15-441/641: Computer Networks
Internet Video Delivery

15-441 Fall 2019
Profs **Peter Steenkiste** & Justine Sherry

Fall 2019
https://computer-networks.github.io/fa19/

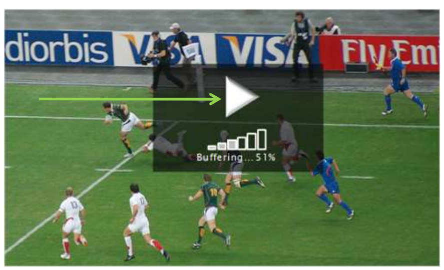
Carnegie Mellon University

Outline

- Background
- Technologies:
  - HTTP download
  - Real-time streaming
  - HTTP streaming
  - Runtime adaptation
  - Video brokers

Based on slides from Hui Zhang

Bad Things to Avoid in Streaming Video



1990 – 2004: 1st Generation Commercial PC/Packet Video Technologies

RealNetworks · Windows Media Player · QuickTime · WebTV

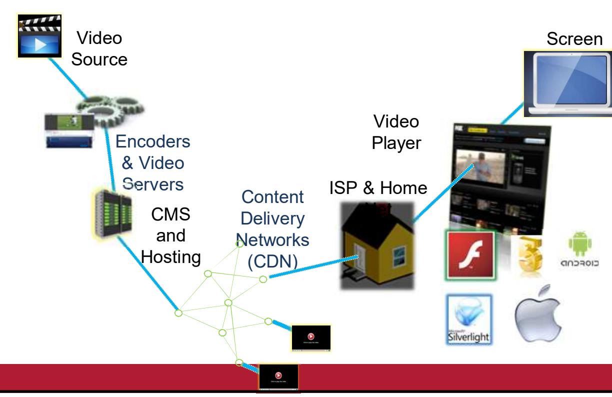
- Simple video playback, no support for rich app
- Not well integrated with Web browser
- No critical mass of compelling content over Internet
- No enough broadband penetration

2005: Beginning of Internet Video Era

2006 – 2013: Video Going Prime Time

Today Internet Video on Multiple Devices

Internet Video Data-plane



Internet Video Requirements

- Smooth/continuous playback
- Elasticity to startup delay: need to think in terms of RTTs
- Elasticity to throughput
  - Multiple encodings: 200Kbps, 1Mbps, 2 Mbps, 6 Mbps, 30Mbps
- Multiple classes of applications with different requirements

	Delay	Bandwidth	Examples
2, N-way conference	< 200 ms	4 kbps audio only, 200 kbps – 5 Mbps video	Skype, Google hangout, Polycom, Cisco
Short form VoD	< 1-5s	300 kbps – 2 Mbps & higher	Youtube
Long form VoD	< 5-30s	500 kbps – 6 Mbps & higher	Netflix, Hulu, Qiyi, HBOGO
Live Broadcast	< 5-10s	500 kbps – 6 Mbps & higher	WatchESPN, MLB
Linear Channel	< 60s	500 kbps – 6 Mbps & higher	DirectTV Live

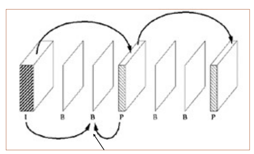
Video Data

- Unlike audio, video compression is essential:
  - Simply too much data - compression ratios from 50 to 500
- Takes advantage of spatial, temporal, and perceptual redundancy
- Temporal redundancy: use past frame(s) to predict future frames
  - Relies on the fact that successive frames are often similar
  - Resulting inter-frame dependencies are broken by inserting independently-encoded "I frames“ (sometimes called key frames)
    - Allows playback from middle of a file and error recovery
- Spatial redundancy: encoding of I frames is based on squares
  - Adjacent pixels often have similar colors
  - Also basis for motion prediction

