# COMP 445
# Data Communications & Computer networks
# Winter 2022

# Application Layer

- ✓ Principles of network applications
- ✓ Web and HTTP
- ✓ Electronic mail
- ✓ DNS
- ✓ P2P applications
- ✓ Video streaming and CDN
- ✓ Sockets

# Application Layer – Part 3

- ✓ P2P Applications
  - ✓ Characteristics
  - ✓ File distribution with BitTorrent
- ✓ Video streaming and Content Delivery Networks
  - ✓ Internet video
  - ✓ HTTP Streaming
  - ✓ CDN

# Learning objectives

- To quantify the differences between file distribution using client-server vs. P2P architectures
- To describe the operation of BitTorrent as on P2P application
- To explain how the video streaming services are implemented and the application-layer protocols involved
- To describe the way multimedia content can be distributed using Content Delivery Networks.

# Application Layer – Part 3

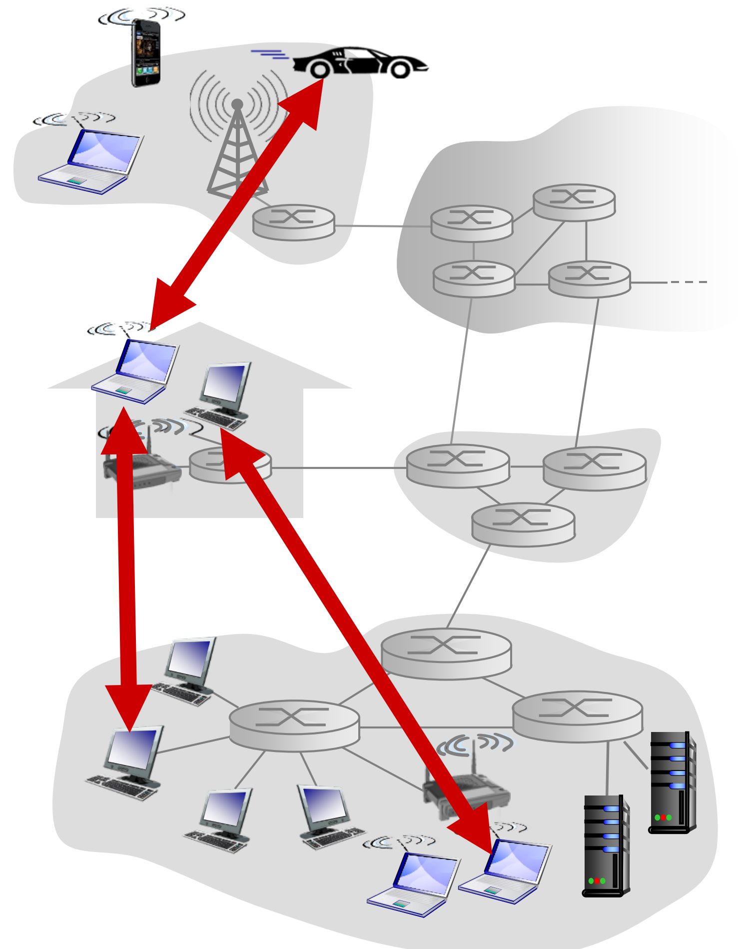
- P2P Applications
  - Characteristics
  - File distribution with BitTorrent
- Video streaming and Content Delivery Networks
  - Internet video
  - HTTP Streaming
  - CDN

# Pure P2P architecture

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

*examples:*

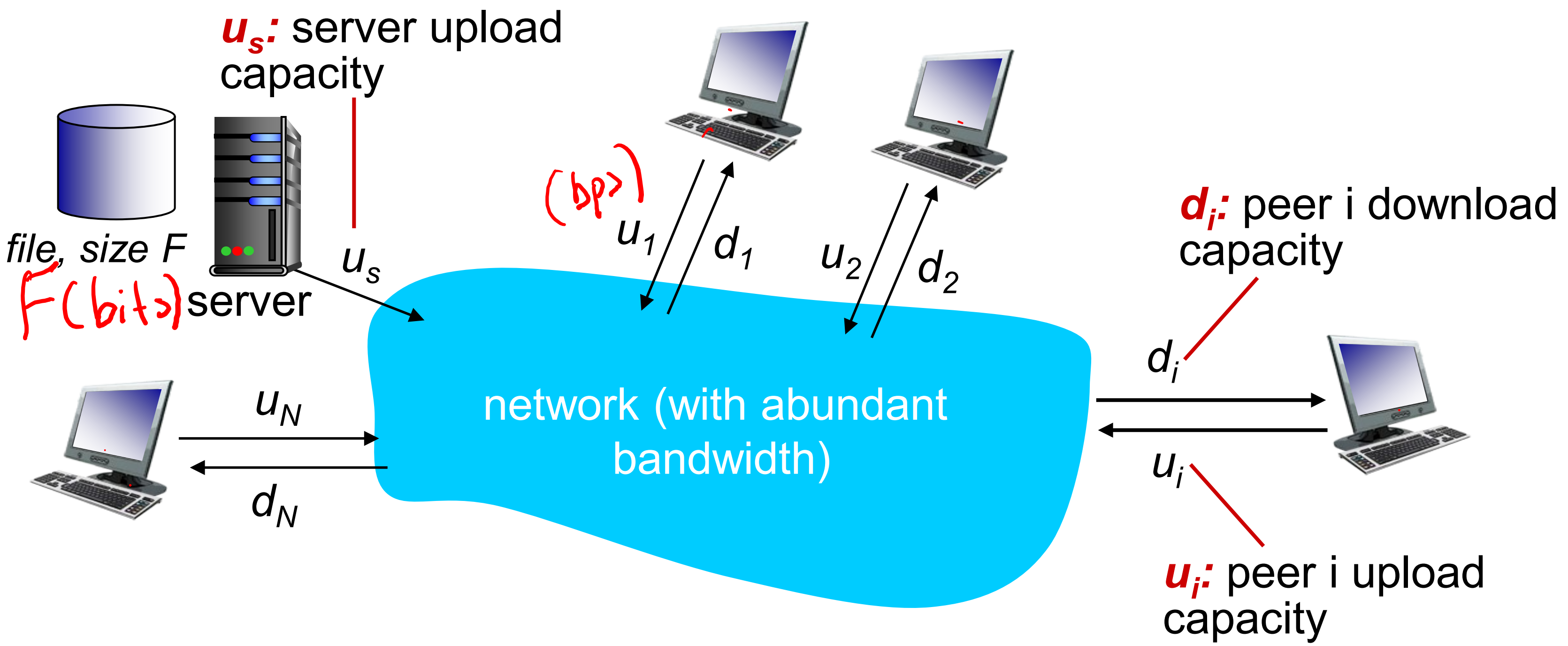
- file distribution (BitTorrent)
- Streaming (KanKan)
- VoIP (Skype)



