Application Layer

Application Layer

- Principles of network applications
- Web and HTTP
- E-mail, SMTP, IMAP
- The Domain Name System: DNS
- P2P applications
- Video streaming, CDNs
- Socket programming

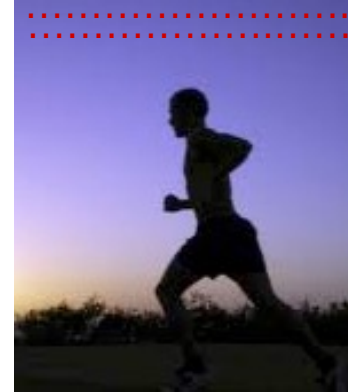# Video Streaming and CDNs: context

- 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
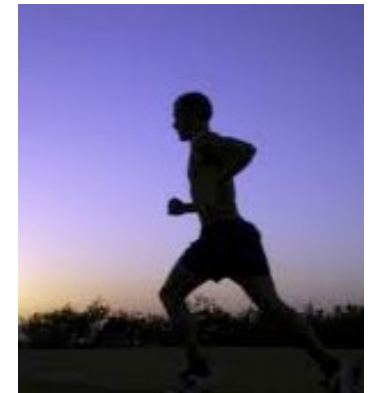
Multimedia: video

- 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)

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*

frame *i+1*

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

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, 64 kbps - 12 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*)
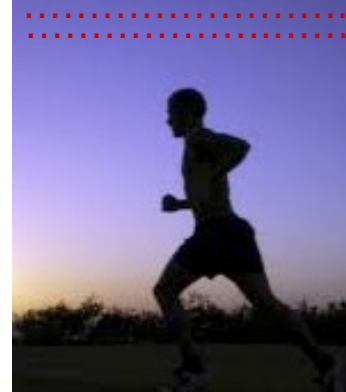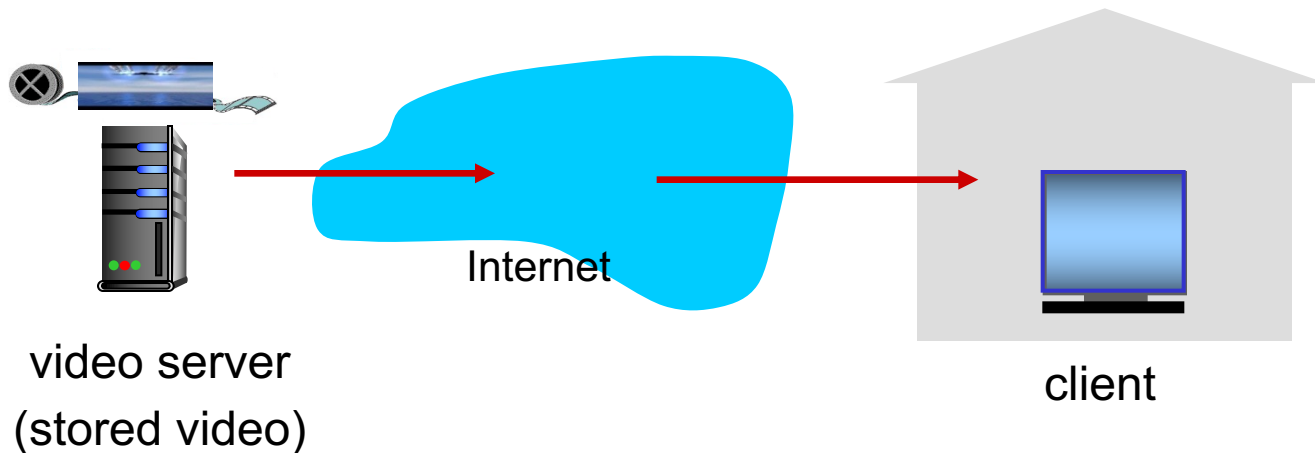
frame *i*