Credit: http://www.icsi.berkeley.edu/PET/GIFS/MPEG_gop.gif

MPEG Video Coding

- Represents a family of coding standards
- Uses three types of frames
- I-frames are Intra-coded frames
  - Do not depend on any other frame
  - Appear periodically in the video
- P-frames are Predicted frames
  - They encode the difference relative to previous I or P frame
  - Appear periodically between successive I-frames
- B-frames are Bi-directionally predicted frames
  - Encode the difference relative to interpolation of previous or next I or P frame


Data dependency

Credit: http://www.icsi.berkeley.edu/PET/GIFS/MPEG_gop.gif

Terminology

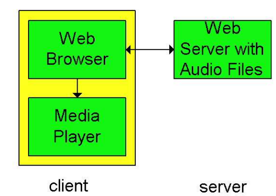
- Bitrate
  - Information stored/transmitted per unit time
  - Usually measured in kbps to mbps
  - Ranges from 200Kbps to 30 Mbps
- Resolution
  - Number of pixels per frame
  - 160x120 to 1920x1080 (1080p) to 4096x2160 (4K)
- FPS (frames per second)
  - 24, 25, 30, or 60

Challenges

- TCP/UDP/IP suite provides best-effort service - no guarantees on bandwidth, latency, or variance of packet delay
- Streaming applications delay of 5 to 10 seconds is typical and has been acceptable - but performance deteriorate if links are congested
- Real-Time Interactive requirements on delay and its jitter have been satisfied by over-provisioning (providing plenty of bandwidth) - what will happen when the load increases?

First Generation: HTTP Download

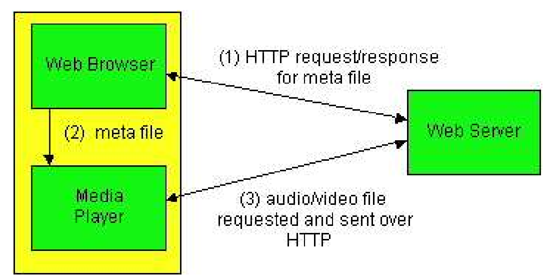
- Browser requests the object(s) and after their reception pass them to the player for display
  - No pipelining: video starts after entire video has been downloaded
- Simple architecture: browser and player are separate applications



First Generation Enhancement: HTTP Progressive Download (2)

- Alternative: set up connection between server and player
  - Player is in charge instead of the browser
- Web browser requests and receives a **Meta File** (a file describing the object) instead of receiving the file itself
- Browser launches the Player and passes it the *Meta File*
- Player sets up a TCP connection with Web Server and downloads or *streams* the file
  - Can start playing as long as it has enough frames

Meta file requests

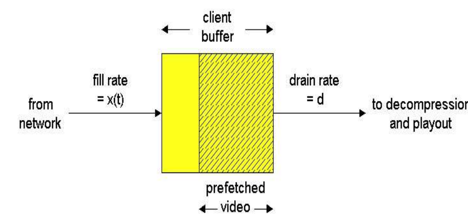


Buffering Continuous Media

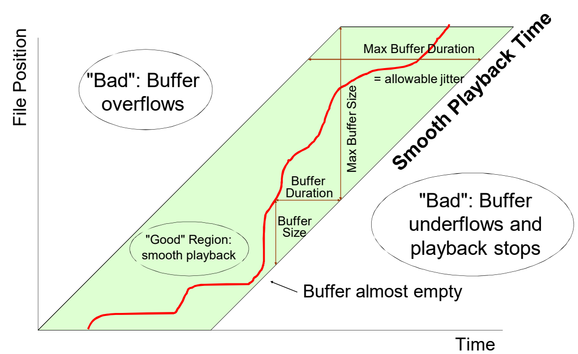
- Jitter = variation from ideal timing
- Media delivery must have very low jitter
  - Video frames every 30ms or so
  - Audio: ultimately samples need <1ns jitter
- But network packets have much more jitter that that!
- Solution: buffers
  - Fill buffer over the network with best effort service
  - Drain buffer via low-latency, local access

