This Lecture will be recorded!!!

Welcome to

CS 3516:
Computer Networks

Prof. Yanhua Li

Time: 9:00am –9:50am M, T, R, and F

Zoom Lecture

Fall 2020 A-term

Quiz 4 on Thursday

domain name registration, DNS hierarchy and Local DNS, DSN message and resource records (RRs) format.

Lab 2 due on Monday 9/21

Available on class website

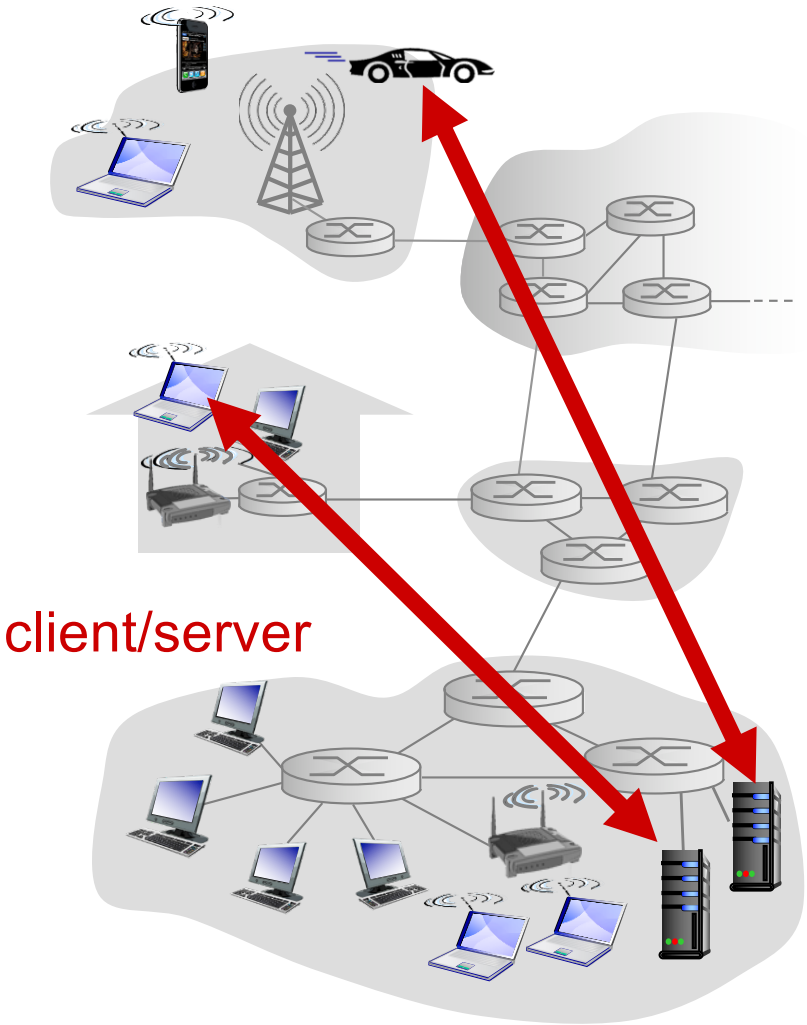
Project 1 due on Thursday 9/17

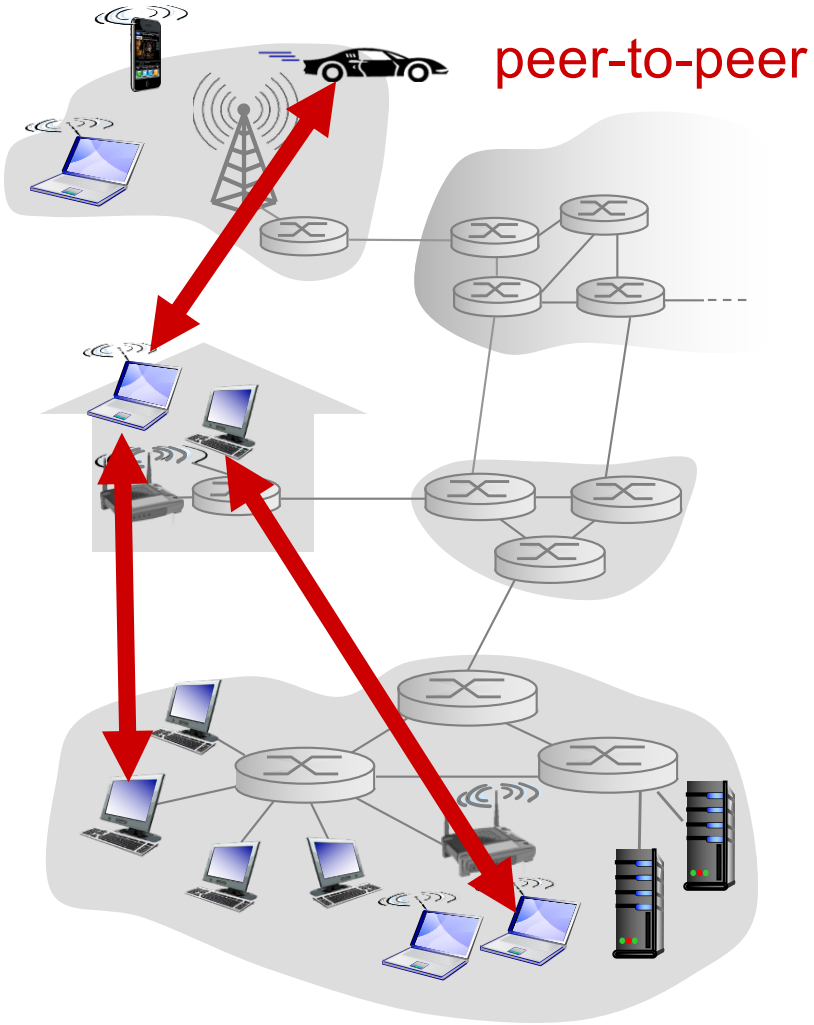
Local DNS, Web Cache, Aut. DNS

2.6 P2P applications

1. P2P vs Client&Server
2. Peer-to-Peer Networks
   - Napster
   - Gnutella
   - BitTorrent

Client-server vs P2P architecture





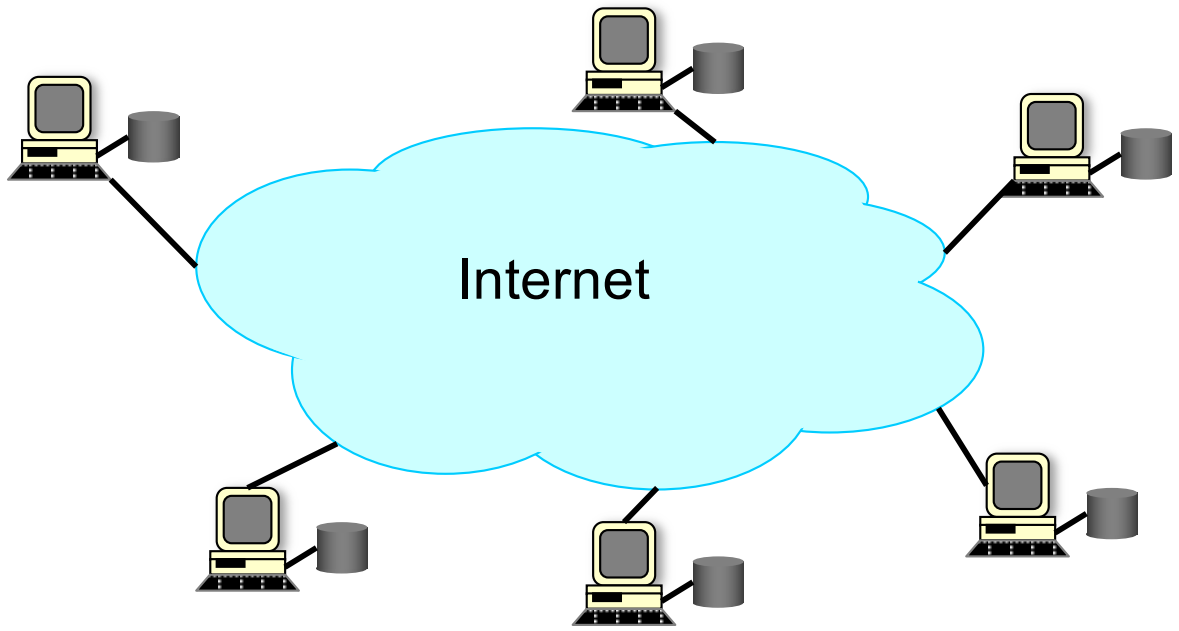
