Application Layer Protocols -2

A/PROF. DUY NGO

Learning Objectives

2.5 P2P applications

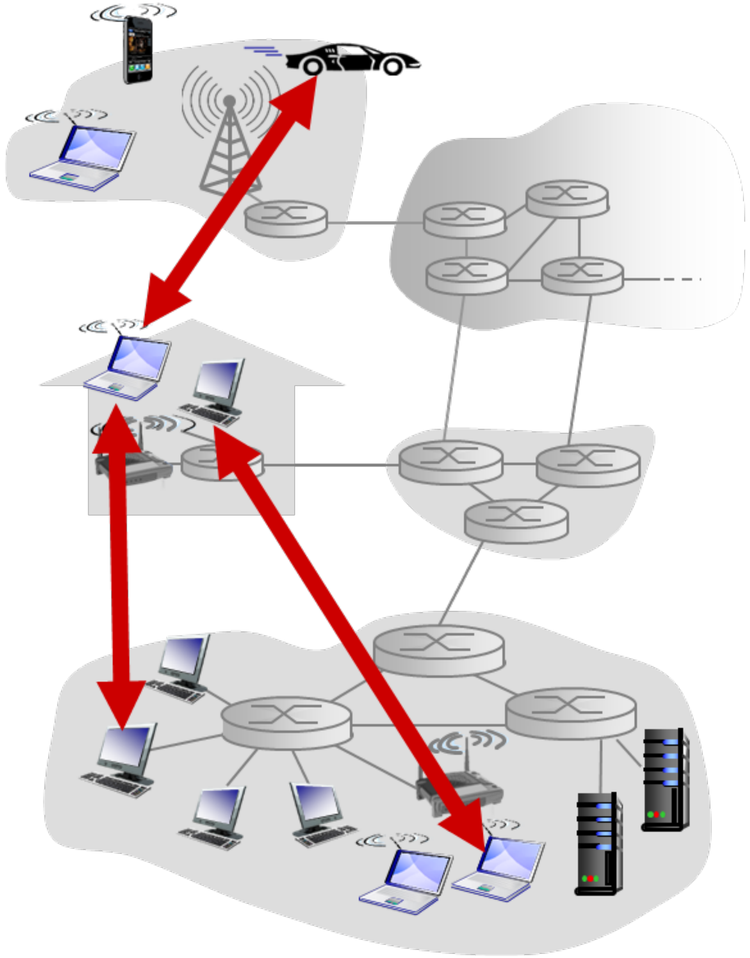
2.6 video streaming and content distribution networks

Pure P2P Architecture

- **Not required a** always-on server
- arbitrary end systems directly communicate
- peers are intermittently connected and change IP addresses

examples:

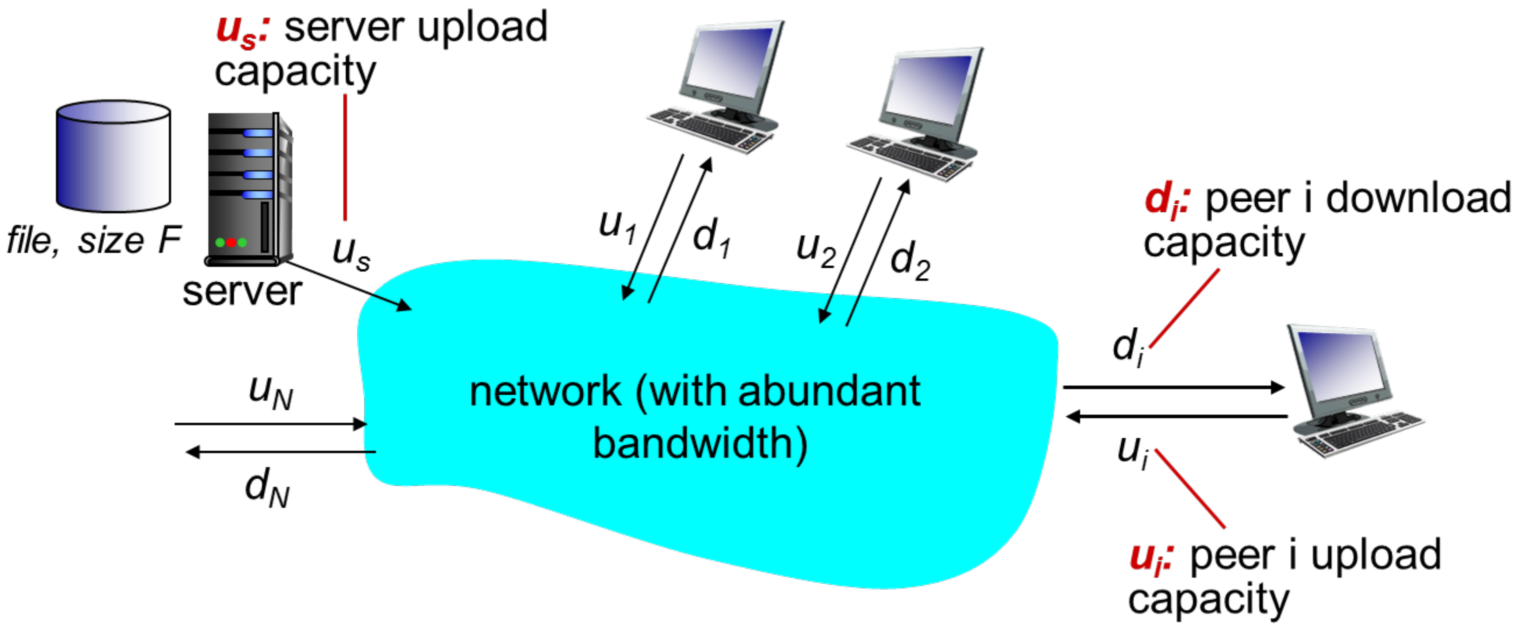
- file distribution (BitTorrent)
- Streaming (KanKan)
- VoIP (Skype)



File Distribution: Client-Server vs P2P

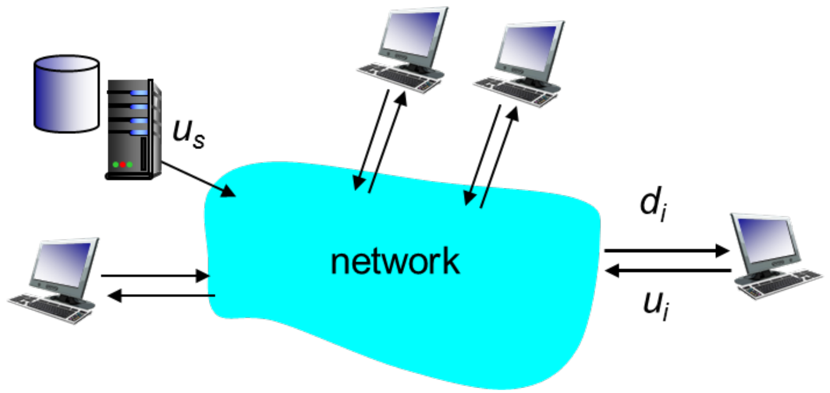
Question: how much time to distribute file (size *F*) from one server to *N* **peers**?

- peer upload/download capacity is limited resource



File Distribution Time: Client-Server

- **server transmission:** must sequentially send (upload) *N* file copies:
  - time to send one copy: $\frac{F}{u_s}$
  - time to send *N* copies: $\frac{NF}{u_s}$
- **client:** each client must download file copy
  - d_{min} = min client download rate
  - min client download time: $\frac{F}{d_{min}}$



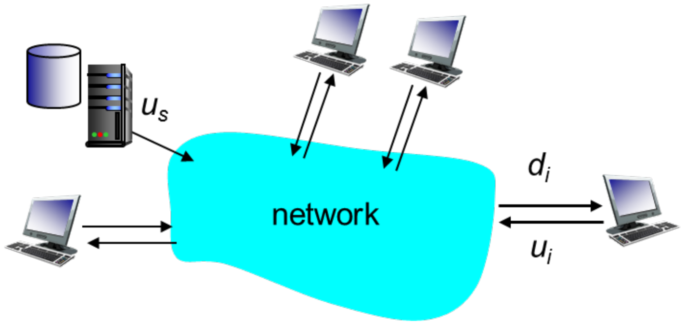
time to distribute F to N clients using client-server approach

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

increases linearly in N

File Distribution Time: P2P

- **server transmission:** must upload at least one copy
  - time to send one copy: $\frac{F}{u_s}$
- **client:** each client must download file copy
  - min client download time: $\frac{F}{d_{min}}$
- **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