HTTP Progressive Download

- With helper application doing the download, playback can start immediately...
- Or after sufficient bytes are buffered
- Sender sends at maximum possible rate under TCP; retransmit when error is encountered; Player uses a much larger buffer to smooth delivery rate of TCP



Streaming, Buffers and Timing

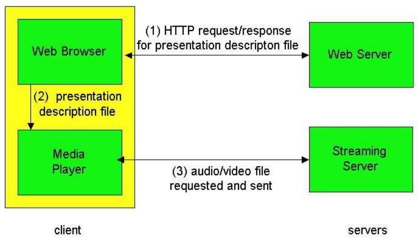


Drawbacks of HTTP Progressive Download

- HTTP connection keeps data flowing as fast as possible to user's local buffer
  - May download lots of extra data if user does not watch the entire video
  - TCP file transfer can use more bandwidth than necessary
- Mismatch between whole file transfer and stop/start/seek playback controls.
  - However: player can use file range requests to seek to video position
- Cannot change video quality (bit rate) to adapt to network congestion

2nd Generation: Real-Time Streaming

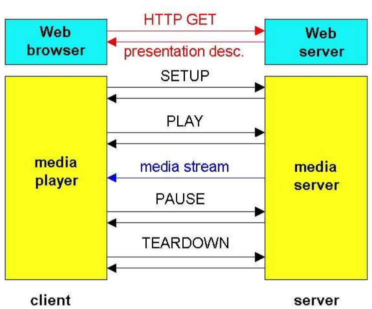
- Replace HTTP + TCP by a custom streaming protocol
  - Application layer protocols - gets around problems with HTTP
  - Allows a choice of UDP vs. TCP



Example: Real Time Streaming Protocol (RTSP)

- For user to control display: rewind, fast forward, pause, resume, etc…
- Out-of-band protocol (uses two connections, one for control messages (Port 554) and one for media stream)
- RFC 2326 permits use of either TCP or UDP for the control messages connection, sometimes called the RTSP Channel
- As before, meta file is communicated to web browser which then launches the Player; Player sets up an RTSP connection for control messages in addition to the connection for the streaming media

RTSP Operation



RTSP Exchange Example

C: SETUP rtsp://audio.example.com/xena/audio RTSP/1.0
Transport: rtp/udp; compression; port=3056; mode=PLAY

Client establishes video session

S: RTSP/1.0 200 1 OK
Session 4231

C: PLAY rtsp://audio.example.com/xena/audio.en/lofi RTSP/1.0
Session: 4231
Range: npt=0 (npt = normal play time)

Client starts the video At the beginning

C: PAUSE rtsp://audio.example.com/xena/audio.en/lofi RTSP/1.0
Session: 4231
Range: npt=37

Client pauses the video

C: TEARDOWN rtsp://audio.example.com/xena/audio.en/lofi RTSP/1.0
Session: 4231

Client ends the session

S: 200 3 OK

RTSP Media Stream

- *Stateful* Server keeps track of client's state
- Client issues Play, Pause, ..., Close
- Steady stream of packets
  - UDP - lower latency
  - TCP - may get through more firewalls, reliable

Drawbacks of RTSP, RTMP

- Web downloads are typically cheaper than streaming services offered by CDNs and hosting providers
  - More complex servers
  - Video was not commodity traffic (at the time) – low volume
- Streaming (non-HTTP) often blocked by routers
- UDP itself often blocked by firewalls
- HTTP delivery can use ordinary proxies and caches
- Conclusion: hard to adapt the Internet to streaming applications
- Alternative: adapt media delivery to the Internet

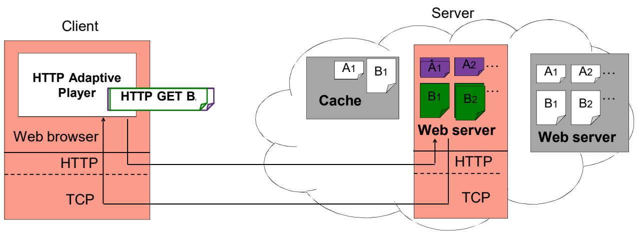
3rd Generation: HTTP Streaming

- Other terms for similar concepts: Adaptive Streaming, Smooth Streaming, HTTP Chunking
- Client-centric architecture with stateful client and stateless server
  - Standard server: Web servers
  - Standard Protocol: HTTP
  - Session state and logic maintained at client
- Video is broken into multiple chunks
- Chunks begin with a keyframe so each chunk is independent of other chunks
- A series of HTTP progressive downloads of chunks
- Playing chunks in sequence gives seamless video

