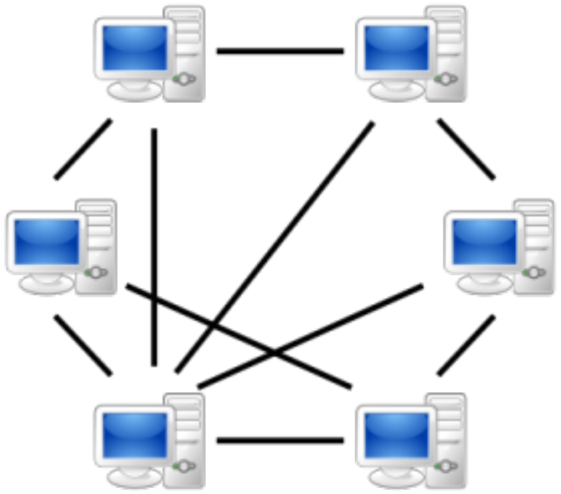
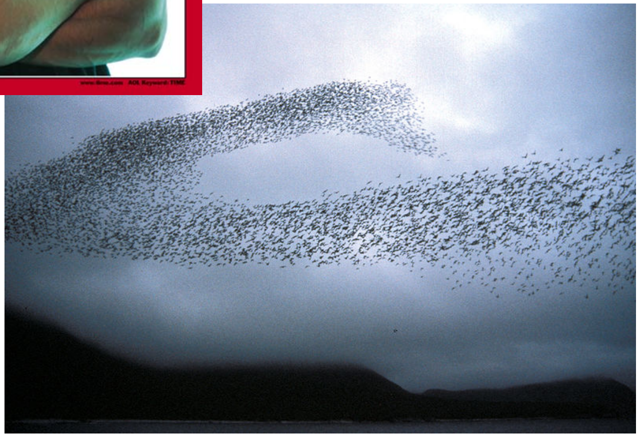
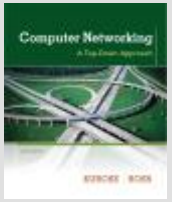
Peer-to-Peer Applications



TIME
What's Next For Napster
How SHAWN FANNING, 19, upended music... and a lot more
Inside a Teen's Stock Scam
The Battle Over Oil Prices

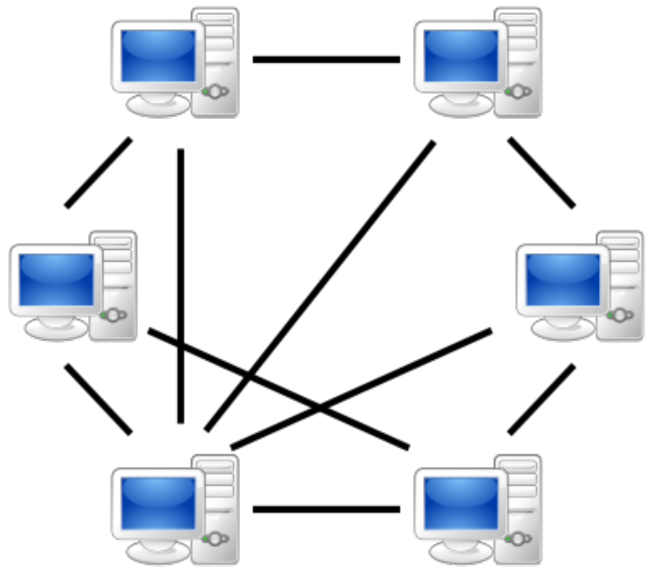


Computer Networking: A Top Down Approach
6th edition
Jim Kurose, Keith Ross
Addison-Wesley



Overview

- Peer-to-peer applications
  - Motivation, types
  - Overlay networks
  - Napster, the rise and fall
  - Performance analysis
    - P2P vs. client-server
  - BitTorrent
  - Distributed hash tables

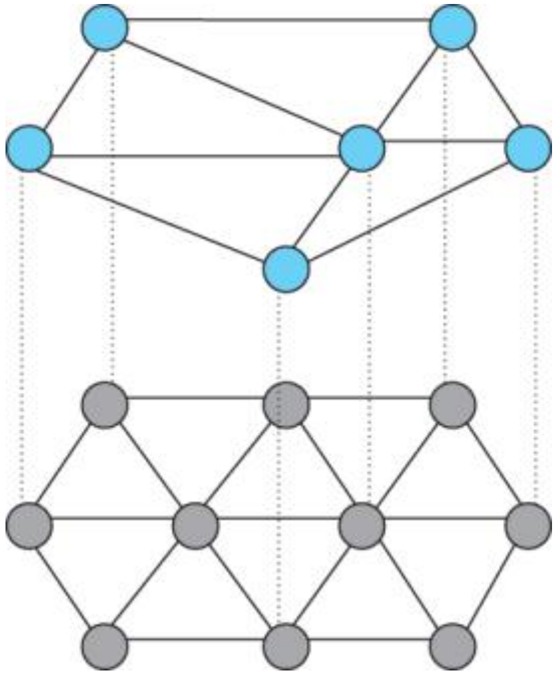


Overlay services: P2P

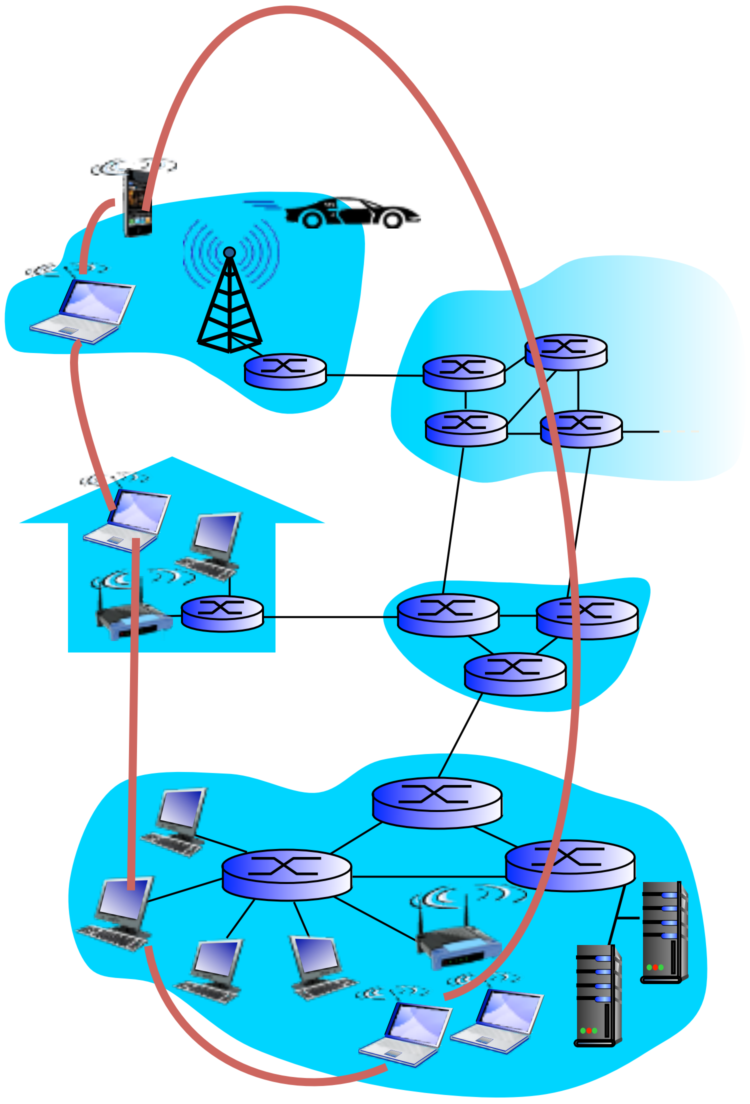
- Peer-to-peer (P2P) networks
  - Community of users pooling resources (storage space, bandwidth, CPU) to provide a service
    - e.g. Sharing MP3 files, Skype
  - Nodes are hosts willing to share
  - Links are tunnels used to transport objects of interest
- Types:
  - Centralized P2P – central server for indexing
  - Pure P2P – all peers are equals
  - Hybrid P2P – some peers are supernodes

Overlay networks

- Overlay networks
  - Logical network running on top of physical network
  - Support alternate routing strategies
  - Experimental protocols


Overlay network

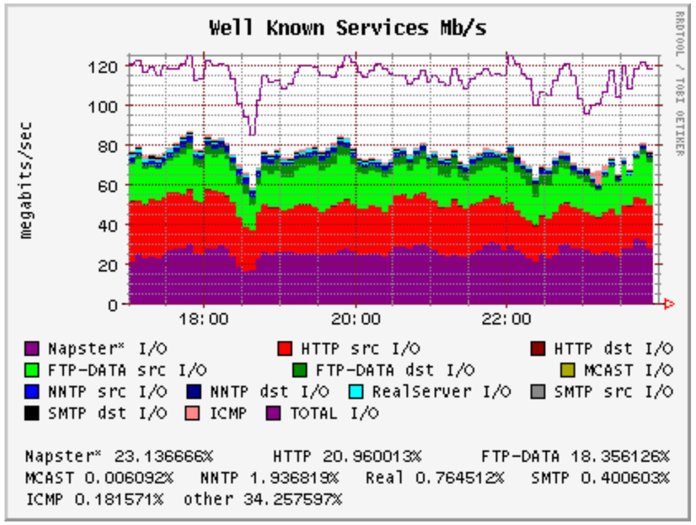
Physical Network, “underlay network”



