Computer Networks
Chapter 2.5

Jong-won Lee

Handong University

Chapter 2: Application layer

Properties

- No central control, no central database
- No hierarchy
  - Every node is both a client and a server
  - The communication between peers is symmetric
- No global view of the system
  - Scalability
- Availability for any peer
- Peers are autonomous
- System globally unreliable
  - Robustness and security issues

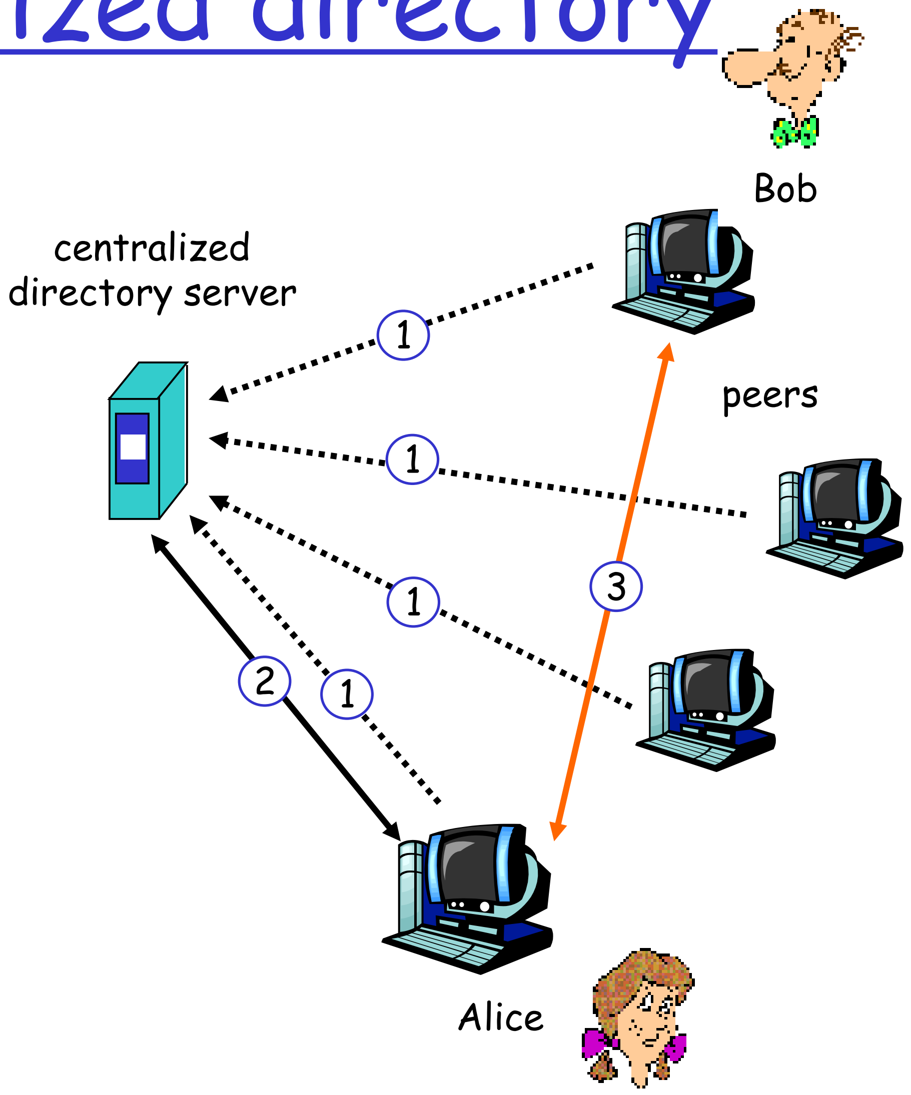
Napster: centralized directory

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



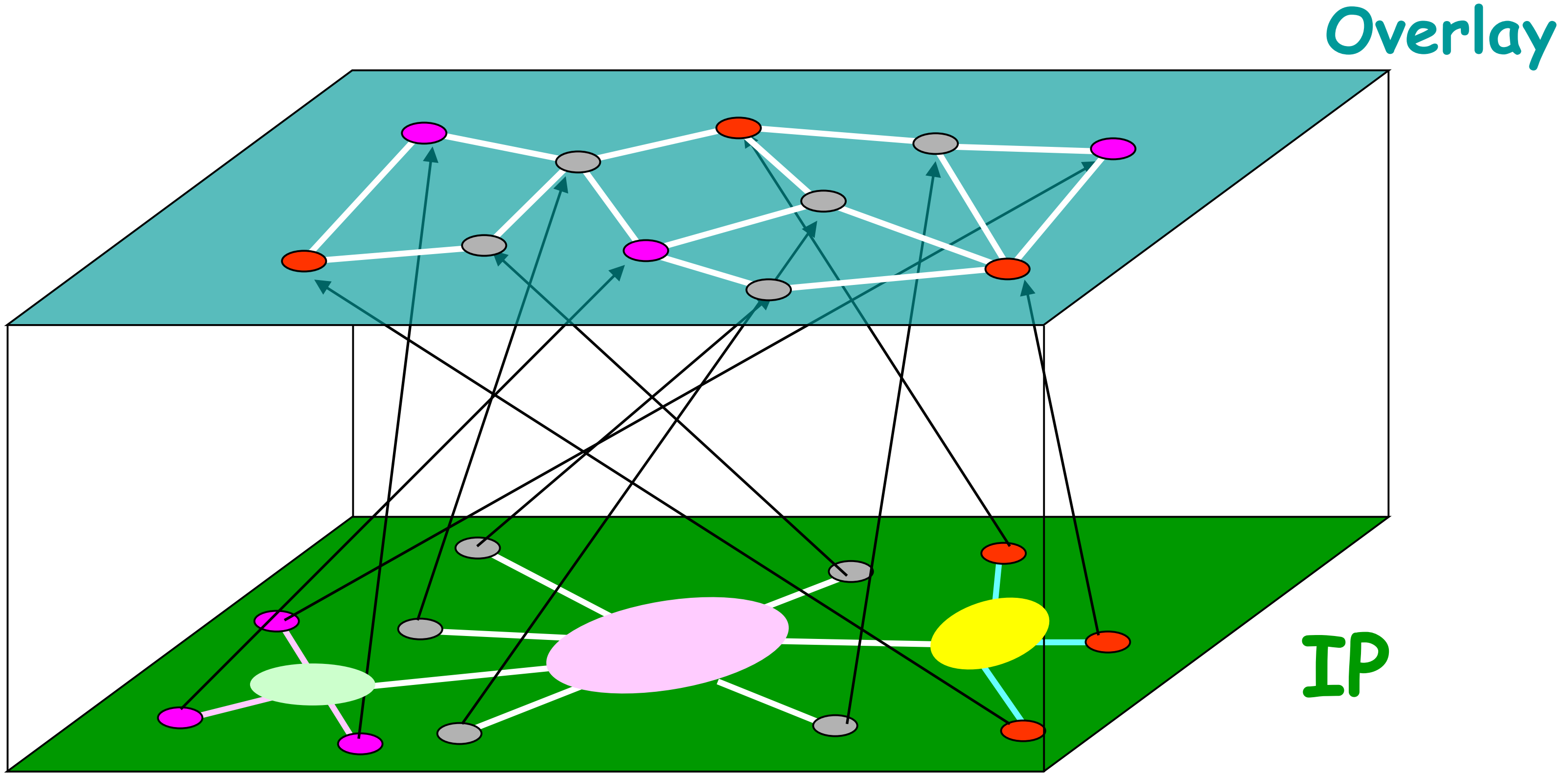
Napster: centralized directory

- File-sharing system
- Almost distributed system
  - The location of a document is centralized
  - The "transfer" is peer-to-peer
  - Fast querying
- Problems
  - Robustness
    - Single point of failure
  - Scalability (?)
    - Performance bottleneck
  - Copyright infringement
    - Sentenced to go out of business

