time: P2P

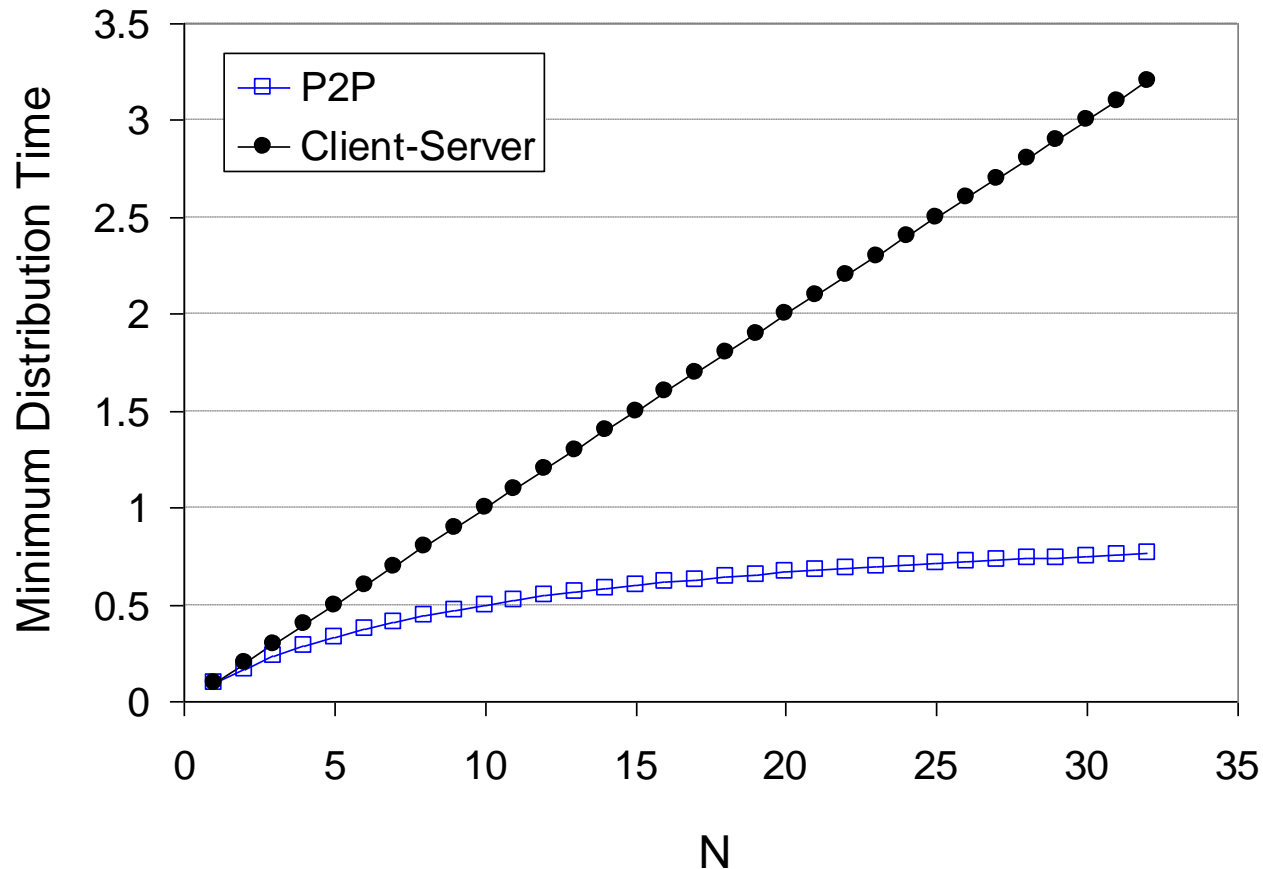
- ❑ server must send one copy: F/u_s time
- ❑ client i takes F/d_i time to download
- ❑ NF bits must be downloaded (aggregate)
 - ❑ fastest possible upload rate: $u_s + \sum u_i$



