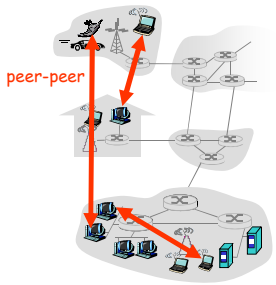
Pure P2P architecture

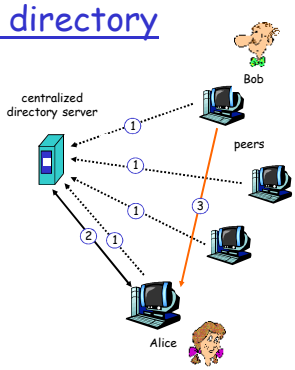
- *no* always-on server
- arbitrary end systems directly communicate
- peers are intermittently connected and change IP addresses
- Three topics:
  - File sharing
  - File distribution
  - Searching for information
  - Case Studies: Bittorrent and Skype



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

- single point of failure
- performance bottleneck
- copyright infringement: "target" of lawsuit is obvious

file transfer is decentralized, but locating content is highly centralized

Query flooding: Gnutella

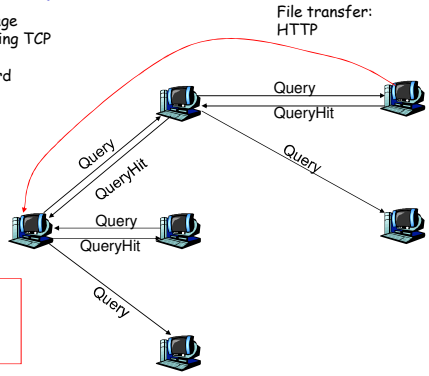
- fully distributed
  - no central server
- public domain protocol
- many Gnutella clients implementing protocol

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

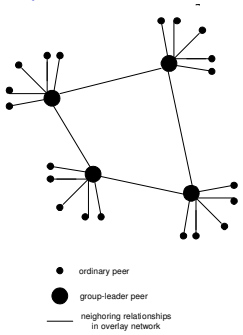
- Query message sent over existing TCP connections
- peers forward Query message
- QueryHit sent over reverse path



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.
  - TCP connection between peer and its group leader.
  - TCP connections between some pairs of group leaders.
- group leader tracks content in its children



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

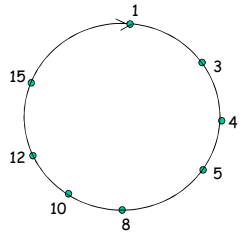
DHT Identifiers

- 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("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 immediate 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

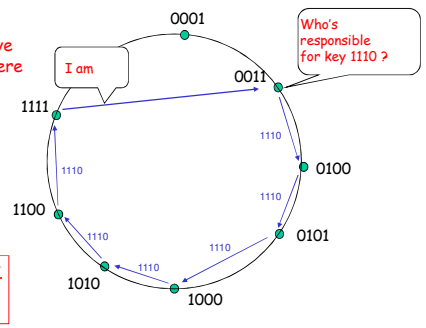
Circular DHT (1)



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

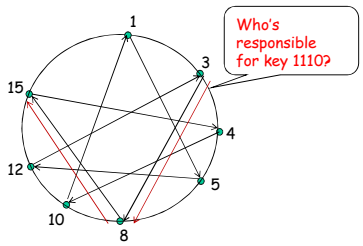
Circle DHT (2)

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



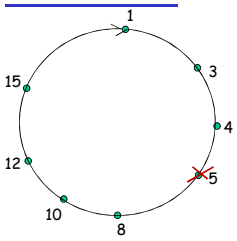
Define closest as closest successor

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

Peer Churn

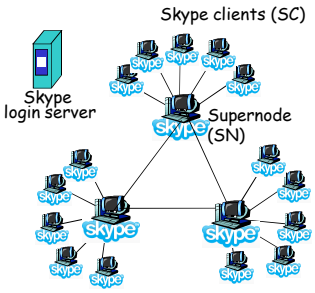


•To handle peer churn, require each peer to know the IP address of its two successors.
• Each peer periodically pings its two successors to see if they are still alive.

- 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?