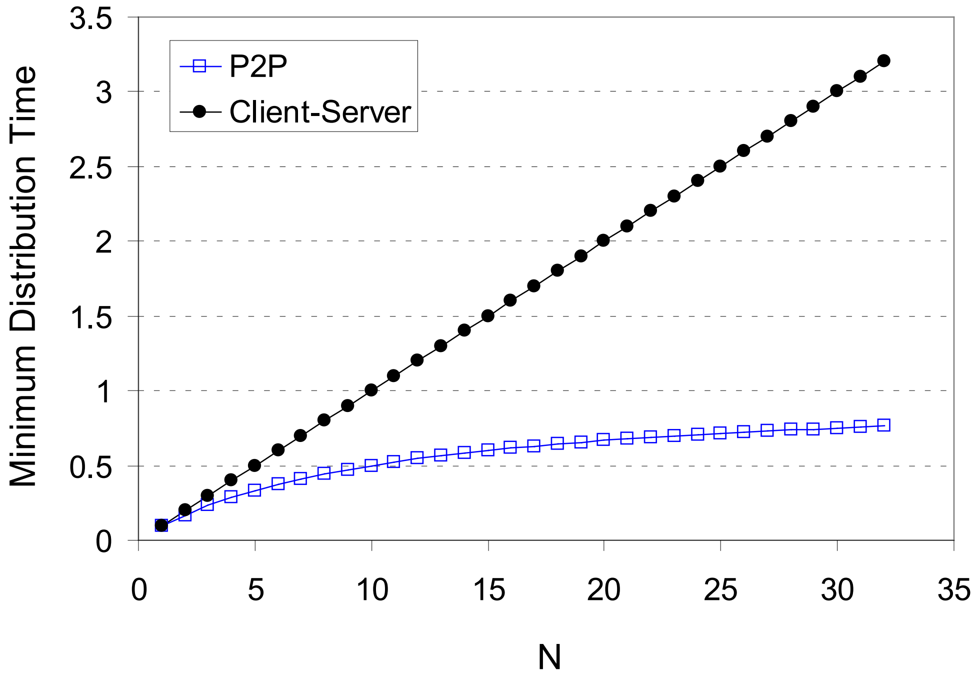
$$D_{P2P} \geq \max\{F/u_s, F/d_{min}, NF/(u_s + \Sigma u_i)\}$$

increases linearly in *N* …

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

Client-Server vs P2P: Example

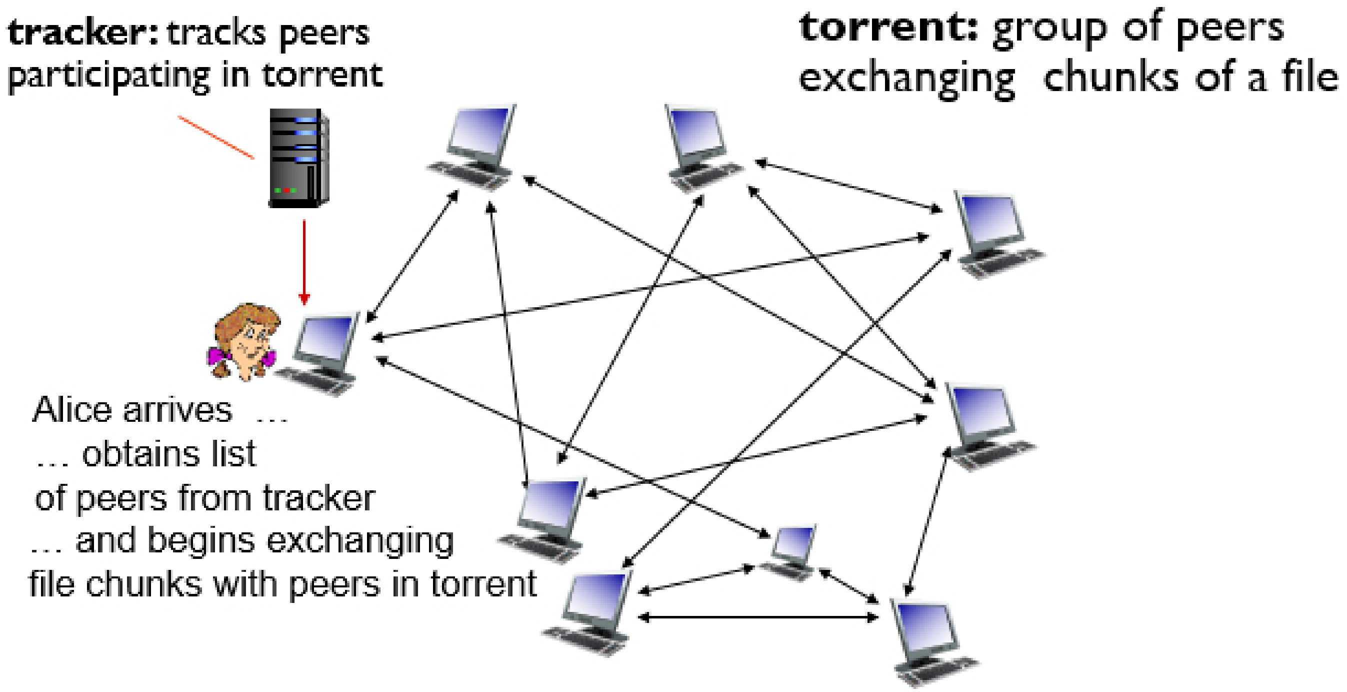
client upload rate = u, F/u = 1 hour, $u_s = 10u$, $d_{min} \geq u_s$



