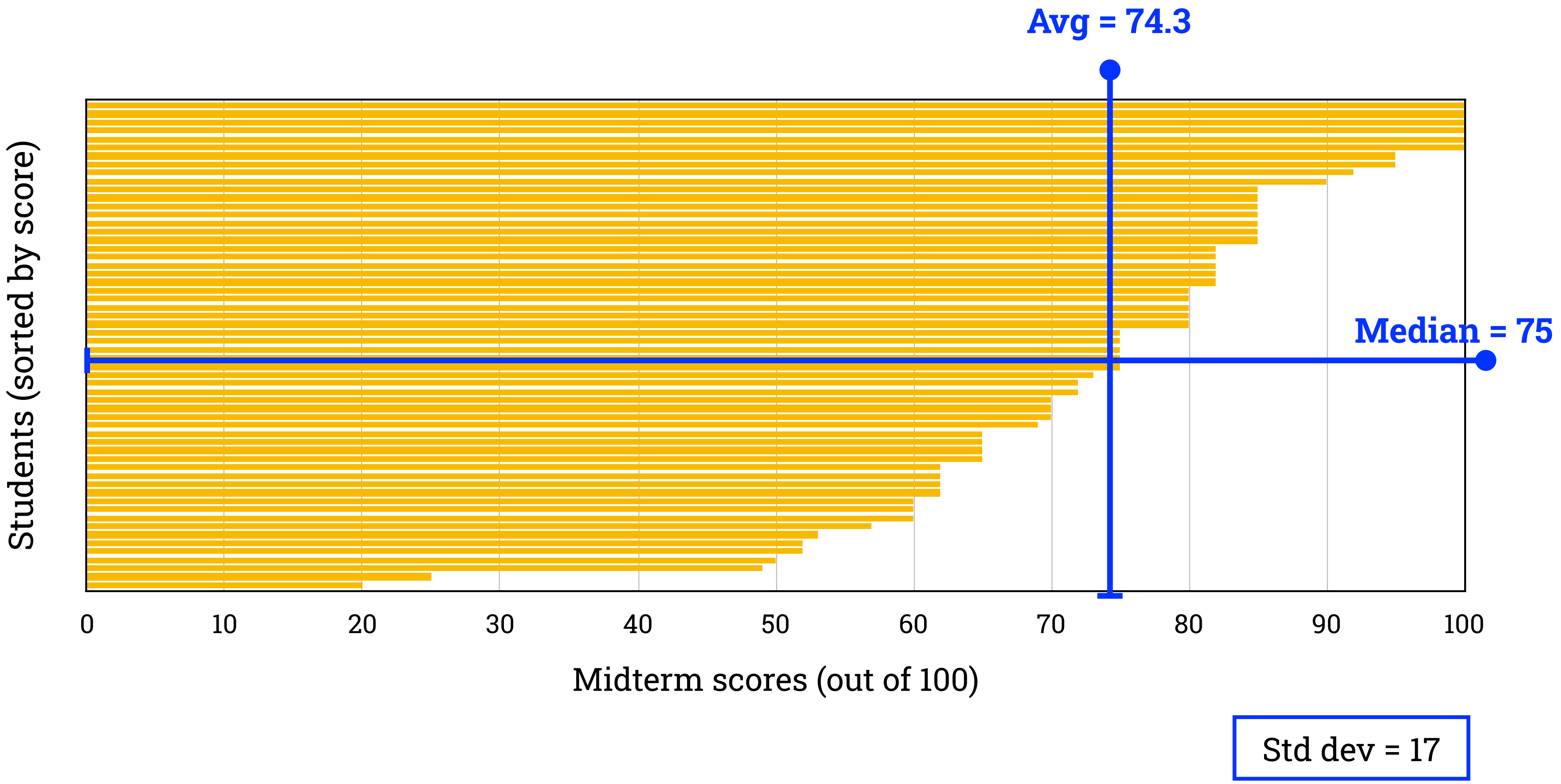
CS3640

Application Layer (5): Video Streaming

Prof. Supreeth Shastri
Computer Science
The University of Iowa