P2P: Napster

- Napster: the rise
  – Created by Shawn Fanning
    - Christmas break, freshmen year at college
  – Allows search and sharing of MP3s
  – January 1999, Napster version 1.0
  – May 1999
    - Company founded
    - Shawn drops out of school
  – September 1999, 1st lawsuits
    - No such thing as bad publicity?
  – By 2000, 80 million users

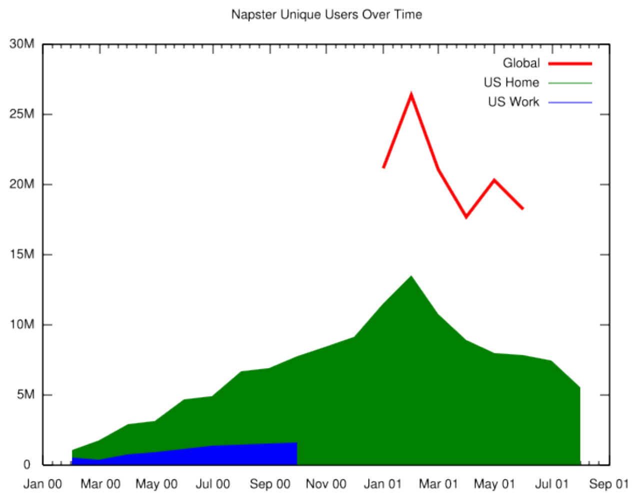
TIME
What's Next For Napster
How SHAWN FANNING, 19, upended music... and a lot more
Inside a Teen's Stock Scam
The Battle Over Oil Prices



UW-Madison, March 9th, 2000

P2P: Napster

- Napster: the fall
  - December 1999, RIAA lawsuit
  - Metallica's "I Disappear" circulates
    - Before official release, starts getting radio play
    - 2000 band files a lawsuit
  - July 2001, shutdown by lawsuits
  - 2002, relaunched as paid service
    - Record labels not keen to license
    - Files bankruptcy
  - Gave rise to many P2P alternatives
  - Forced industry out of stone age
    - iTunes



Napster users peak, Feb 2001.

Napster technology

- User installs software
  - Registers name, password, local dir with music
- Client contacts central Napster server
  - Connects via TCP
  - Provides list of music in user's directory
  - Napster updates its database
- Client searches for music
  - Napster identifies currently online client with file
  - Provides IP addresses so client can download directly

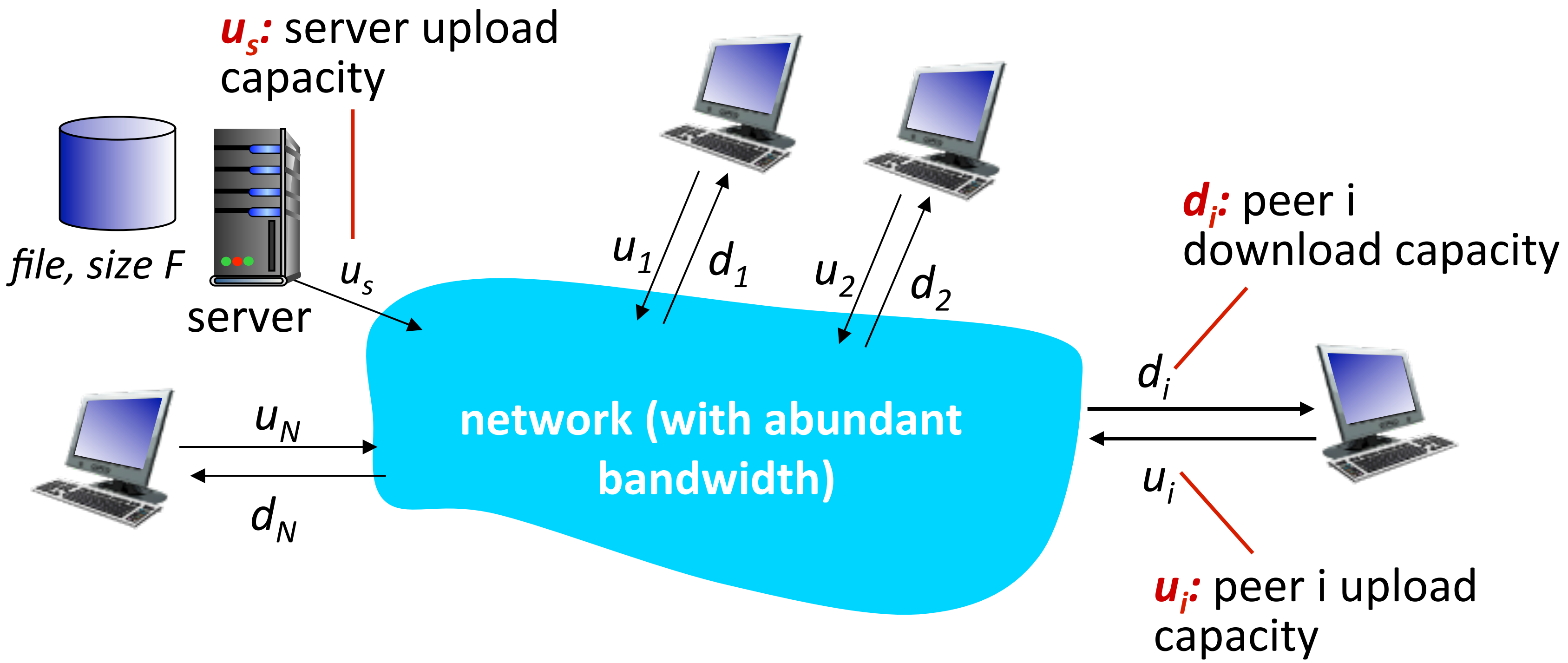
Napster technology

- Central server continually updated
  - Easy to track music currently available and from what peer
  - Good source to prove copyright infringement
  - Single point of failure, performance bottleneck
- Peer-to-peer transfer
  - Key idea of P2P: heavy lifting done between peers
  - No need for Napster to provision lots of capacity
    - Just enough to support indexing/search needs of clients
- Proprietary protocol

File distribution: client-server vs. P2P

Question: Time to distribute file (size *F*) from one server to *N* *peers*?

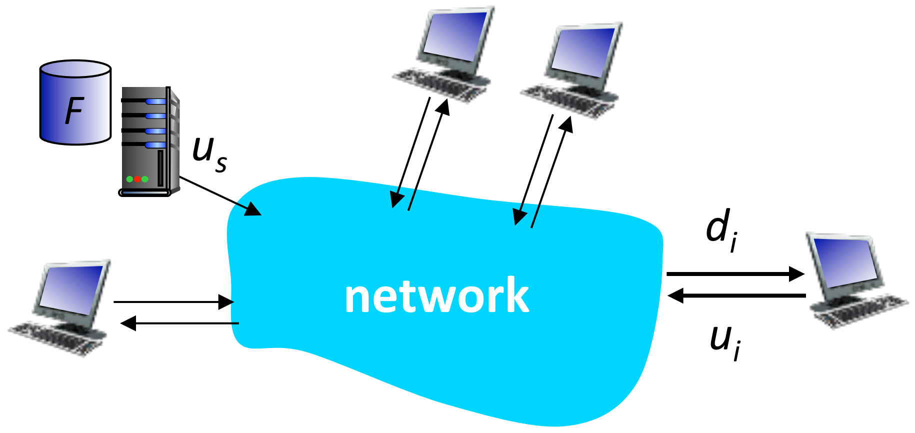
– Peer upload/download capacity is limited resource



u_s: server upload capacity
file, size F
server
u_s
u_1
d_1
u_2
d_2
d_i: peer i download capacity
network (with abundant bandwidth)
u_N
d_N
d_i
u_i
u_i: peer i upload capacity

File distribution time: client-server

- *Server transmission:* must sequentially send (upload) *N* file copies:
  - Time to send one copy: F/u_s
  - Time to send N copies: NF/u_s



