CSCD 433/533
Advanced Networks
Winter 2017

Lecture 15
Multimedia Applications

Reading: Book 6.4 and 7.4

Topics

- Multimedia
  - Attributes
  - Streaming, Stored
  - Steaming Live
  - Real-time Interactive
  - RTSP
  - Microsoft Smooth Streaming

Introduction

- Multimedia Applications
  - Delay sensitive
    - Delay is a problem
    - We notice delay of 100 ms
  - Loss tolerant
- Other Applications
    - Web, FTP, email, SSH
      - Delay annoying but data integrity most important
      - Can't Lose any data!

Introduction to Multimedia

- Delivery of Multimedia is challenging in today's Internet
  - Want timely delivery of content
  - Want quality delivery of content
  - Want content to get there – no interruptions of display

Demands of Multimedia

- Can existing protocols UDP and TCP guarantee end-to-end delays for packets?
  - No. Not generally.
  - Need something more than typical transport protocols
  - Multimedia was not an original application of historic Internet
- Yet, in spite of “best effort” delivery
  - Multimedia is successful

Example:

- Skype – 7 million users on-line at any time
- WOW - 11-12 million monthly users

Characteristics of Multimedia Applications

Application	Data loss	Throughput	Time Sensitive
File transfer	no loss	elastic	no
E-mail	no loss	elastic	no
Web documents	no loss	elastic	no
Real-time audio/video	loss-tolerant	audio: 5kbps -1Mbps	yes, 100's ms
		video:10kbps-5Mbps	yes, few secs
Stored audio/video	loss-tolerant	same as above	yes, 100's ms
Interactive games	loss-tolerant	few kbps up	yes and no
Instant messaging	no loss	elastic	

elastic – no hard requirement

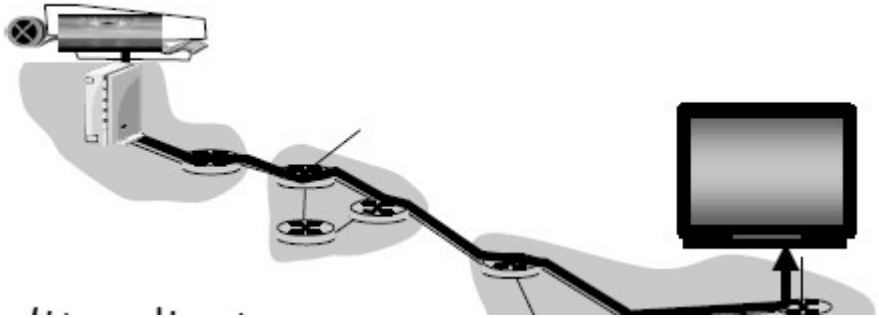
Three General Types of Multimedia Applications

1. Streaming Stored Audio/Video
2. Streaming Live Audio/Video
3. Real-time interactive Audio/Video

Examples of these?

1. Streaming Stored - YouTube, CNN – Uses Adobe Flash Player, Progressive download
2. Streaming Live - Internet Radio, IPTV, Videocams
3. Real-time interactive – Microsoft Net-meeting, Polycom, Skype voice and video

Multimedia Types



Characteristics

1. Streaming Stored

- Play begins seconds after file starts to download
- Doesn't have to wait for entire file to download
- Continuous Playout
  - **Plays according to original timing of recording**
- Flash, Apple Live Stream, others

VCR-like functionality: pause, rewind, FF, push slider bar - works

- 10 sec initial delay OK
- 1-2 sec until command effect OK
- RTSP (Real Time Streaming Protocol) often used

Streaming Stored Audio and Video Summary

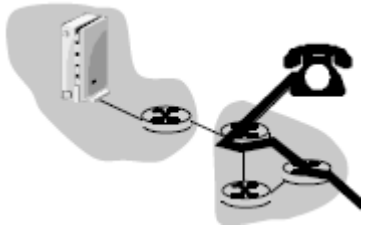
- Client-server system
  - Server stores audio and video files
  - Client requests files, plays them as they download, and performs VCR-like functions (e.g., rewind and pause)
- Playing data at right time
  - Server divides data into segments
  - … labels each segment with timestamp or frame id
  - … so client knows when to play data
- Avoids starvation at client
  - Data must arrive quickly enough
  - … otherwise the client cannot keep playing