P2P File Distribution: BitTorrent (1 of 2)

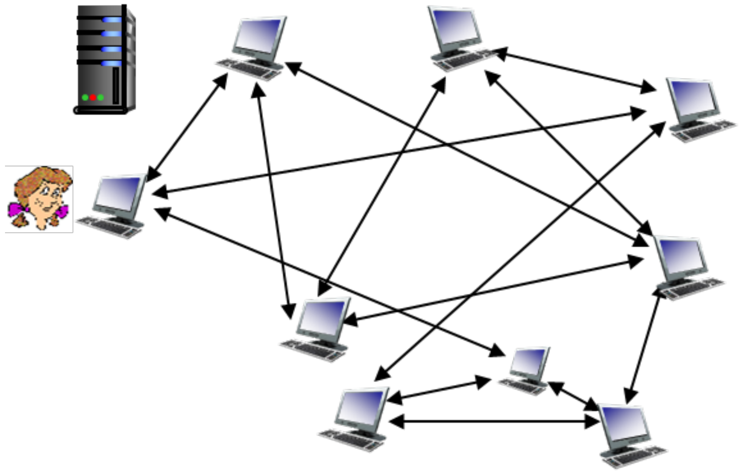
file divided into 256Kb chunks

peers in torrent send/receive file chunks



P2P File Distribution: BitTorrent (2 of 2)

- 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

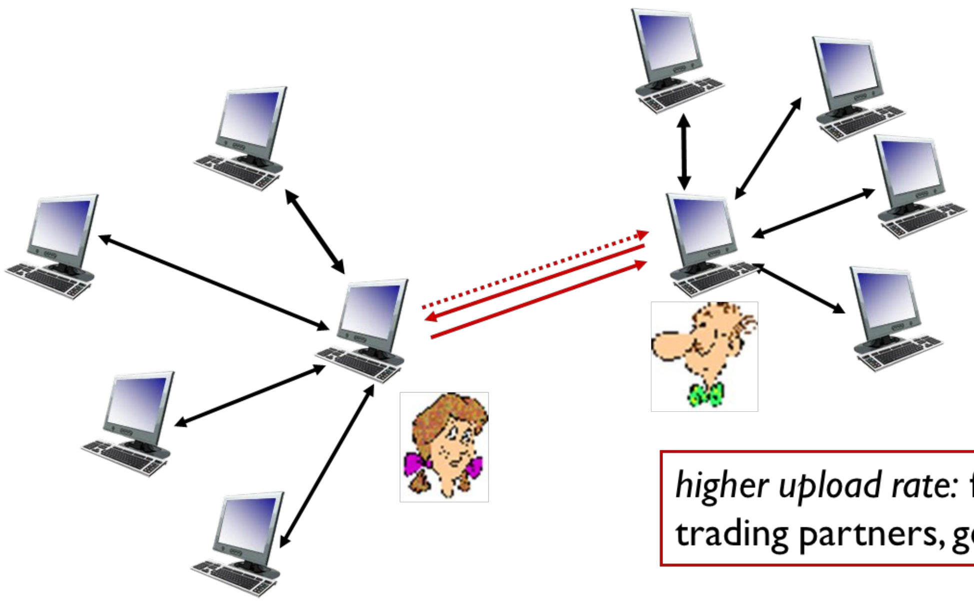
- Alice sends chunks to those four peers currently sending her chunks **at highest rate**
  - other peers are choked by Alice (do not receive chunks from her)
  - re-evaluate top 4 every10 secs
- every 30 secs: randomly select another peer, starts sending chunks
  - “optimistically unchoke” this peer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