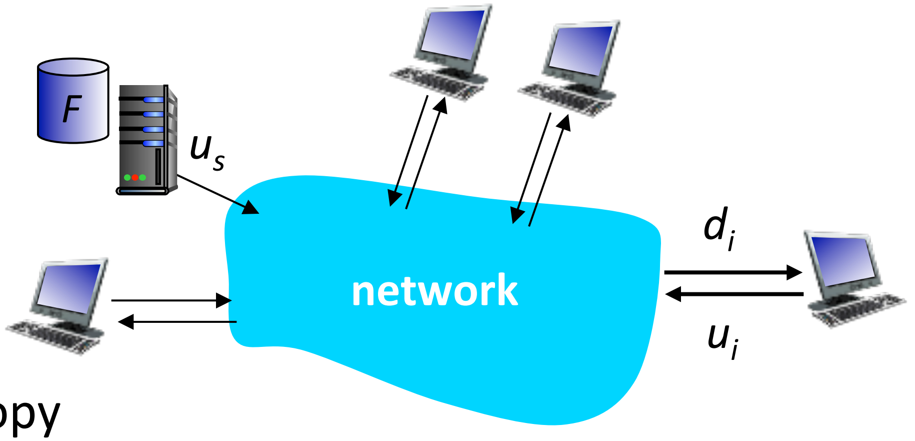
- *Client:* each client must download file copy
  - d_{min} = min client download rate
  - Min client download time: F/d_{min}

Time to distribute F to N clients using client-server approach

$$D_{c\text{-}s} \geq \max\{ NF/u_s,\ F/d_{min} \}$$

increases linearly in N

File distribution time: P2P



- Server transmission:
  - Must upload at least one copy
  - Time to send one copy: F/u_s
- Client:
  - Each client must download file copy
  - Min client download time: F/d_{min}
- Clients:
  - Aggregate download of *NF* bits
  - Max upload rate (limiting max download rate) is $u_s + \Sigma u_i$

Time to distribute F to N clients using P2P approach

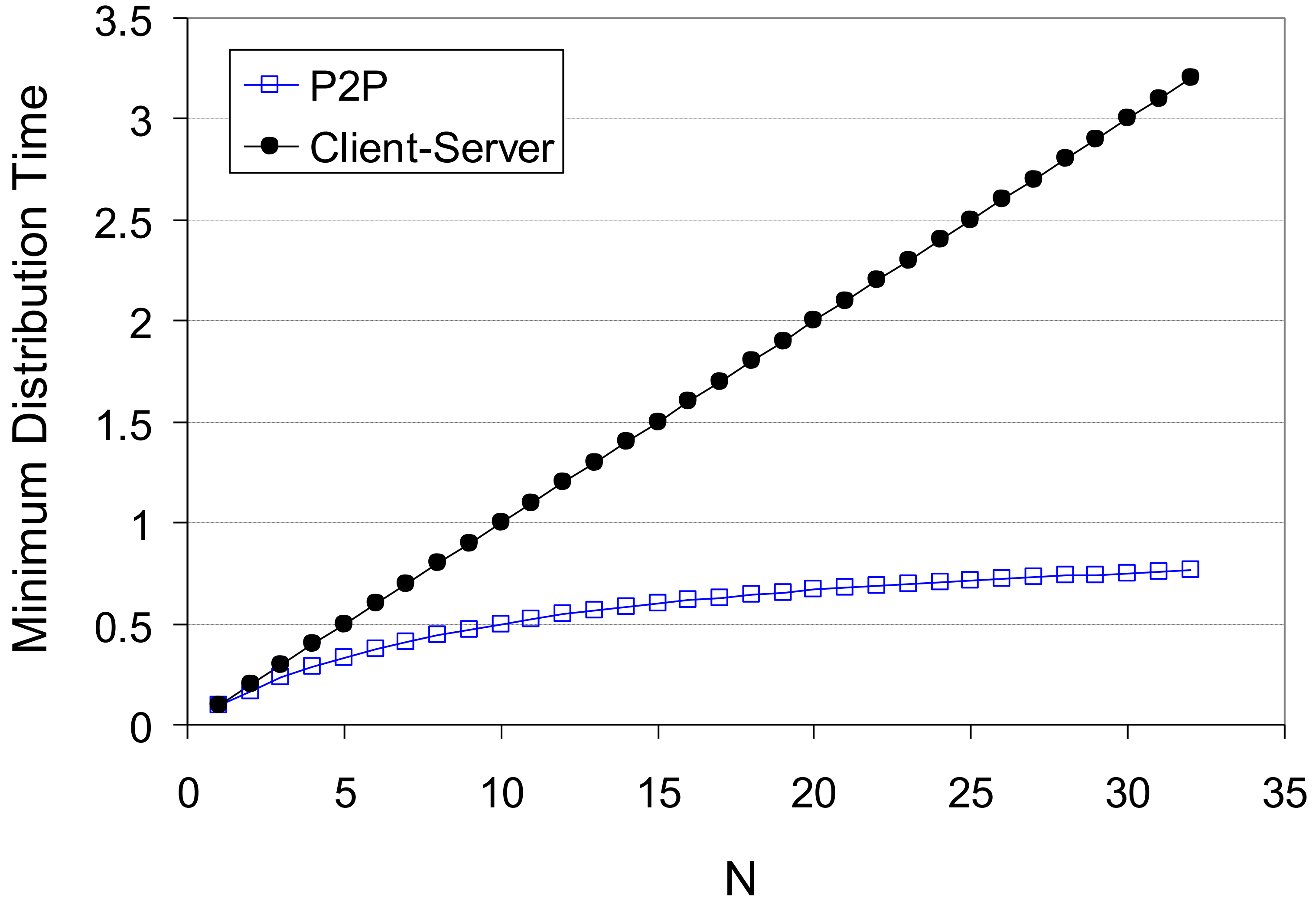
$$D_{P2P} \geq max\{ F/u_s, F/d_{min}, NF / (u_s + \Sigma u_i) \}$$

increases linearly in *N*…

… but so does this, as each peer brings service capacity

Client-server vs. P2P example

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

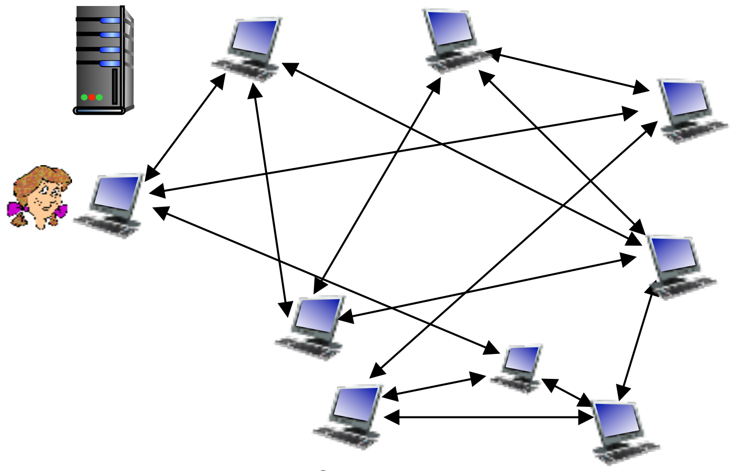


P2P: BitTorrent

- BitTorrent protocol
  - 2001, Bram Cohen releases first implementation
  - Now supported by many different clients
  - 2011, ~100 million users
- Motivations:
  - Serve up popular content fast
    - Popularity exhibits temporal locality
    - Efficient fetching, not searching
    - Distribute same file to many peers
    - Single publisher, many downloaders
  - Measures to prevent free-loading

BitTorrent process

- File divided into many 256KB chunks
  - Peers exchange the pieces by uploading and downloading to each other
  - Seed: peer with entire file

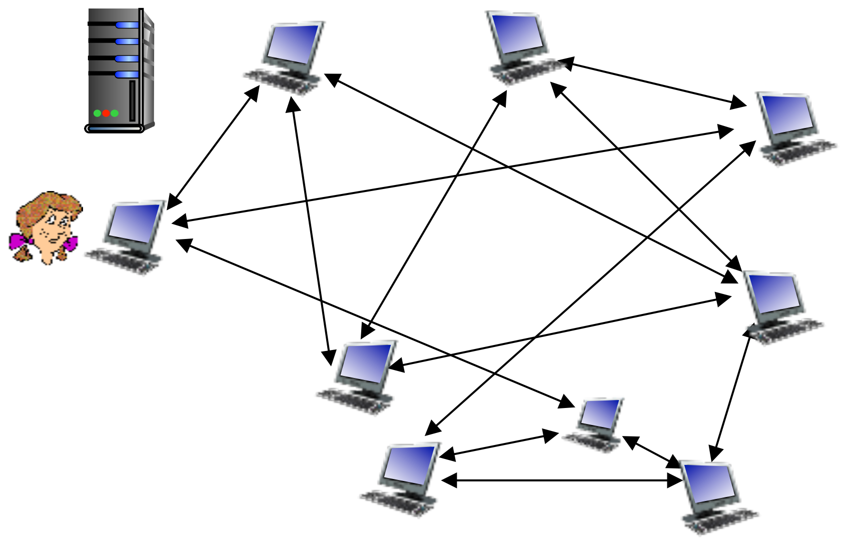


http://youtu.be/w8_JHgVNsA8

- Process:
  - Users find torrent of interest, open in client
  - Client contacts the *tracker* listed in torrent file
  - Gets list of peers currently transferring the file
  - Joins the *swarm*
    - Peers currently with some/all of the file