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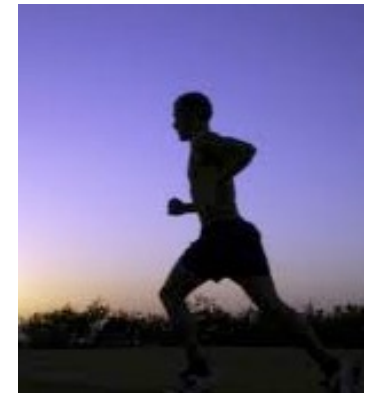
frame *i+1*

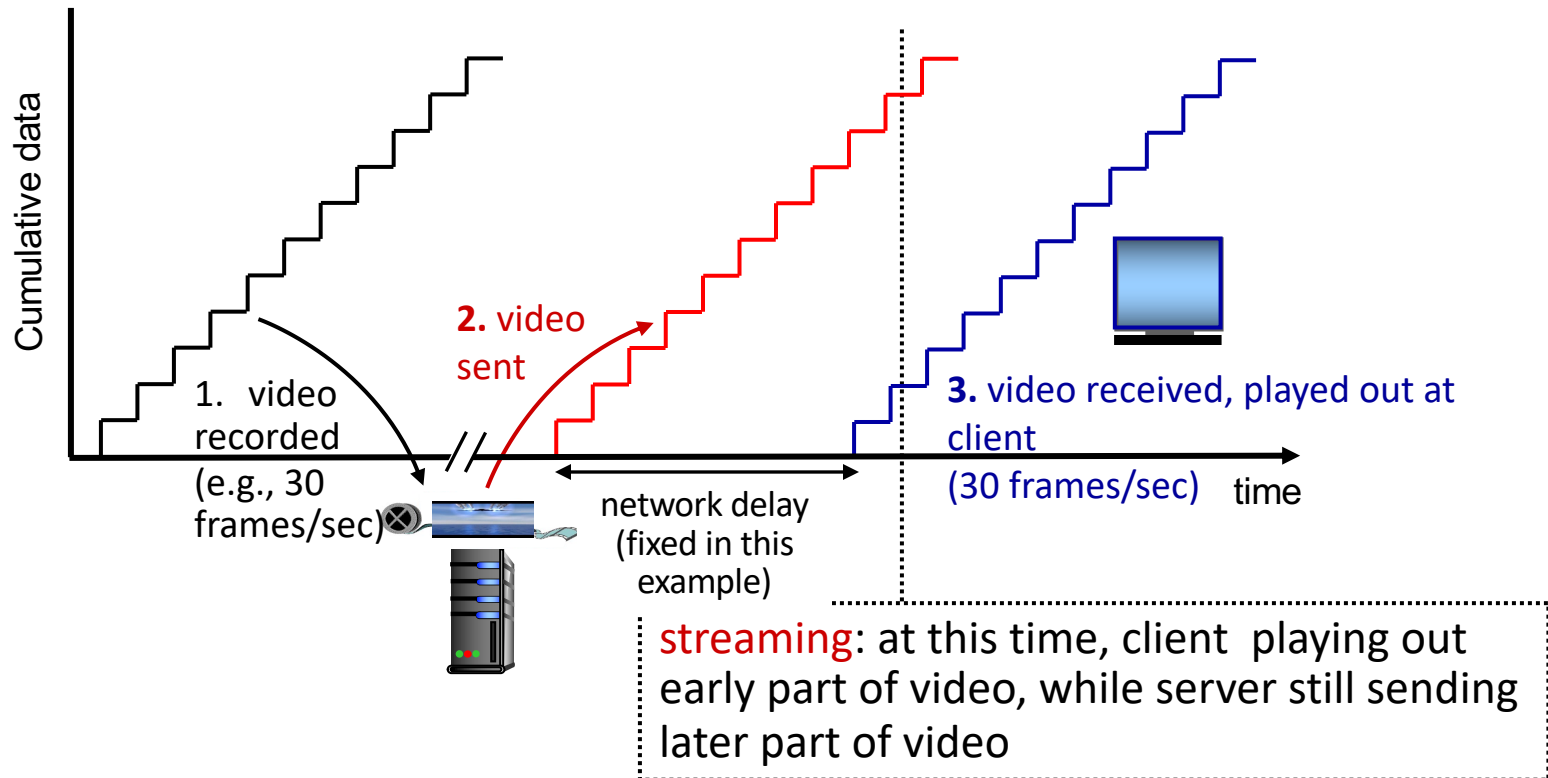# Streaming stored video:

simple scenario:

video server
(stored video)

client

Main challenges:

- server-to-client bandwidth will *vary* over time, with changing network congestion levels (in house, access network, network core, video server)
- packet loss, delay due to congestion will delay playout, or result in poor video quality

Streaming stored video

Cumulative data

2. video sent

1. video recorded (e.g., 30 frames/sec)

network delay (fixed in this example)

3. video received, played out at client (30 frames/sec)

time

streaming: at this time, client playing out early part of video, while server still sending later part of video

Streaming stored video: challenges

- **continuous playout constraint**: during client video playout, playout timing must match original timing
  - … but network delays are variable (jitter), so will need client-side buffer to match continuous playout constraint
- other challenges:
  - client interactivity: pause, fast-forward, rewind, jump through video
  - video packets may be lost, retransmitted
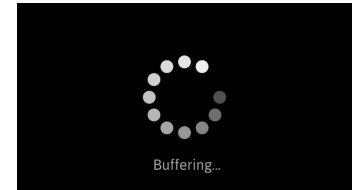
Streaming stored video: playout buffering
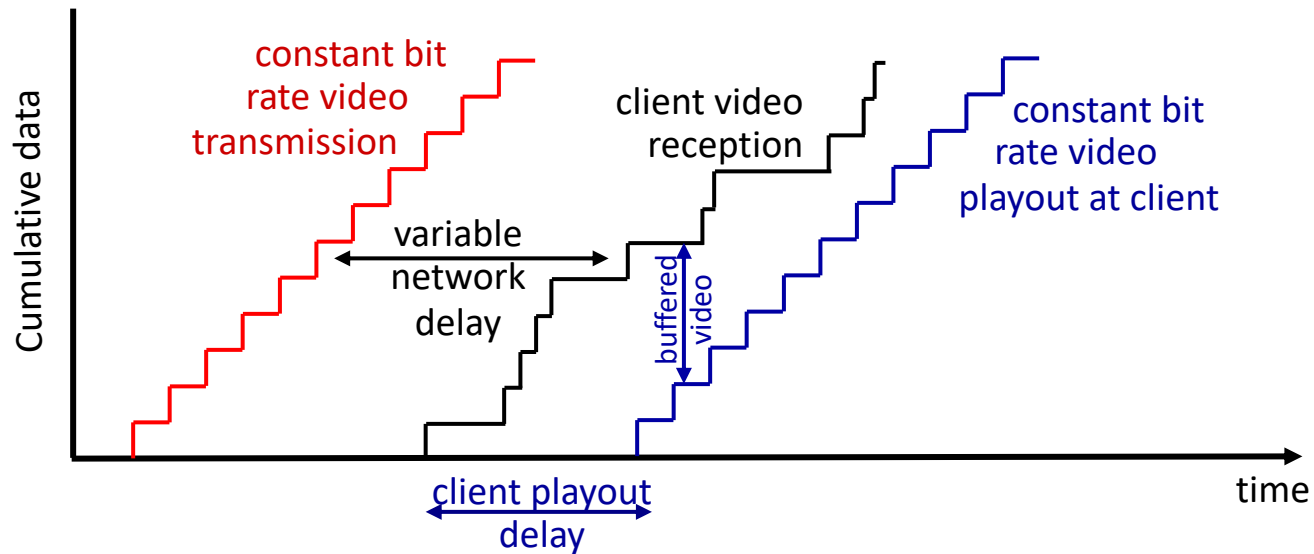
- *client-side buffering and playout delay:* compensate for network-added delay, delay jitter