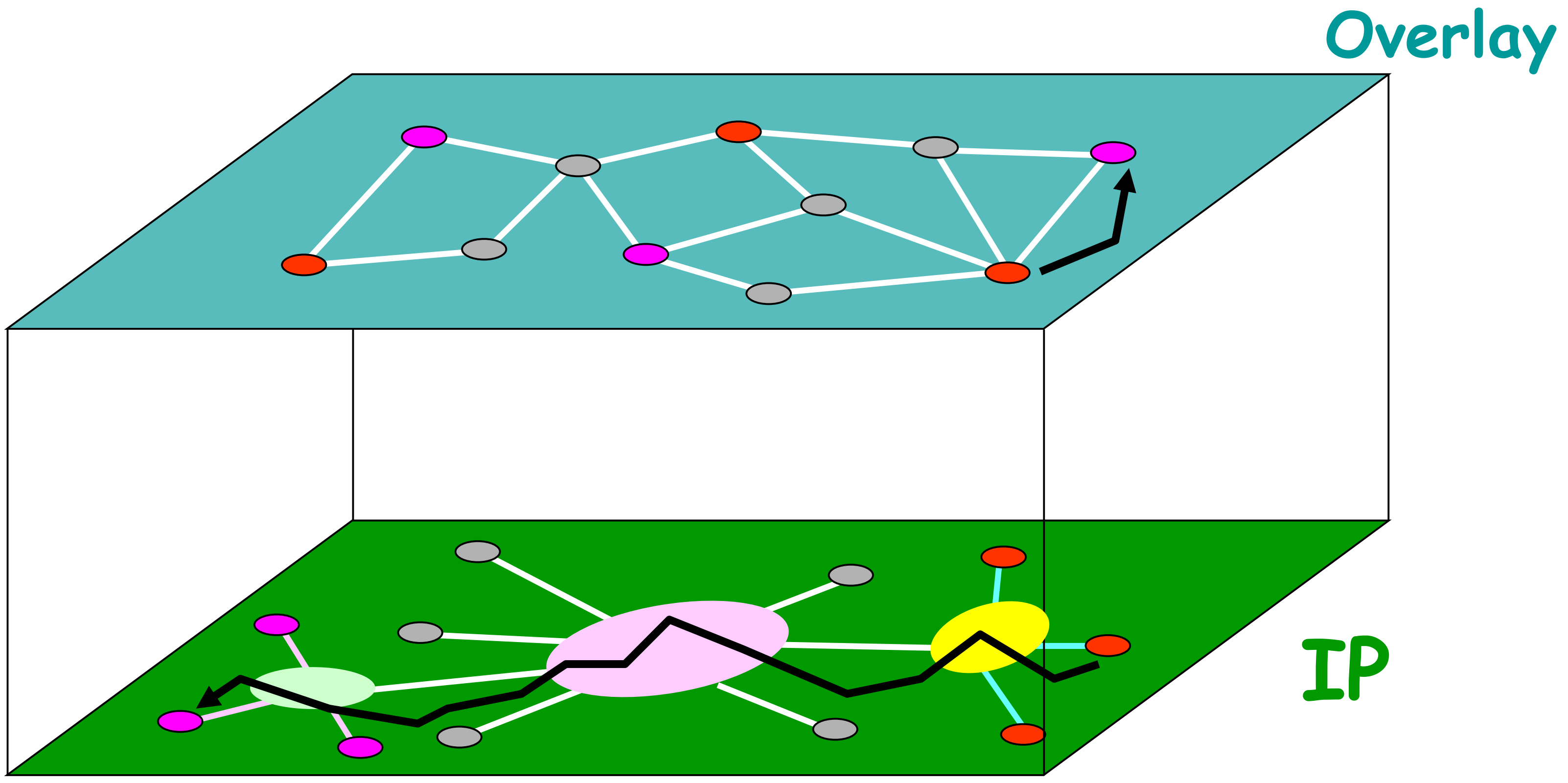
Gnutella: Query flooding

- fully distributed P2P protocol
    - no central server
- public domain protocol
    - many Gnutella clients implementing protocol
- Gnutella protocol
    - Based on broadcast/back-propagation mechanism over an overlay network
    - Message types
        - Ping/Pong: for group membership
        - Query/Query Hit: for search