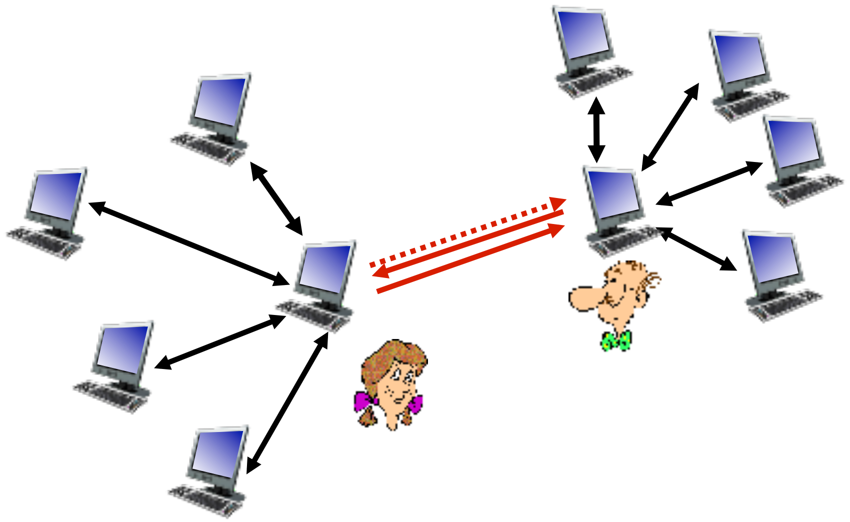
BitTorrent process

- Peer joining torrent:
  - Has no chunks, but will accumulate them over time from other peers
  - Registers with tracker to get list of peers, connects to subset of peers, "neighbors"



- While downloading, peer uploads chunks to other peers
- Peer may change peers with whom it exchanges chunks
- *Churn:* peers may come and go
- Once peer has entire file it may (selfishly) leave or (altruistically) remain in torrent

BitTorrent: requesting, sending file chunks



Requesting chunks:

- At any given time, different peers have different subsets of file chunks
- Periodically, Alice asks each peer for list of chunks that they have
- Alice requests missing chunks from peers, rarest first

Sending chunks: tit-for-tat

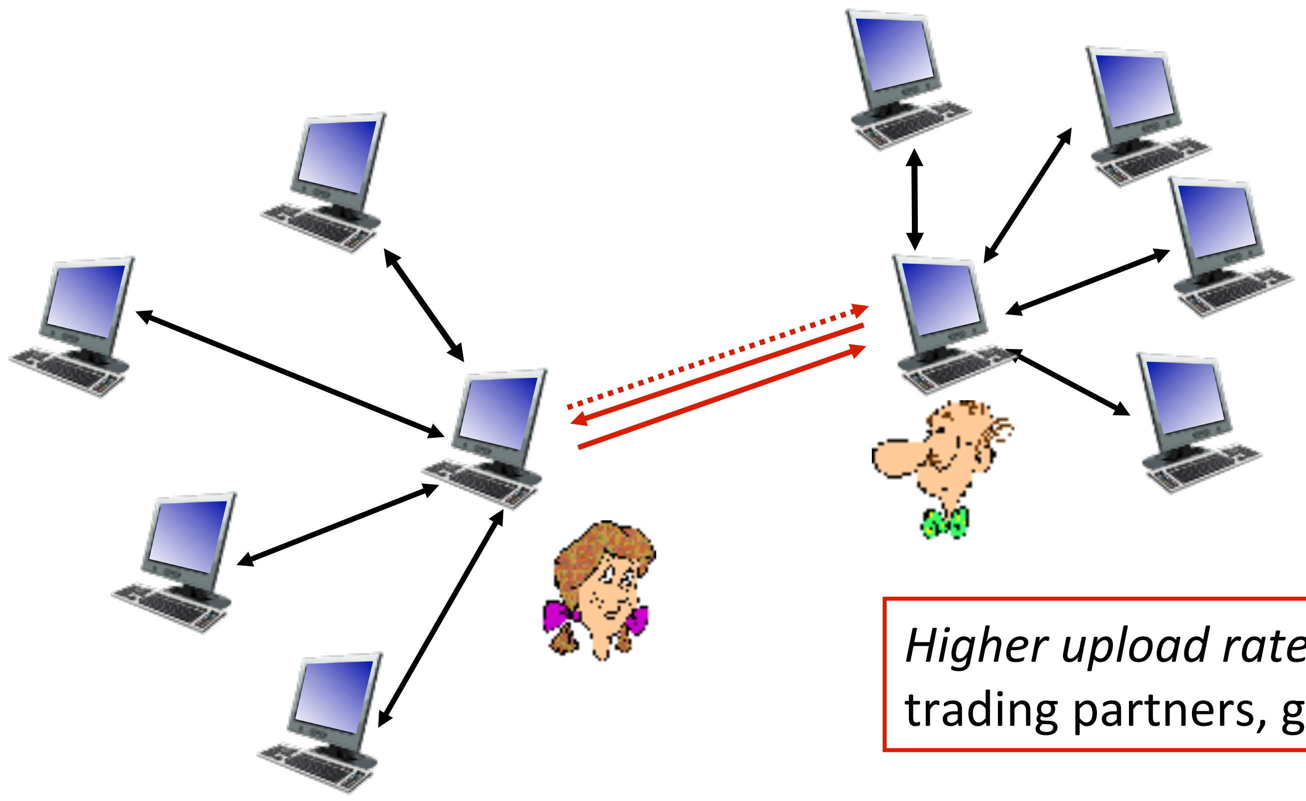
- Alice sends chunks to 4 peers currently sending her chunks *at highest rate*
  - Other peers are choked by Alice
  - Re-evaluate top 4 every 10 seconds
- Every 30 secs: randomly select another peer, starts sending chunks
  - "Optimistically unchoke" this peer
  - Newly chosen peer may join top 4

BitTorrent: tit-for-tat

(1) Alice "optimistically unchokes" Bob

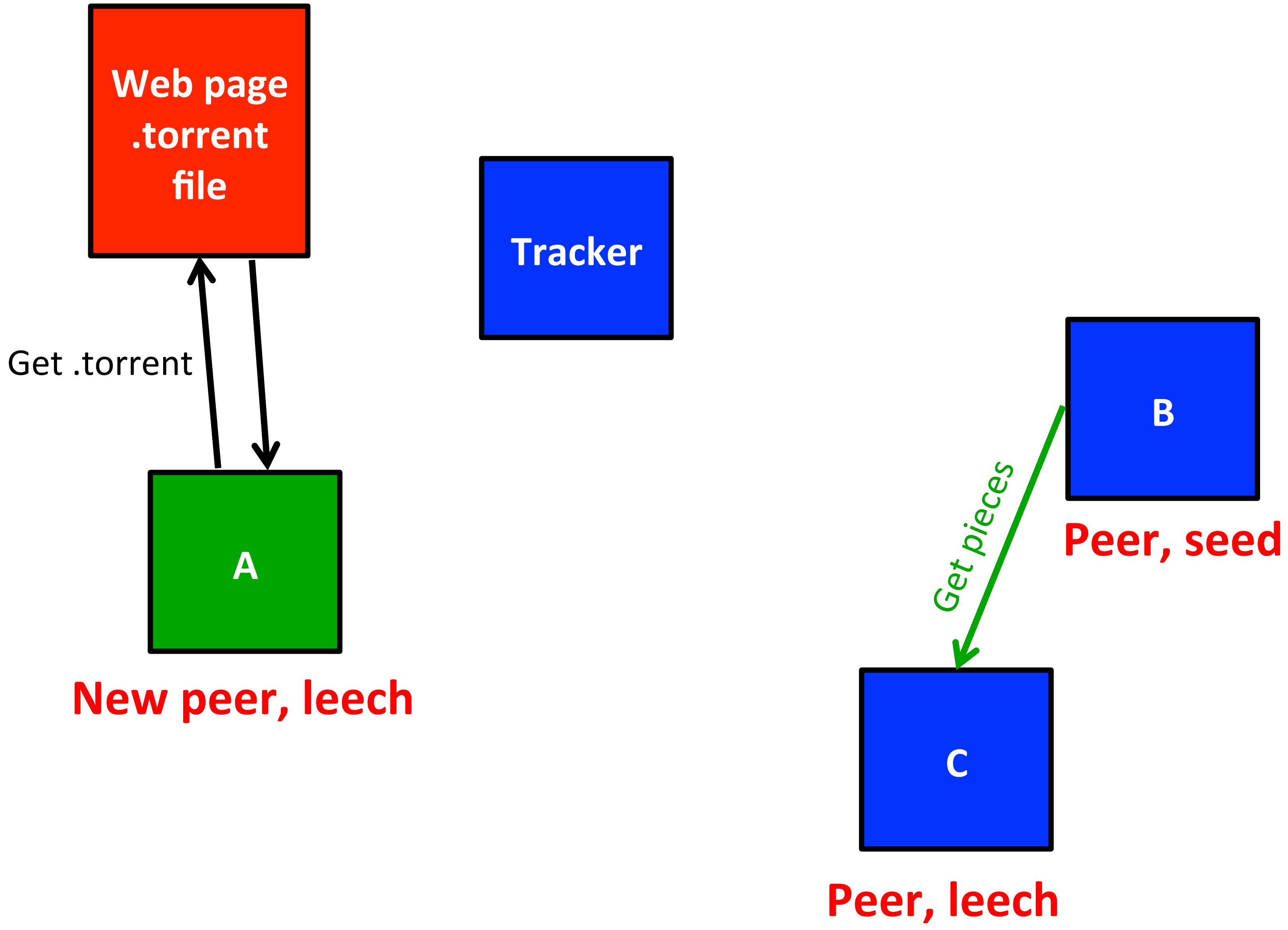
(2) Alice becomes one of Bob's top-four providers; Bob reciprocates