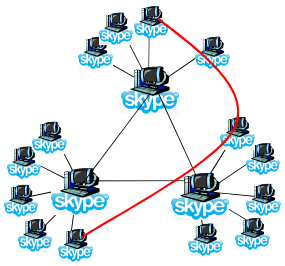
P2P Case study: Skype

- 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



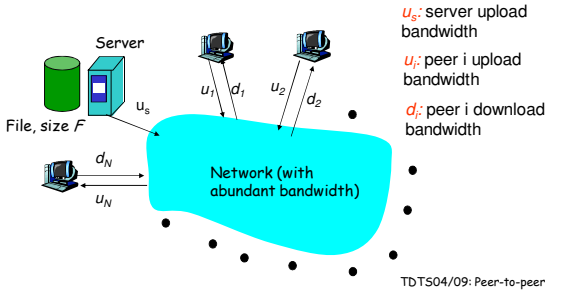
Peers as relays

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



File Distribution: Server-Client vs P2P

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



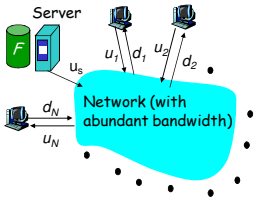
u_s: server upload bandwidth

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

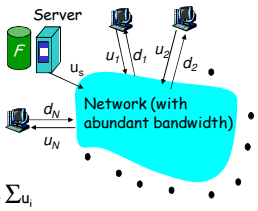


Time to distribute F to N clients using client/server approach $= d_{cs} = \max\{ NF/u_s, F/\min(d_i) \}$

increases linearly in N (for large N)

File distribution time: P2P

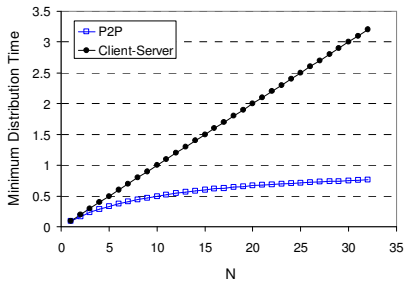
- server must send one copy: F/u_s time
- client i takes F/d_i time to download
- NF bits must be downloaded (aggregate)
  - fastest possible upload rate: $u_s + \Sigma u_i$



$d_{P2P} = \max\{ F/u_s, F/\min(d_i), NF/(u_s + \Sigma u_i) \}$

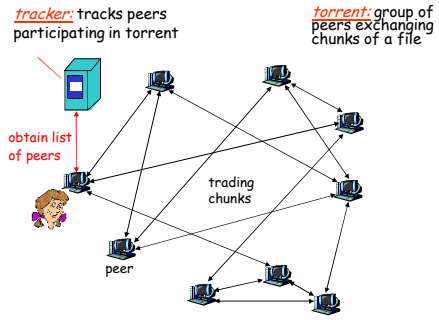
Server-client vs. P2P: example

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



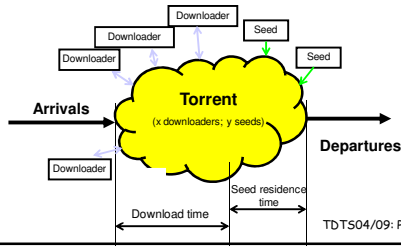
File distribution: BitTorrent

- P2P file distribution



BitTorrent-like systems

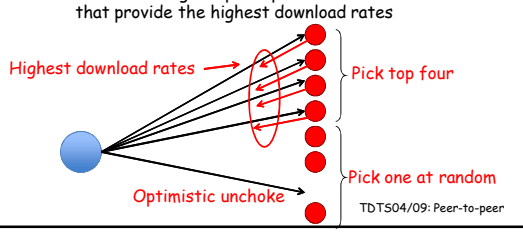
- File split into many smaller pieces
- Pieces are downloaded from both seeds and downloaders
- Distribution paths are dynamically determined
  - Based on data availability



Download using BitTorrent

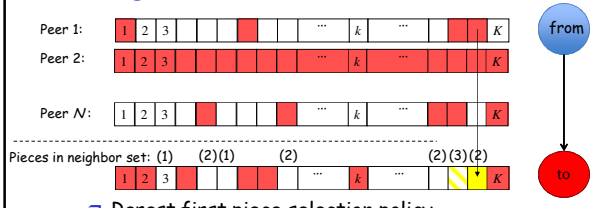
Background: Incentive mechanism

- 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

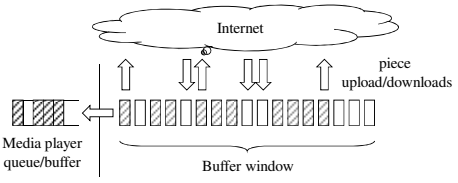
Background: Piece selection



- 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

Live Streaming

using BT-like systems



- Live streaming (e.g., CoolStreaming)
  - All peers at roughly the same play/download position
    - High bandwidth peers can easily contribute more …
  - (relatively) Small buffer window
    - Within which pieces are exchanged

Peer-assisted VoD streaming
Some research questions …

- Can BitTorrent-like protocols provide scalable on-demand streaming?
- How sensitive is the performance to the application configuration parameters?
  - Piece selection policy
  - Peer selection policy
  - Upload/download bandwidth
- What is the user-perceived performance?
  - Start-up delay
  - Probability of disrupted playback

ACM SIGMETRICS 2008; IFIP Networking 2007; IFIP Networking 2009