$$d_{p2p} = \max \left\{ F/u_s, F/\min(d_i), NF/(u_s + \sum u_i) \right\}$$

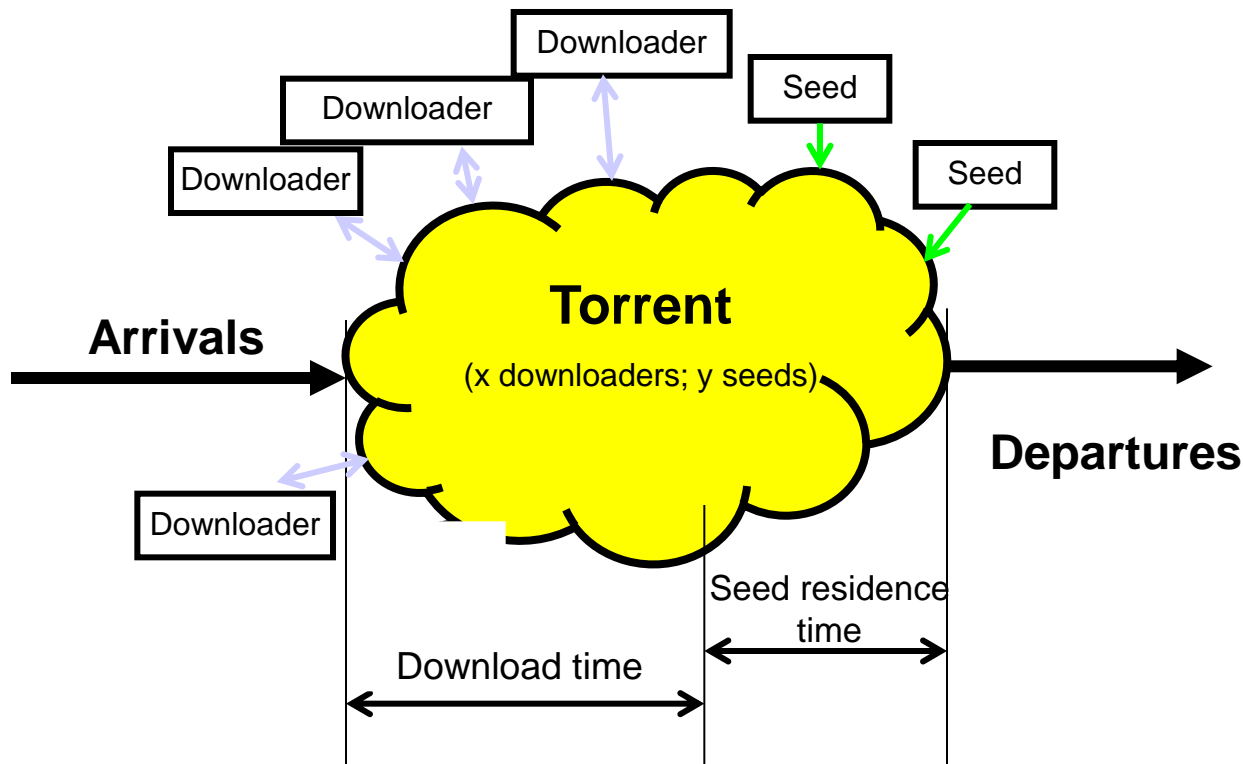
Server-client vs. P2P: example

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



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