Video Streaming and CDNs: Context

- video traffic: major consumer of Internet bandwidth
  - Netflix, YouTube: 37%, 16% of downstream residential ISP traffic
  - ~1B YouTube users, ~75M Netflix users
- challenge: scale - how to reach ~1B users?
  - single mega-video server won't work (why?)
- challenge: heterogeneity
  - different users have different capabilities (e.g., wired versus mobile; bandwidth rich versus bandwidth poor)
- **solution:** distributed, application-level infrastructure

Multimedia: Video (1 of 2)

video: sequence of images displayed at constant rate

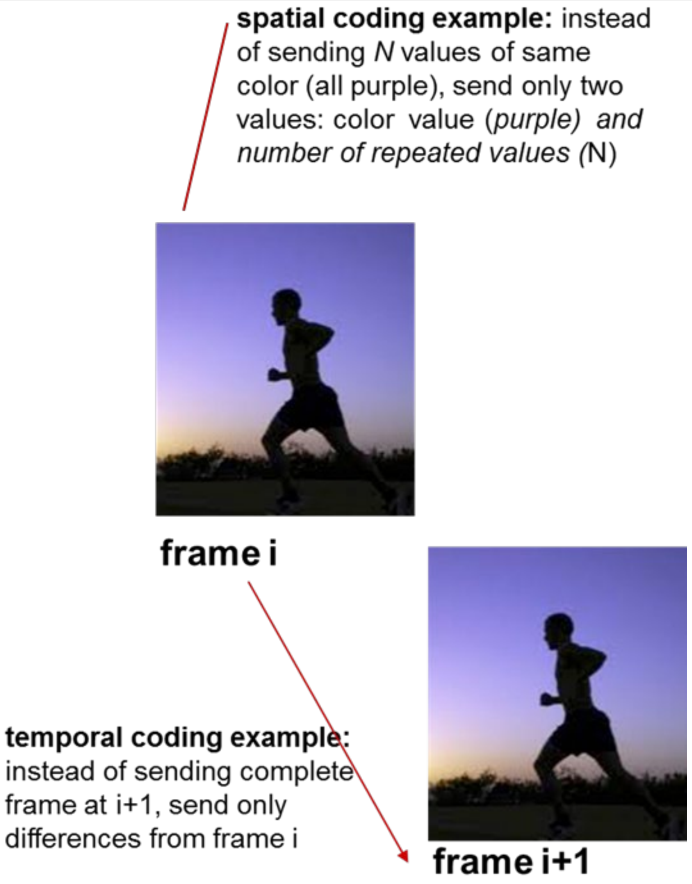
- e.g., 24 images/sec

digital image: array of pixels

- each pixel represented by bits

coding: use redundancy **within** and **between** images to decrease # bits used to encode image

- spatial (within image)
- temporal (from one image to next)