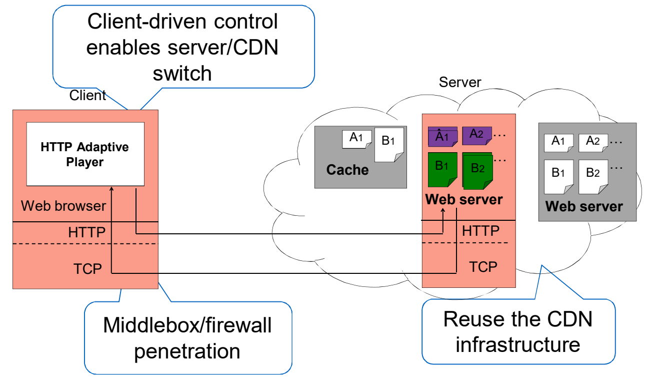
Adaptive Bit Rate with HTTP Streaming

- Encode video at different levels of quality/bandwidth
- Client can adapt by requesting different sized chunks
  - I.e., if downloading a chunk takes too much time, choose a lower bit rate for the next chunk
- Chunks of different bit rates must be synchronized
  - All encodings have the same chunk boundaries and all chunks start with key frames, so you can make smooth splices to chunks of higher or lower bit rates

HTTP Chunking Protocol



Reasons for Wide Adoption

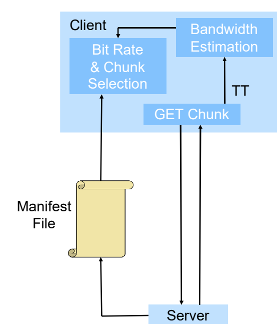


Bit Rate Selection

- Each chunk represents a certain play time
  - Transfer time of chunk must be shorter than the play time
- Learn from previous chunk transfers what the available bandwidth is on network path from server to client
  - Use this to estimate predicted transfer time (PTT) of future chunks
  - General approach to adapting bit rate:
    - Decrease bit rate if PTT is close to/higher than play time
    - Increase bit rate if PTT is significantly lower than play time
  - Many variants: what thresholds, hysteresis, etc.

Bit Rate Selection - Implementation

- Manifest file lists list multiple URLs for each chunk, one for each different bit rates
- Client estimates PTT for the chunk based on previous transfer times
- Selects best bit rate
  - PTT is below threshold
  - QoE considerations
  - Buffer status, …



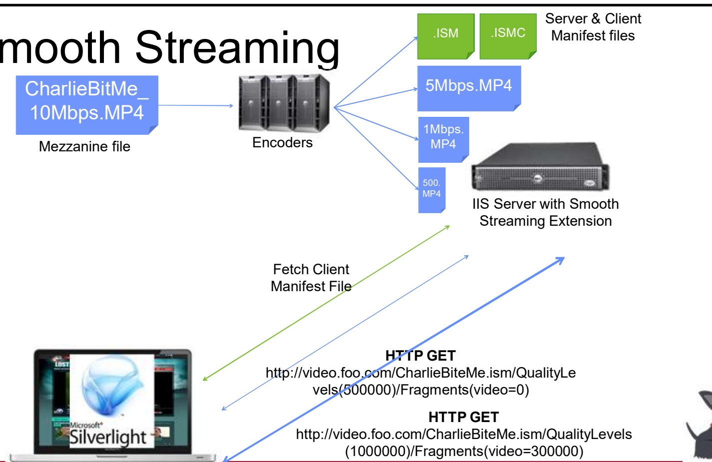
Advantages of HTTP Streaming

- Easy to deploy: it's just HTTP!
  - Work with existing caches/proxies/CDN/Firewall
- Very cost effective
  - Uses commodity web infrastructure
- Fast startup by downloading lowest quality/smallest chunk
- Bitrate switching is seamless
- Many small files
  - Small with respect to the movie size
  - Large with respect to TCP
    - 5-10 seconds of 1Mbps – 3Mbps → 0.5MB – 4MB per chunk