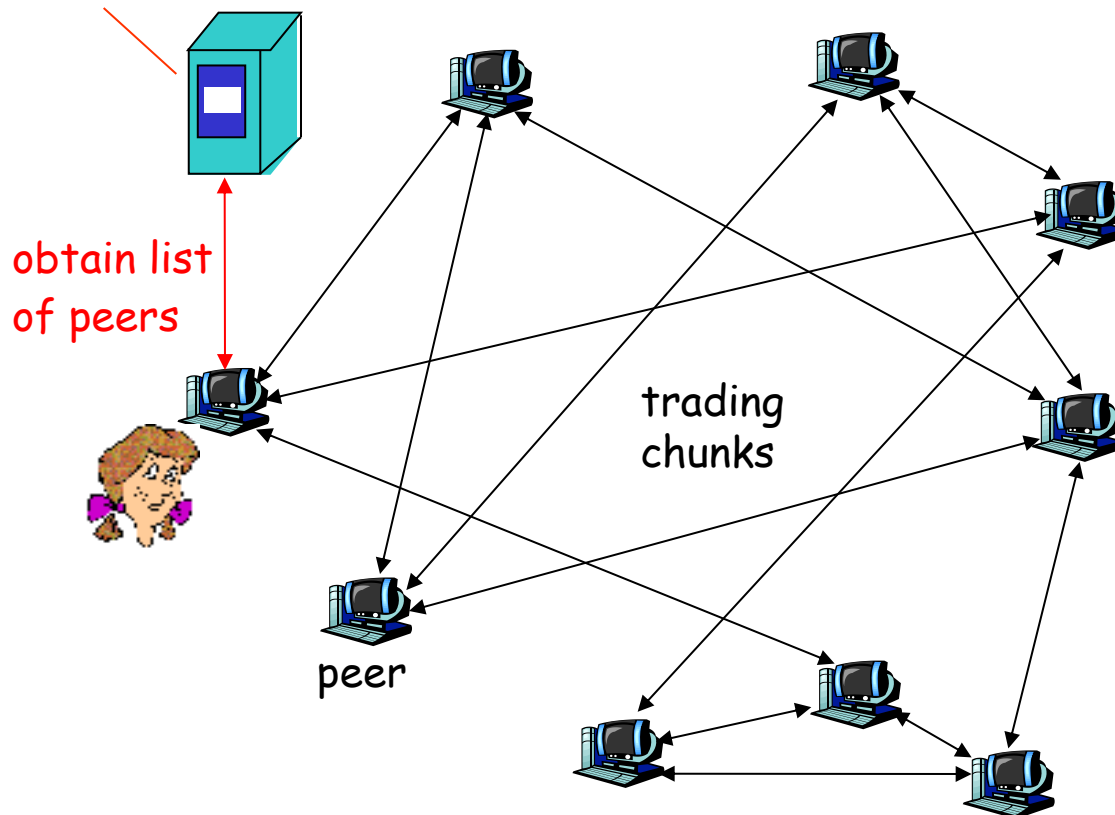


File distribution: BitTorrent

❑ P2P file distribution



tracker: tracks peers participating in torrent

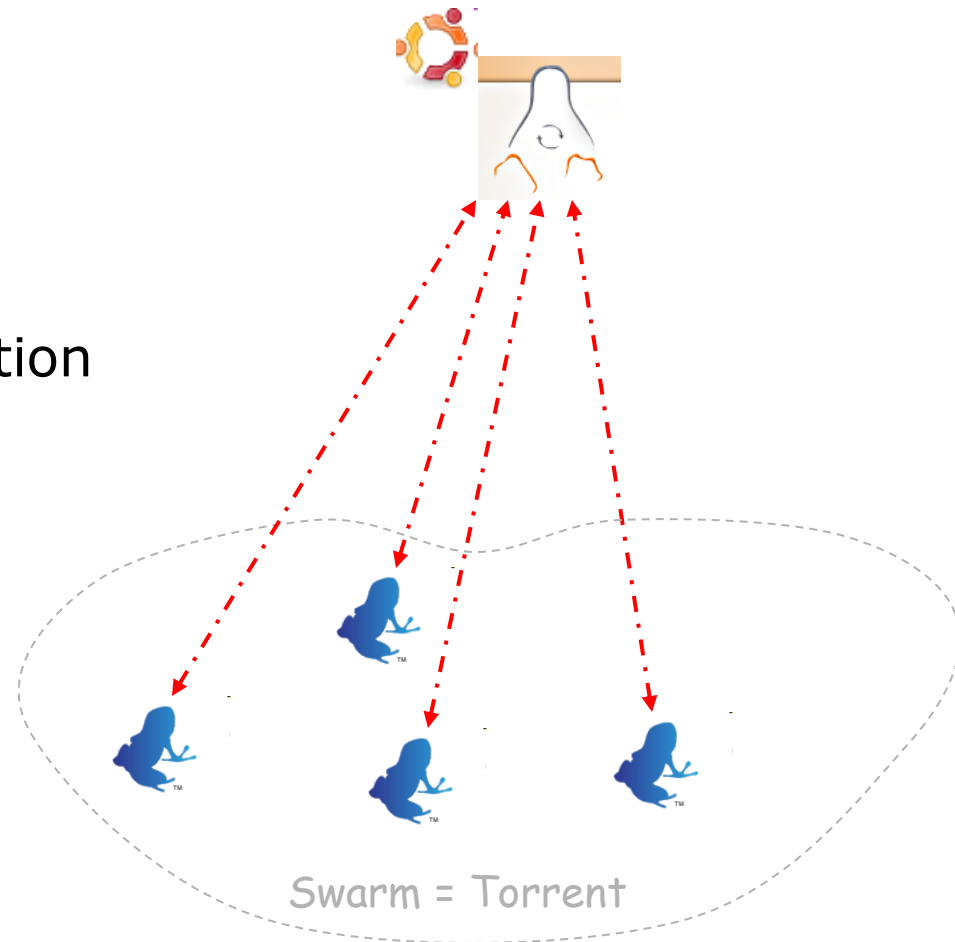
torrent: group of peers exchanging chunks of a file



