PES UNIVERSITY
CELEBRATING 50 YEARS

COMPUTER NETWORKS

TEAM NETWORKS

Department of Computer Science and Engineering

COMPUTER NETWORKS

Application Layer

Department of Computer Science and Engineering

Unit – 2 Application Layer

2.3 The Domain Name System

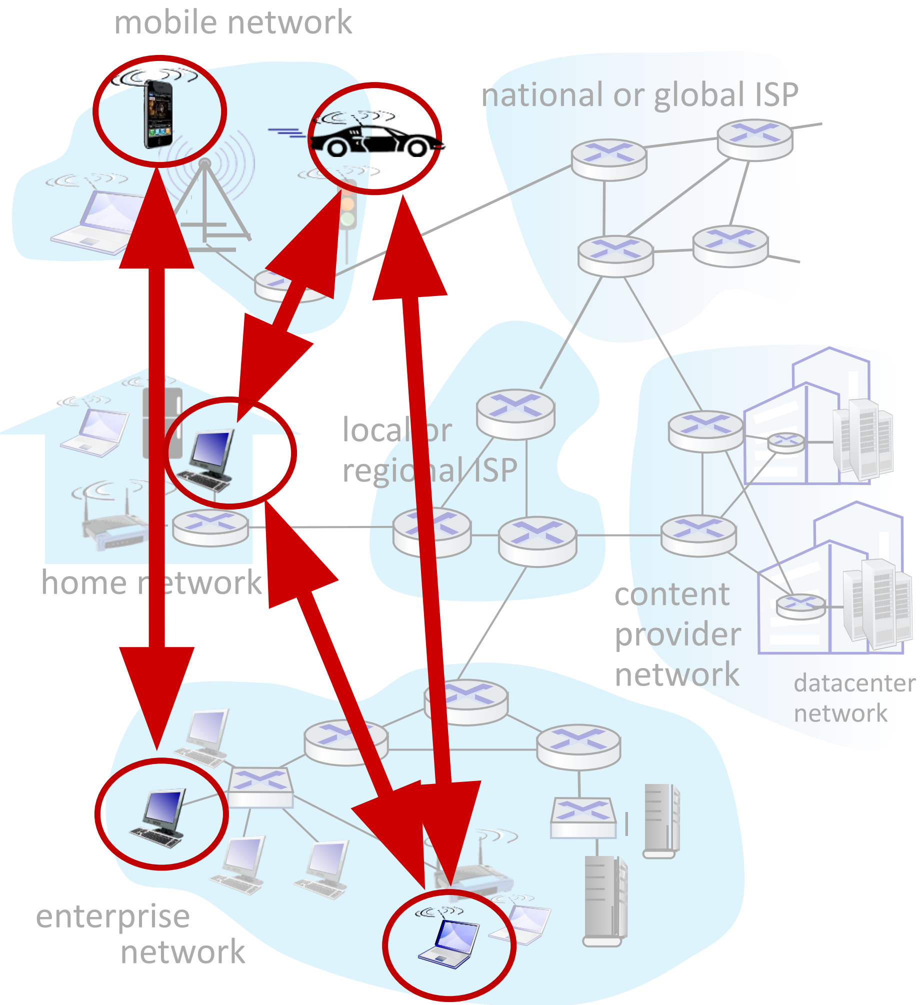
2.4 P2P Applications

2.5 Socket Programming with TCP & UDP

2.6 Other Application Layer Protocols

Peer-to-peer (P2P) architecture

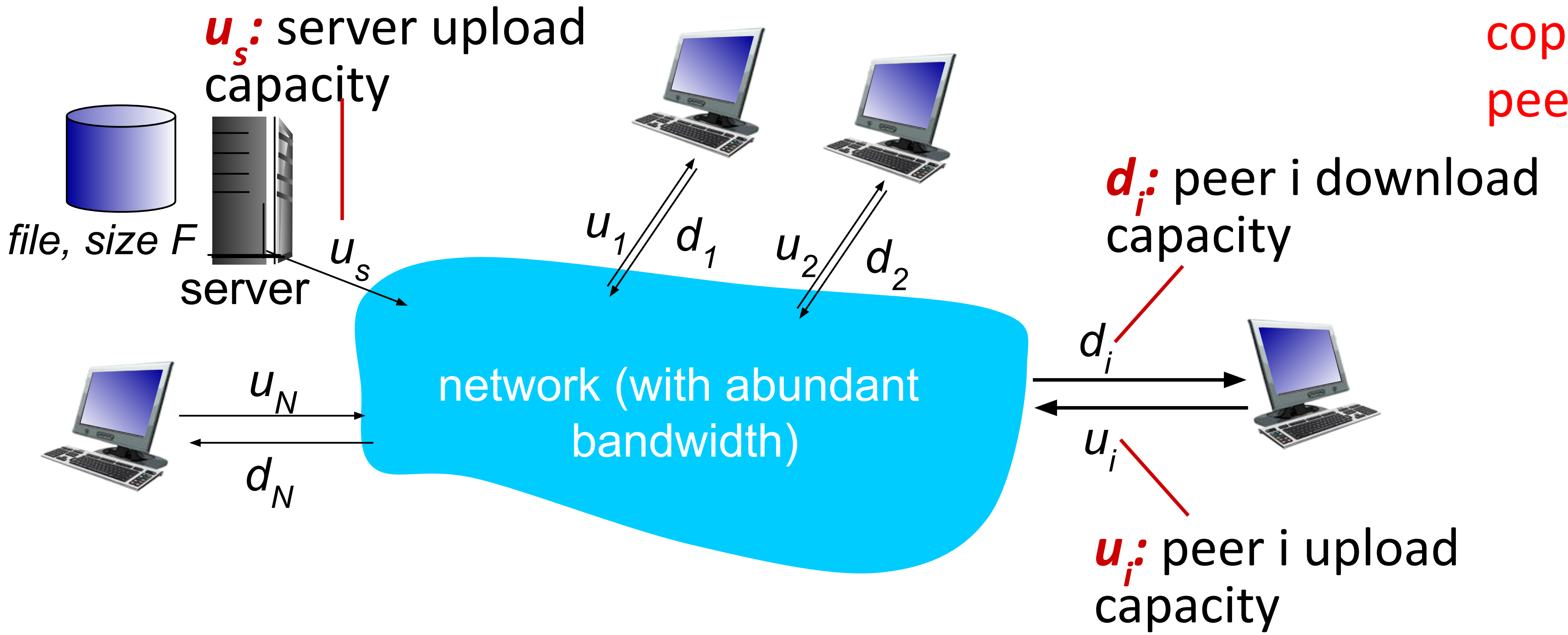
- *no* always-on server
- arbitrary end systems directly communicate