Multimedia Types

2. Streaming Live

- Live TV or radio from anywhere in world
- Since, not stored, can't fast forward, but can
  - Pause, rewind, stop
- Delays up to 10's of seconds can be tolerated
- Less stringent requirements than real-time Interactive applications

Multimedia Types



3. Real-time Interactive

- Communicate in real time
- Most stringent time requirements

- Internet telephony/Video conferencing
  - Skype, Net meeting, Skype Live Video, Polycom
  - Delays < 150 ms not perceived
  - Delays between 150 ms and 400 ms acceptable
  - Delays > 400 ms can result in frustrating, if not completely unintelligible voice conversations

  - **Example of Live WebCam:**

https://www.skylinewebcams.com/en/webcam/italia/veneto/venezia/rio-di-palazzo.html

How to Deliver Multi-media Content

- What tricks can we do to accommodate multimedia?
  - **Send audio/video over UDP, not TCP**
  - **Delay playback by 100 ms or more**
    - Reduces effect of network induced jitter
  - **Time stamp packets at sender**
  - **Identify packets** - Number them
  - **Stored audio/video**
    - Pre-fetch data during playback when client storage and extra bandwidth available
  - **Send redundant information** to mitigate packet loss

How to Deliver Multi-media Content

- Specifically
  - Sequencing of packets
  - Time Stamping
  - **How does it help us?**

1	2	3	...

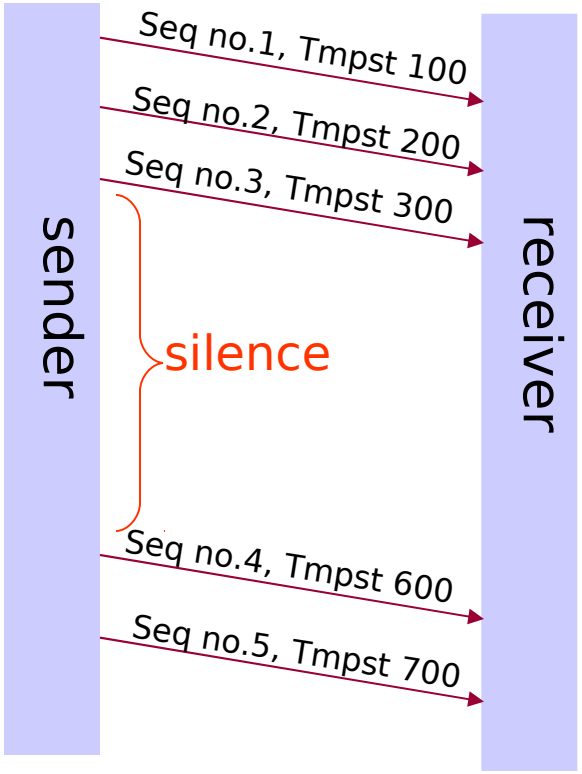
10ms 15 ms 18 ms 25 ms

Details Streaming Stored Content

How does Sequence number and Timestamp help ?

Audio silence example:

- Consider audio data
  - What should sender do during silence?
    - Does not send anything
  - Why might this cause problems?
    - Receiver cannot distinguish between loss and silence



Solution:

After receiving no packets for a while, next packet received at the receiver will reflect a big jump in timestamp, but have the correct sequence number

Thus, receiver knows there was a gap

Streaming Stored Audio and Video

- Server sends file to client on demand
  - Client requests stored compressed multimedia applications
  - Resides on Servers
  - Either Web servers or Streaming Media servers tailored to multimedia streaming