Chapter 2: outline

2.6 P2P applications

1. P2P vs Client&Server
2. Peer-to-Peer Networks
   - Napster
   - Gnutella
   - BitTorrent

Peer-to-Peer Networks: How Did it Start?

- A killer application: Napster from 1999
  - Free music over the Internet
- Key idea: share the content, storage *and* bandwidth of individual (home) users



Model

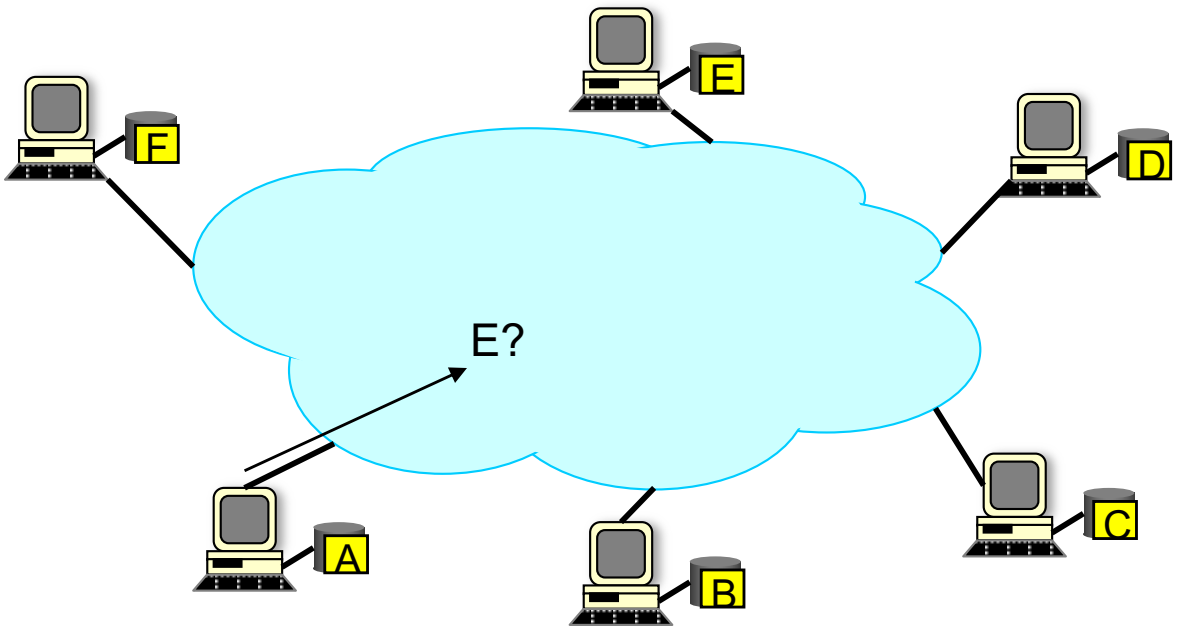
- Each user stores a subset of files
- Each user has access (can download) to files from all users in the system

Challenges

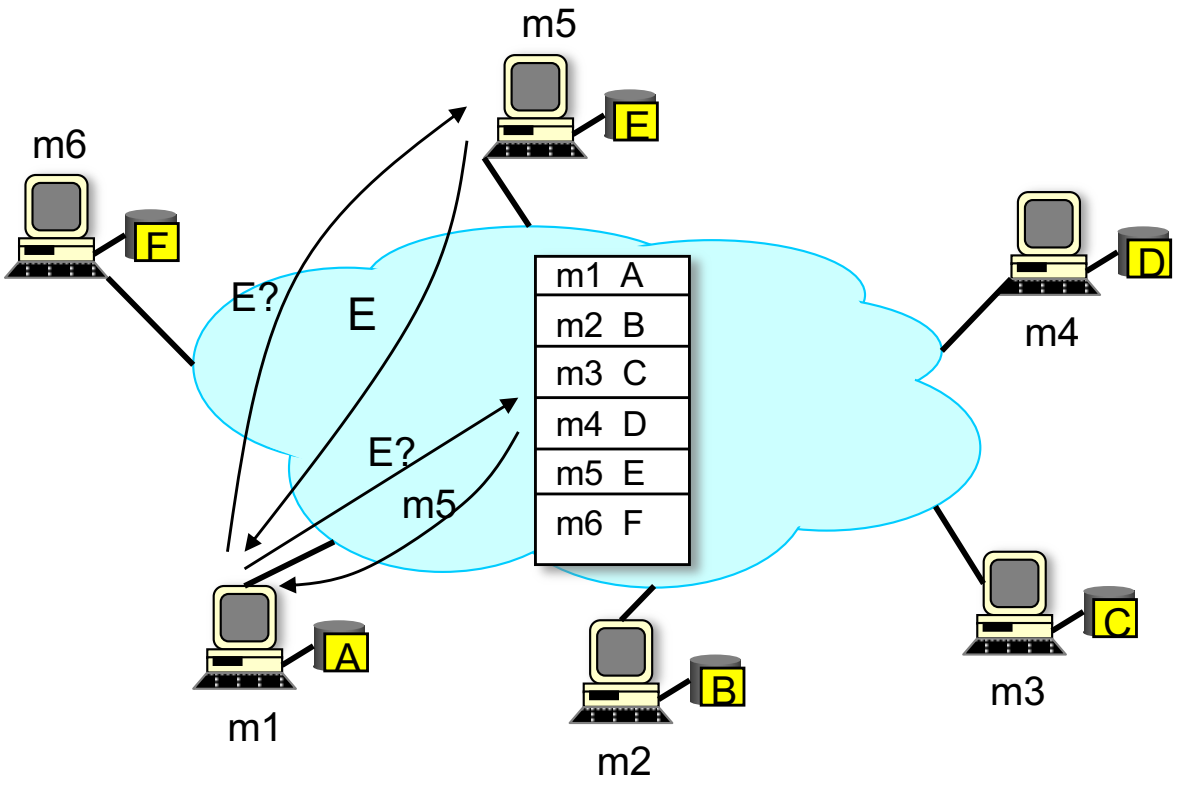
- Scale: up to hundred of thousands or millions of machines
- Dynamicity: machines can come and go any time