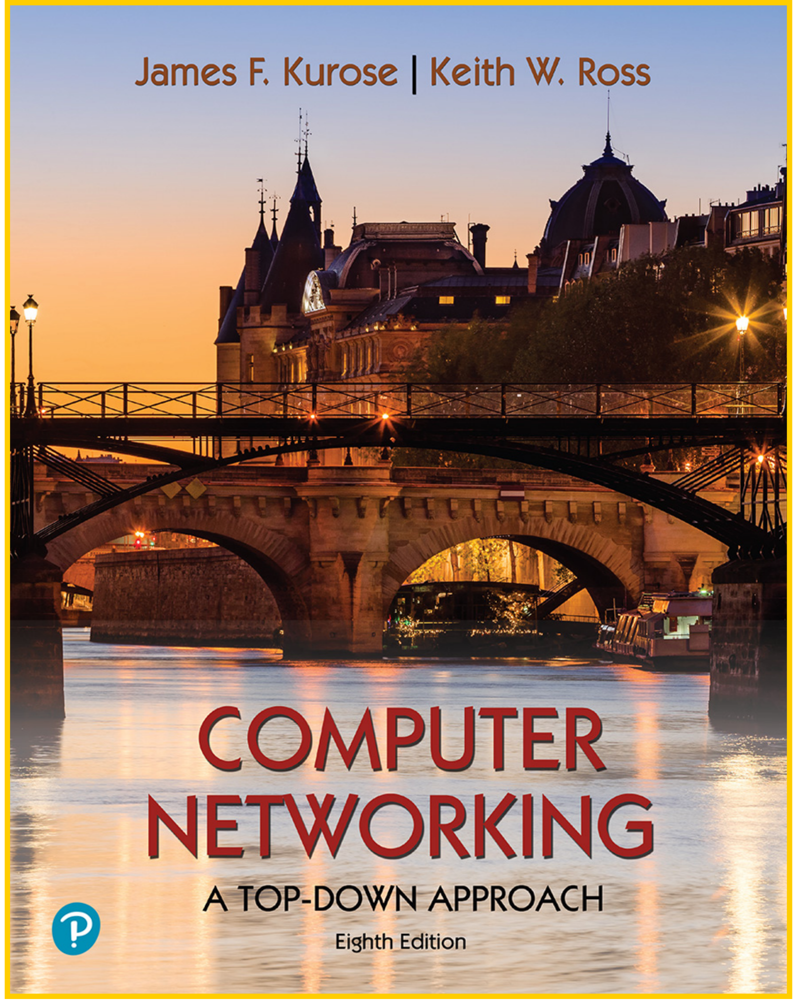
Written-1 Distribution



Lecture goals

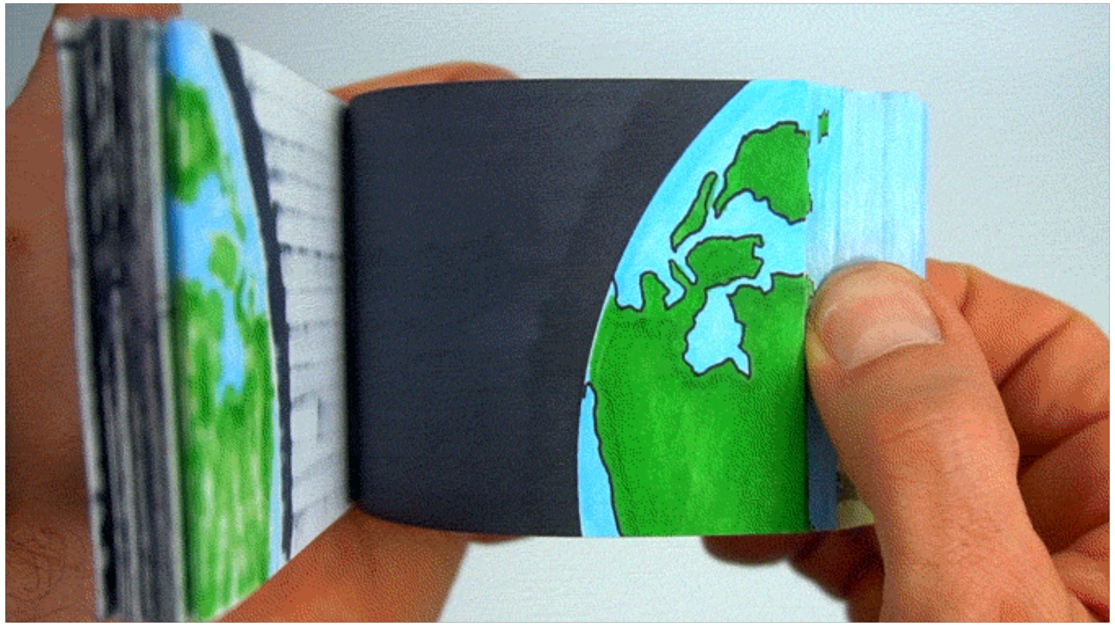
Technical overview of how video streaming on Internet is implemented