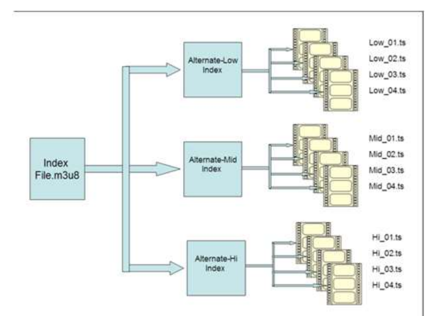
Example of HTTP Streaming Protocols

- Apple HLS: HTTP Live Streaming
- Microsoft IIS Smooth Streaming: part of Silverlight
- Adobe HDS: HTTP Dynamic Streaming
- *DASH: Dynamic Adaptive Streaming over HTTP*

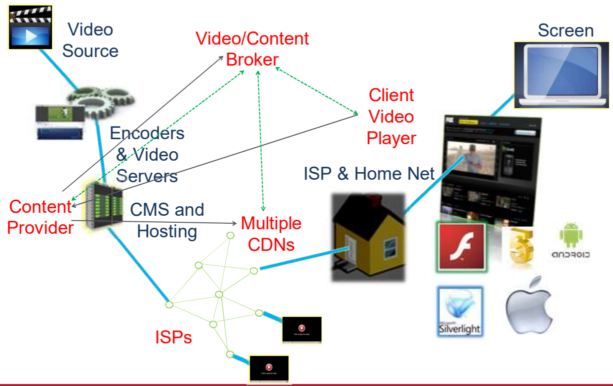
Smooth Streaming



Example of HLS Meta Data



Internet Video Today: Video Brokers

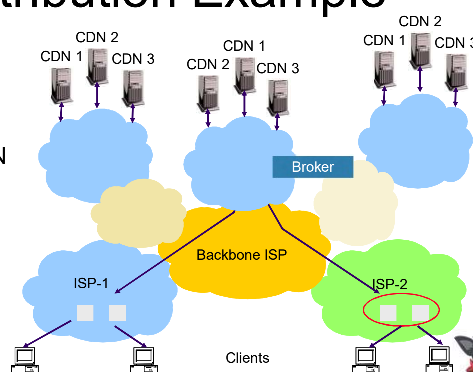


Video Broker

- Content broker has agreement with multiple CDNs
  - Each CDN has many geographically distributed data centers
  - Uses its own solution for server selection
- Video broker selects what CND each client should use
  - Can also specify bit rates, other parameters
- Brokers use a “big data” approach:
  - Get reports from each client on the performance they experience, e.g., bit rates, bandwidth, latency, ..
  - Clients in same part of the network should have similar experience
  - Used to give instructions to all clients

Video Distribution Example

- Huge number of clients effectively sample paths and CDNs
- Input for broker to pick CDN
  - But CDN picks the cluster!
- Many other considerations
  - Cost, CDN load, ..
- Broker, ISPs, CDNs each make decisions without explicit coordination



Important Points

- NOT all contents are the same
- Video is fundamentally different from transaction traffic
- In the last few years, we have seen a video revolution
  - video is more than 60% Internet traffic today,
  - video will be more than 90% Internet traffic in 2-3 years
  - Next: premium video (4K), 3D video, mobile video,…
- Solution builds on commodity web technology
  - Cost effective, least likely to run into problems (firewalls, ..)
- Focus is on: Quality, scalability, mobility, security, usability
  - Chunk-based bit rate adaptation is a key technology
  - Massive replication to achieve scalability