Streaming multimedia: DASH

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

Streaming video= encoding+ DASH+ playout buffering

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

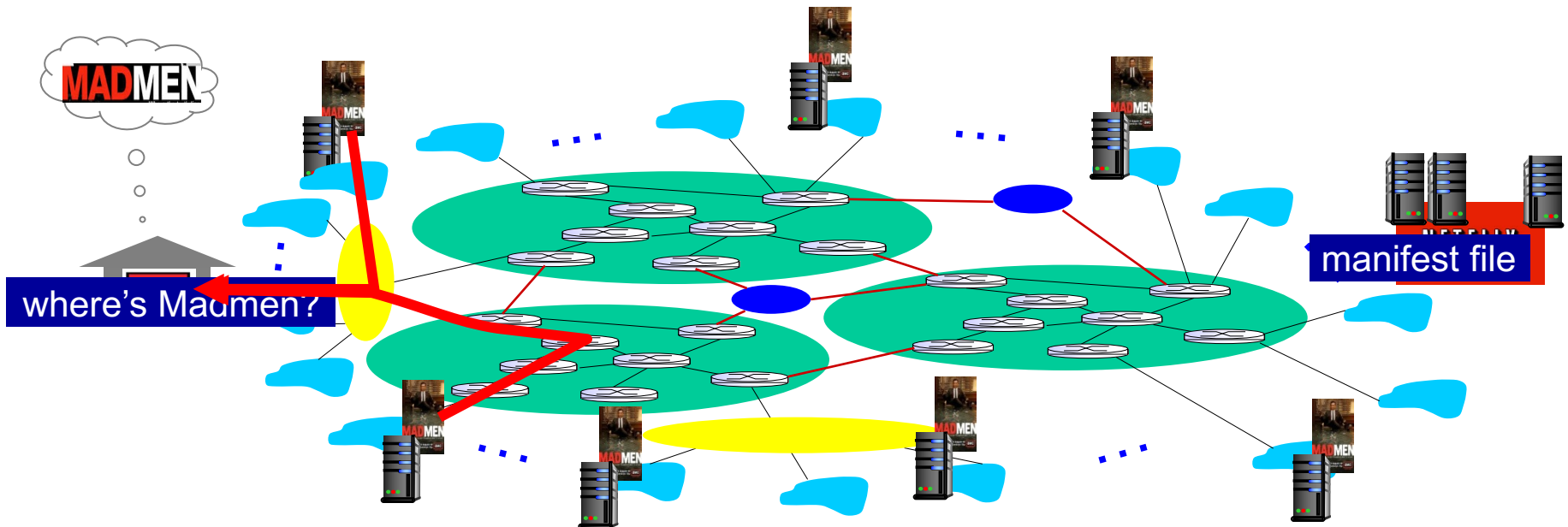….quite simply: this solution *doesn't scale*