- *Characteristics of video*
- *Internet streaming*
- *Content Distribution Networks*
- *Case study: Netflix and YouTube*



Chapter 2.6

Basics of video



courtesy: https://andymation.squarespace.com/

- **Video:** a sequence of images displayed at constant rate to create the illusion of motion
- **Frame rate**: the number of still pictures per unit of time.
- The *minimum* frame rate to achieve a comfortable illusion of a moving image is ~16 frames/second
- National Television System Committee (NTSC) standard requires a rate of **30 frames/sec**

Digital video

- **Digital image:** an array of pixels, each of which represents the luminance and color of that area.
- Two types of **redundancy**: spatial (within an image) and temporal (across images)

spatial redundancy: *instead of storing N values of same color, we could store only two values: color (black) and number of repeated values (N)*

temporal redundancy: *instead of storing the complete frame at i+1, store only the differences from frame i*

frame i

frame i+1

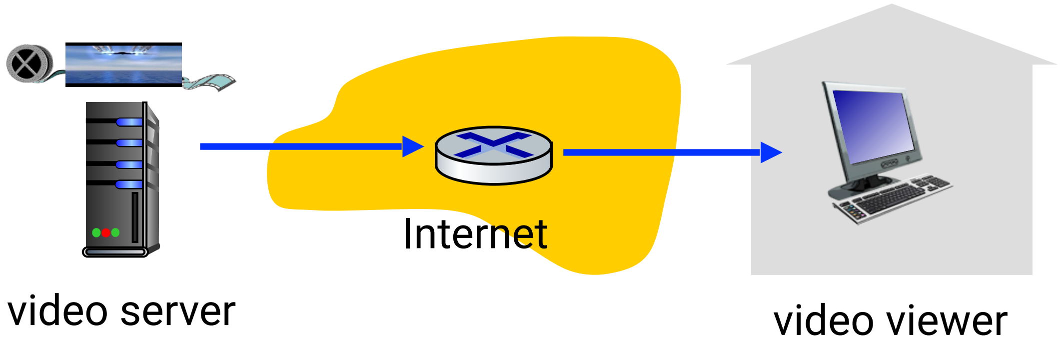
Video encoding

Key idea: *exploit the redundancy to reduce the video size*

- Video codec: a system of technologies used for encoding videos at the origin and then decoding before viewing
  - ➡ Examples: MPEG-1 (CD-ROM), MPEG-2 (DVD), MPEG-4 and AVC (Internet video)
- Video codecs employ both lossless and lossy compression techniques
  - ➡ **Variable length encoding** (such as Huffman coding), which maps frequently occurring symbols to shorter length codes
  - ➡ **Color subsampling**, which separates luminance (brightness) from chrominance (color) information, and then downsamples color information since human vision is less sensitive to color than brightness
- Codecs allow control over output bit rate
  - ➡ **CBR** (constant bit rate), if the video is encoded at a fixed rate
  - ➡ **VBR** (variable bit rate), when video encoding rate changes over time