Background



Peer discovery in BitTorrent

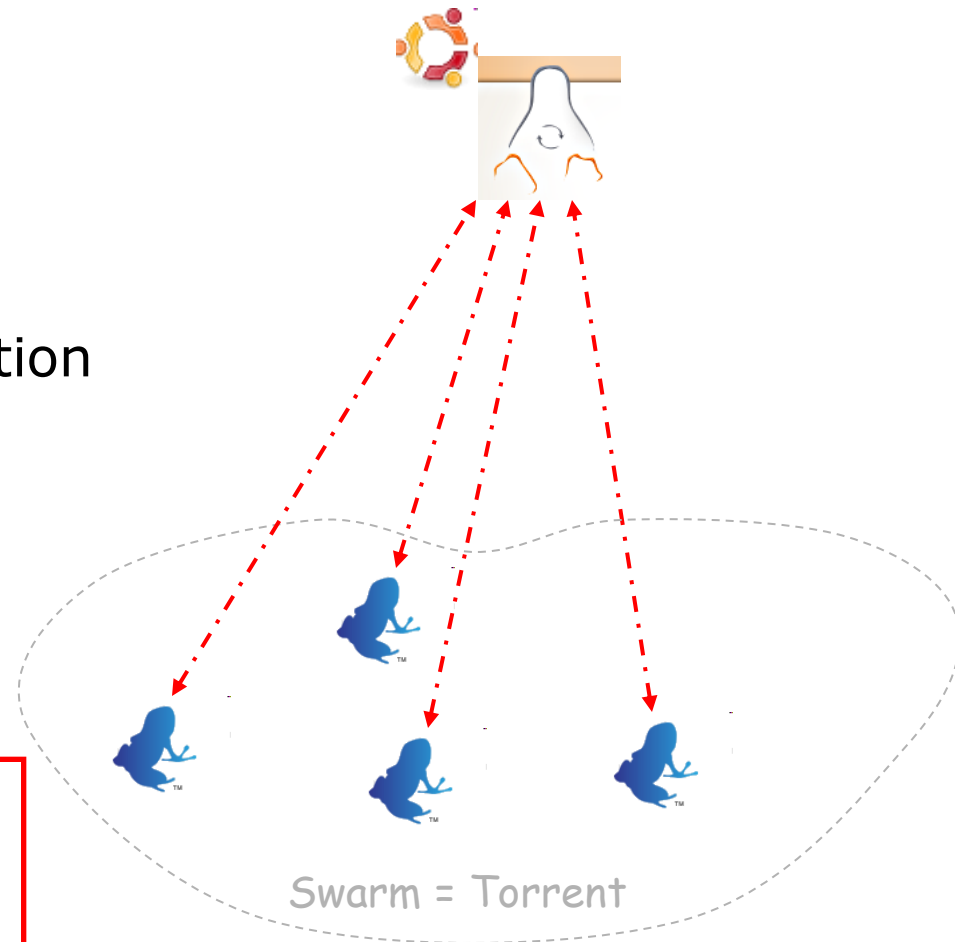
- ❑ Torrent file 
 - “announce” URL
- ❑ Tracker 
 - Register torrent file
 - Maintain state information
- ❑ Peers
 - Obtain torrent file
 - Announce
 - Report status
 - Peer exchange (PEX)
- ❑ Issues
 - Central point of failure
 - Tracker load