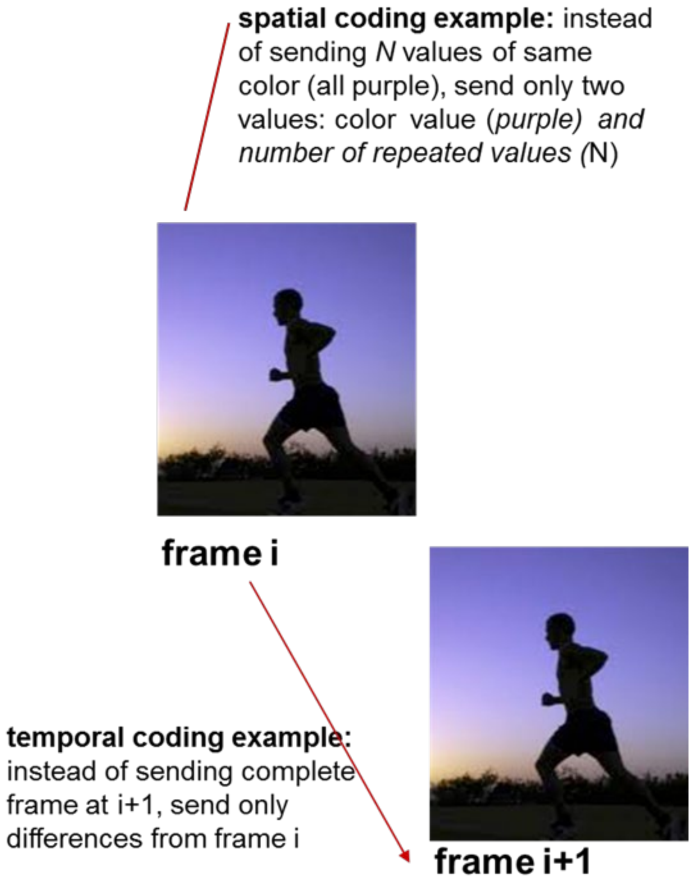
Multimedia: Video (2 of 2)

CBR: (constant bit rate): video encoding rate fixed

VBR: (variable bit rate): video encoding rate changes as amount of spatial, temporal coding changes

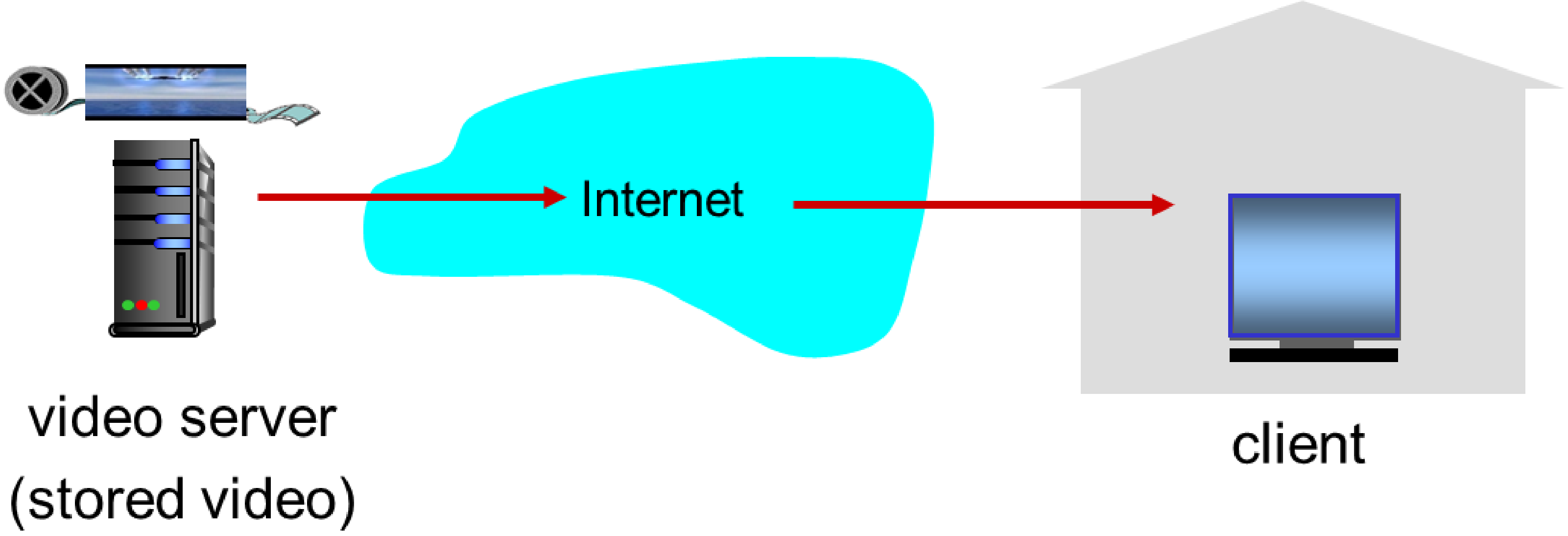
examples:

- MPEG1 (CD-ROM) 1.5 Mbps
- MPEG2 (DVD) 3-6 Mbps
- MPEG4 (often used in Internet, < 1 Mbps)



Streaming Stored Video

simple scenario:



Streaming Multimedia: DASH (1 of 2)

DASH: Dynamic, **A**daptive **S**treaming over **H**TTP

server:

- divides video file into multiple chunks
- each chunk stored, encoded at different rates
- **manifest file:** provides URLs for different chunks

client:

- periodically measures server-to-client bandwidth
- consulting manifest, requests one chunk at a time
  - chooses maximum coding rate sustainable given current bandwidth
  - can choose different coding rates at different points in time (depending on available bandwidth at time)

Streaming Multimedia: DASH (2 of 2)

“intelligence” at client: client determines

- **when** to request chunk (so that buffer starvation, or overflow does not occur)
- **what encoding rate** to request (higher quality when more bandwidth available)
- **where** to request chunk (can request from URL server that is “close” to client or has high available bandwidth)

Content Distribution Networks (1 of 2)

challenge: how to stream content (selected from millions of videos) to hundreds of thousands of **simultaneous** users?

option 1: single, large "mega-server"
- single point of failure
- point of network congestion
- long path to distant clients
- multiple copies of video sent over outgoing link

....quite simply: this solution **doesn't scale**

Content Distribution Networks (2 of 2)

option 2: store/serve multiple copies of videos at multiple geographically distributed sites **(CDN)**

- **enter deep:** push CDN servers deep into many access networks
  - close to users
  - used by Akamai, 1700 locations
- **bring home:** smaller number (10's) of larger clusters in POPs near (but not within) access networks
  - used by Limelight

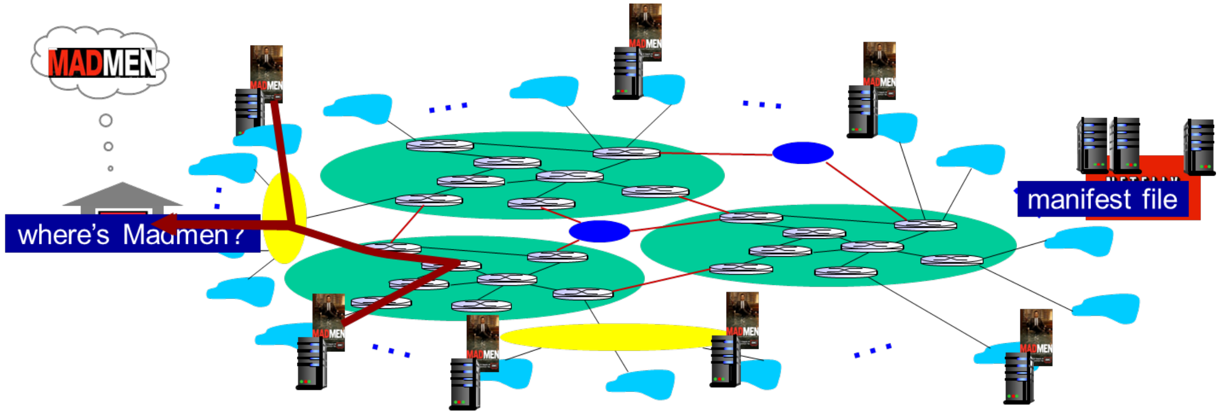
Content Distribution Networks (CDNs) (1 of 2)