Internet **Streaming**

Challenges of Internet Streaming



Data Volume

Netflix, YouTube, Amazon Prime account for 80% of residential ISP traffic (2020)

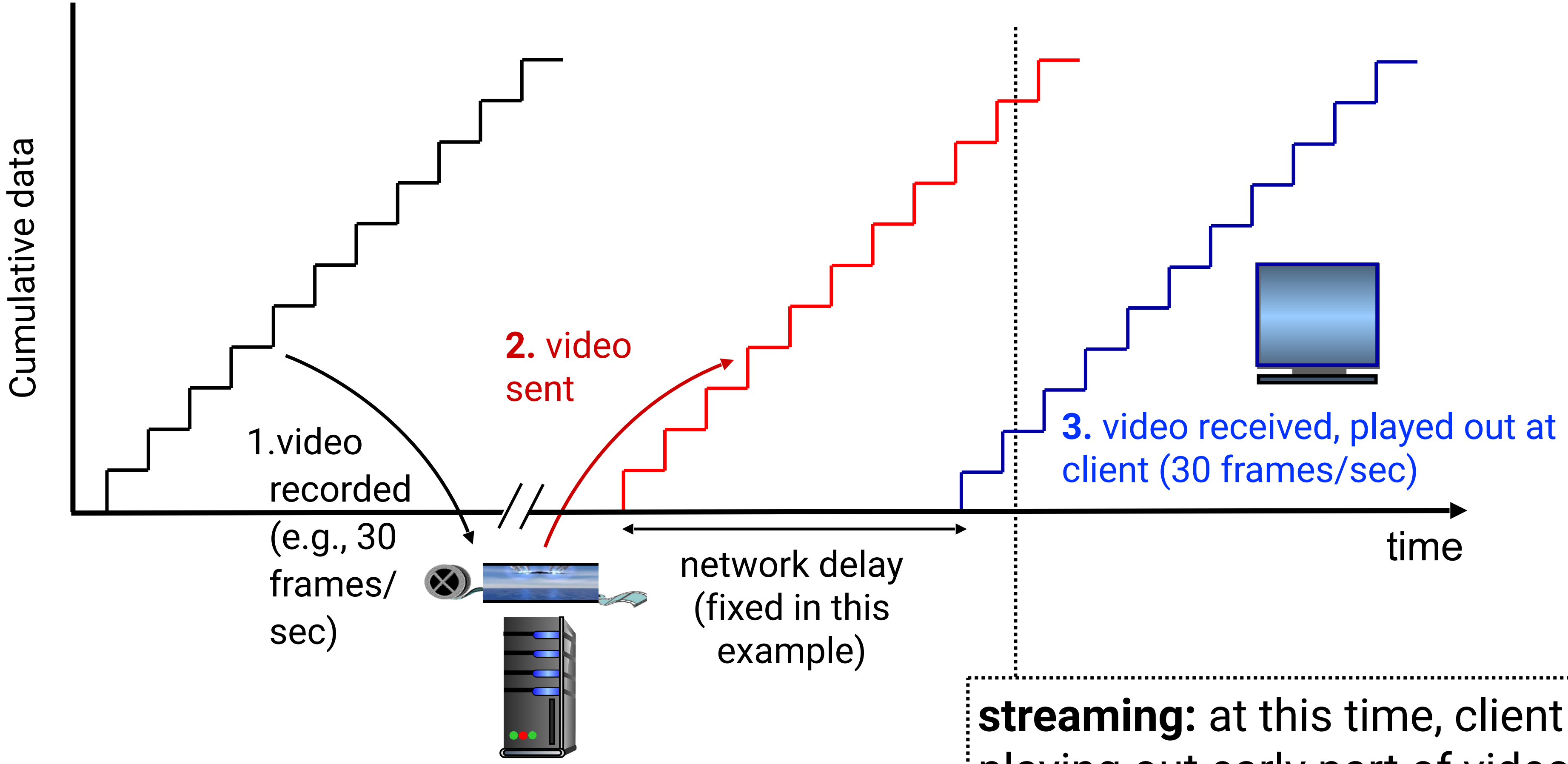
Scale

How to efficiently stream video to millions - billions of users worldwide?