# File distribution: client-server vs P2P

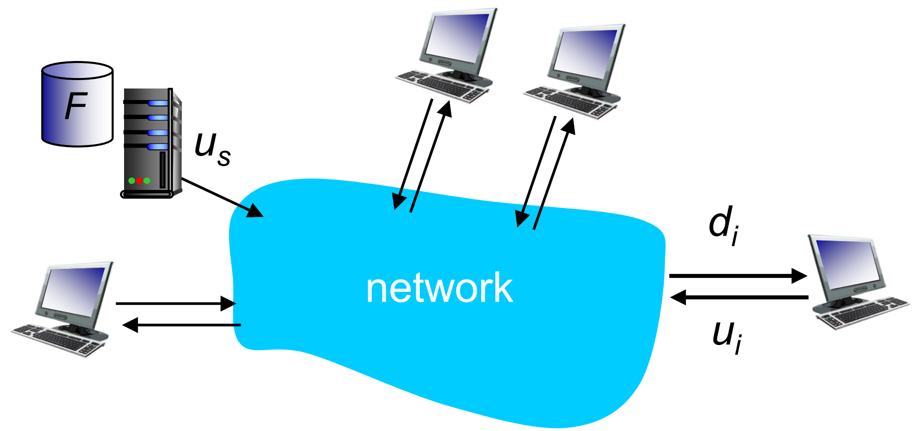
*Question:* how much time to distribute file (size *F*) from one server to *N* *peers*?

- 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: $F/u_s$
  - time to send $N$ copies: $\boxed{NF/u_s}$
- *client:* each client must download file copy
  - $d_{min}$ = min client download rate
  - min client download time: $F/\boxed{d_{min}}$




access network
↓
bottleneck

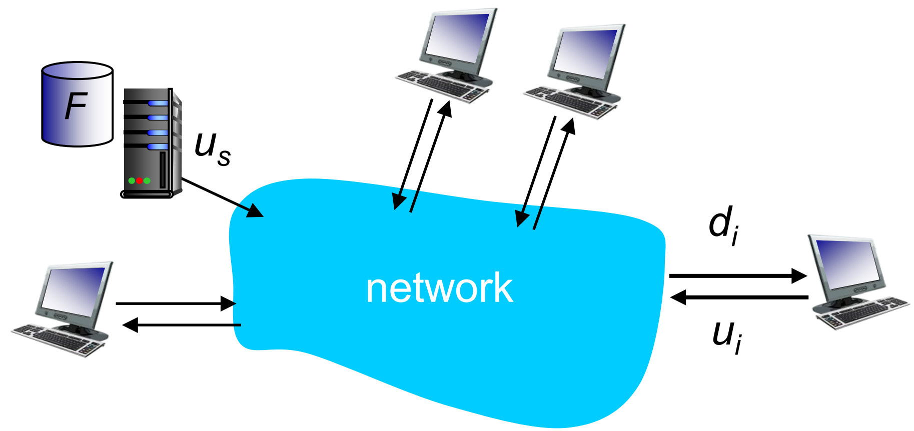
*time to distribute F to N clients using client-server approach*