Main Challenge

- Find where a particular file is stored



Napster: Example

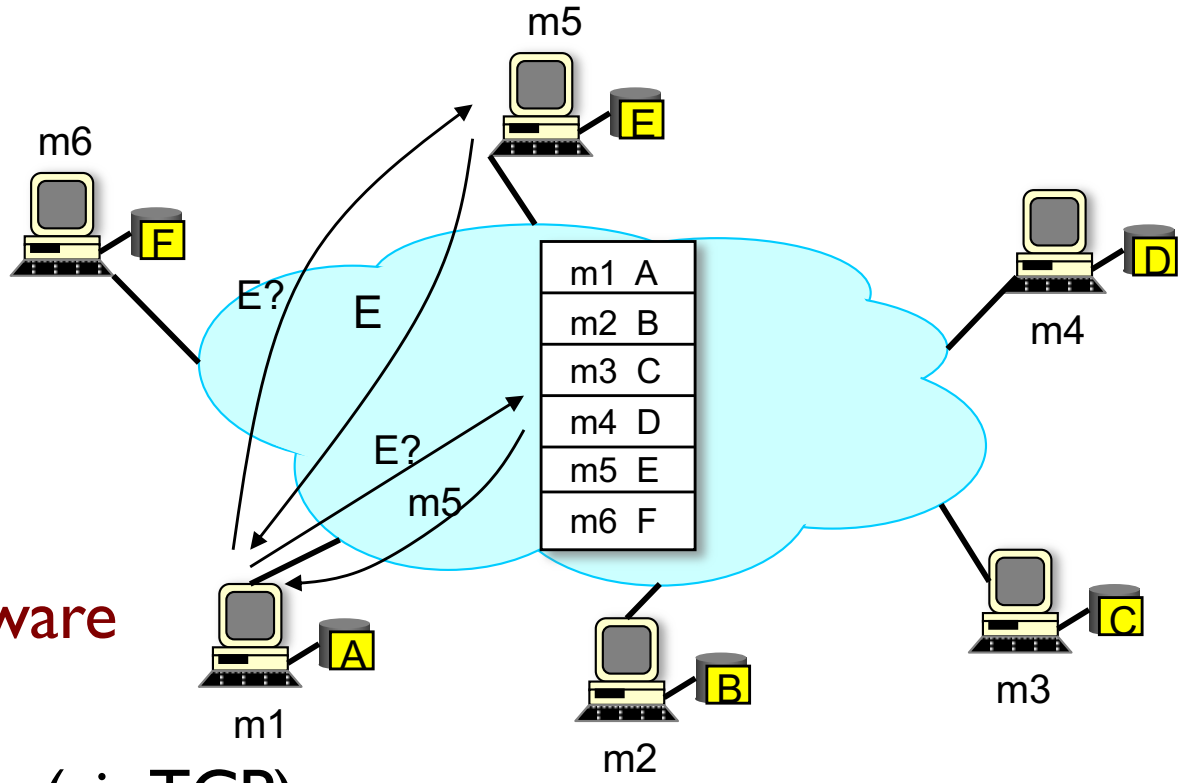


Peer-to-Peer Networks: Napster

- Napster history: the rise
  - January 1999: Napster version 1.0
  - May 1999: company founded
  - September 1999: first lawsuits
  - 2000: 80 million users
- Napster history: the fall
  - Mid 2001: out of business due to lawsuits
  - 2003: growth of pay services like iTunes
- Napster history: the resurrection
  - 2003: Napster reconstituted as a pay service
- Now
  - Music focused online service, for more details: https://en.wikipedia.org/wiki/Napster

Shawn Fanning, Northeastern freshman

Napster: Example



- User installing the software
- Client contacts Napster (via TCP)
- Client searches on a title or performer
- Client requests the file from the chosen supplier

Napster: Limitations of Central Directory

?

- Single point of failure
- Performance bottleneck
- Copyright infringement

File transfer is decentralized, but locating content is highly centralized

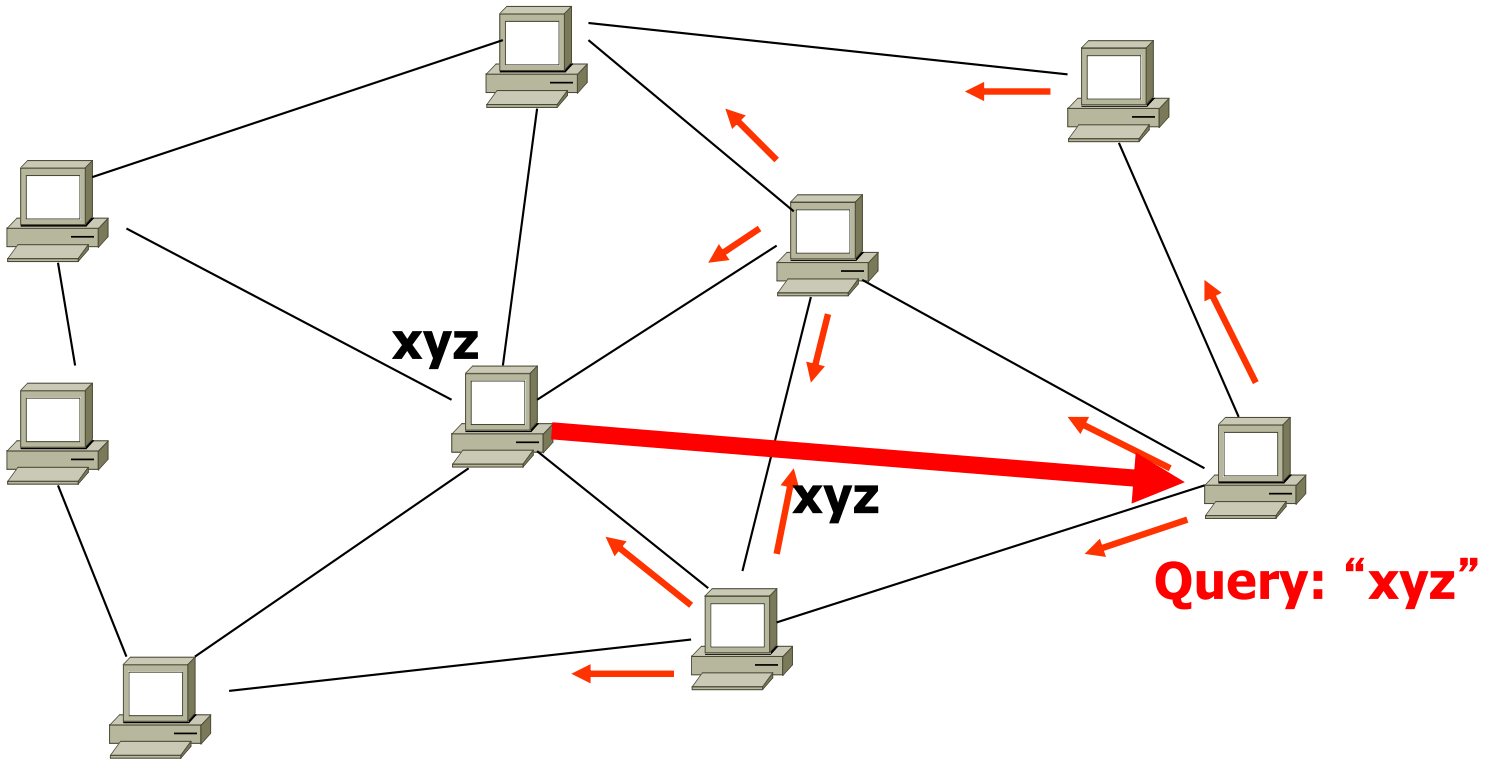
- So, later P2P systems were more distributed

Peer-to-Peer Networks: Gnutella

- Gnutella history
  - 2000: J. Frankel & T. Pepper released Gnutella
  - Soon after: many other clients (e.g., Morpheus, Limewire, Bearshare)
  - 2001: protocol enhancements, e.g., "ultrapeers"
- Query flooding
  - Join: contact a few nodes to become neighbors
  - Publish: no need!
  - Search: ask neighbors, who ask their neighbors
  - Fetch: get file directly from another node

