

Application Layer: P2P File Distribution

EECS 3214

Slides courtesy of J.F. Kurose and K.W. Ross, All Rights Reserved

29-Jan-18

1-1

Chapter 2: outline

2.1 principles of network applications

2.2 Web and HTTP

2.3 electronic mail

- SMTP, POP3, IMAP

2.4 DNS

2.5 P2P applications

2.6 video streaming and content distribution networks (later)

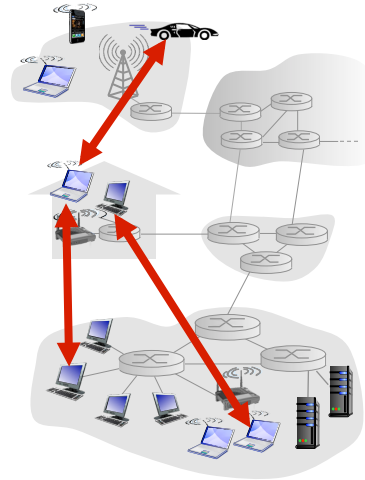
Application Layer 2-2

Pure P2P architecture

- no always-on server
- arbitrary end systems directly communicate
- peers are intermittently connected and change IP addresses

examples:

- file distribution (BitTorrent)
- Streaming (Xunlei KanKan)
- VoIP (Skype)

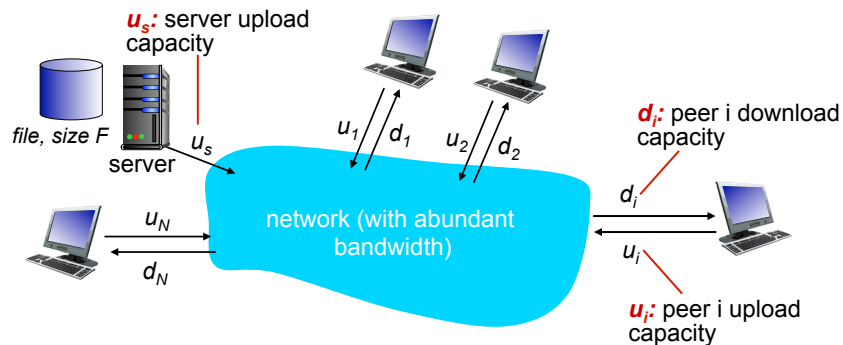


Application Layer 2-3

File distribution: client-server vs P2P

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

- peer upload/download capacity: limited resource (bottlenecks)



Application Layer 2-4

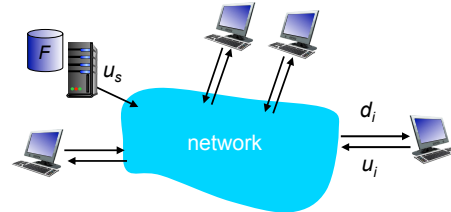
File distribution time: client-server

- **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

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

increases linearly in N

Application Layer 2-5

File distribution time: P2P

- **server transmission:** must upload at least one copy

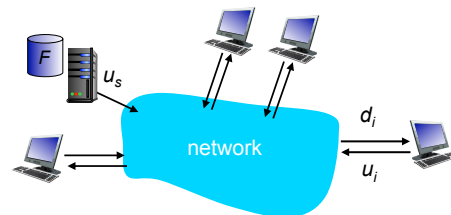
- time to send one copy: F/u_s

- **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 $u_s + \sum u_i$



time to distribute F
to N clients using
P2P approach

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

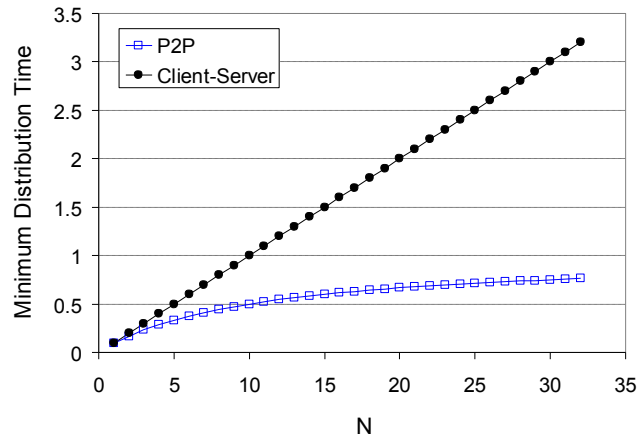
increases linearly in N ...