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

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 $u_s + \Sigma u_i$



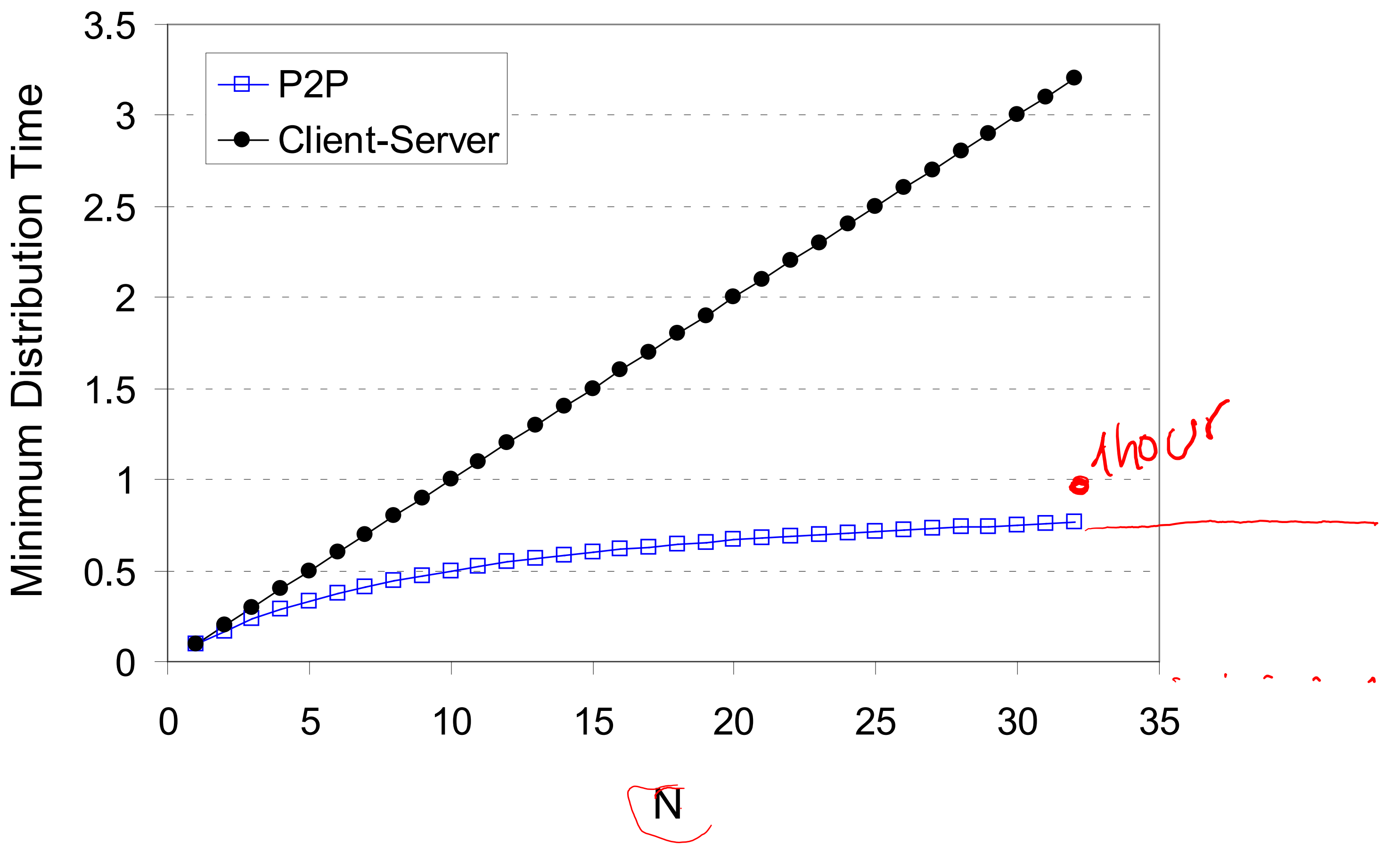

*time to distribute F to N clients using P2P approach*

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

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, $u_s = 10u$, $d_{min} \geq u_s$



# Application Layer – Part 3

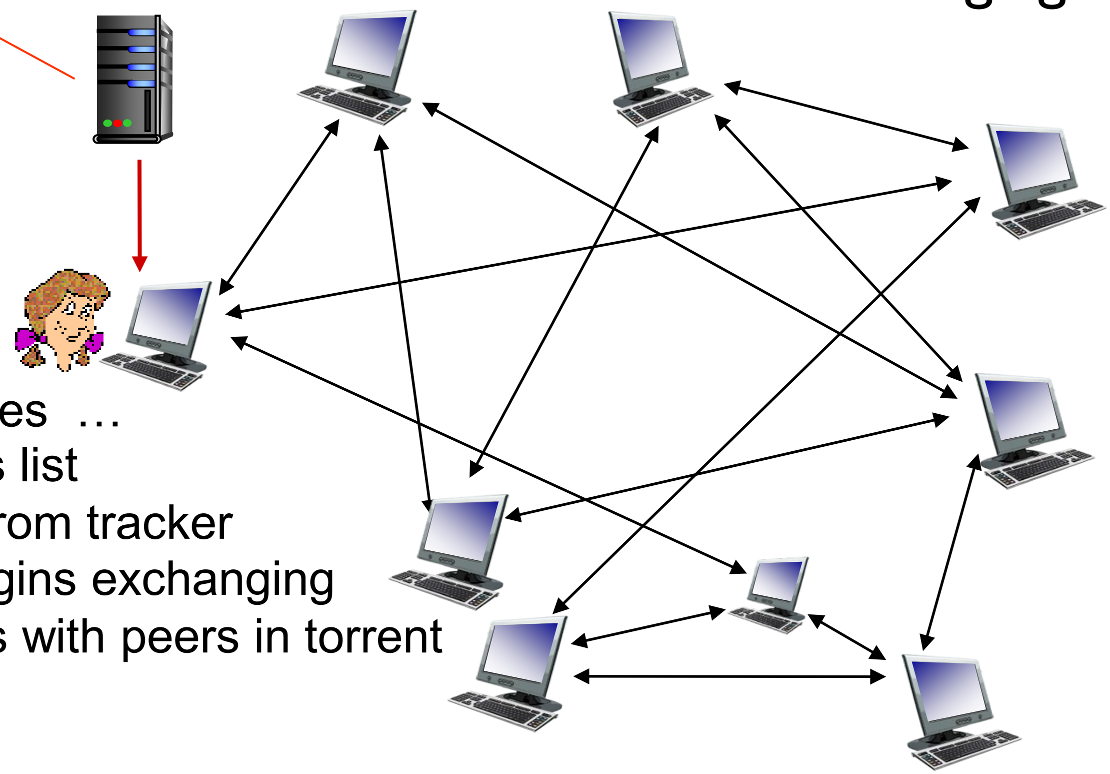
- ✓ P2P Applications
  - ✓ Characteristics
  - ✓ File distribution with BitTorrent
- ✓ Video streaming and Content Delivery Networks
  - ✓ Internet video
  - ✓ HTTP Streaming
  - ✓ CDN