Heterogeneity

Users have different device capabilities and bandwidths, both of which may vary with time

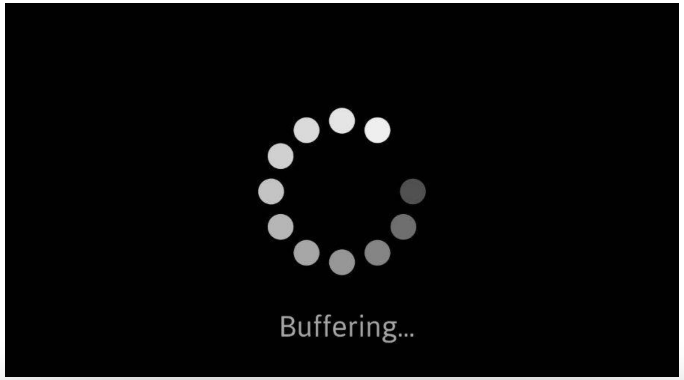
Streaming stored video



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

Streaming stored video

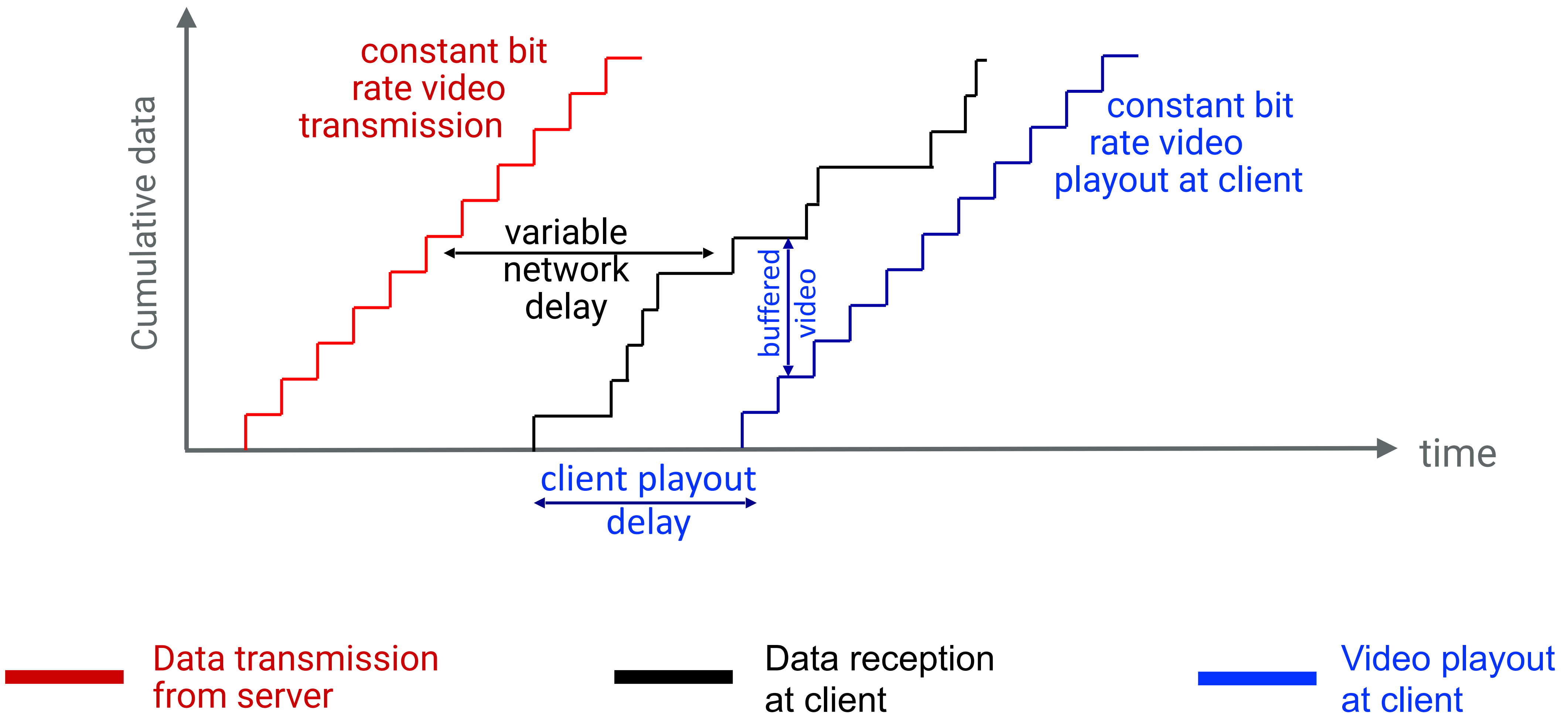
What could go wrong?