Overlay Networks



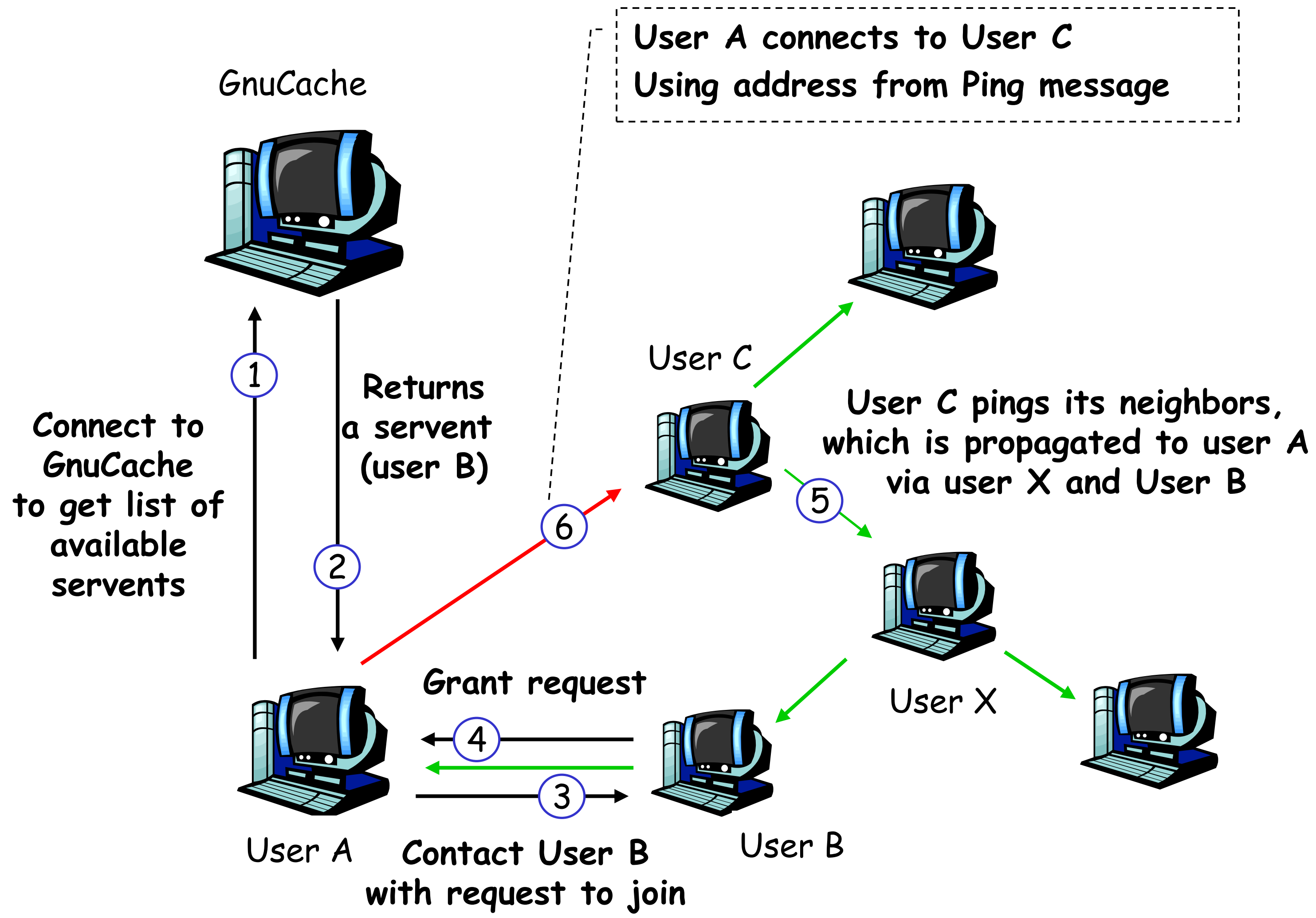
Overlay Networks



Gnutella: Join operation

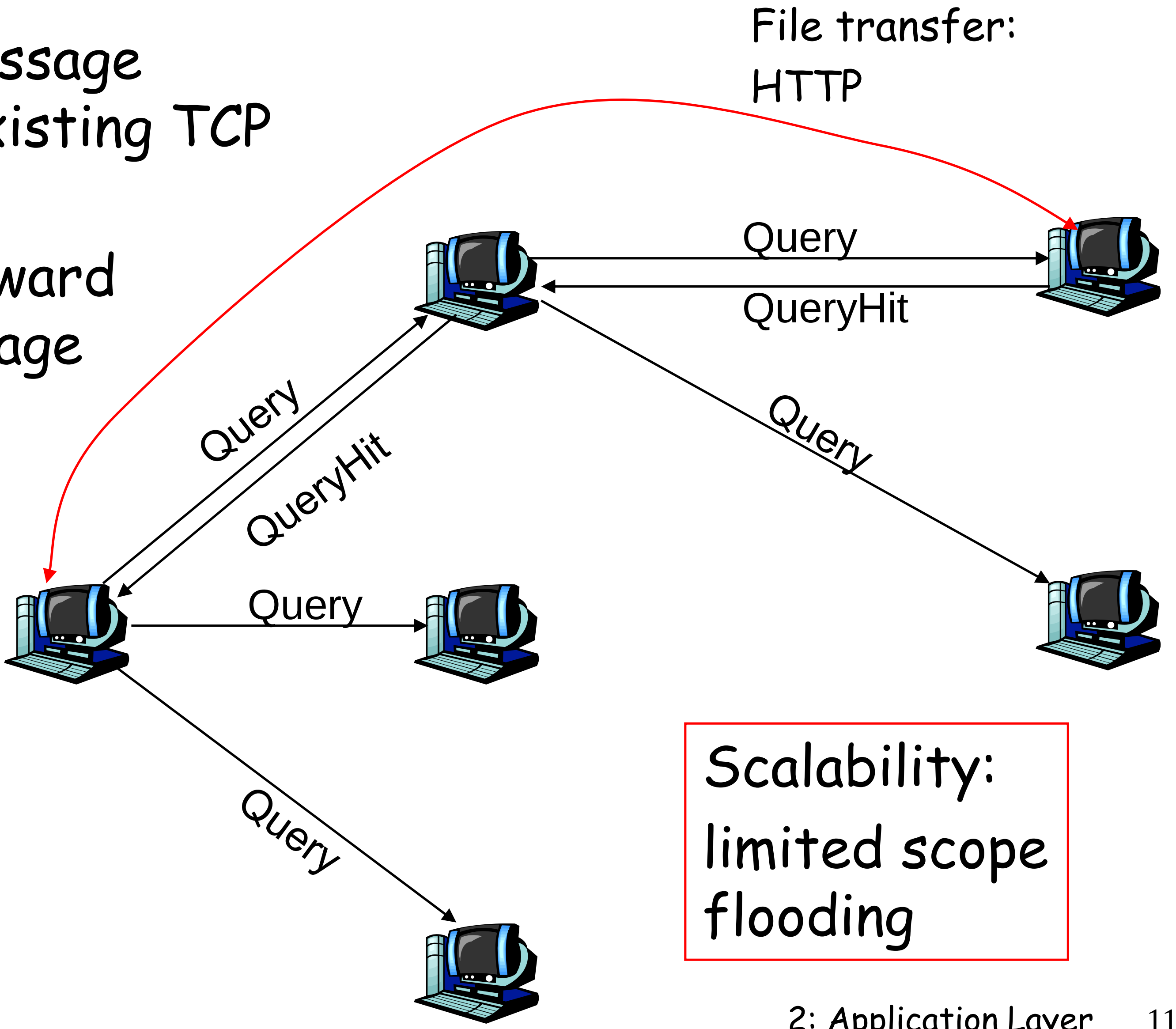
1. Joining User A must find some other peer in Gnutella network: use list of candidate peers. In this case, the user connects to a *GnuCache* server.
2. *GnuCache* returns a list of nodes on the Gnutella network. User A chooses one of these (User B) and attempts to contact it
3. User A sends a "Gnutella Connect" (see section 4) to User B to request to join the Gnutella network
4. User B accepts and returns a "Gnutella OK" to User A. User A is now part of the Gnutella network and is connected to one other Gnutella node.
5. Peers in Gnutella *Ping* their neighbors with their information periodically. Such *Ping* messages are not only replied to with *Pong* messages but they are propagated along to all other interconnected *servents*.
6. In this way, User A finds out about User C because User C has propagated its Ping message over to User A. The Ping messages contain the address of the sender. Typically Gnutella *servents* connect to around 3 other *servents* in the network.

Gnutella: Join operation


User A connects to User C
Using address from Ping message
GnuCache
1
Connect to GnuCache to get list of available servents
Returns a servent (user B)
2
User C
User C pings its neighbors, which is propagated to user A via user X and User B
5
6
User X
Grant request
4
3
User A
Contact User B with request to join
User B