- peers request service from other peers, provide service in return to other peers
  - *self scalability* – new peers bring new service capacity, and new service demands
- peers are intermittently connected and change IP addresses
  - complex management
- examples: P2P file sharing (BitTorrent), media streaming (Spotify), VoIP (Skype)



File distribution: client-server vs P2P

Q: how much time to distribute file (size *F*) from one server to *N peers*?

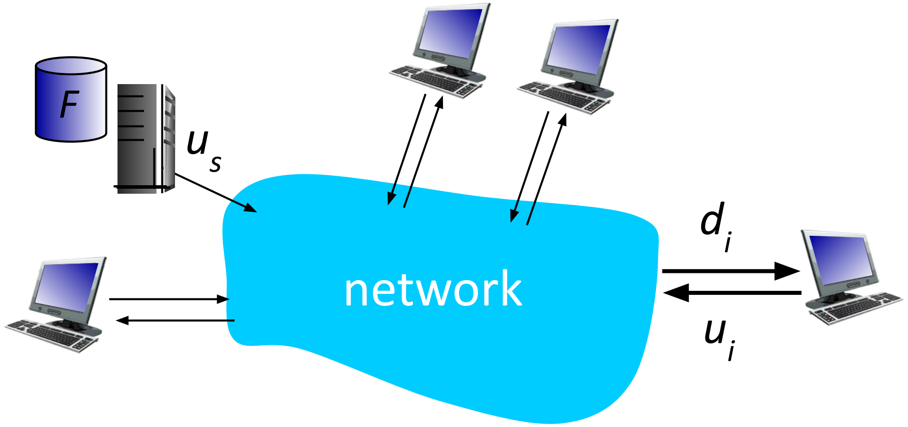
- peer upload/download capacity is limited resource