Streaming Stored Audio and Video

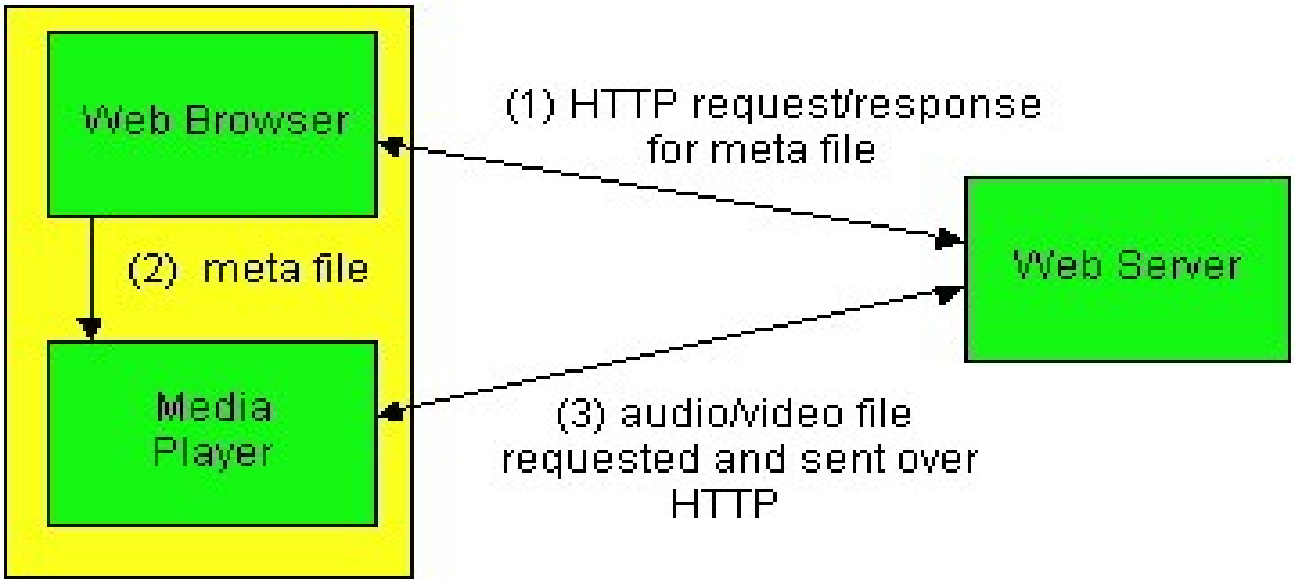
1. First type of Delivery - Web Server Delivery

- One way to deliver streaming content
- With Web Server, multimedia file sent encapsulated in HTTP response over TCP
  - So, you ask for a movie file ... Want to watch CNN Olympics recap
    - Browser's job, request metafile from server
    - Has URL of file(s), type of encoding for player
    - **Metafile** passed to media player
    - **Media player** contacts Web Server,coordinates playing of file
    - Web Server sends packets of media file to player

Initiating Streams from Web Servers

Media Player in addition to Web browser

- Web server returns a meta file describing the object
- Browser launches media player and passes meta file
- The player sets up its own connection to Web server



Streaming Server

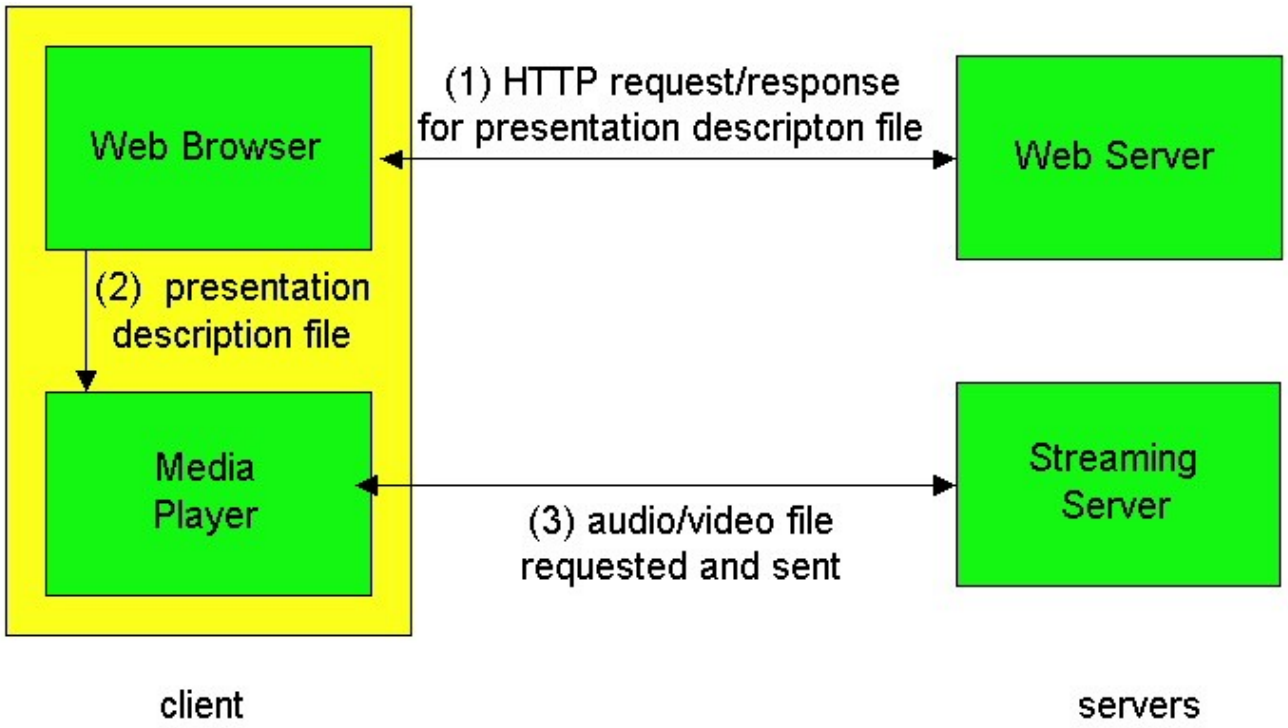
2. Second Type of Delivery - Two Servers