Gnutella: Search

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



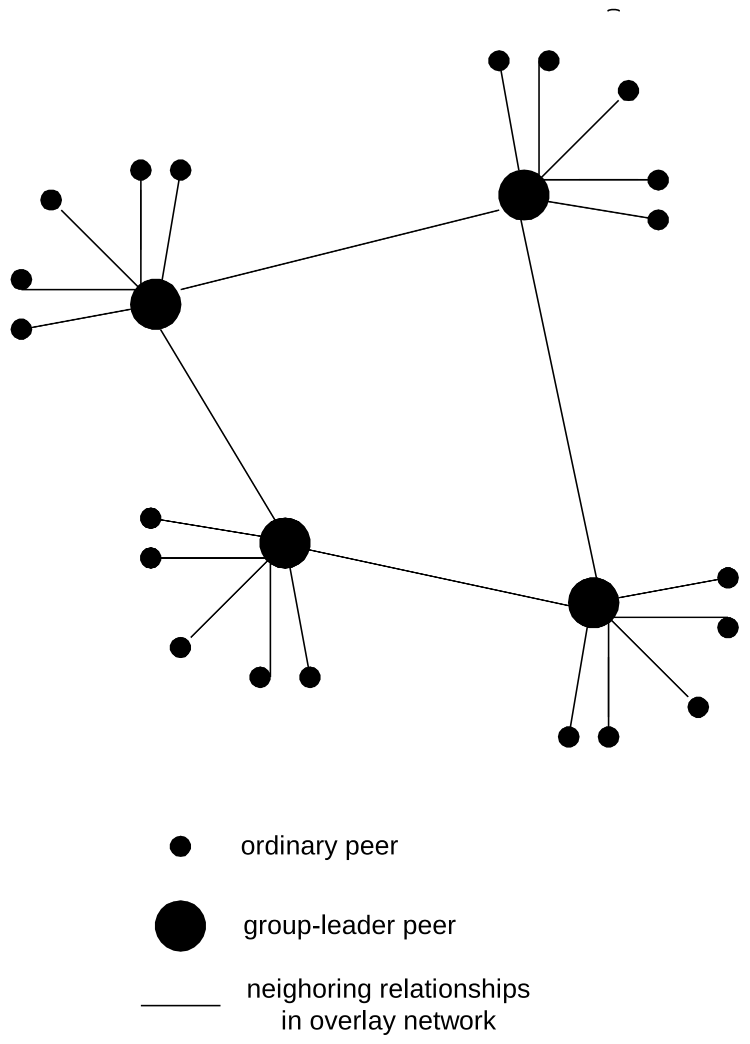
Scalability: limited scope flooding

Gnutella

- Gnutella is simple, however
-
- Excessive query traffic
  - Can be controlled through (limited-scope query flooding)
-
  - Limited flooding reduces the number of peers
- Maintenance of overlay network
  - TCP connection between peer neighbors should be maintained even when there is no traffic
- Free riding
  - Most Gnutella users do not provide files to share
  - 47% of all responses are returned by top 1% of hosts

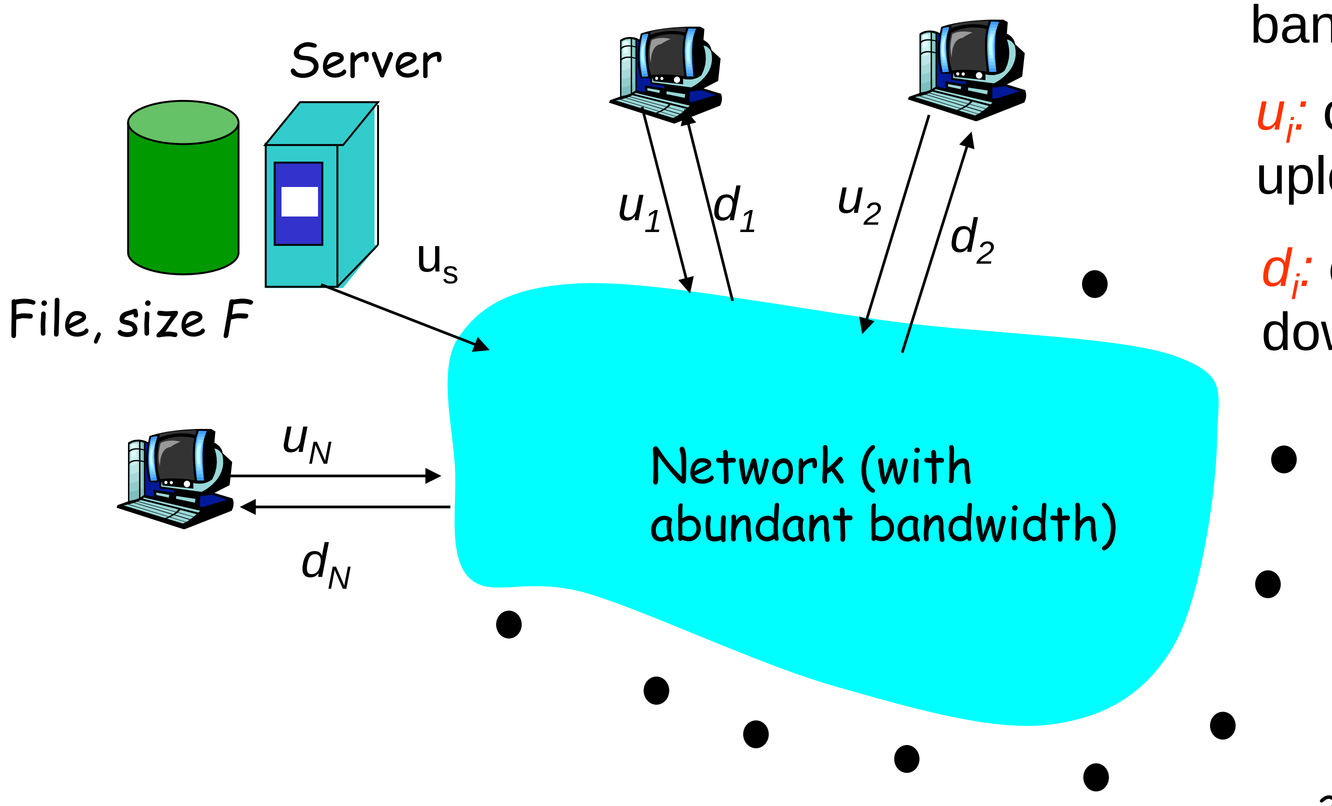
Hierarchical Overlay

- between centralized index, query flooding approaches
- each peer is either a *group leader (super-peer)* 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



Comparing Client-server, P2P architectures

Question : How much time to distribute file initially at one server to N other computers?