Gnutella

- Ad-hoc topology
- No guarantees on recall



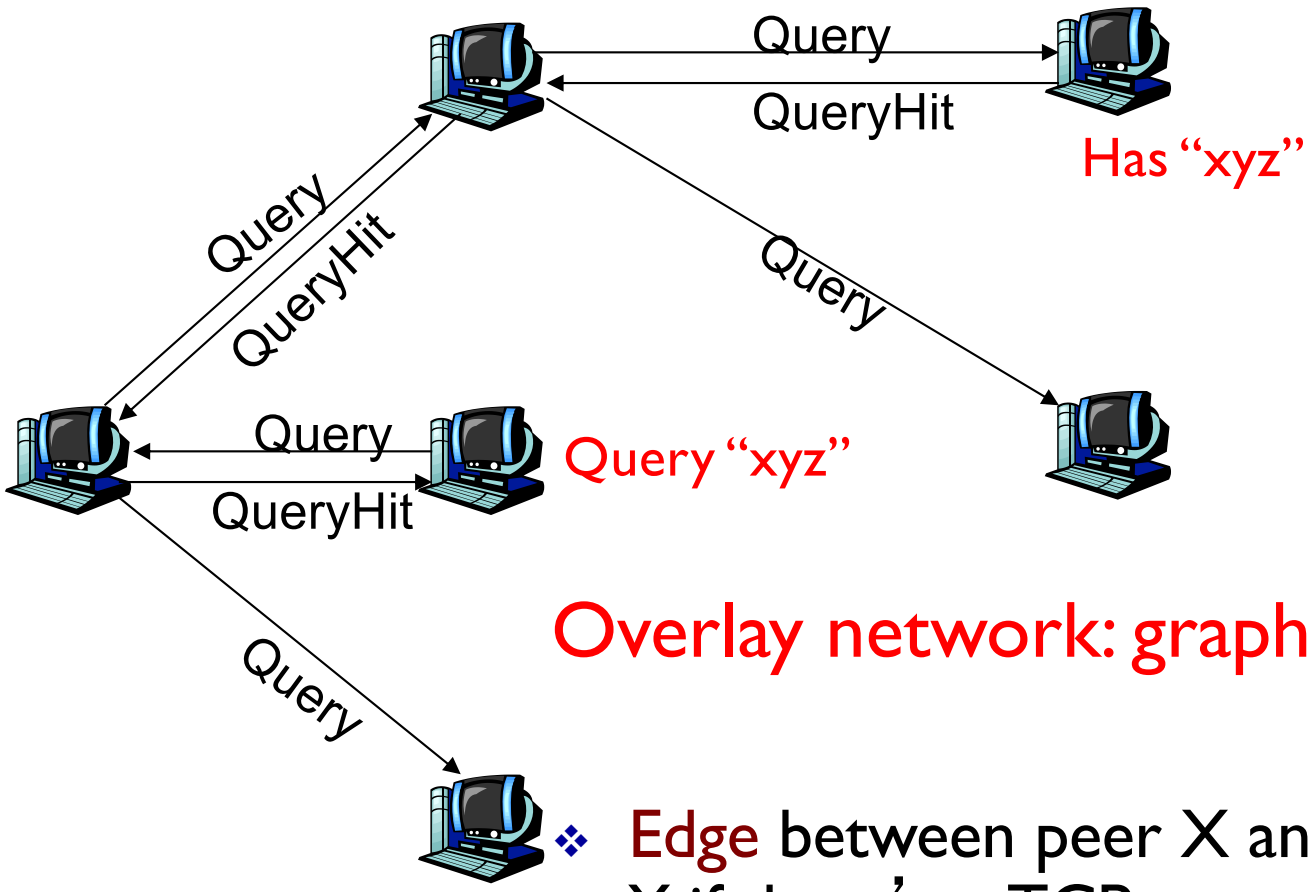
- Queries are flooded for bounded number of hops (TTL)

Gnutella: Protocol

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

Scalability:
limited scope flooding

File transfer:
HTTP



Overlay network: graph

- Edge between peer X and Y if there's a TCP connection

Gnutella: Pros and Cons

- Advantages
  - Fully decentralized, Highly robust
- Disadvantages
  - Not scalable; the entire network can be swamped with request
    - Search time may be quite long