Continuous playout constraint: during video playout in the client device, the playout timing must match the original timing

- **network challenges**: network delays are variable (jittery), so the client will need to buffer data in order to meet the continuous playout constraint
- **playback challenges**: client interactivity due to stop/pause, fast-forward, rewind, jump through video

Solution: Client-side buffering and delayed playout



Dynamic Adaptive Streaming over HTTP (**DASH**)

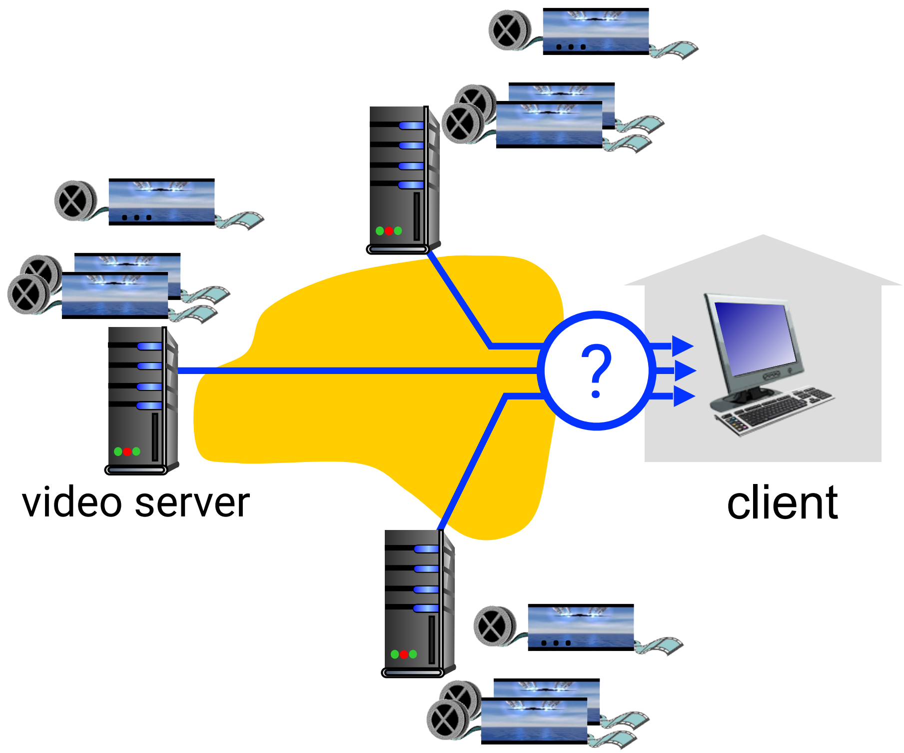
DASH server

- Divides video file into multiple chunks
- Encodes each chunk at multiple different rates
- Stores different rate encodings in different files
- Sends client a **manifest file** w/ URLs for these files

DASH-aware browser

- Estimates server-to-client bandwidth periodically
- Consulting manifest file, it requests one chunk at a time
  - ➡ goal is to choose the maximum coding rate sustainable with current bandwidth
  - ➡ ability to choose different coding rates at different points in time

Dynamic Adaptive Streaming over HTTP (**DASH**)



Intelligence at client: client determines

- **when** to request chunk (to avoid both buffer starvation and buffer overflow)
- **what encoding rate** to request (to maximize video quality when bandwidth available)
- **where** to request chunk (to minimize latency to the destination CDN server)

Content Distribution Networks

Challenge: you are an Internet video company. How to stream video content (selected from ~millions of videos) to millions of simultaneous users?