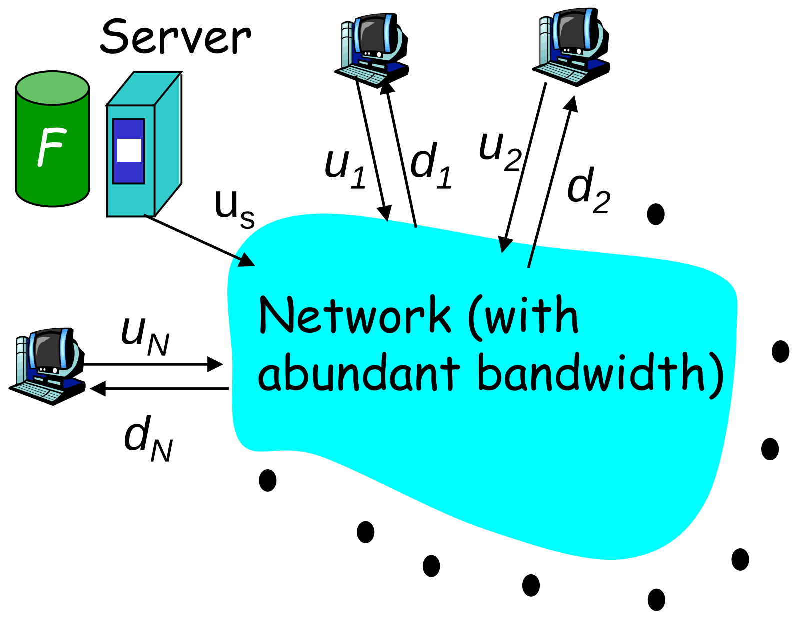
u_s: server upload bandwidth

u_i: client/peer i upload bandwidth

d_i: client/peer i download bandwidth

Client-server: file distribution time

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

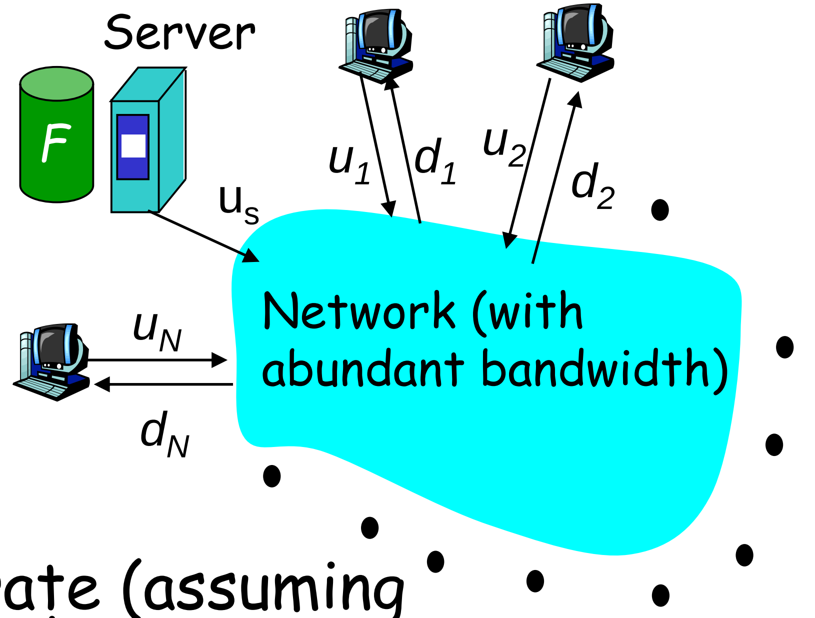


Time to distribute F to N clients using client/server approach $= d_{cs} = \max_i \{ NF/u_s, F/\min(d_i) \}$

increases linearly in N (for large N)

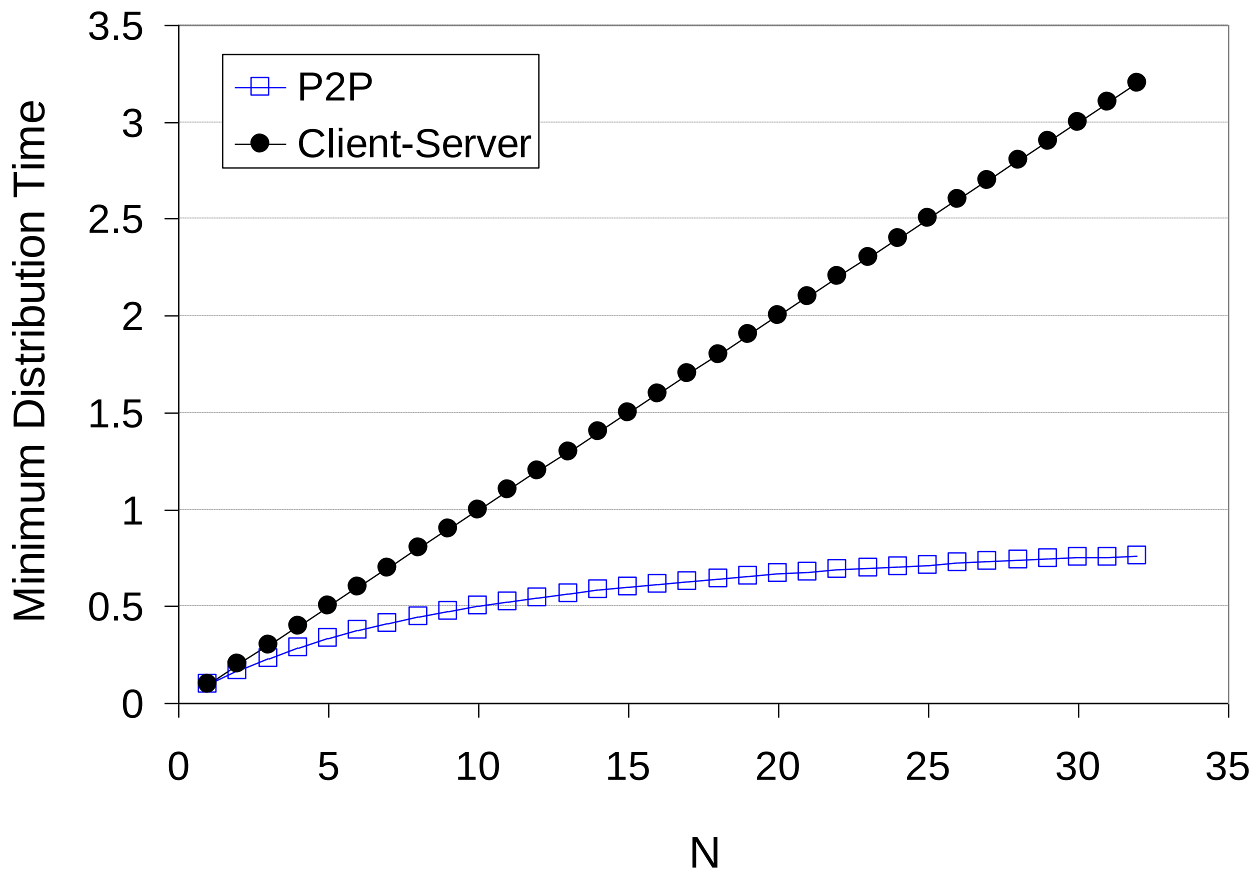
P2P: file distribution time

- 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 (assuming all nodes sending file chunks to same peer): $u_s + \sum_{i=1,N} u_i$