Background



Peer discovery in BitTorrent

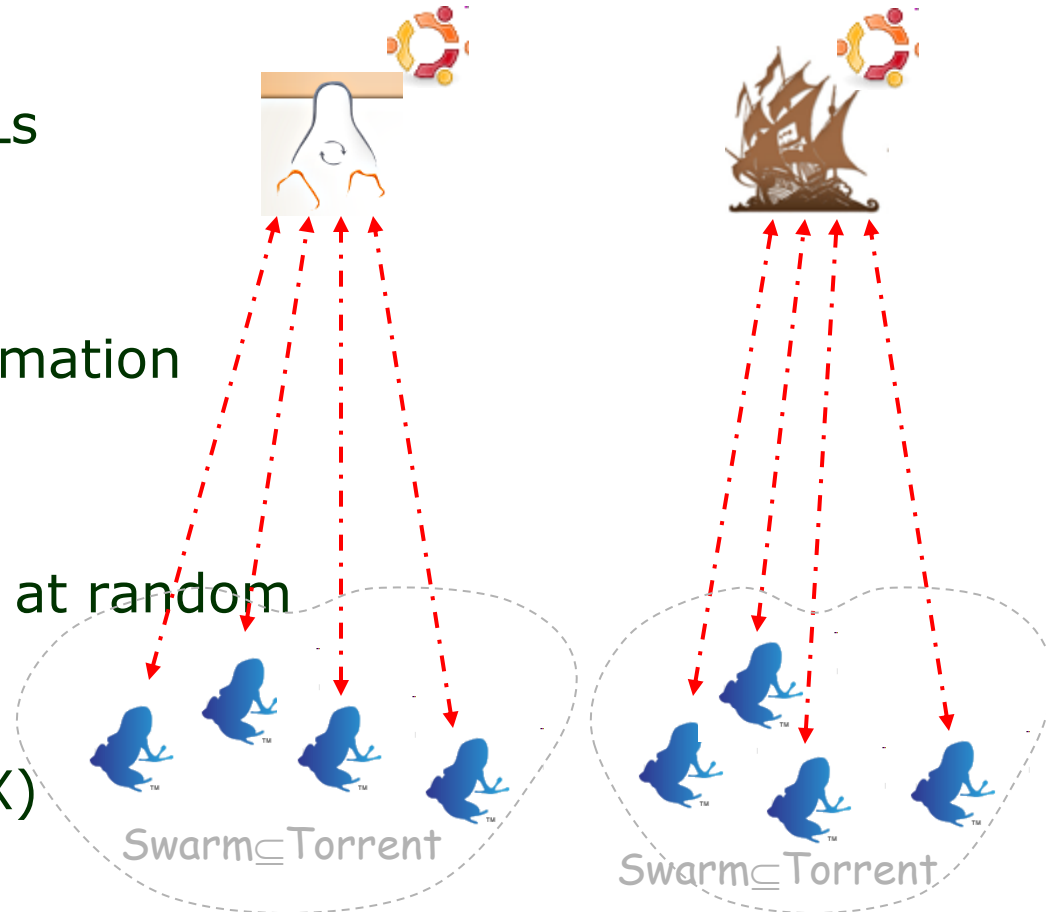
- ❑ Torrent file 
 - “announce” URL
- ❑ Tracker 
 - Register torrent file
 - Maintain state information
- ❑ Peers
 - Obtain torrent file
 - Announce
 - Report status
 - Peer exchange (PEX)
- ❑ Issues
 - Central point of failure
 - Tracker load