(3) Bob becomes one of Alice's top-four providers

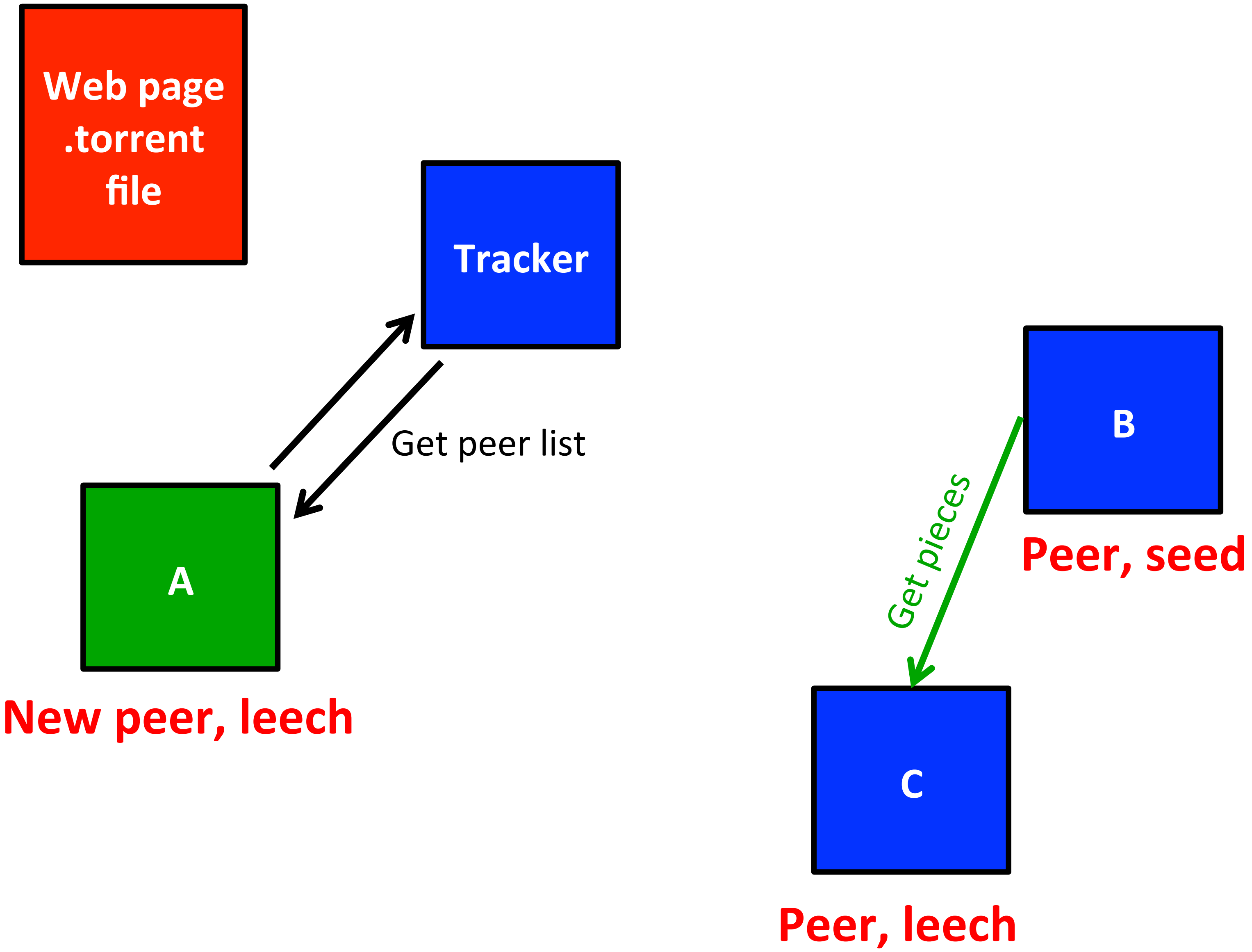


Higher upload rate: find better trading partners, get file faster!