$$d_{P2P} >= \max \{ F/u_s, F/\min_i(d_i) , NF/(u_s + \sum_{i=1,N} u_i) \}$$

Comparing Client-server, P2P architectures

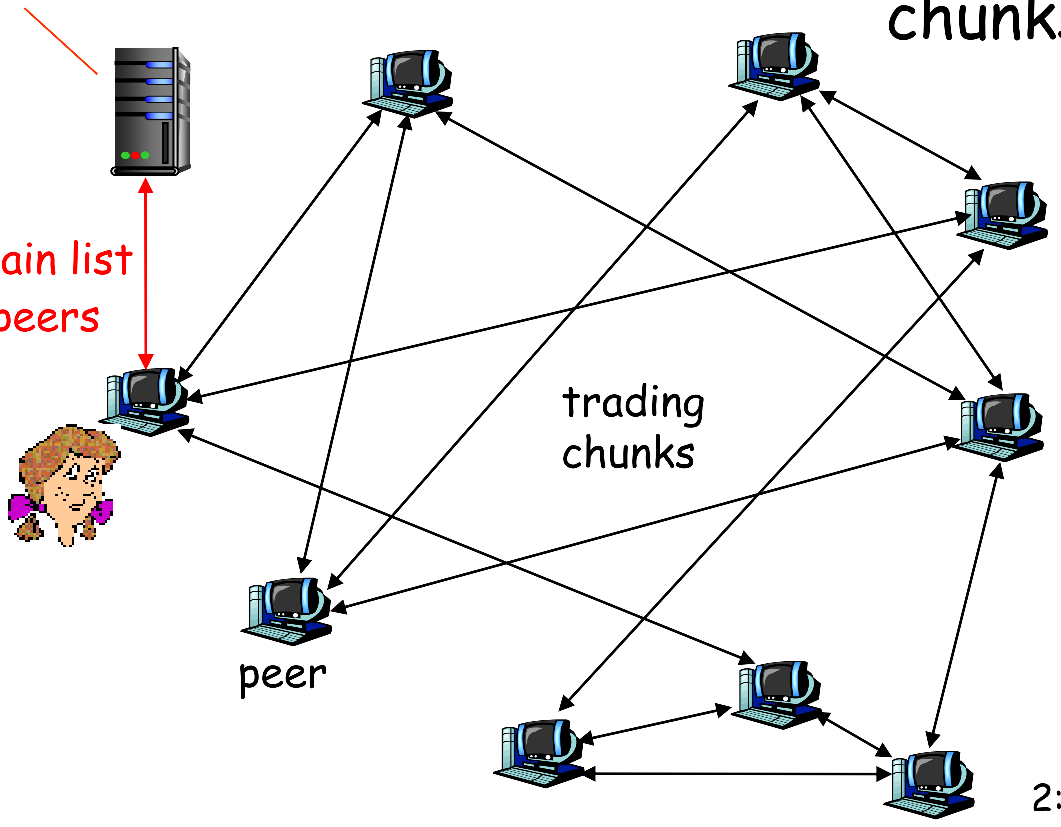


P2P Case Study: BitTorrent

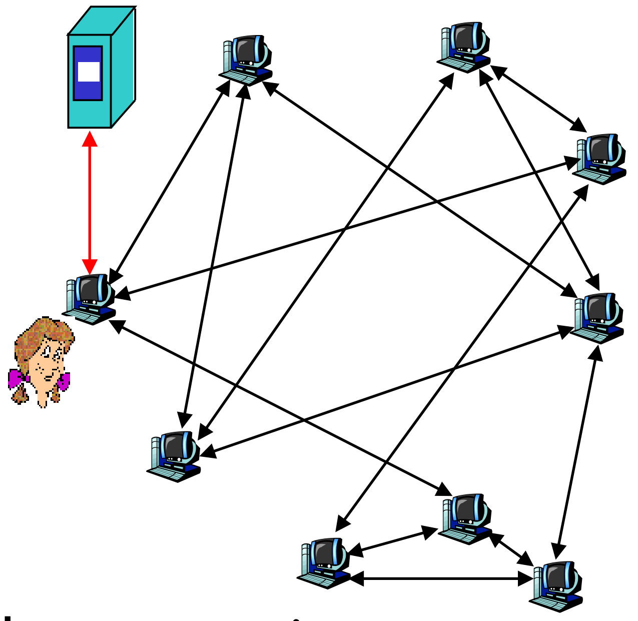
- P2P file distribution

tracker: tracks peers participating in torrent

torrent: group of peers exchanging chunks of a file



BitTorrent (1)



- file divided into 256KB *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 (2)

Requesting chunks

- at any given time, different peers have different subsets of file chunks
- periodically, Alice asks each neighbor for list of chunks that they have.
- Alice issues requests for her missing chunks
  -

Sending Chunks:

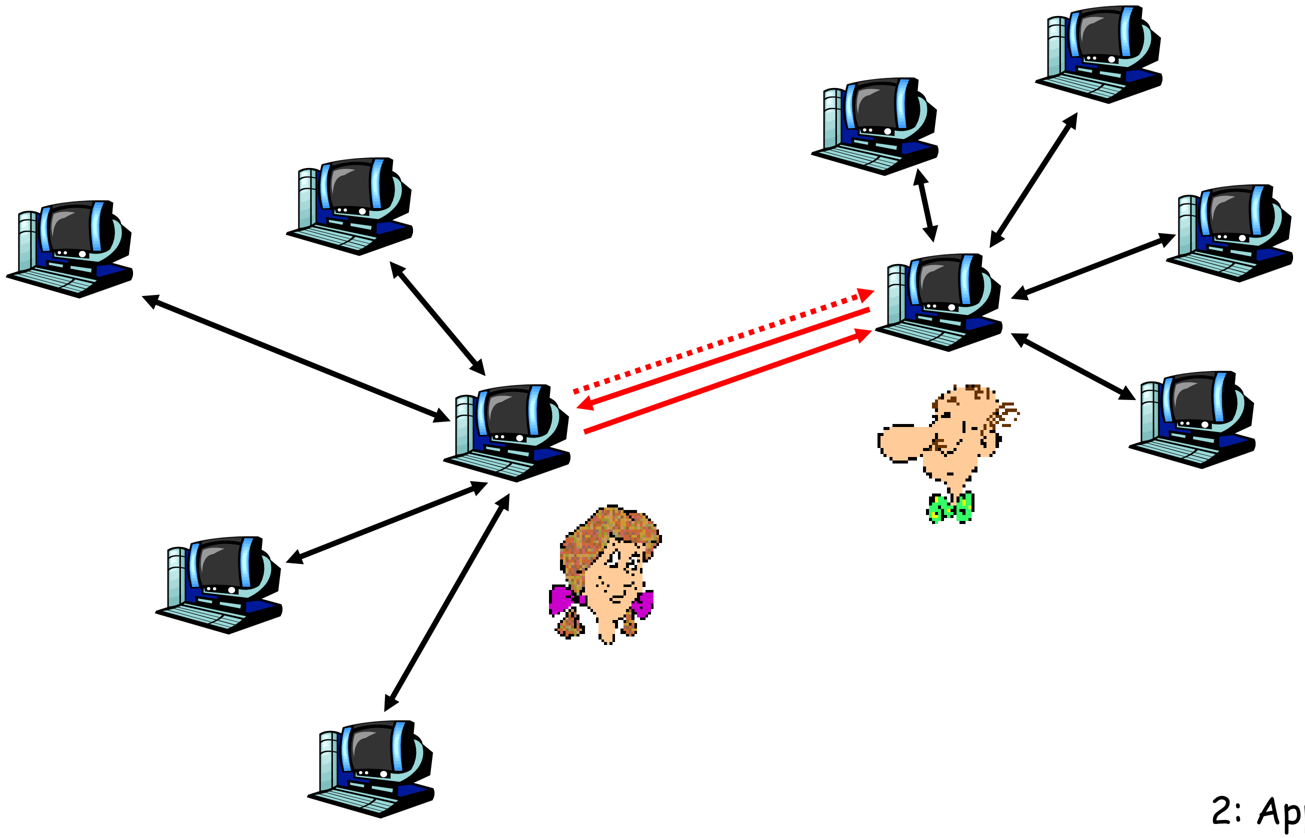
- Alice sends chunks to four neighbors currently sending her chunks *at the highest rate*
  - re-evaluate every 10 secs
- every 30 secs: , starts sending chunks
  - newly chosen peer may join top 4

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



Distributed Hash Table (DHT)