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
d_i
u_i
u_N
d_N
network (with abundant bandwidth)
u_i: peer i upload capacity

The **distribution time** is the time it takes to get a copy of the file to all *N* peers.

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 = i.e., d_{min} = min {d1, dp,……, dN }
  - min client download time: F/d_{min}



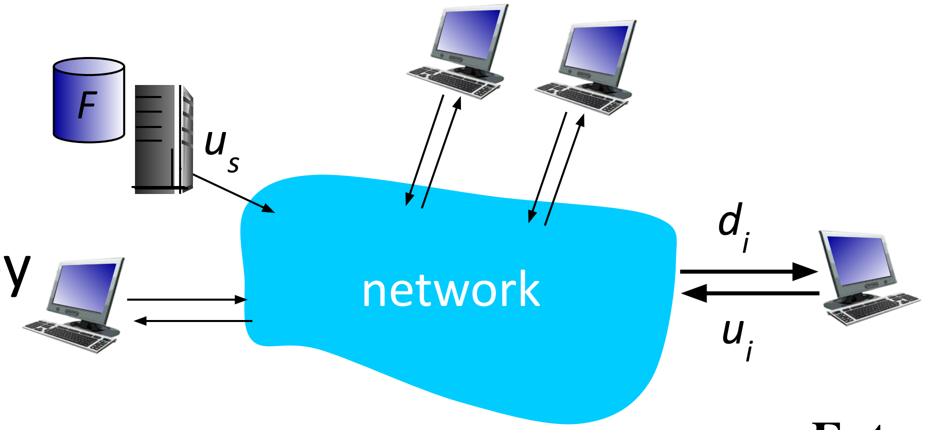
time to distribute F to N clients using client-server approach

$$D_{c-s} > max\{NF/u_{s,},F/d_{min}\}$$

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:* as aggregate must download *NF* bits
  - max upload rate (limiting max download rate) is $u_s + \Sigma u_i$

Total upload capacity of the system as a whole

Eqtn - provides a lower bound for the minimum distribution time for the P2P architecture.