# P2P file distribution: BitTorrent

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

*tracker:* tracks peers participating in torrent

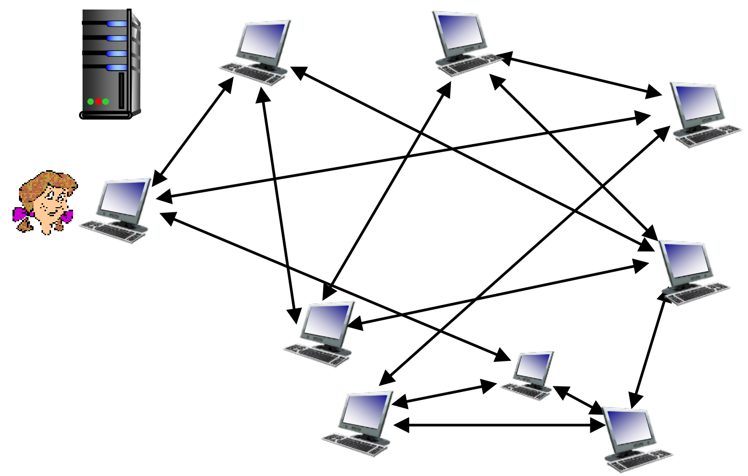
*torrent:* group of peers exchanging chunks of a file



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

TCP

*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*

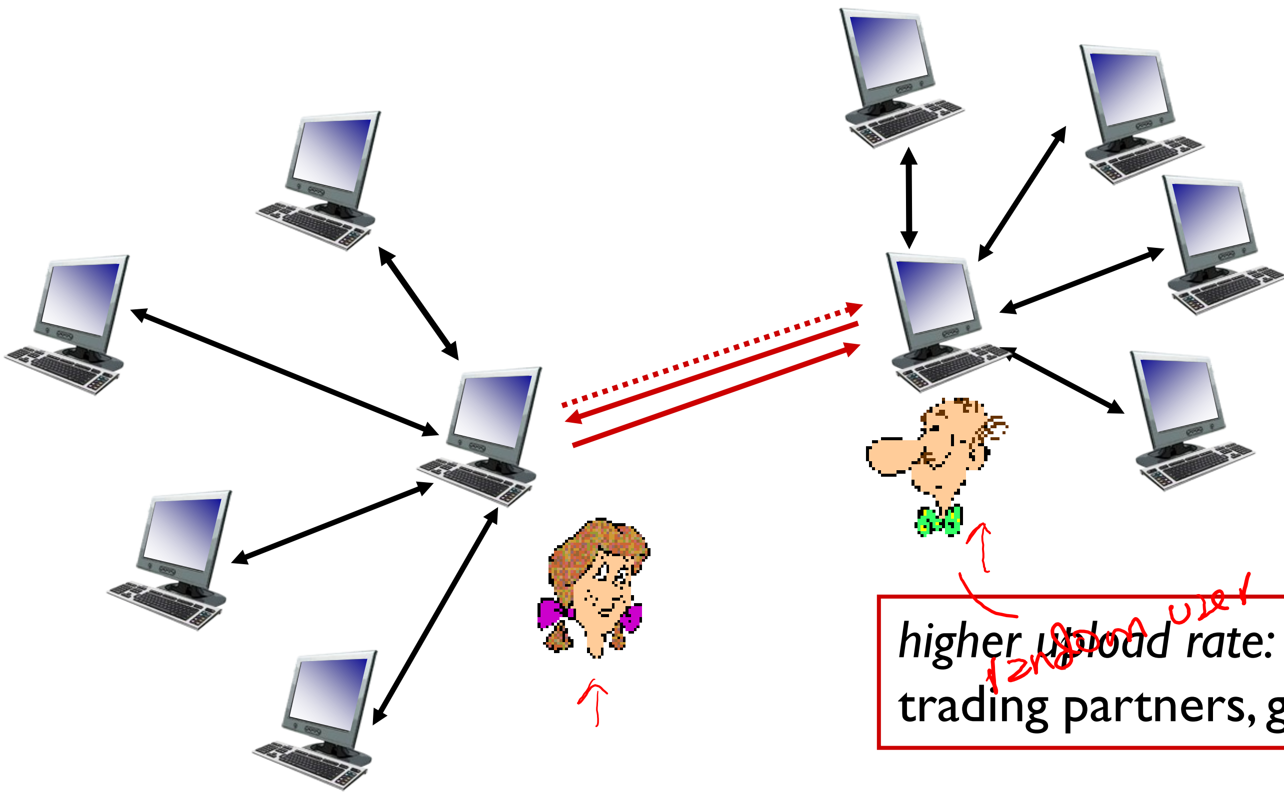
- Alice sends chunks to those four peers currently sending her chunks *at highest rate*
  - other peers are choked by Alice (do not receive chunks from her)
  - re-evaluate top 4 every10 secs
- every 30 secs: randomly select another peer, starts sending chunks
  - "optimistically unchoke" this peer
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



random user

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

# Application Layer – Part 3

- P2P Applications
  - Characteristics
  - File distribution with BitTorrent
- Video streaming and Content Delivery Networks
  - Internet video
  - HTTP Streaming
  - CDN

# Video Streaming and CDNs: context

- video traffic: major consumer of Internet bandwidth
  - Forecasted as 81% of consumer Internet traffic in 2021 (Source: Cisco)
  - Mobile Internet video traffic (usage measured from smartphones) accounts for a 49%, as of May 2021 (source: Visual Capitalist)
    - YouTube 48%, TikTok 16%, FB video 15%
  - ~116M Disney subscribers, ~73M Netflix subscribers in North America (source: Forbes.com)