CDN: stores copies of content at CDN nodes

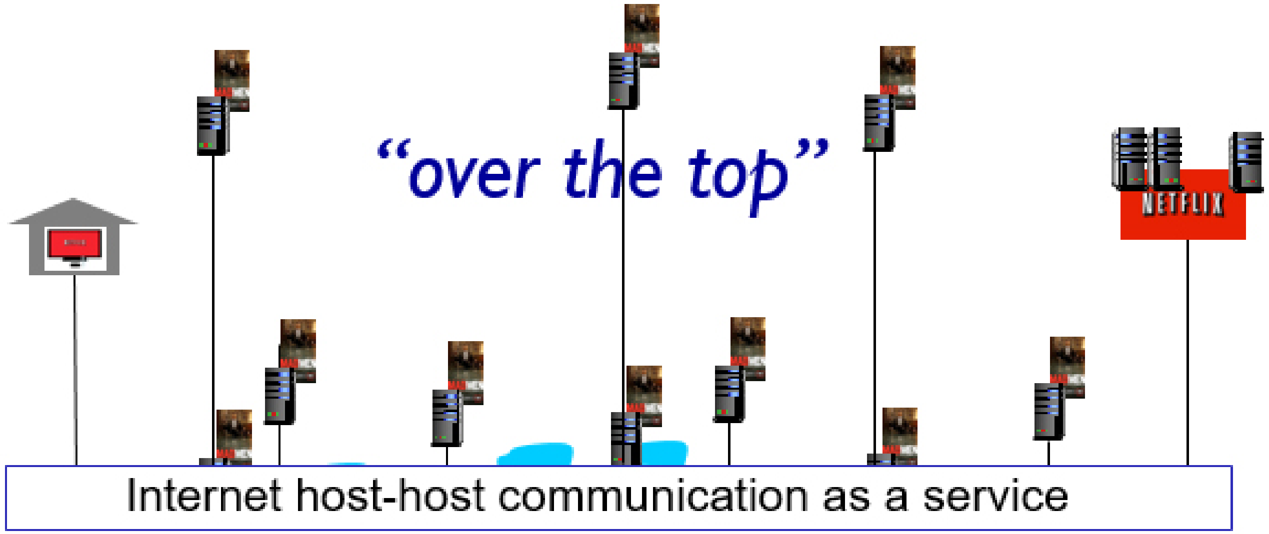
- e.g. Netflix stores copies of MadMen

subscriber requests content from CDN

- directed to nearby copy, retrieves content
- may choose different copy if network path congested



Content Distribution Networks (CDNs) (2 of 2)



OTT challenges: coping with a congested Internet

- from which CDN node to retrieve content?
- viewer behavior in presence of congestion?
- what content to place in which CDN node?

CDN Content Access: A Closer Look

Bob (client) requests video http://netcinema.com/6Y7B23V

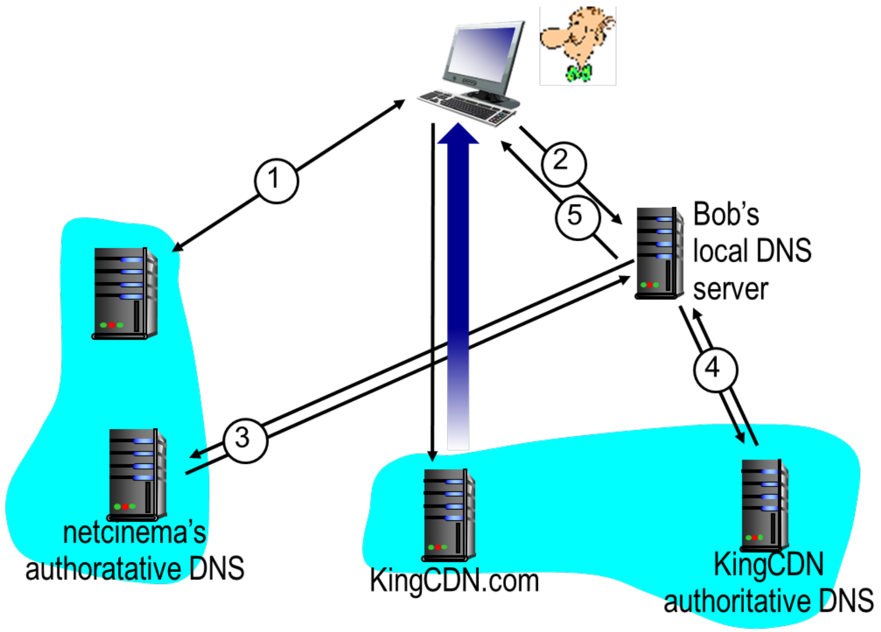
– video stored in CDN at http://KingCDN.com/NetC6y&B23V

1. Bob gets URL for video http://netcinema.com/6Y7B23V from netcinema.com web page

2. resolve http://netcinema.com/6Y7B23V via Bob's local DNS

3. netcinema's DNS returns URL http://KingCDN.com/NetC6y&B23V

4&5. Resolve http://KingCDN.com/NetC6y&B23 via KingCDN's authoritative DNS, which returns IP address of KingCDN server with video



Case Study: Netflix