time to distribute F to N clients using P2P approach

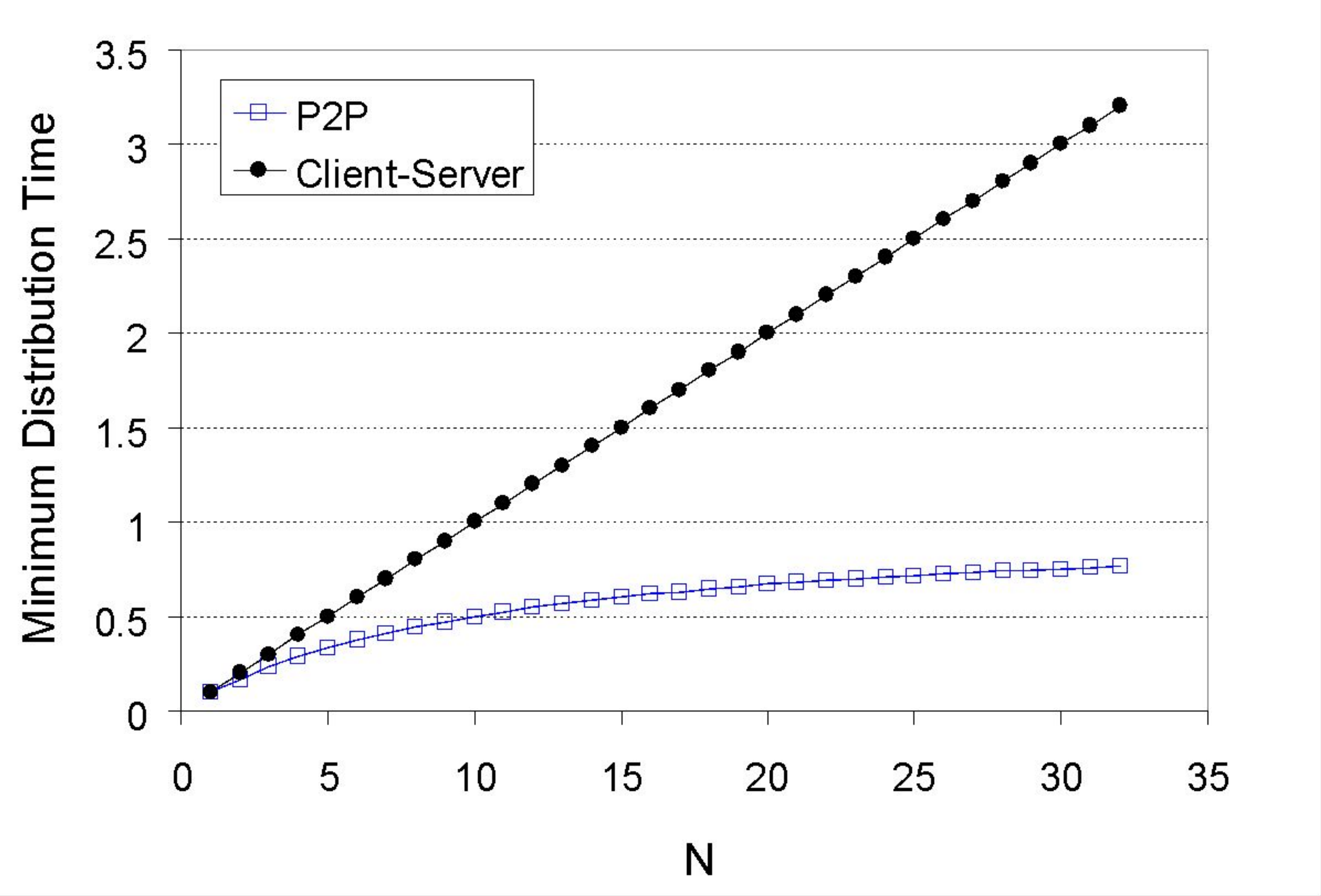
$$D_{P2P} > max\{F/u_s, F/d_{min}, NF/(u_s + \Sigma u_i)\}$$