# Video Streaming and CDNs: context

- 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

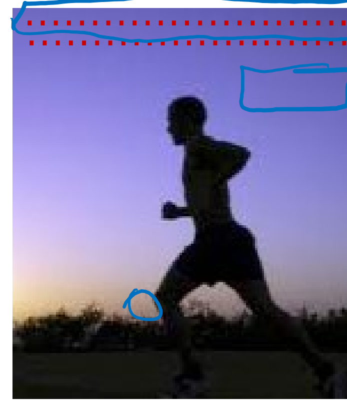
# Multimedia: video

- video: sequence of images displayed at constant rate
  - e.g., 24 images/sec / 30 fps
- 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)

low-quality 100 kpbs
HD 3 Mbps, 4K video 10 Mbps

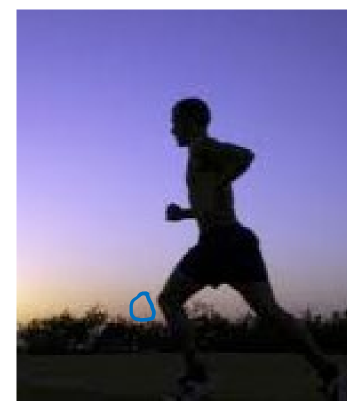
*spatial coding example:* instead of sending *N* values of same color (all purple), send only two values: color value (*purple*) *and number of repeated values* (N)

luminence & color (10101110)
# bits



frame *i*

*temporal coding example:* instead of sending complete frame at i+1, send only differences from frame i

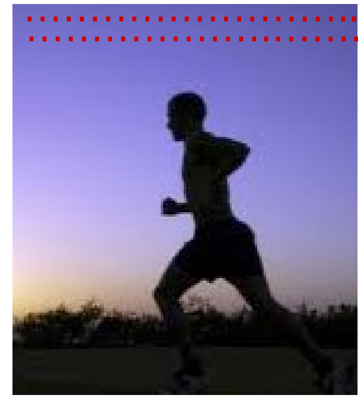


frame *i+1*

# Multimedia: video

- CBR: (constant bit rate): video encoding rate fixed
- VBR: (variable bit rate): video encoding rate changes as amount of spatial, temporal coding changes
- examples:
  - MPEG 1 (CD-ROM) 1.5 Mbps
  - MPEG2 (DVD) 3-6 Mbps
  - MPEG4 (often used in Internet, < 1 Mbps)

*spatial coding example:* instead of sending *N* values of same color (all purple), send only two values: color value (*purple*) *and number of repeated values* (N)



frame *i*

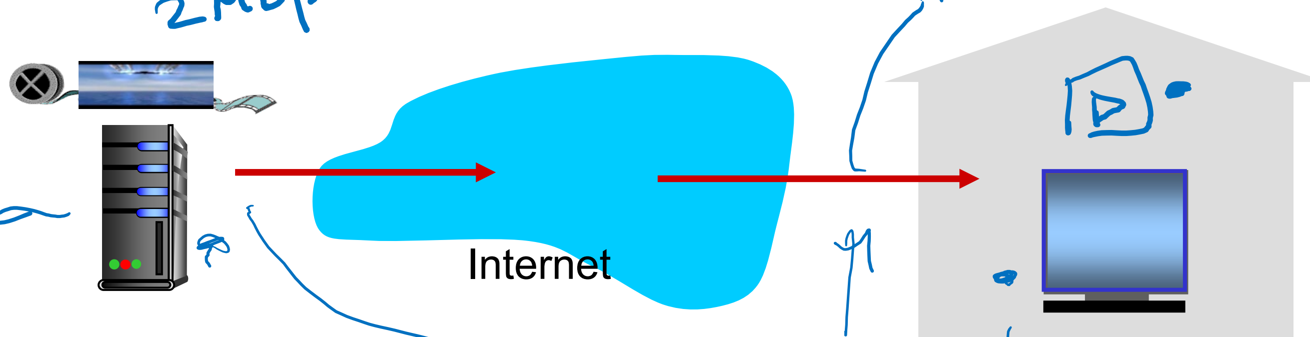
*temporal coding example:* instead of sending complete frame at i+1, send only differences from frame i

frame *i+1*

# Streaming stored video:

simple scenario: 2 Mbps — 67 minutes, 1 GB

Residential ISP




Versions
350 kbps
1 Mbps
3 Mbps

video server
(stored video)

2 Mbps

client

End-to-end throughput

CDN
- Residential ISP
- IXP
- Data center

# Application Layer – Part 3

- ✓ P2P Applications
  - ✓ Characteristics
  - ✓ File distribution with BitTorrent
- ✓ Video streaming and Content Delivery Networks
  - ✓ Internet video
  - ✓ HTTP Streaming
  - ✓ CDN

# Streaming multimedia: DASH

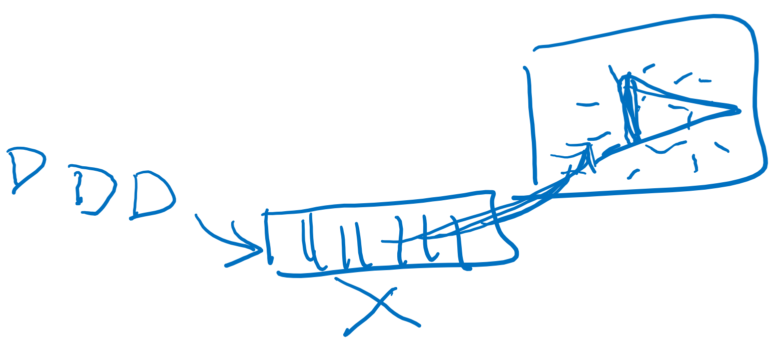
- *DASH: D*ynamic, *A*daptive *S*treaming over *H*TTP
- *server:*
  - divides video file into multiple chunks
  - each chunk stored, encoded at different rates
  - *manifest file:* provides URLs for different chunks
- *client:*
  - periodically measures server-to-client bandwidth
  - consulting manifest, 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