Chapter 2: outline

2.6 P2P applications

1. P2P vs Client&Server
2. Peer-to-Peer Networks
   - Napster
   - Gnutella
   - BitTorrent

P2P design challenges

- Large file
- Free Riding
- Scalability
  - Publish/Download

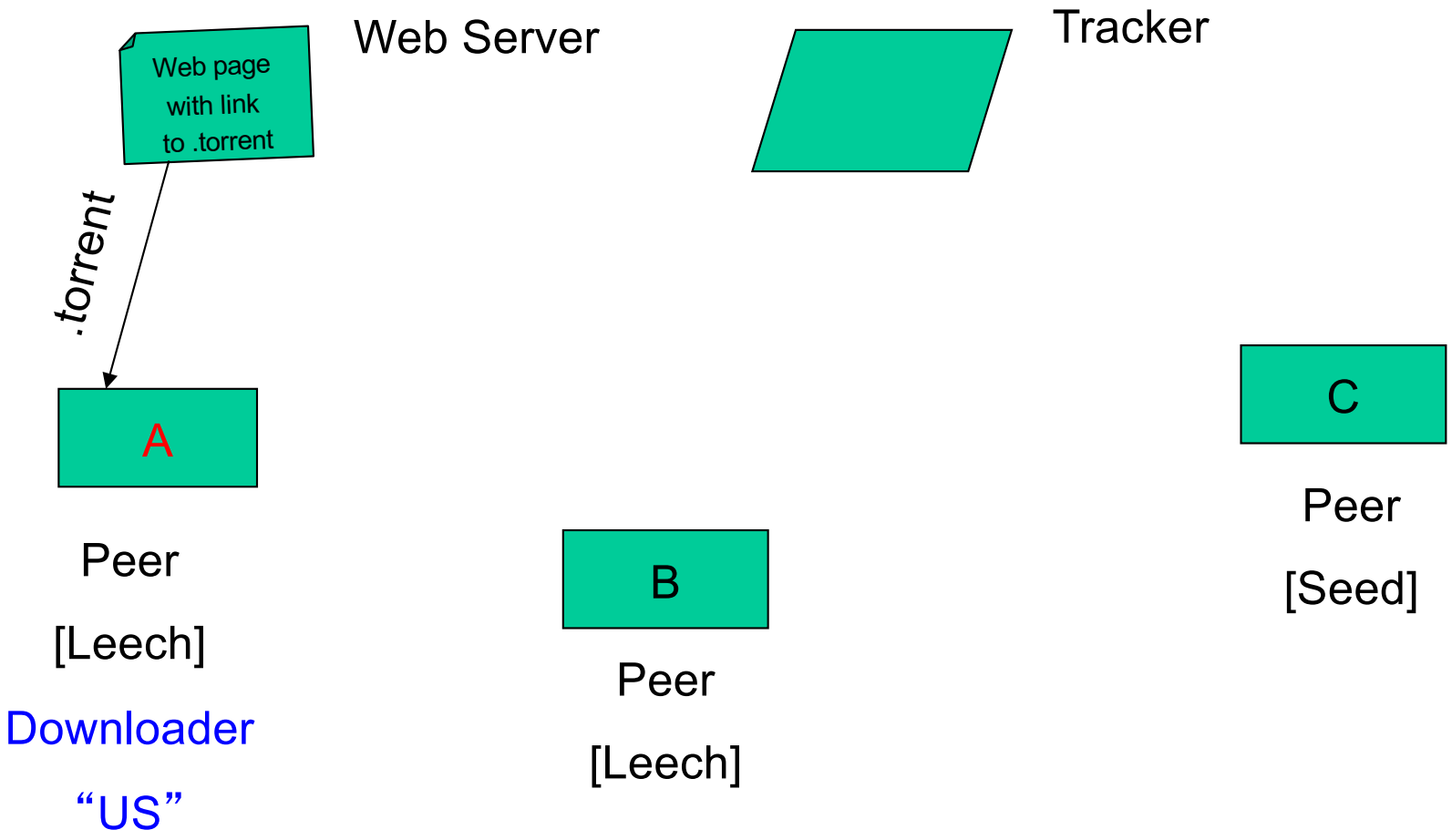
BitTorrent: Simultaneous Downloading

- Divide large file into many pieces (256Kbytes)
  - Replicate different pieces on different peers
  - A peer with a complete piece can trade with other peers
  - Peer can (hopefully) assemble the entire file
- Allows simultaneous downloading
  - Retrieving different parts of the file from different peers at the same time

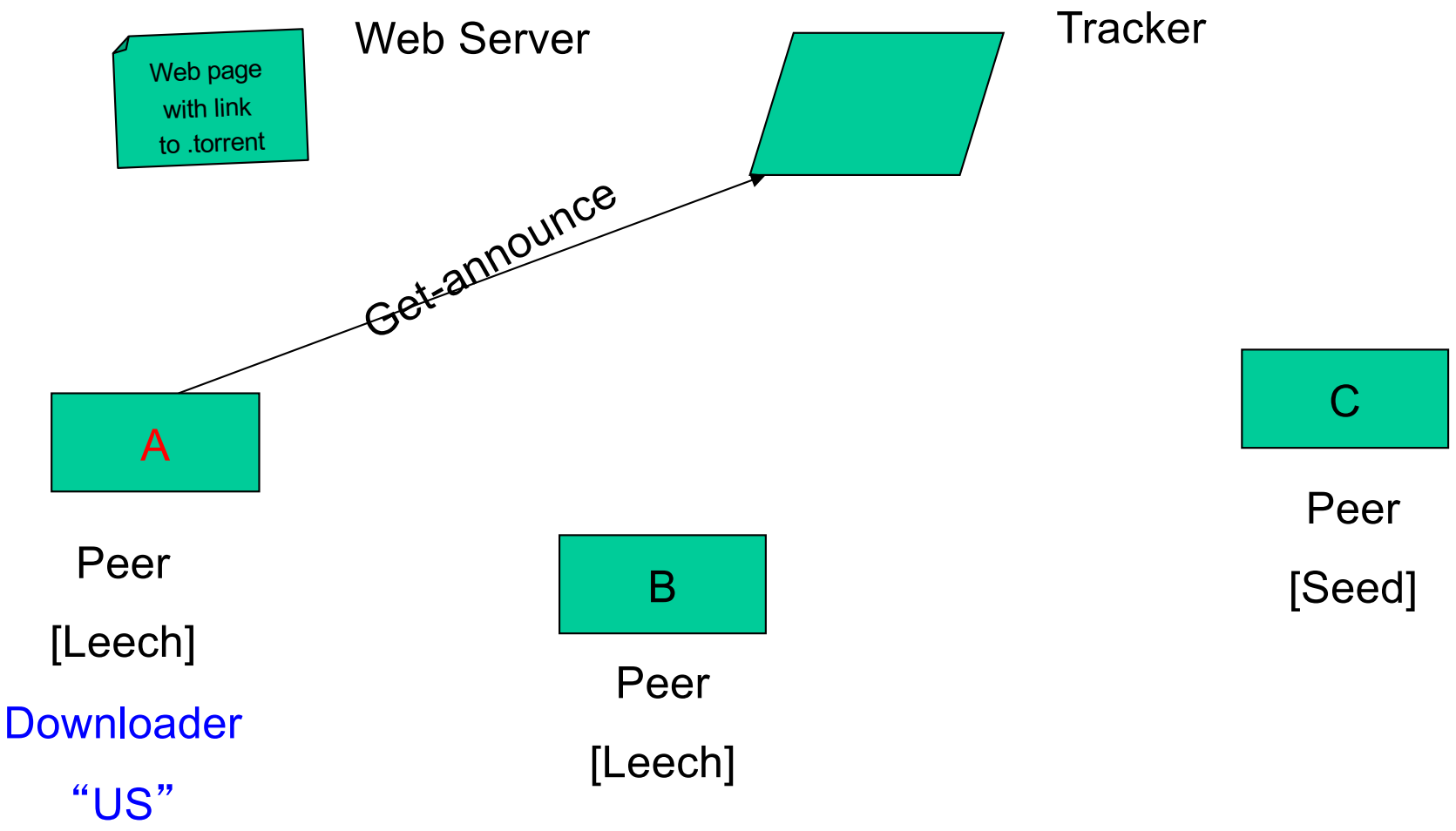
BitTorrent Components

- Seed
  - Peer with entire file
  - Fragmented in pieces
- Leech
  - Peer with an incomplete copy of the file
- Torrent file
  - Passive component
  - Stores summaries of the pieces to allow peers to verify their integrity
- Tracker
  - Allows peers to find each other
  - Returns a list of random peers