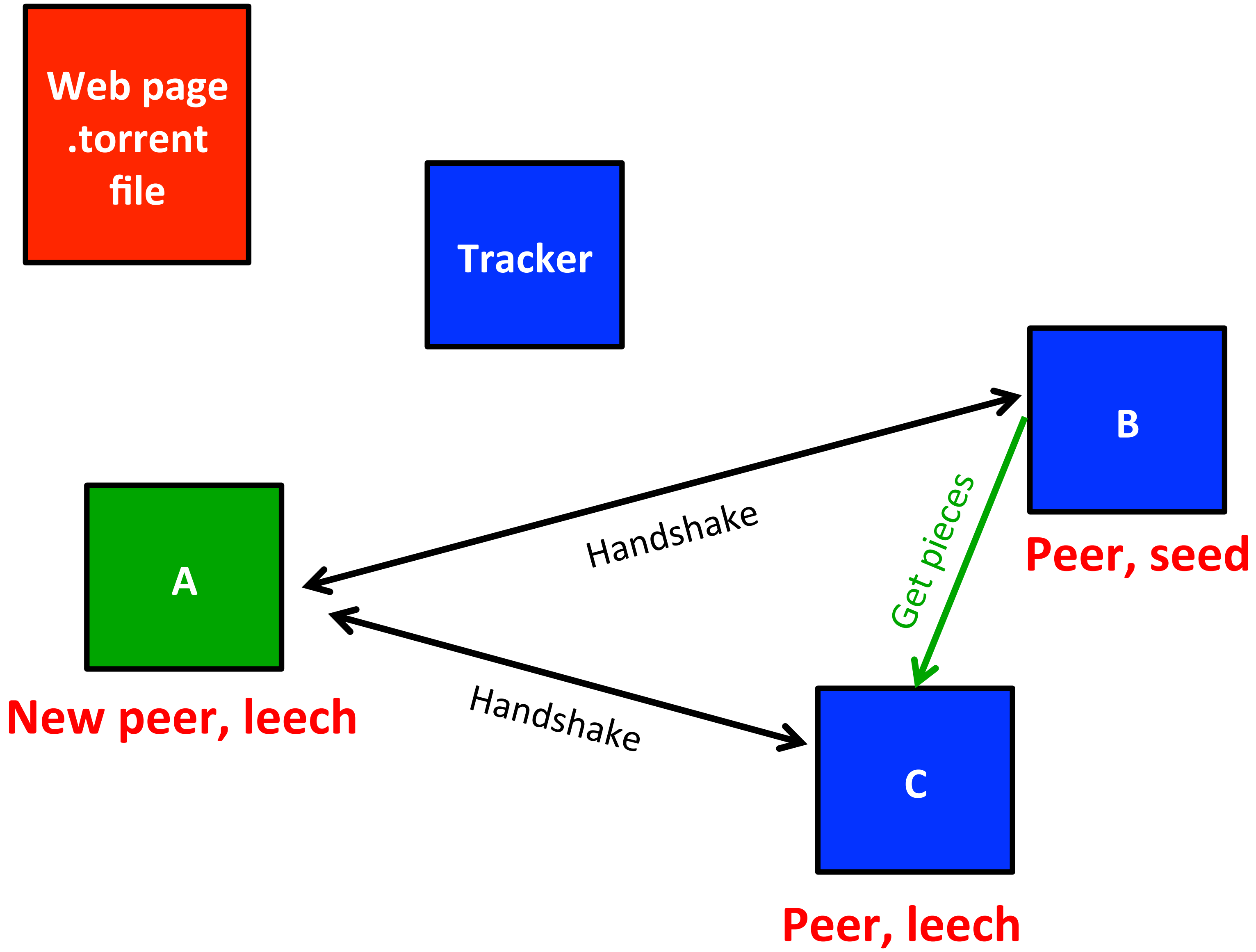
BitTorrent process



BitTorrent process

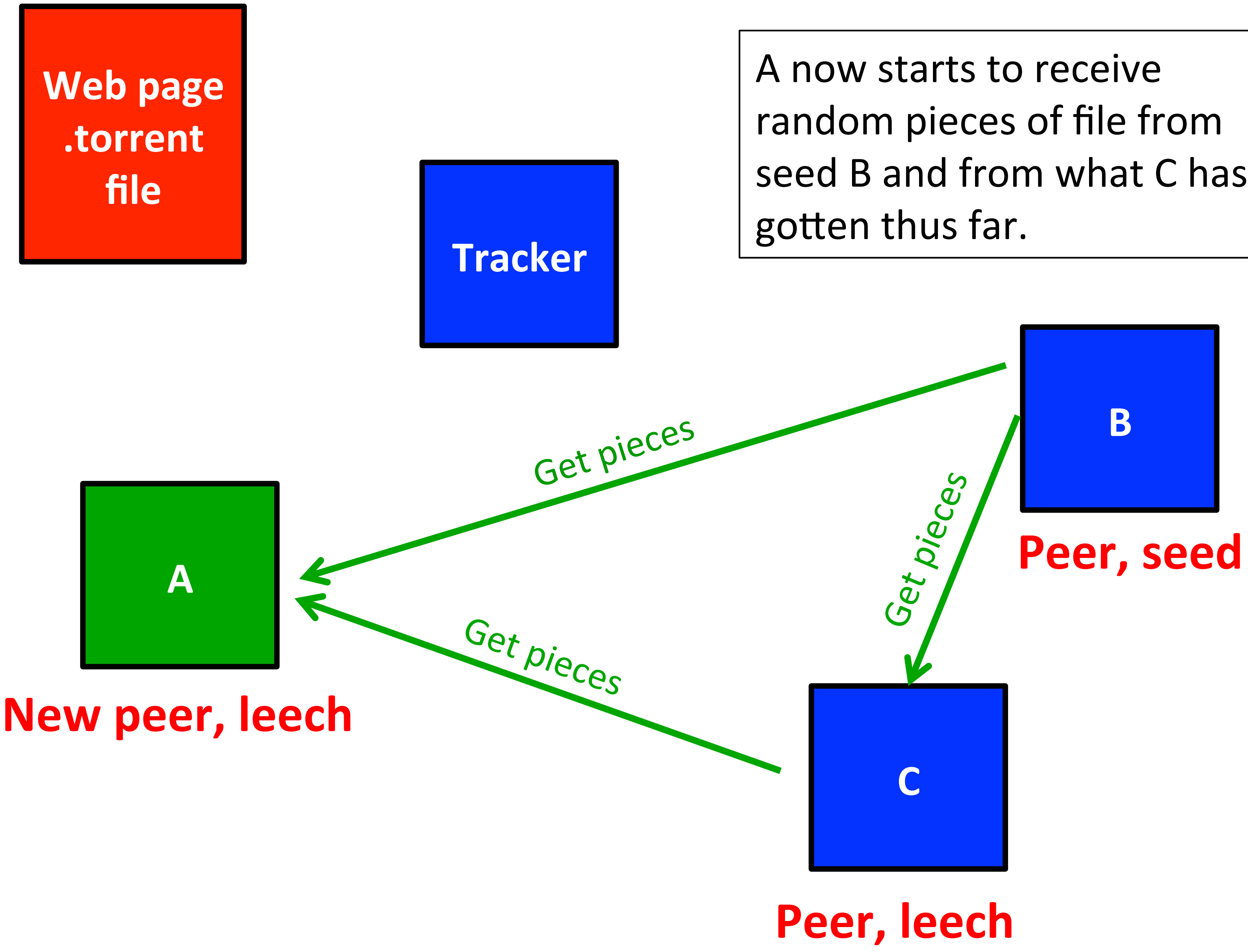


BitTorrent process



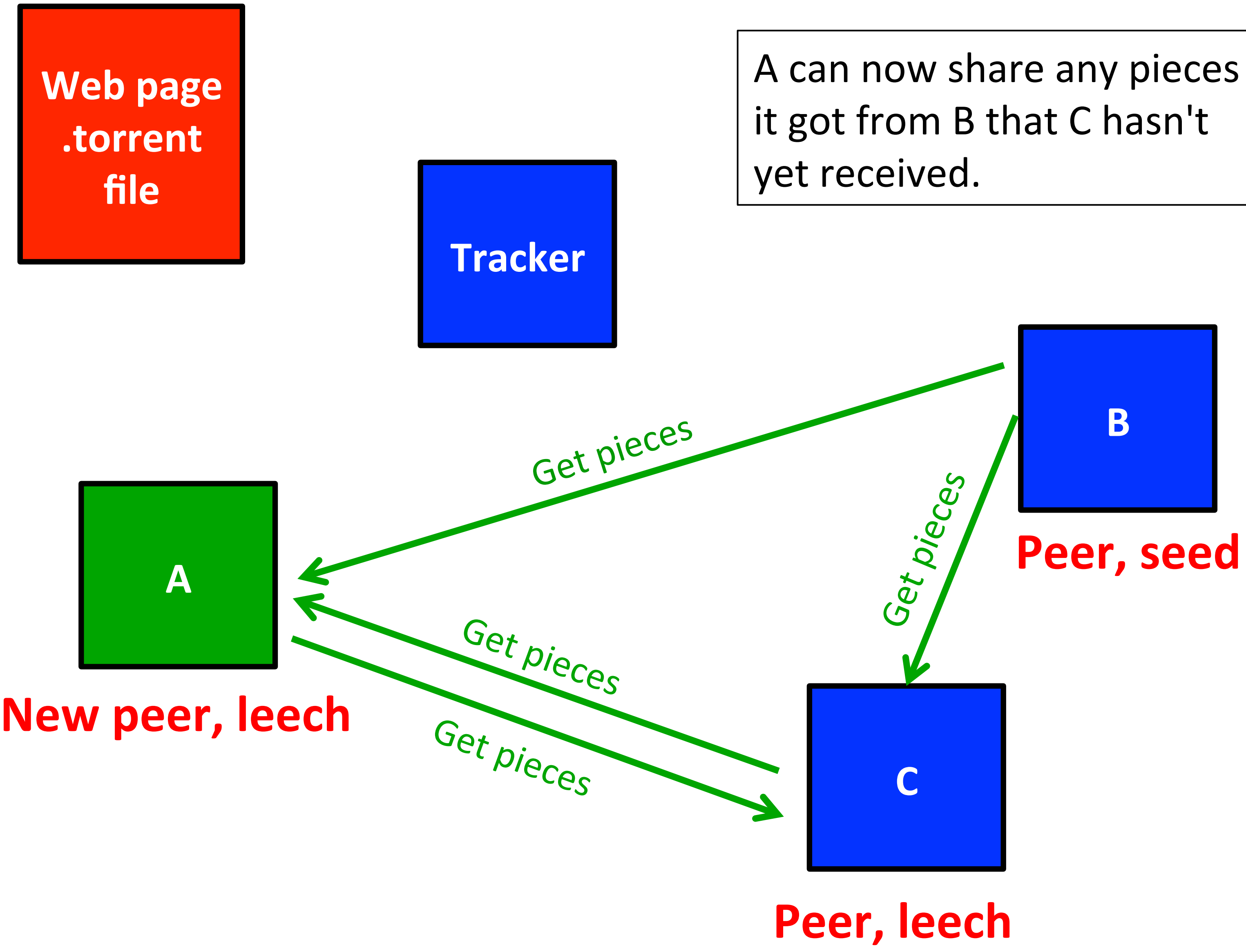
BitTorrent process

A now starts to receive random pieces of file from seed B and from what C has gotten thus far.