- *DASH: Dynamic, Adaptive Streaming over HTTP*
- *"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)

HTTP GET
(IP address)

End end throughput



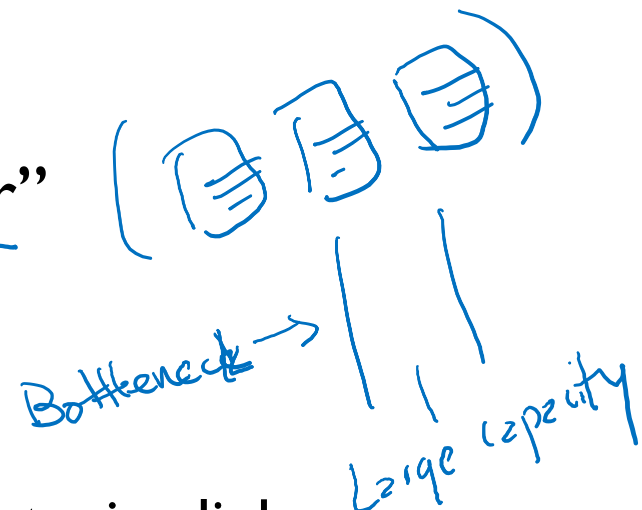
# Application Layer – Part 3

- ✓ P2P Applications
  - ✓ Characteristics
  - ✓ File distribution with BitTorrent
- ✓ Video streaming and Content Delivery Networks
  - ✓ Internet video
  - ✓ HTTP Streaming
  - ✓ CDN

# Content distribution networks

- *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

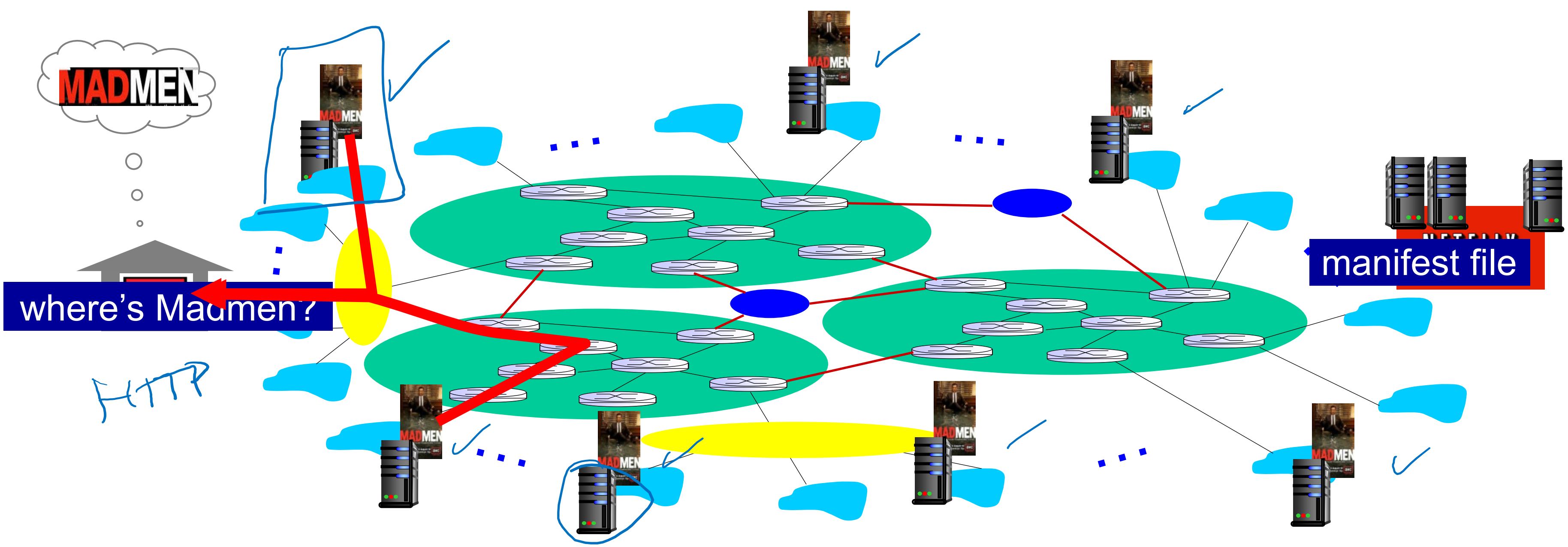
- *challenge:* how to stream content (selected from millions of videos) to hundreds of thousands of simultaneous users?

- *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 (IXP)
    - used by Limelight