Option-1: build a single massive datacenter

- Single point of failure
- Contributes to network congestion
- Long path to distant clients

Option-2: content distribution networks (CDNs)

- a network of geographically distributed servers, each of which stores copies of different videos
- Make DASH clients stream video from CDN servers (instead of the origin server)
- Examples: Akamai, Limelight, Level-3

CDN placement strategies

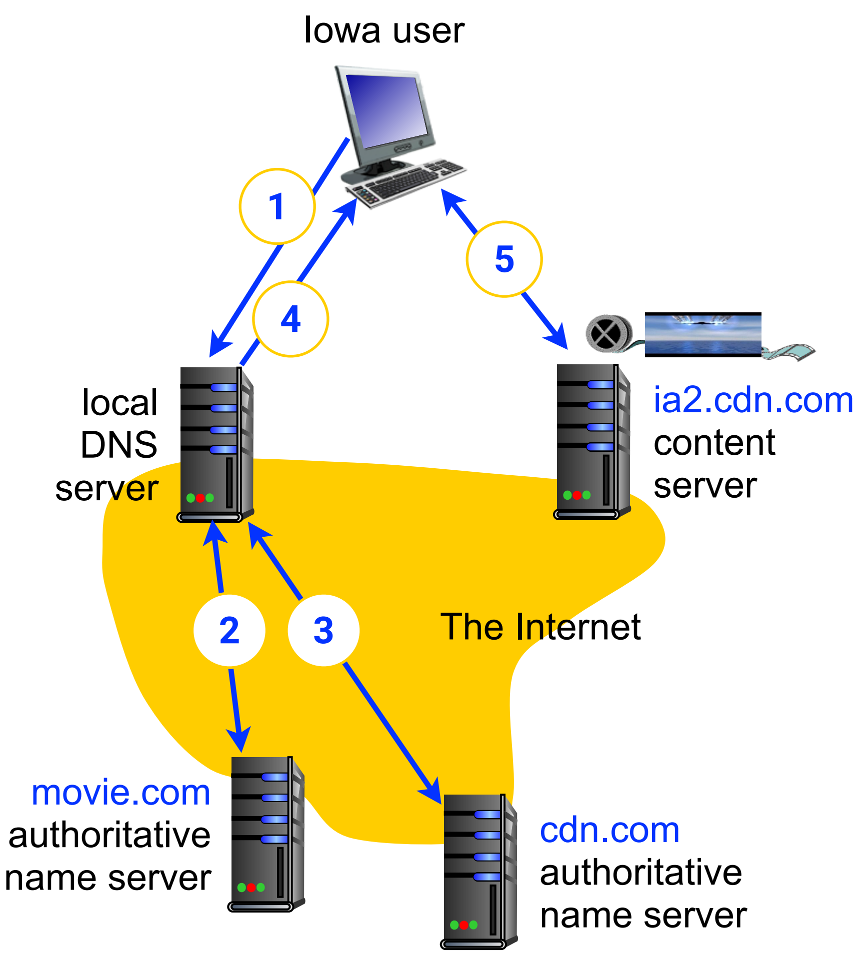
1. Enter Deep into the network

- **Goal**: get CDN servers as close to the users as possible
- This requires deploying small number of CDN servers in access ISPs all around the world (i.e., a large number of installations)
- Decreases latency (due to proximity), improves throughput (by avoiding hopping across the Internet core)
- Example: *Akamai*

2. Bring ISPs Home

- **Goal**: place CDN servers close to Internet exchange points (IXPs)
- This requires deploying large CDN clusters at a small number of IXPs
- Lowers overhead of maintenance and management (but affects latency and throughput)
- Example: *Limelight networks*