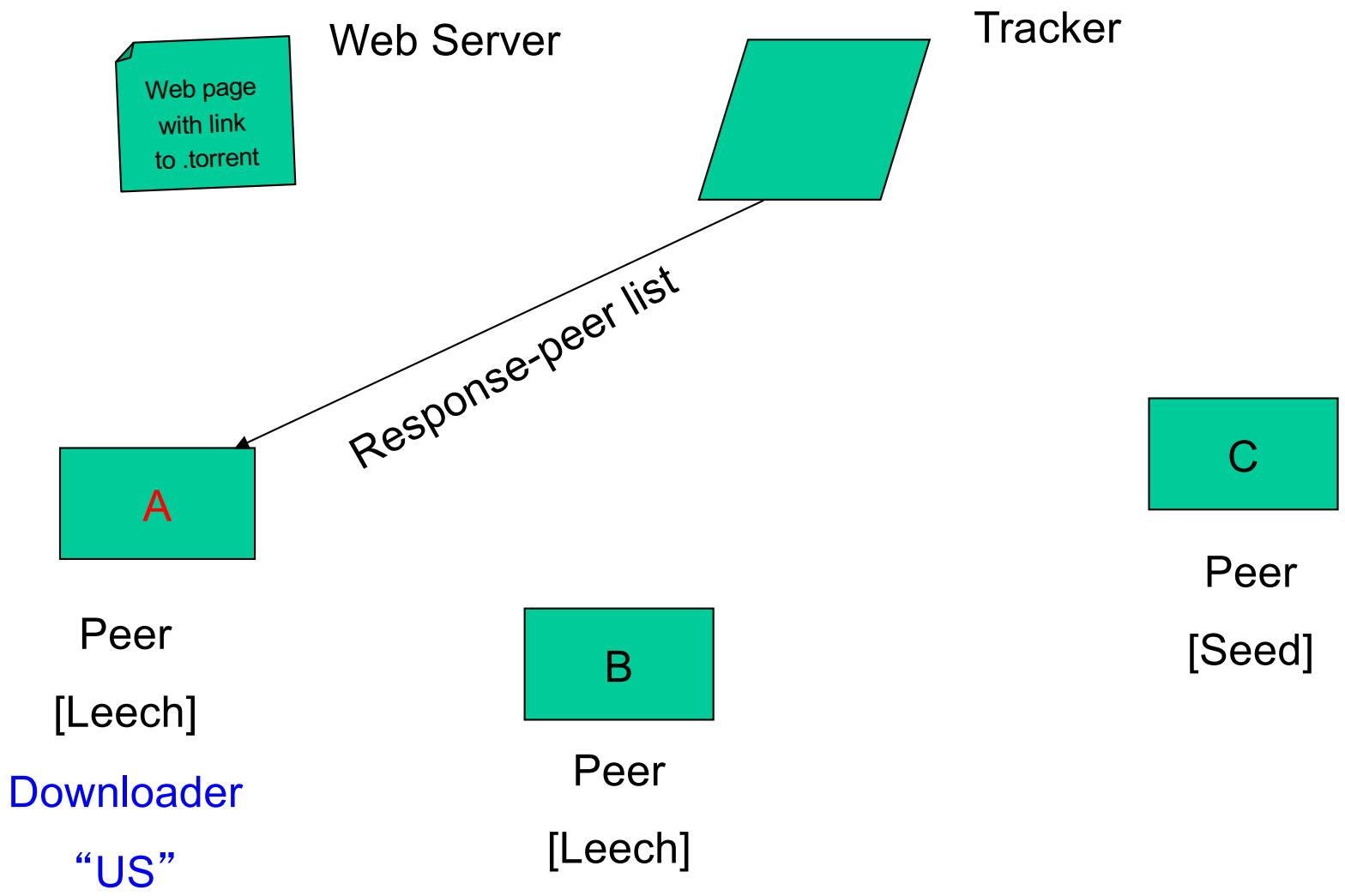
BitTorrent: Overall Architecture



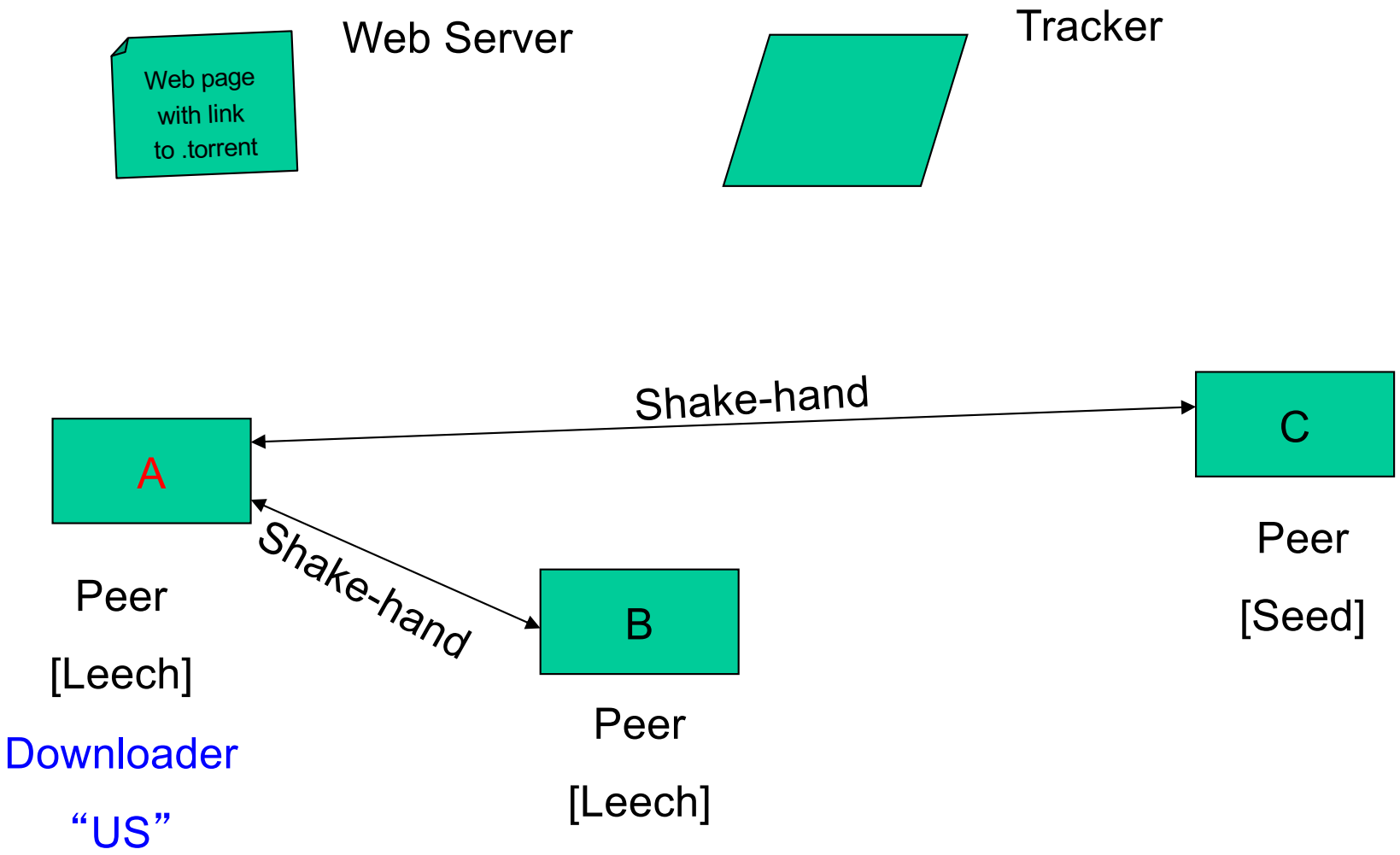
BitTorrent: Overall Architecture



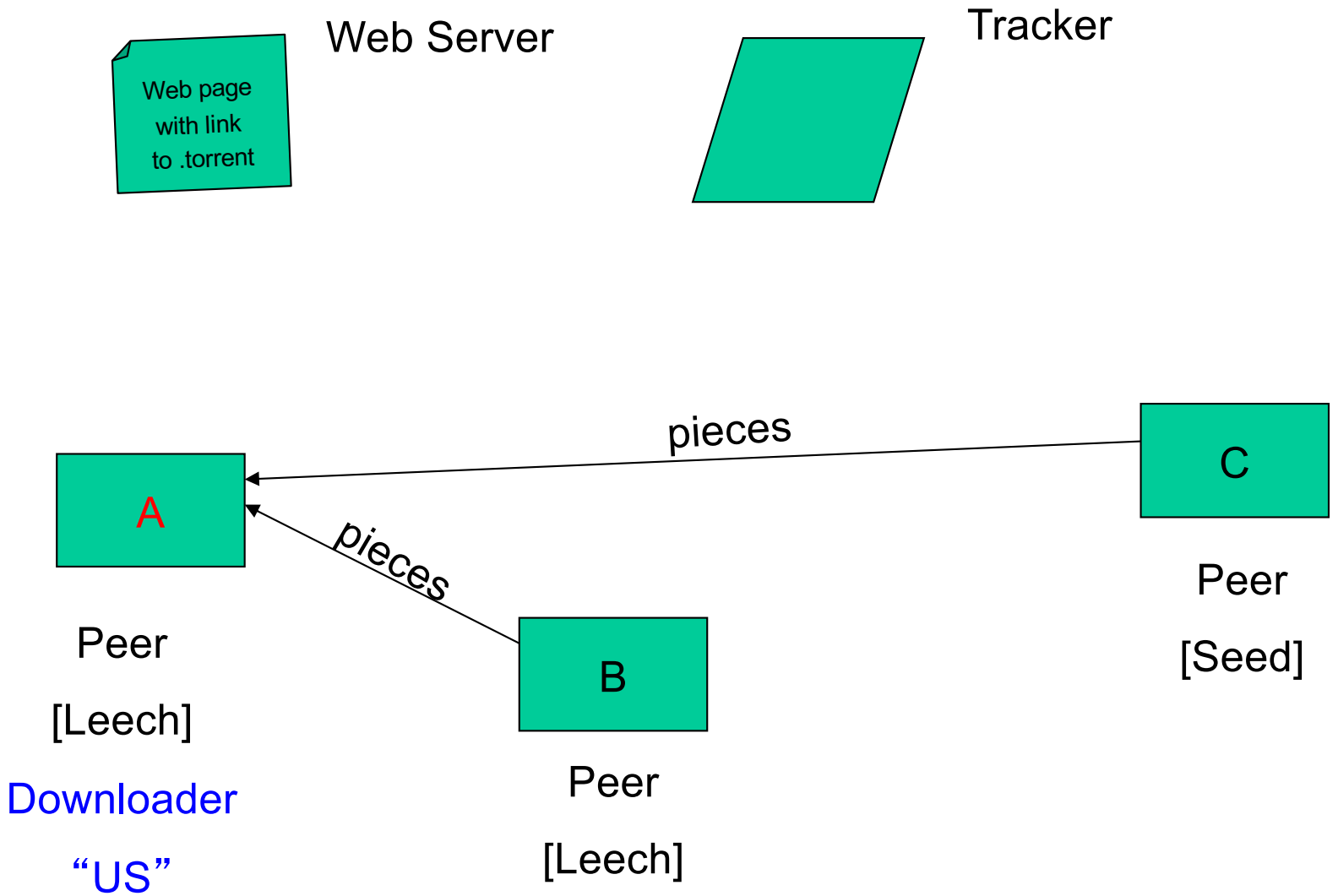
BitTorrent: Overall Architecture



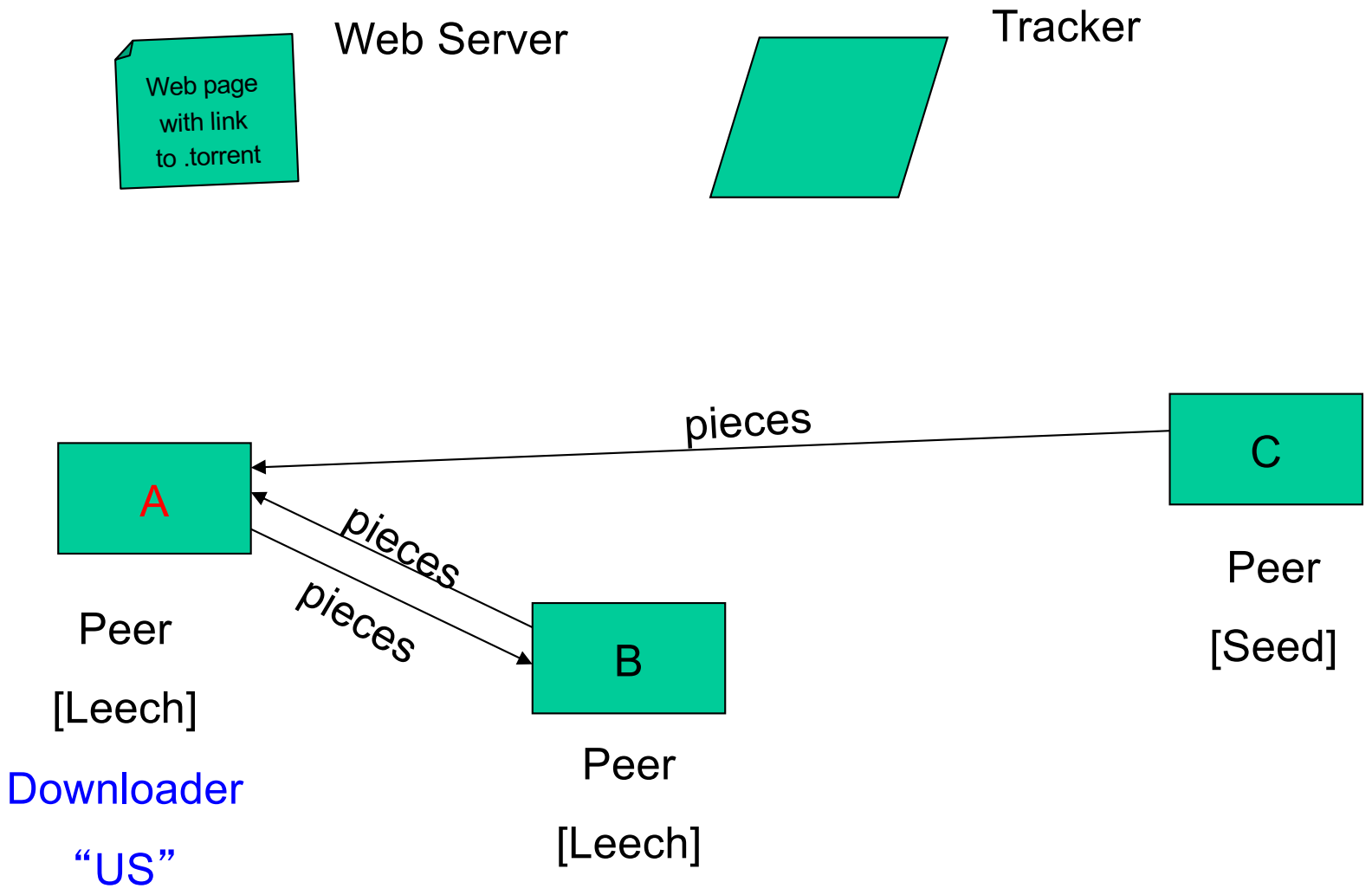
BitTorrent: Overall Architecture



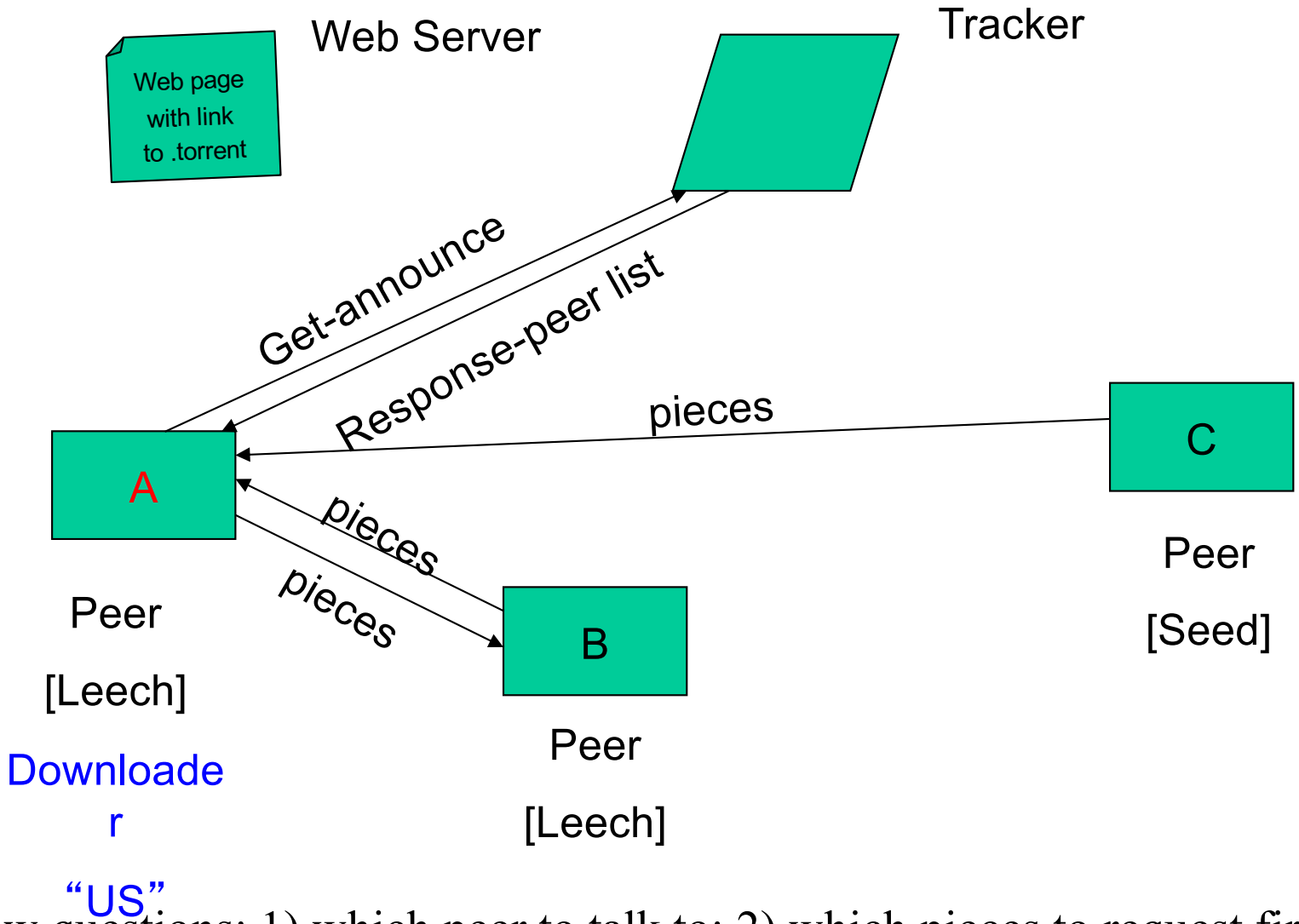
BitTorrent: Overall Architecture



BitTorrent: Overall Architecture



BitTorrent: Overall Architecture



A few questions: 1) which peer to talk to; 2) which pieces to request first; 3) how to motivate other peers to contribute

Free-Riding Problem in P2P Networks