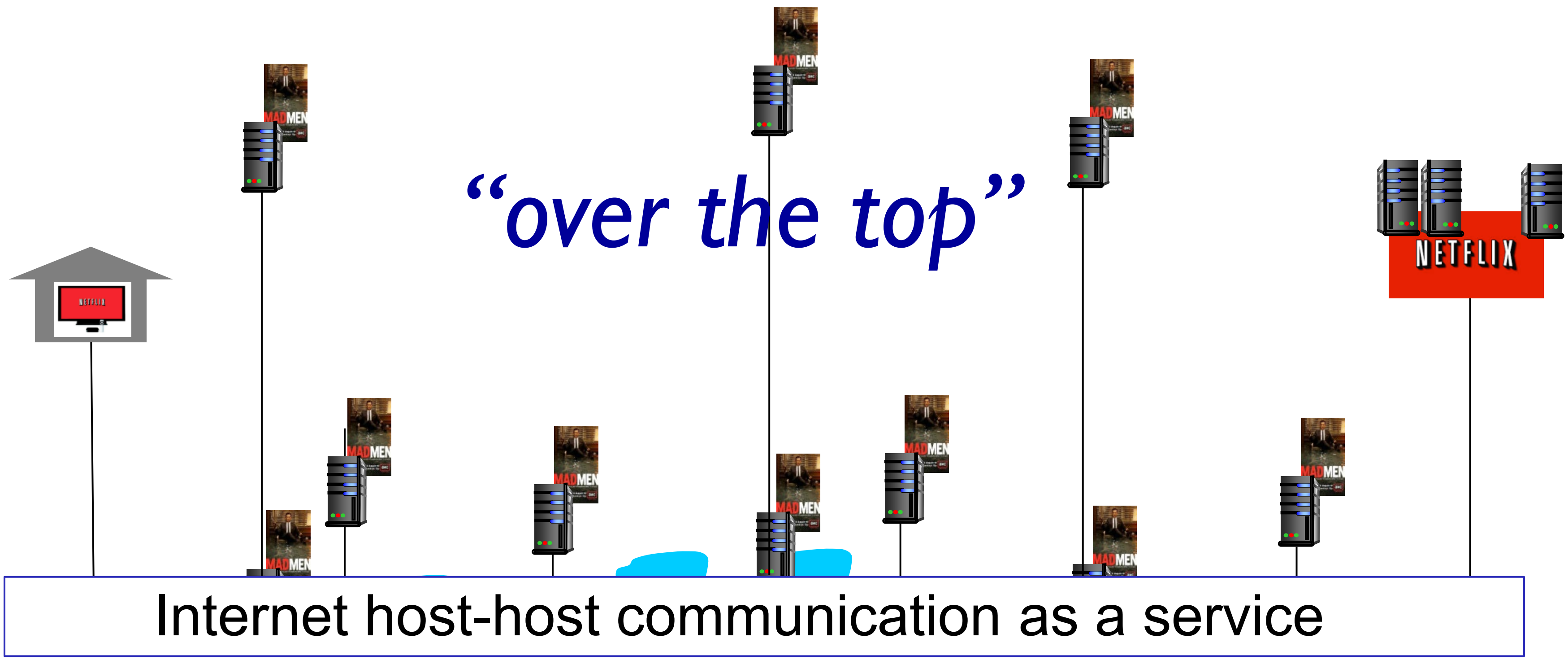
ISP1 ISP2 ISP3

# Content Distribution Networks (CDNs)

- 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)




*OTT challenges:* coping with a congested Internet

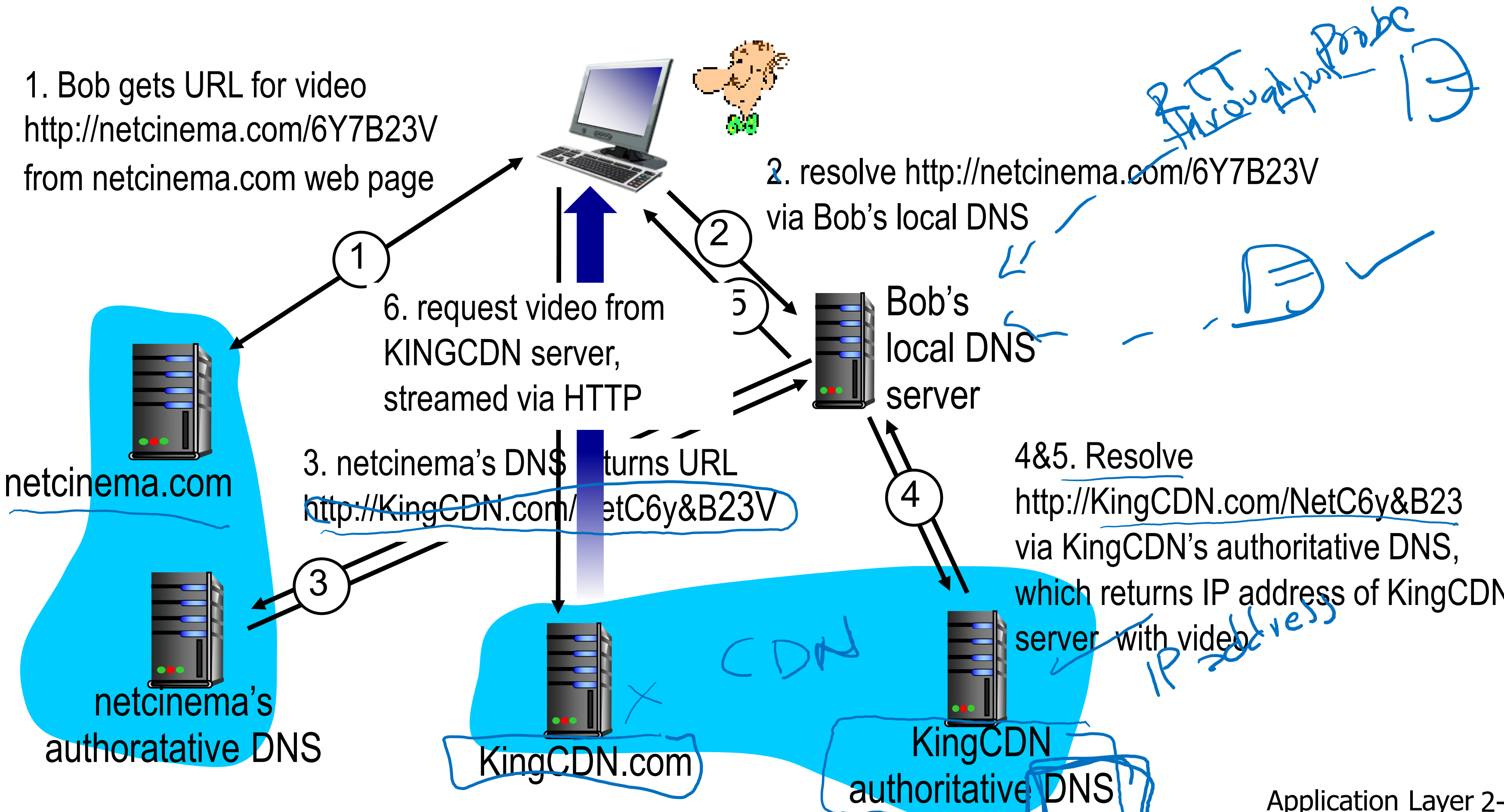
- from which CDN node to retrieve content?
- viewer behavior in presence of congestion?
- what content to place in which CDN node?