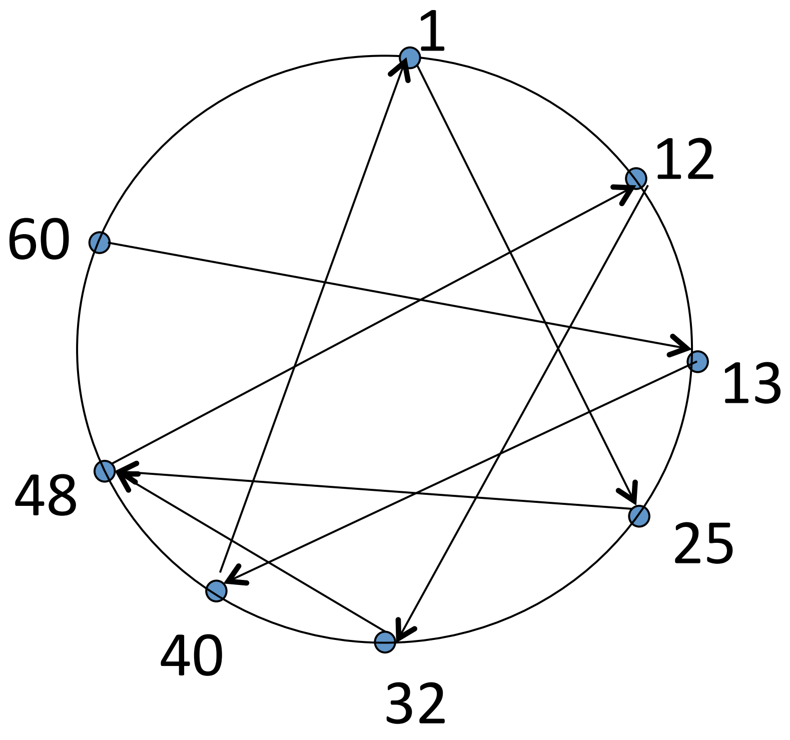
BitTorrent process

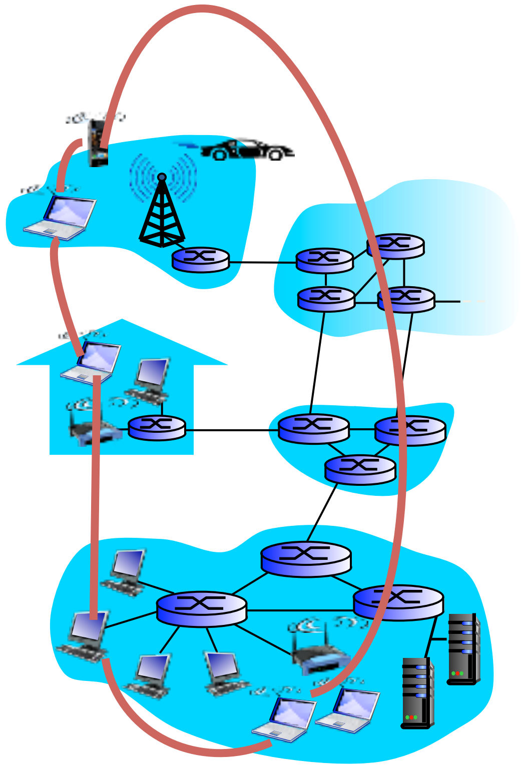
A can now share any pieces it got from B that C hasn't yet received.



Distributed Hash Table (DHT)

- Hash table
- DHT paradigm
- Circular DHT and overlay networks
- Peer churn





Simple database

- Simple database with (key, value) pairs:
  - Key: human name
  - Value: social security #

Key	Value
John Washington	132-54-3570
Diana Louise Jones	761-55-3791
Xiaoming Liu	385-41-0902
Rakesh Gopal	441-89-1956
Linda Cohen	217-66-5609
…….	………
Lisa Kobayashi	177-23-0199

  - Key: movie title
  - Value: IP address of system storing movie

Hash Table

- More convenient:
  - Store/search on numerical representation of key
  - Key = hash(original key)

Original Key	Key	Value
John Washington	8962458	132-54-3570
Diana Louise Jones	7800356	761-55-3791
Xiaoming Liu	1567109	385-41-0902
Rakesh Gopal	2360012	441-89-1956
Linda Cohen	5430938	217-66-5609
…….		………
Lisa Kobayashi	9290124	177-23-0199

Distributed Hash Table (DHT)

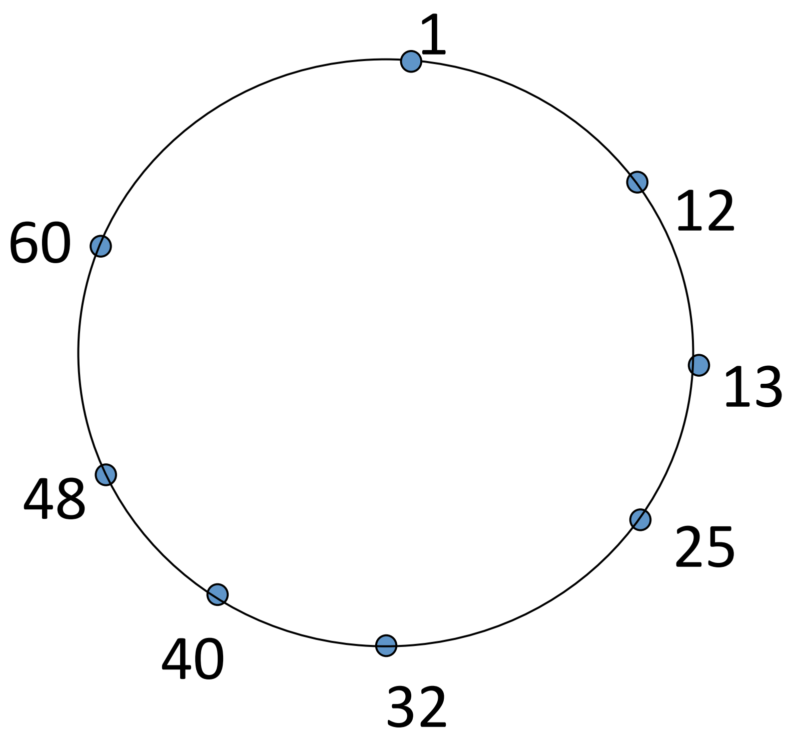
- Distribute (key, value) pairs over millions of peers
  - Pairs are evenly distributed over peers
- Any peer can query database with a key
  - Database returns value for the key
  - To resolve query, small number of messages exchanged among peers
- Peer only knows a small number of other peers
- Robust to peers coming and going, *churn*

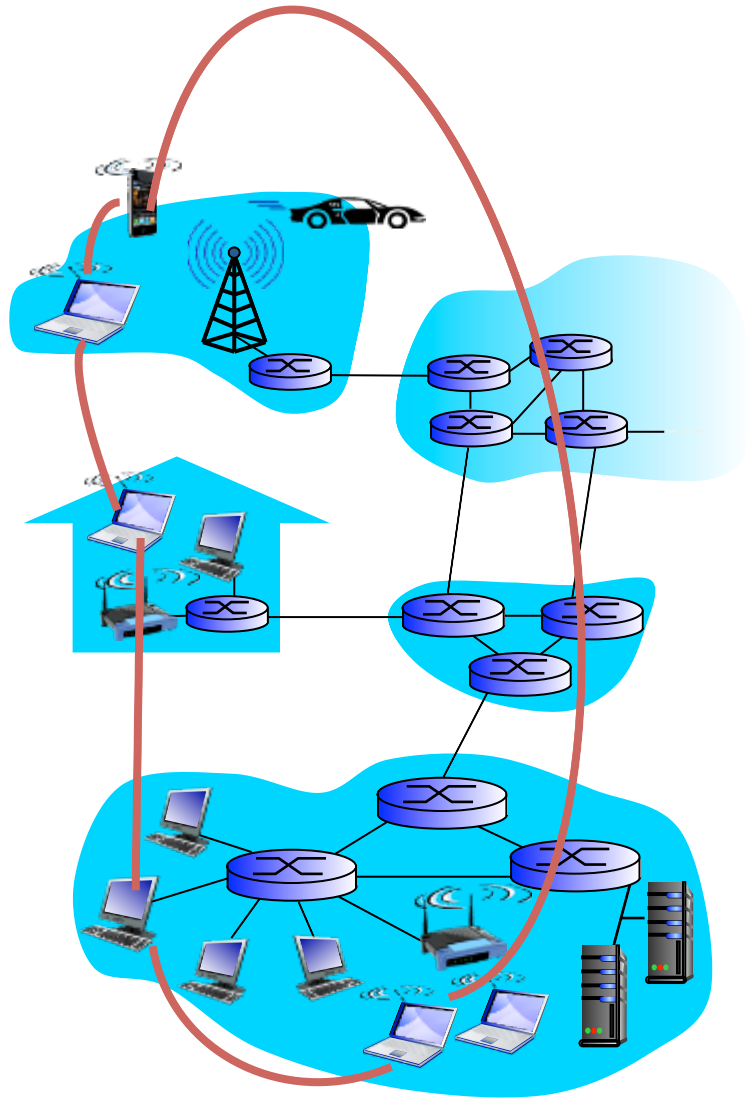
Assign key-value pairs to peers

- Rules:
  - Assign key-value pair to the peer that has the *closest* ID
  - Closest is the *immediate successor* of the key
- Example:
  - ID space {0, 1, 2, 3, …, 63}
  - 8 peers: 1, 12, 13, 25, 32, 40, 48, 60
  - If key = 51, then assigned to peer 60
  - If key = 60, then assigned to peer 60
  - If key = 61, then assigned to peer 1