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
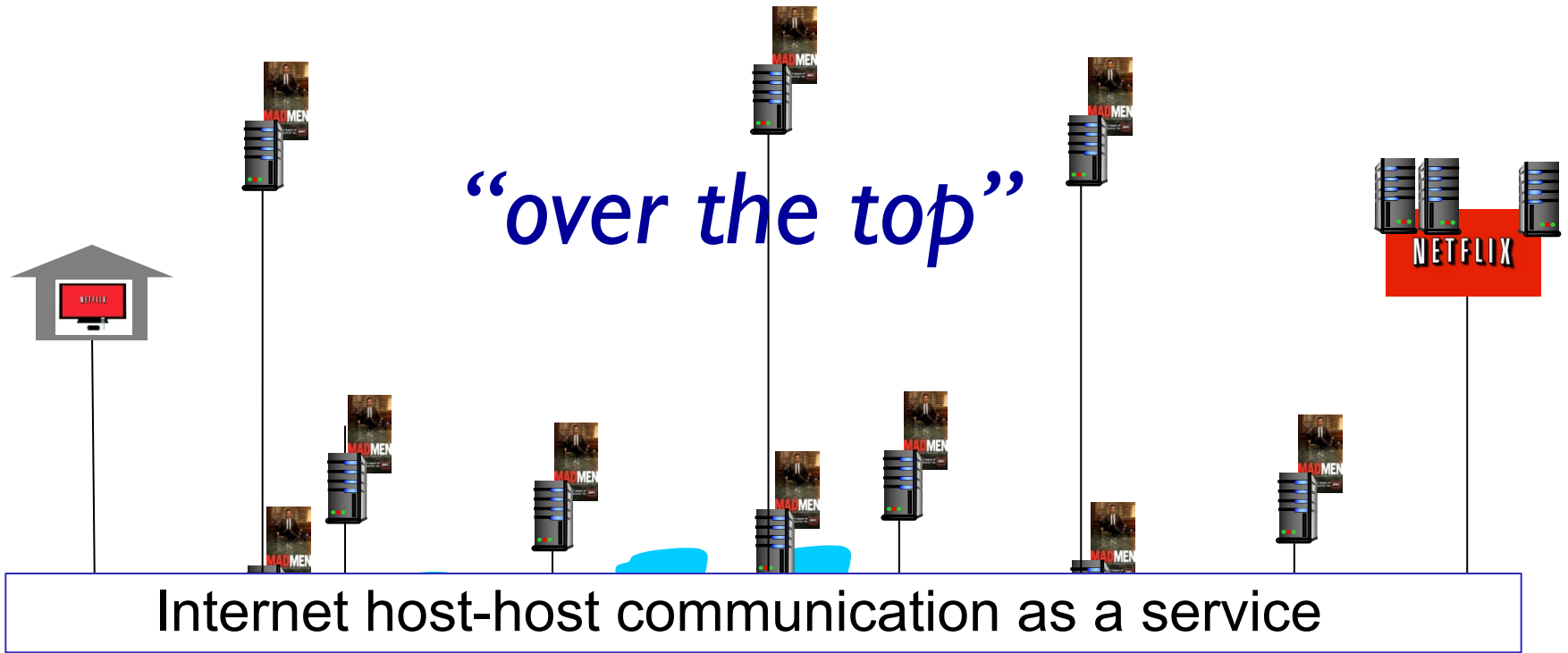   - used by Limelight

Content Distribution Networks (CDNs)

- CDN: stores copies of content (e.g. Madmen) at CDN nodes
- subscriber requests content, service providers return manifest
  - Using manifest, client retrieves content at highest supportable rate
  - may choose different copy if network path congested

MADMEN

where's Madmen?

manifest file

Content Distribution Networks (CDNs)

"over the top"