- Vast majority of users are free-riders
  - Most share no files and answer no queries
  - Others limit # of connections or upload speed
- A few "peers" essentially act as servers
  - A few individuals contributing to the public good
  - Making them hubs that basically act as a server

- BitTorrent prevent free riding
  - Allow the fastest peers to download from you
  - Occasionally let some free loaders download

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

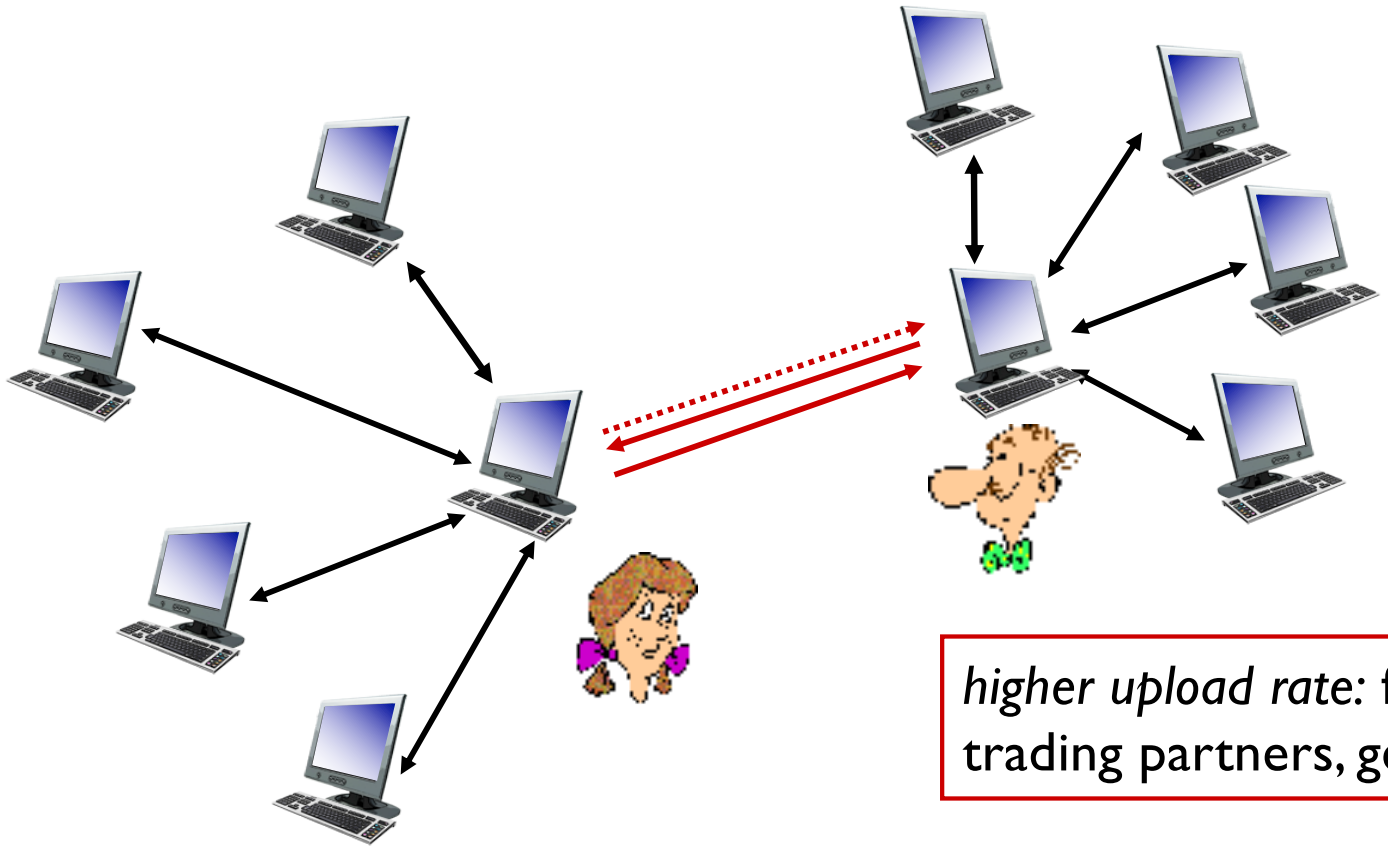
- Alice sends chunks to those four peers currently sending her chunks *at highest rate*
  - other peers are choked by Alice (do not receive chunks from her)
  - re-evaluate top 4 every10 secs
- every 30 secs: randomly select another peer, starts sending chunks
  - "optimistically unchoke" this peer
  - 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 !

Chapter 2: summary

our study of network apps now complete!

- application architectures
  - client-server
  - P2P
- application service requirements:
  - reliability, throughput, delay, security
- Internet transport service model
  - connection-oriented, reliable: TCP
  - unreliable, datagrams: UDP

- specific protocols:
  - HTTP
  - DNS
  - P2P: BitTorrent
  - SMTP, POP, IMAP

Questions?