increases linearly in *N*

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

Client-server vs. P2P: example

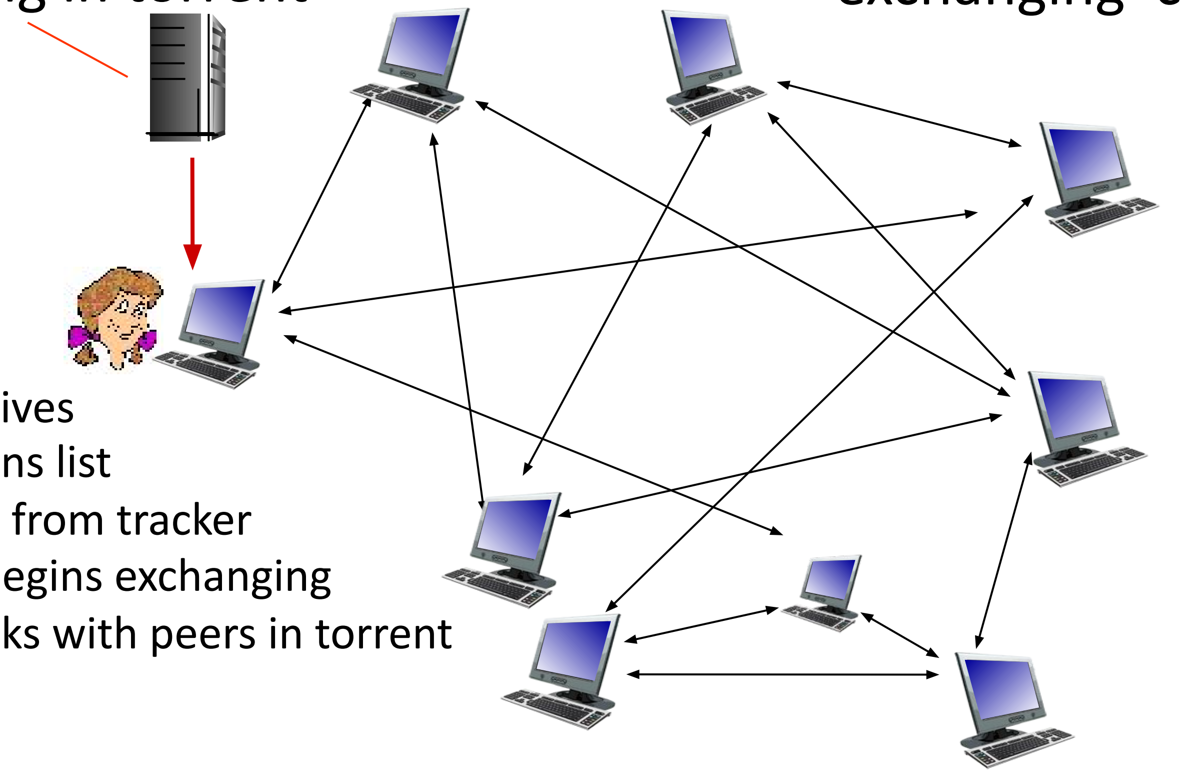
Client (all peers) upload rate = *u*, *F/u* = 1 hour, $u_s = 10u$, $d_{min} \geq u_s$



- A peer can transmit the entire file in one hour.
- The server transmission rate is 10 times the peer upload rate.
- Peer download rates are set large enough so as not to have an effect.

P2P file distribution: BitTorrent

- file divided into 256Kb chunks
- peers in torrent send/receive file chunks



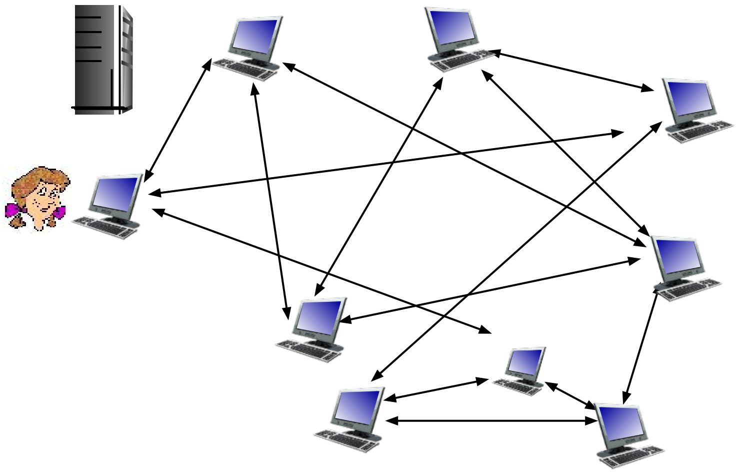
tracker: tracks peers participating in torrent

torrent: group of peers exchanging chunks of a file

Alice arrives
… obtains list
of peers from tracker
… and begins exchanging
file chunks with peers in torrent

P2P file distribution: BitTorrent

- 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 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 every 10 secs
- every 30 secs: randomly select another peer, starts sending chunks
  - "**optimistically unchoke**" this peer
  - newly chosen peer may join top 4