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

Application Layer 2-6

Client-server vs. P2P: example

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



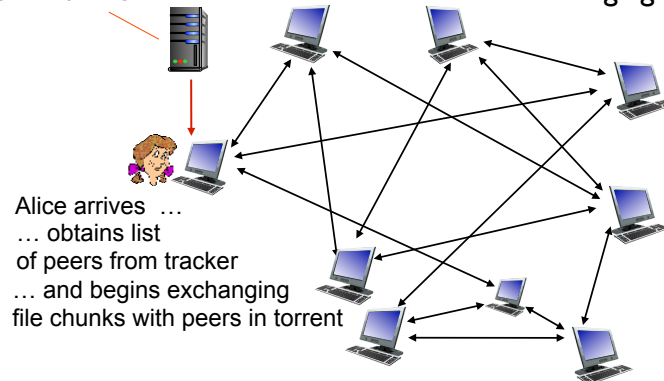
Application Layer 2-7

P2P file distribution: BitTorrent

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

tracker: tracks peers participating in torrent

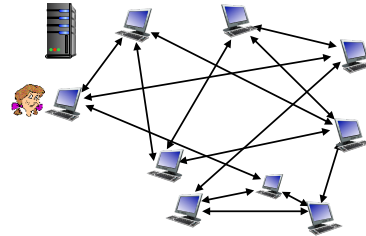
torrent: group of peers exchanging chunks of a file



Application Layer 2-8

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



Application Layer 2-9

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

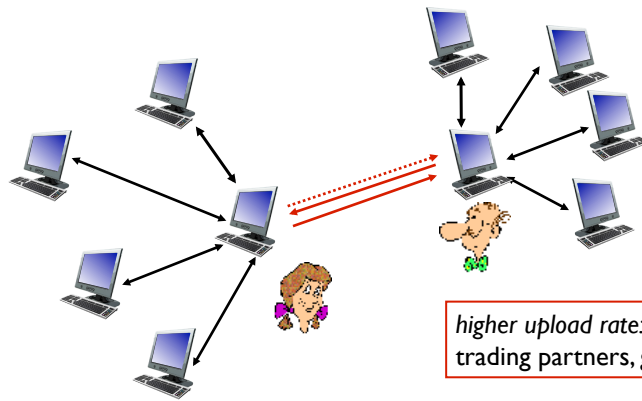
- Alice sends chunks to those four peers currently sending her chunks *at highest rate*
 - other peers are choked by Alice (do not receive chunks from her)
 - re-evaluate top 4 every 10 secs
- every 30 secs: randomly select another peer, starts sending chunks
 - “optimistically unchoke” this peer
 - newly chosen peer may join top 4

Application Layer 2-10

BitTorrent: tit-for-tat

Ultimate goal: find partners with compatible rates.

- (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



Application Layer 2-11

Chapter 2: summary

- application architectures
 - client-server
 - P2P
- application service requirements:
 - reliability, bandwidth, delay
- Internet transport service model
 - connection-oriented, reliable: TCP
 - unreliable, datagrams: UDP
- specific protocols:
 - HTTP
 - SMTP, POP, IMAP
 - DNS
 - P2P: BitTorrent
- video streaming, CDNs (later)
- socket programming: TCP, UDP sockets

Application Layer 2-12

Chapter 2: summary

most importantly: learned about protocols!

- typical request/reply message exchange:
 - client requests info or service
 - server responds with data, status code
 - message formats:
 - *headers*: fields giving info about data
 - *data*: info(payload) being communicated
- important themes:*
 - centralized vs. decentralized
 - stateless vs. stateful
 - reliable vs. unreliable message transfer
 - “complexity at network edge”

Application Layer 2-13