Background



Multi-tracked torrents

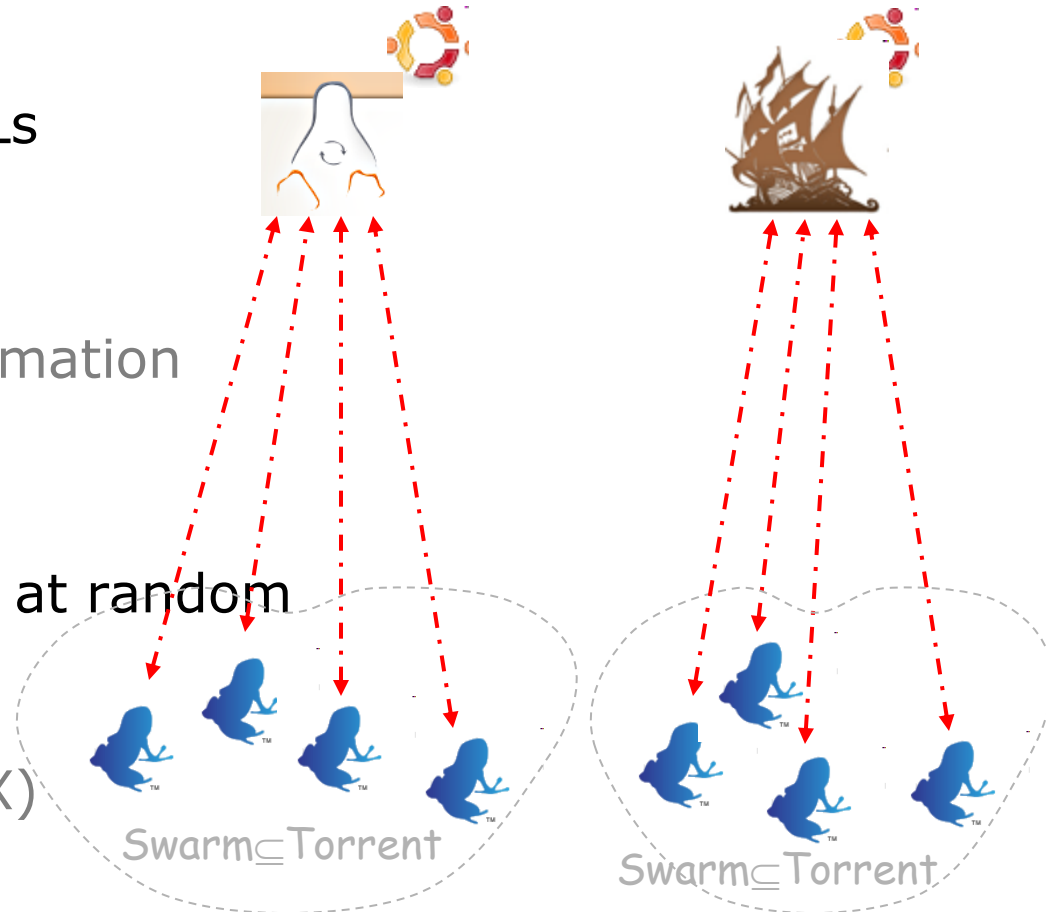
- ❑ Torrent file 
 - "announce-list" URLs
- ❑ Trackers 
 - Register torrent file
 - Maintain state information
- ❑ Peers
 - Obtain torrent file
 - Choose **one** tracker at random
 - Announce
 - Report status
 - Peer exchange (PEX)
- ❑ Issue
 - Multiple smaller swarms