- An ordinary hashtable, which is …

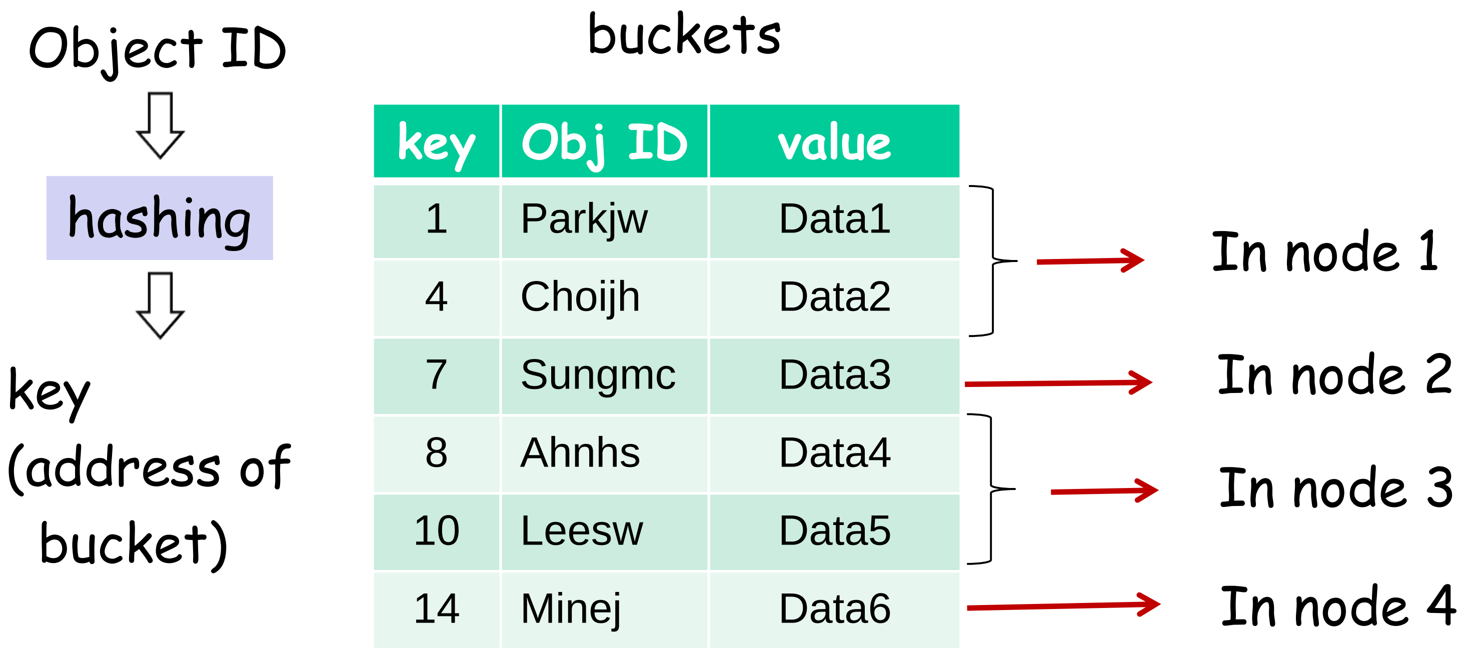
$$\begin{tikzcd}\text{Object ID} \arrow[d] \\ \boxed{\text{Hashing function}} \arrow[d] \\ \text{key (address of bucket)}\end{tikzcd}$$

buckets

key	Obj ID	value
1	Parkjw	Data1
4	Choijh	Data2
7	Sungmc	Data3
8	Ahnhs	Data4
10	Leesw	Data5
14	Minej	Data6

Distributed Hash Table (DHT)

- An ordinary hashtable, which is distributed



key	Obj ID	value
1	Parkjw	Data1
4	Choijh	Data2
7	Sungmc	Data3
8	Ahnhs	Data4
10	Leesw	Data5
14	Minej	Data6

Distributed Hash Table (DHT)

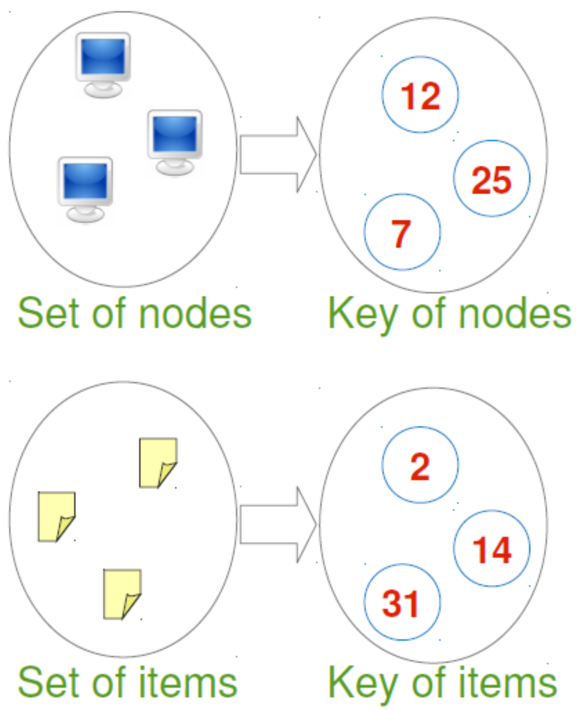
- DHT = a distributed P2P database
  - Distributes data among a set of nodes according to predefined rules
- Database has (object ID, value) pairs;
  - Object ID: ss number; value: human name
  - Object ID: movie title; value: IP address
- Peers query DB with object ID or key
  - DB returns values that match the key
  - In DHT-based networks each peer has a partial knowledge about the whole network. This knowledge can be used to route the queries to the responsible nodes using effective and scalable procedures.

Q: how to assign keys to peers?

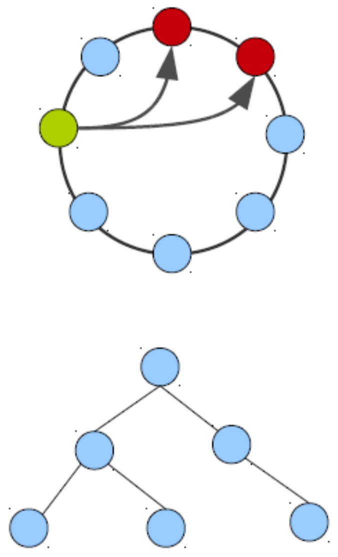
- central issue:
  - assigning (key, value) pairs to peers.
- basic idea:
  - convert each key to an integer
  - Assign integer to each peer
  - put (key,value) pair in the peer that is closest to the key

DHT

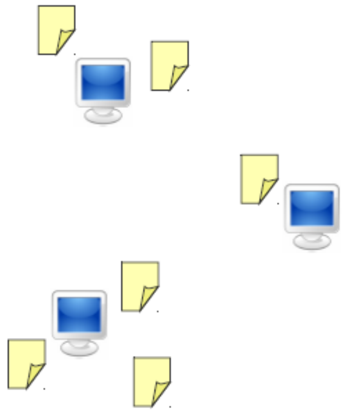
- Peer IDs and object IDs are in the same range
  - Assign an integer to each peer or each object in range [0,2^n-1].
    - key = hash(object name)



1. Decides on common key space for nodes and items



2. Connects the nodes smartly



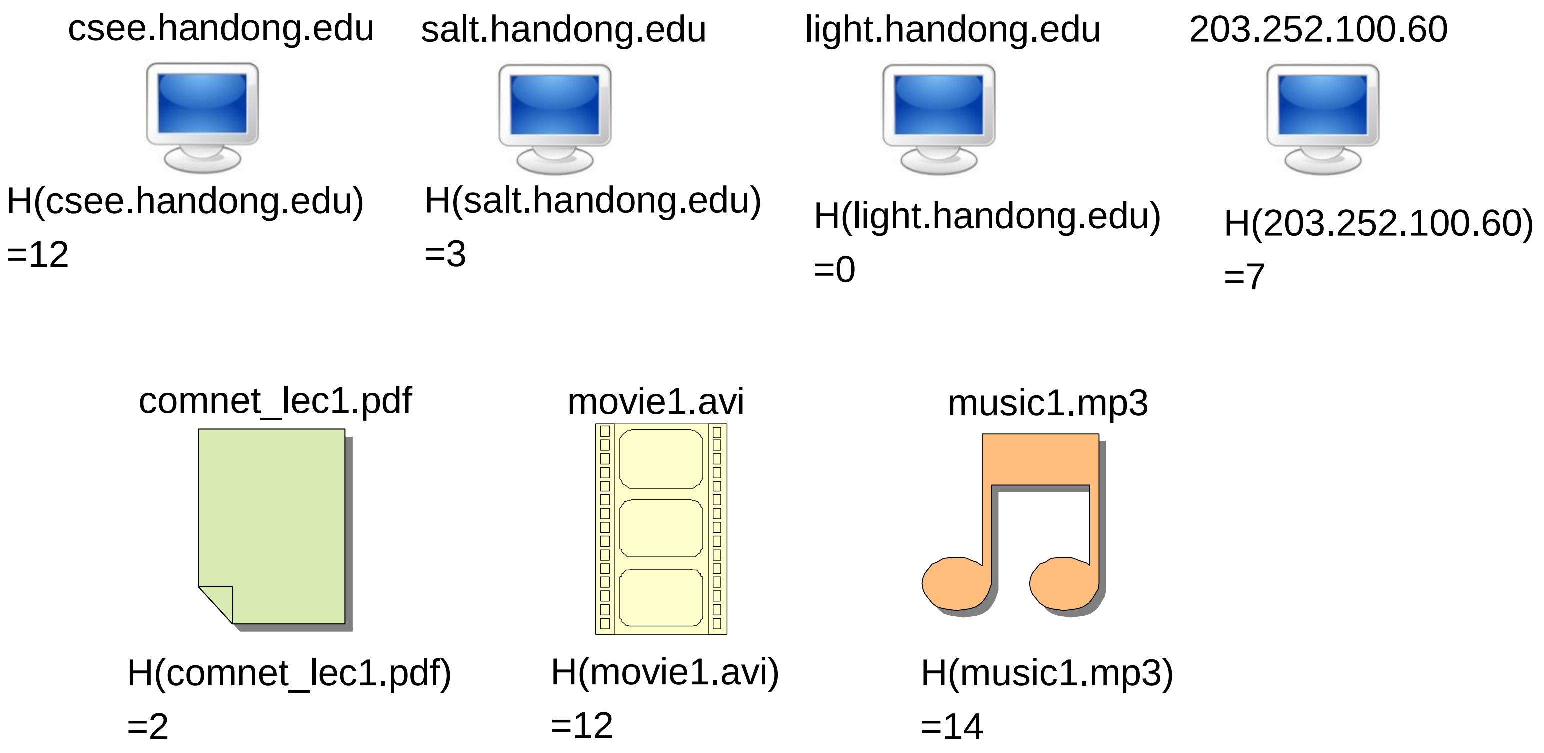
3. Make a strategy for assigning items to nodes