*more .. in chapter 7*

# CDN content access: a closer look

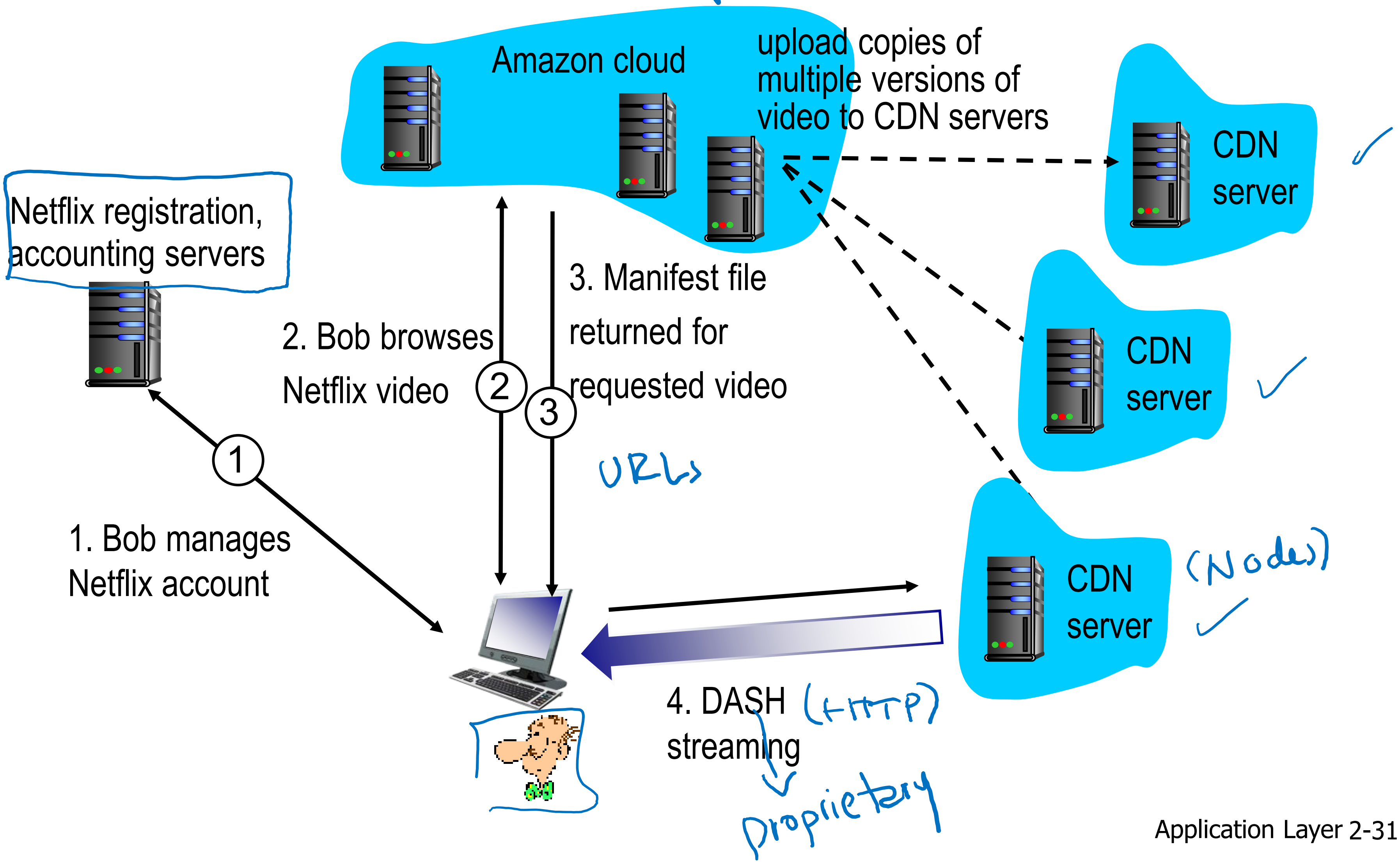
## Bob (client) requests video http://netcinema.com/6Y7B23V

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



# Case study: Netflix

Process → versions



# Case study: YouTube and Kankan

- *YouTube*
  - Private CDN to distribute videos
  - Uses pull-caching and DNS redirect
  - Cluster selection based on measured RTT with load balancing
  - Manual selection of video version
- *Kankan*
  - Based on P2P   X client-server
  - Similar to BitTorrent
  - Request for play-first chunks, to ensure smooth playback of videos

# References

Figures and slides are taken/adapted from:


- Jim Kurose, Keith Ross, “Computer Networking: A Top-Down Approach”, 7th ed. Addison-Wesley, 2012. All material copyright 1996-2016 J.F Kurose and K.W. Ross, All Rights Reserved