Circular DHT

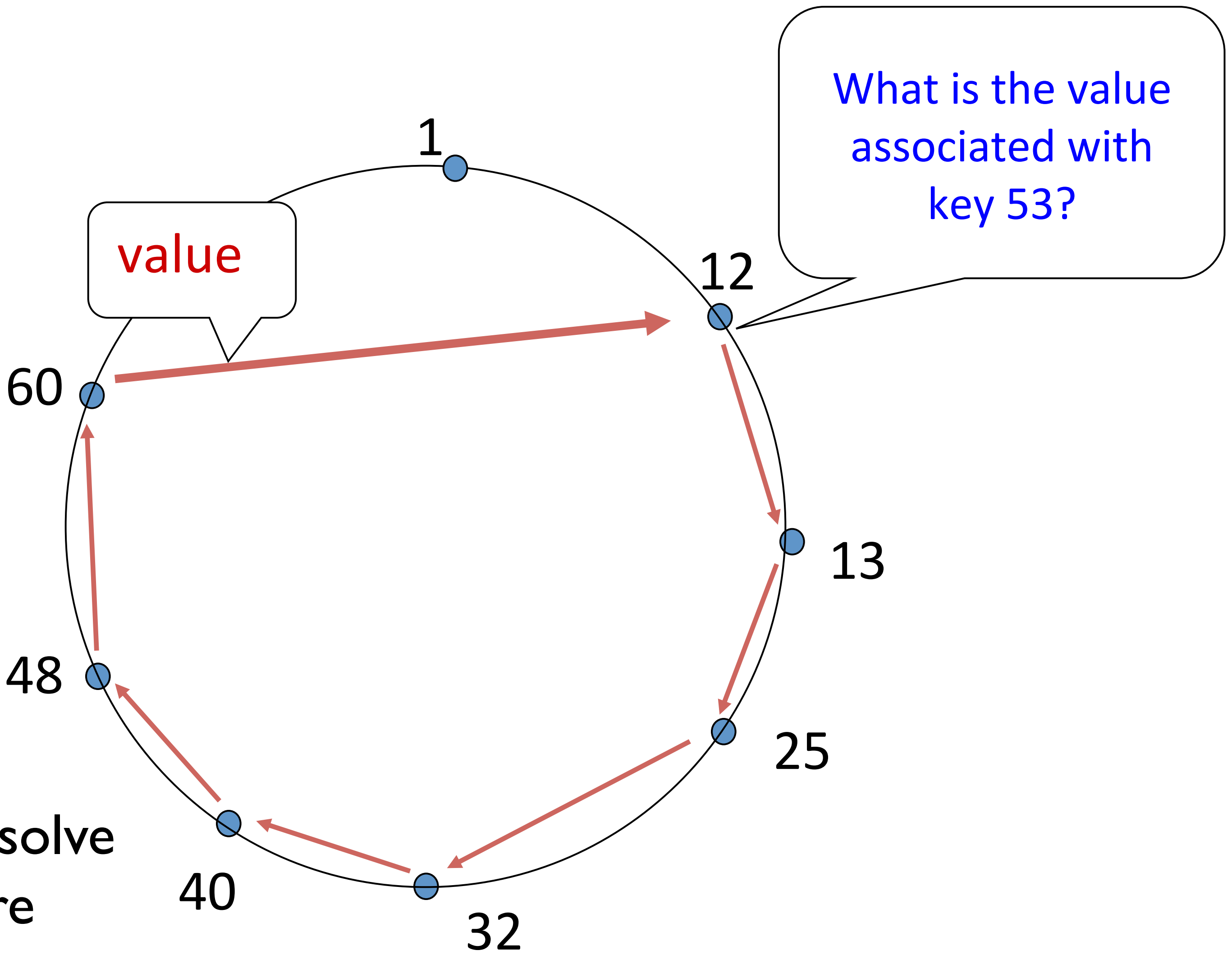
- Each peer *only* aware of immediate successor and predecessor





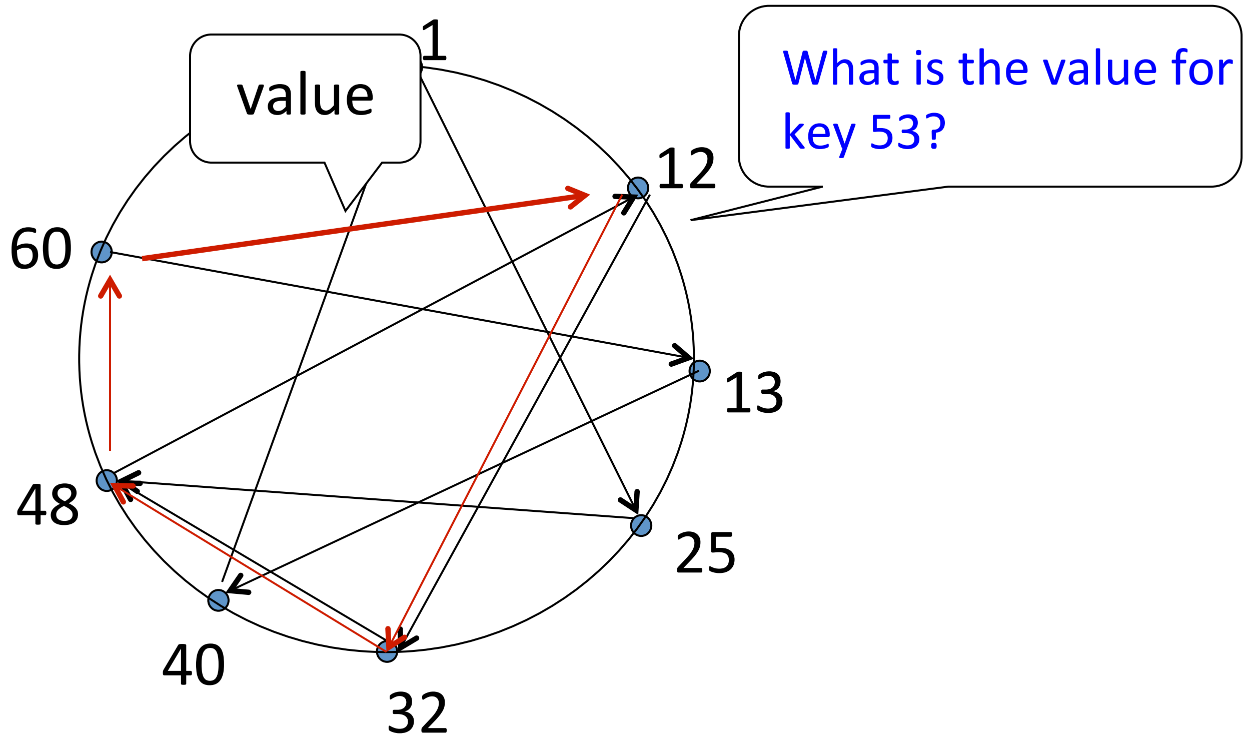
"overlay network"

Resolving a query



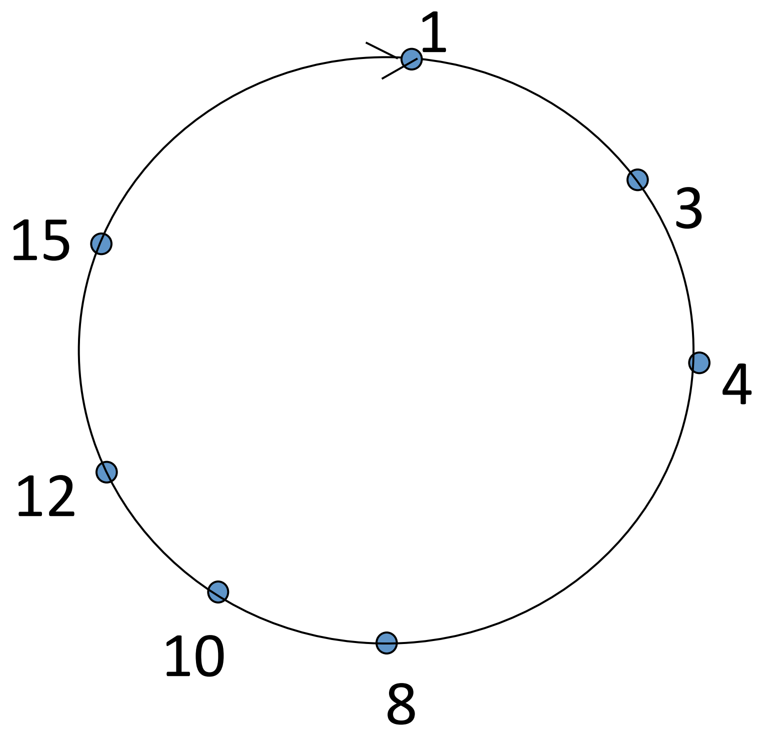
O(*N*) messages
on average to resolve
query, when there
are *N* peers

Circular DHT with shortcuts



- Each peer keeps track of IP addresses of predecessor, successor, and short cuts
- Reduced from 6 to 3 messages
- Possible to design shortcuts with *O(log N)* neighbors, *O(log N)* messages in query

Peer churn



Handling peer churn:

- Peers may come and go (churn)
- Each peer knows address of its two successors
- Each peer periodically pings its two successors to check aliveness
- If immediate successor leaves, choose next successor as new immediate successor

Example: peer 5 abruptly leaves

- Peer 4 detects peer 5's departure; makes 8 its immediate successor
- 4 asks 8 who its immediate successor is; makes 8's immediate successor its second successor.

Summary

- Peer-to-peer applications
  - Use an overlay network
    - Logical network on top of existing physical network
  - Scale better than client-server model
    - Clients share chunks using their upload/download links
  - Finding things:
    - May be centralized (e.g. Napster)
    - Decentralized via a distributed hash table