Background

Multi-tracked torrents

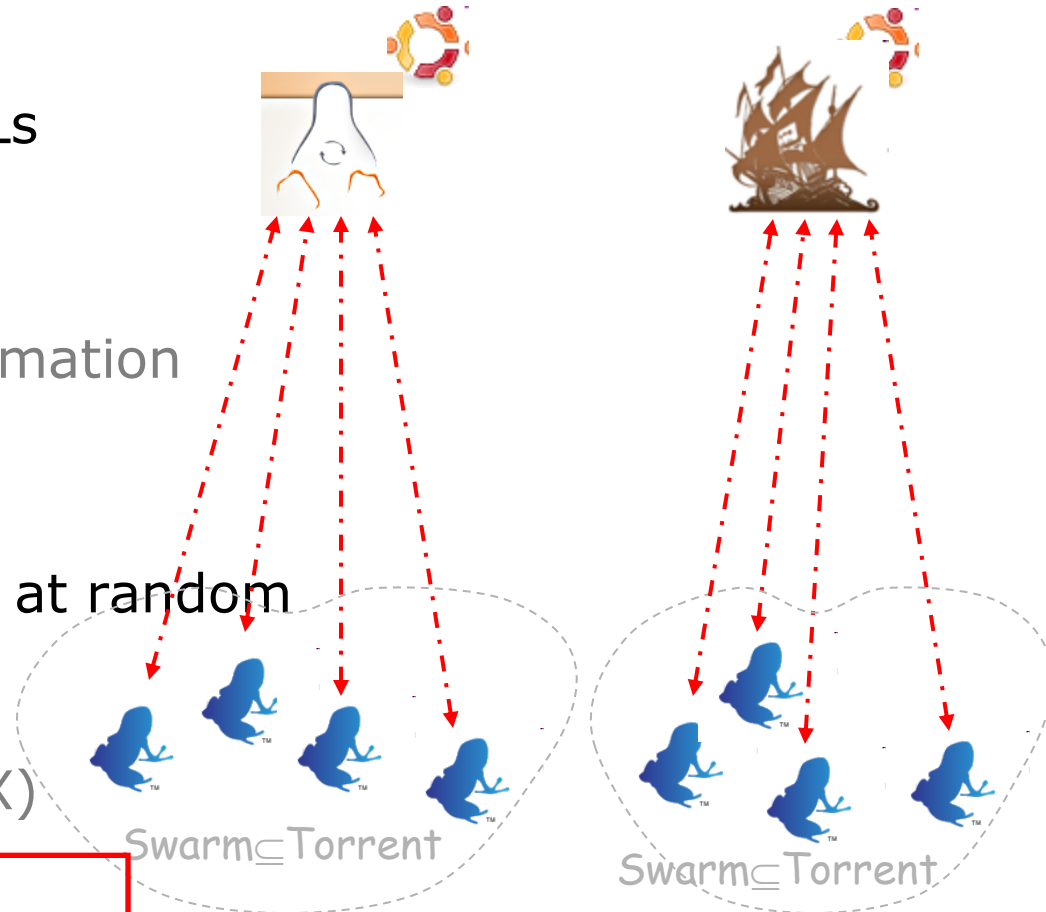
- ❑ Torrent file 
 - "announce-list" URLs
- ❑ Trackers 
 - Register torrent file
 - Maintain state information
- ❑ Peers
 - Obtain torrent file
 - Choose **one** tracker at random
 - Announce
 - Report status
 - Peer exchange (PEX)
- ❑ Issue
 - Multiple smaller swarms



Background

Multi-tracked torrents

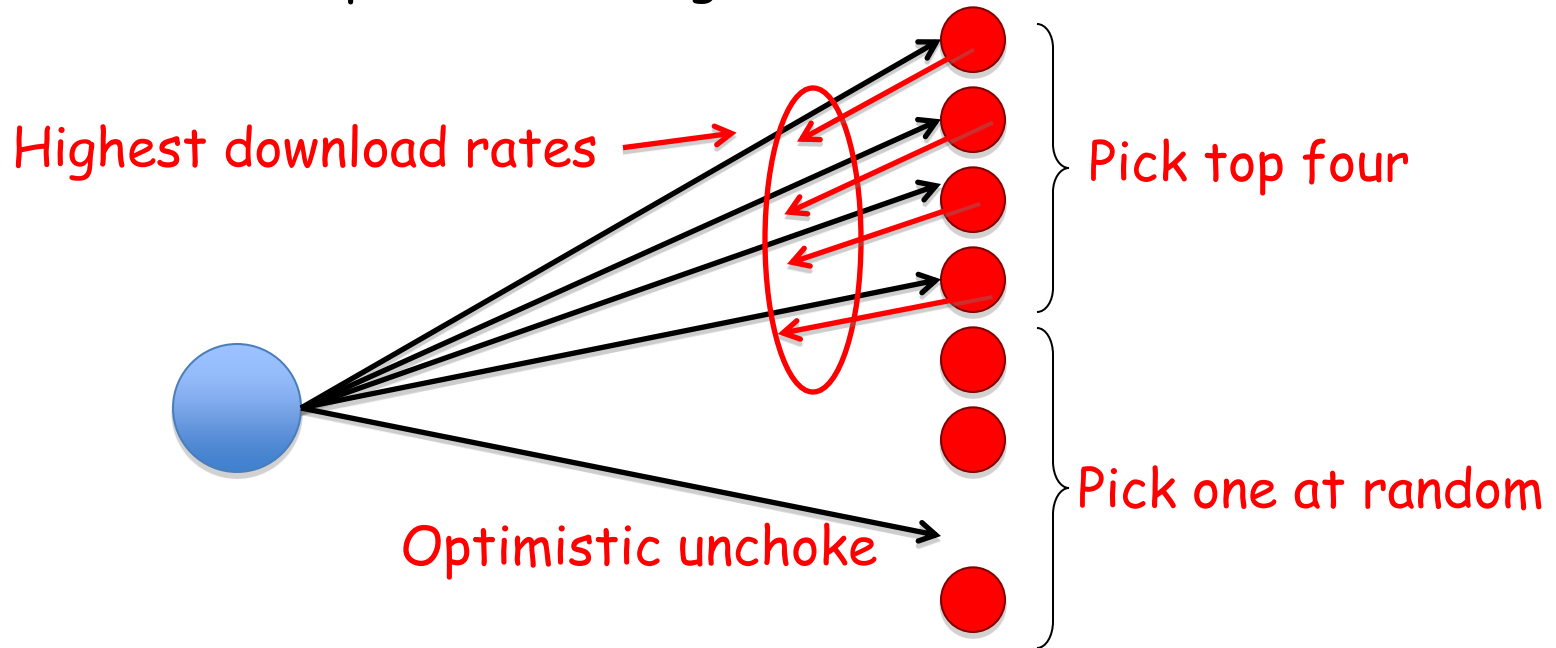
- ❑ Torrent file
 - "announce-list" URLs
- ❑ Trackers
 - Register torrent file
 - Maintain state information
- ❑ Peers
 - Obtain torrent file
 - Choose **one** tracker at random
 - Announce
 - Report status
 - Peer exchange (PEX)
- ❑ Issue
 - Multiple smaller swarms