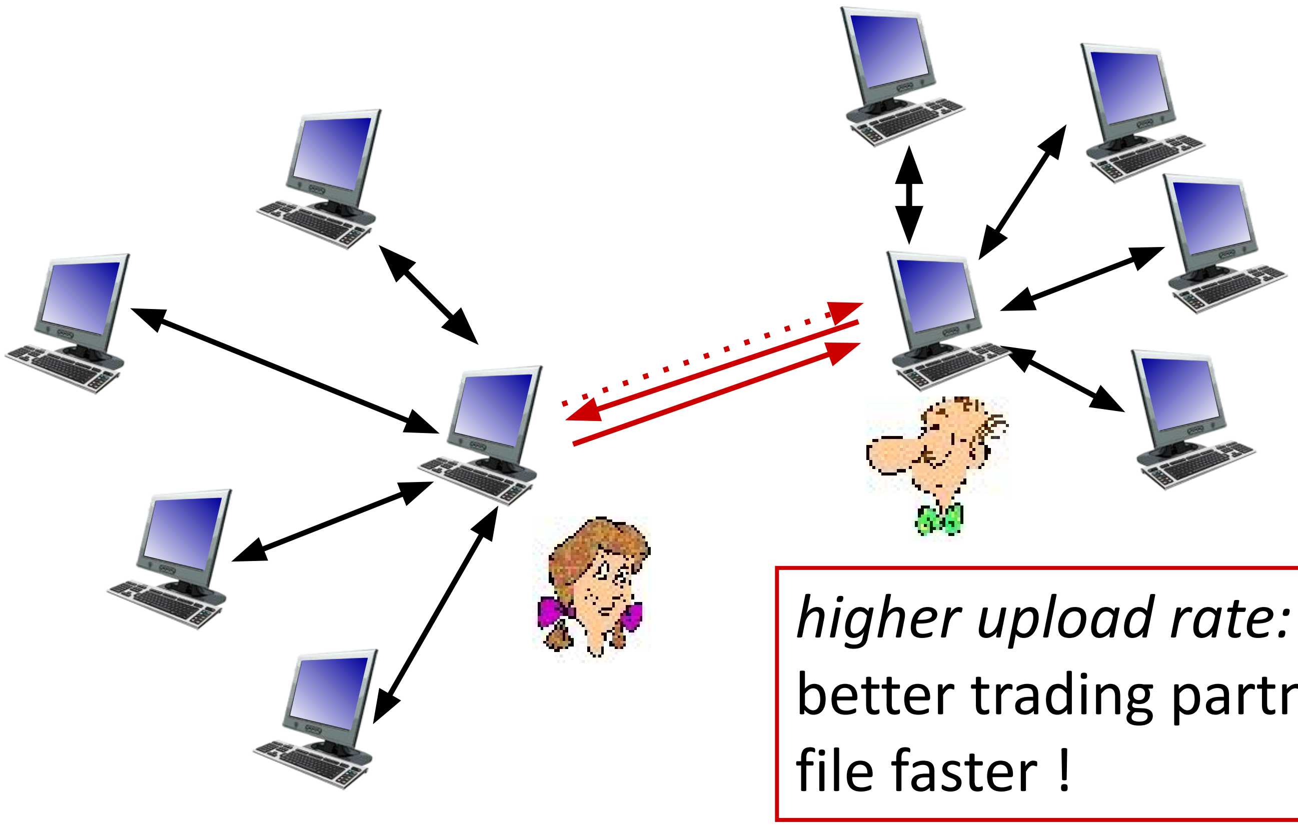
BitTorrent: tit-for-tat

(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

Pieces (mini-chunks), pipelining, random first selection, endgame mode, and anti-snubbing



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

Suggested Readings

- BitTorrent (BTT) White Paper – https://www.bittorrent.com/btt/btt-docs/BitTorrent_(BTT)_White_Paper_v0.8.7_Feb_2019.pdf
- Peer-to-peer networking with BitTorrent – http://web.cs.ucla.edu/classes/cs217/05BitTorrent.pdf
- Torrents Explained: How BitTorrent Works – https://youtu.be/urzQeD7ftbI

Thank You
For Your Attention

THANK YOU

TEAM NETWORKS

Department of Computer Science and Engineering