CDN operation



1. **DNS lookup**: User visits *movie.com*, and user's host sends a DNS query to the local DNS server
2. **CDN handover**: local DNS server relays the query to an authoritative name server for *movie.com*. Instead of an IP address, it returns the name of its CDN operator
3. **Content server selection**: local DNS server sends another query to *cdn.com*, to which the CDN responds with the IP address of the chosen/nearby content delivery server
4. **DNS response**: local DNS forwards the IP address of the content serving CDN node, *ia2.cdn.com*
5. **Content flow**: client establishes a *TCP* connection with the local content server, and sends *HTTP GET* request for the video

Real-world CDNs

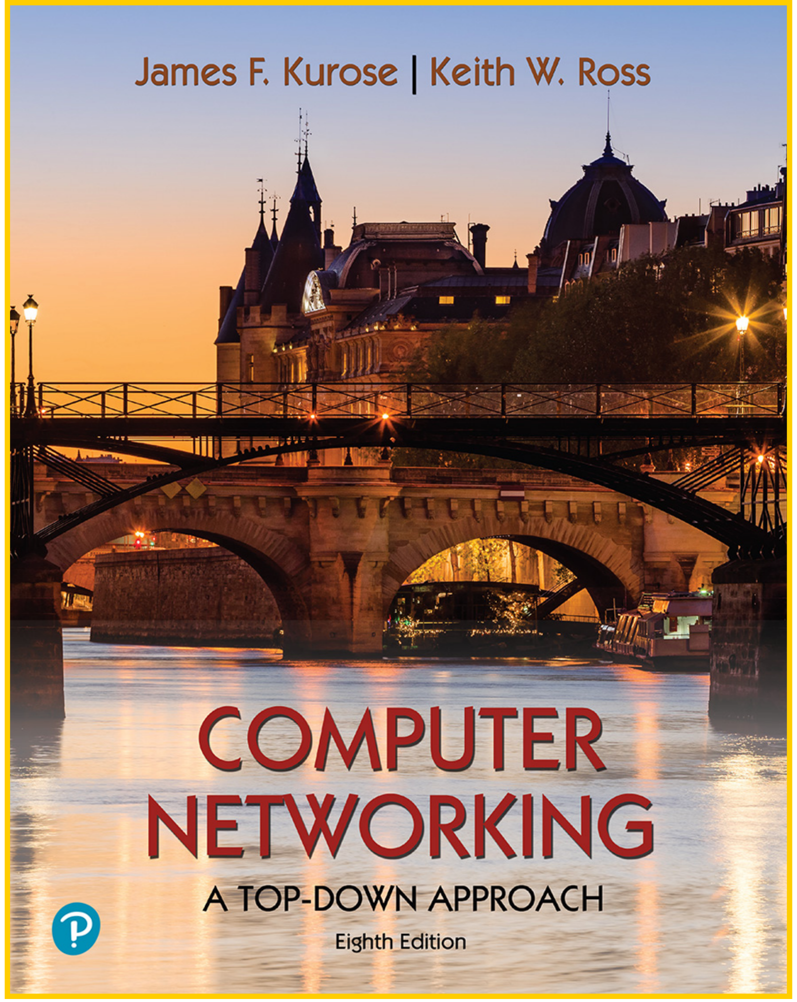
	NETFLIX	YouTube
Operator	2007 - 2012: Akamai, Limelight, Level 3 2012 onwards: Self owned and operated	Google CDN (since 2006)
Architecture	Hybrid: ~200 IXP locations, ~100 ISPs. Offered free of cost to any ISP/IXP	Hybrid: several hundred IXP and ISP locations. Offered free of cost to any ISP/IXP
CDN content	Preprocessing on Amazon cloud; Netflix refreshes CDN content daily (i.e push caching)	Preprocessing on Google data centers; Employs DNS redirect and pull caching (lecture-7)
Streaming	DASH w/ manifest file	HTTP Streaming, choice of bit rate left to users

Ponder about: network **neutrality** | network **economics** | network **principles**

Next lecture

how to build client-server applications that communicate using sockets

- *Socket programming*
- *TCP sockets*
- *UDP sockets*



Chapter 2.7

Spot Quiz (ICON)