DHT

- Consistent hashing
  - A scheme that provides hashing table functionality in a way that the addition or removal of one node does not significantly change the mapping of keys to nodes

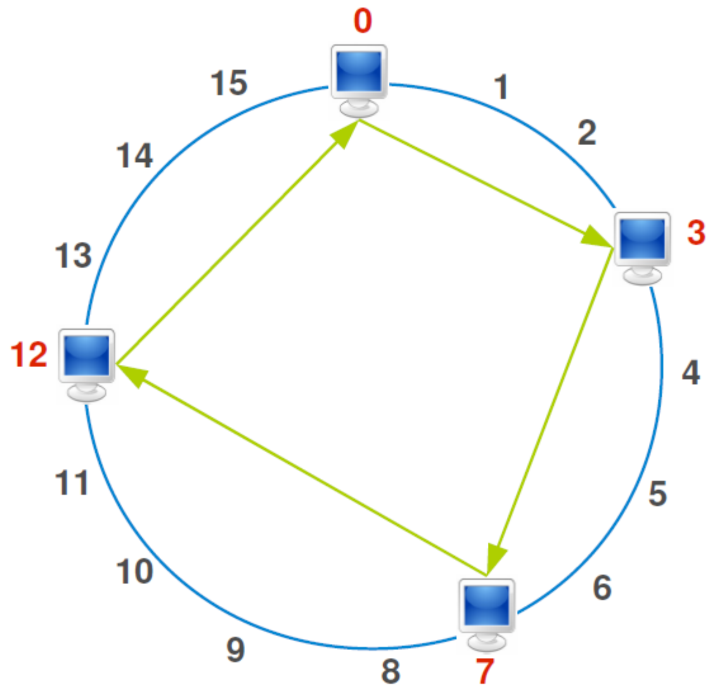
Consistent hashing using a ring

- The keys of Peer(node) IDs and object IDs are in the same range
  - Ex.: the size of key space = 16 ([0, 15])



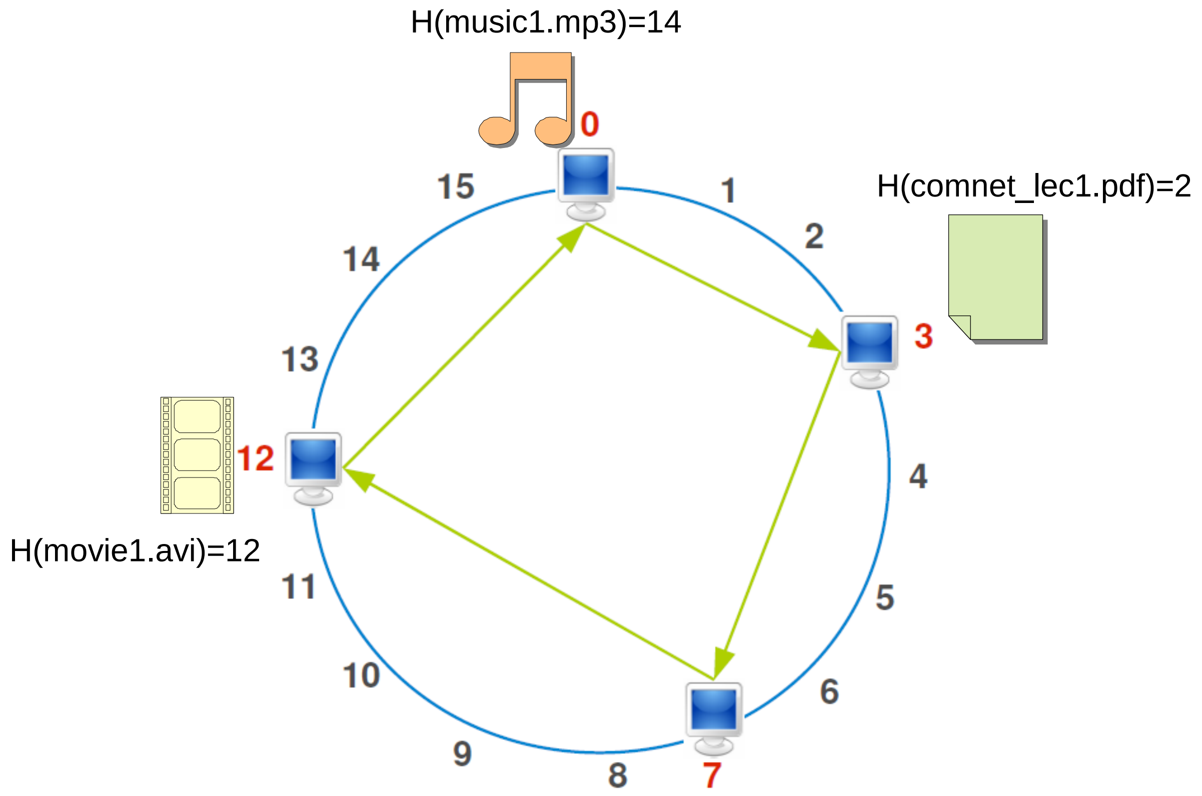
Consistent hashing using a ring

- Node connections
  - The successor node of node i is the first node with ID greater than i.
  - The ID range is considered as a circular space.



Consistent hashing using a ring

- The policy that each item is assigned to a node
  - An item with ID x is assigned at the node with ID succ(x).
  - Def. succ(x) is the first node on the ring with ID greater than or equal to x in the fashion of circular space.



Consistent hashing using a ring

- If each node knows its successor node, the two operations, get() and put(), would be simply done by searching them sequentially.
  - Put(hash(item),value), get(item)

- The average number of messages to resolve a query
  - O(N) : N = the number of nodes
  - Possible to reduce to O(logN)
    - How? (refer to Chord scheme)

- Consider how to handle peer