Download using BitTorrent

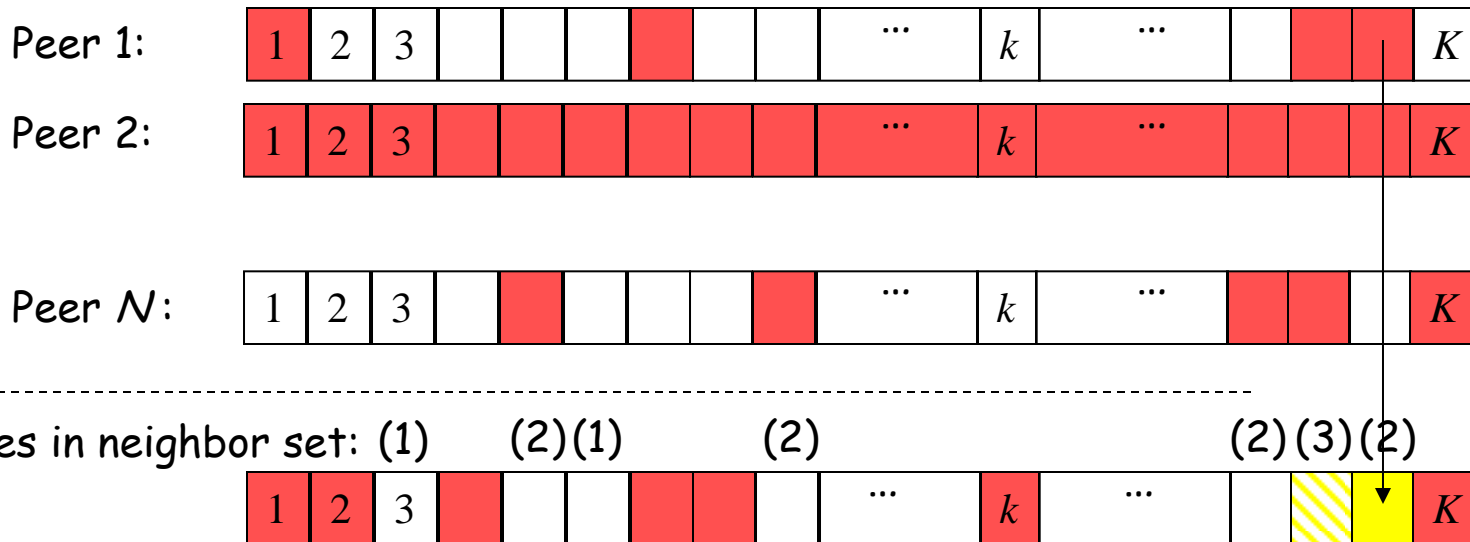
Background: Incentive mechanism

- ❑ 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