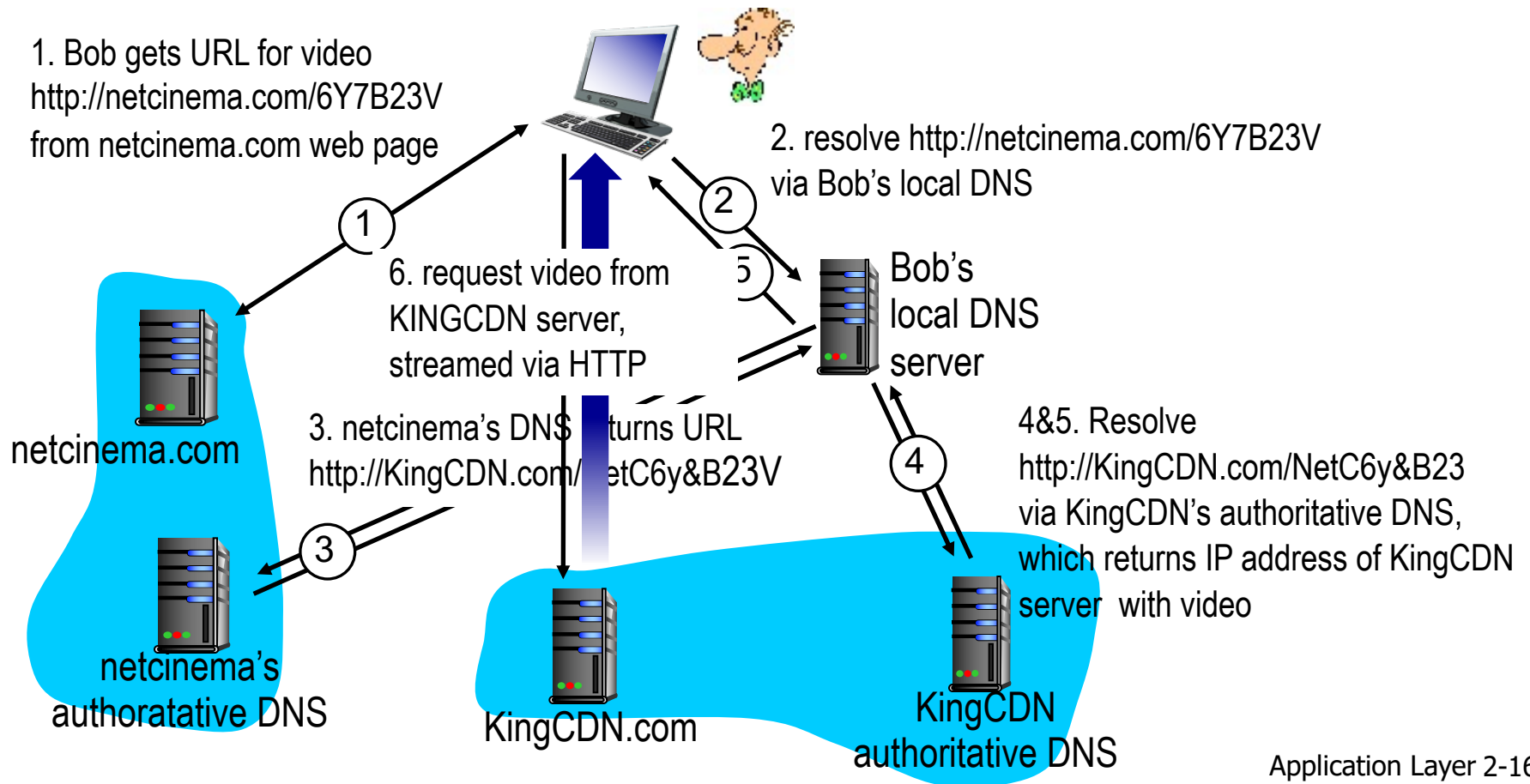Internet host-host communication as a service

OTT challenges: coping with a congested Internet
- from which CDN node to retrieve content?
- 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

6. request video from
KINGCDN server,
streamed via HTTP

netcinema.com

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

netcinema's
authoratative DNS

KingCDN.com

Bob's
local DNS
server

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

KingCDN
authoritative DNS

Case study: Netflix

Netflix registration, accounting servers

Amazon cloud

upload copies of multiple versions of video to CDN servers

CDN server

CDN server

CDN server

3. Manifest file returned for requested video

2. Bob browses Netflix video

1. Bob manages Netflix account

4. DASH streaming

Application Layer

❖ Principles of network applications

❖ Web and HTTP

❖ E-mail, SMTP, IMAP

❖ The Domain Name System: DNS

❖ P2P applications

❖ Video streaming, CDNs

❖ Socket programming