- Web server serves **Web pages** and **meta file**
- Streaming Server serves audio/video files
- Can be run on same end system or different systems

- Difference from just Web Server Architecture
  - Media player gets file from streaming server
  - Media player and streaming server can use own protocols
    - Not limited to HTTP and TCP or UDP!!!!

Using a Streaming Server

- Typically Uses other Protocols
  - Web server returns a meta file describing the object
  - Player requests the data using a different protocol



Streaming Server

Lots of options with Streaming Server

1. Audio/Video – Over UDP

- Sent at a constant rate
  - Equals drain rate at receiver
  - Encoded compression rate of media
  - As soon as client gets compressed media from network, decompresses it and plays it back

2. Same as First Option, Delay playout for 2 – 5 secs

- Want to eliminate network induced jitter
- Puts it into buffer, After few seconds of pre-fetched media, begins to drain the buffer

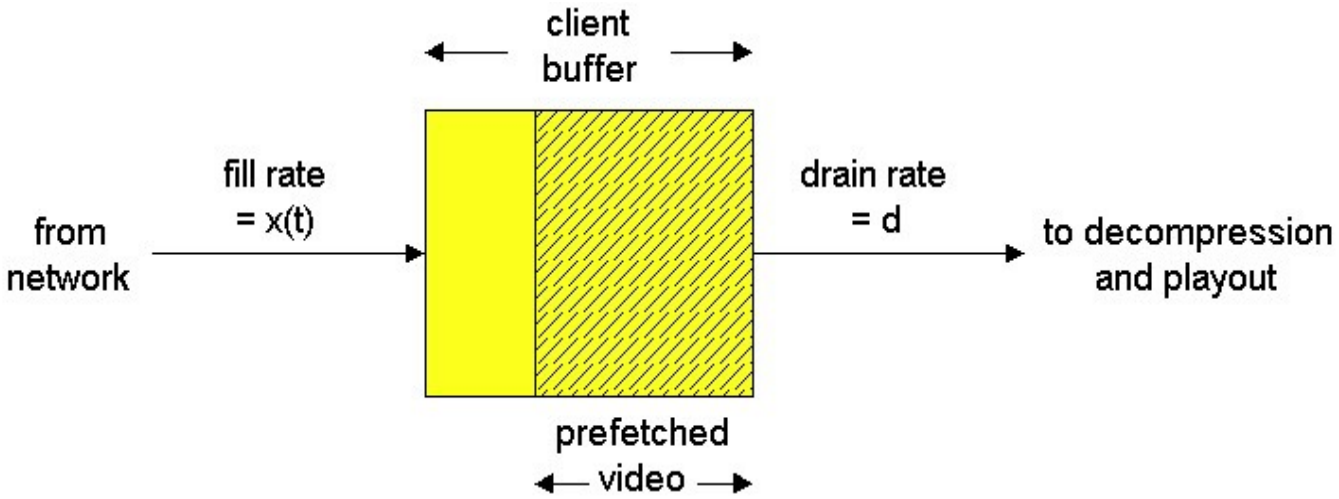
Streaming Server

3. Media sent over TCP

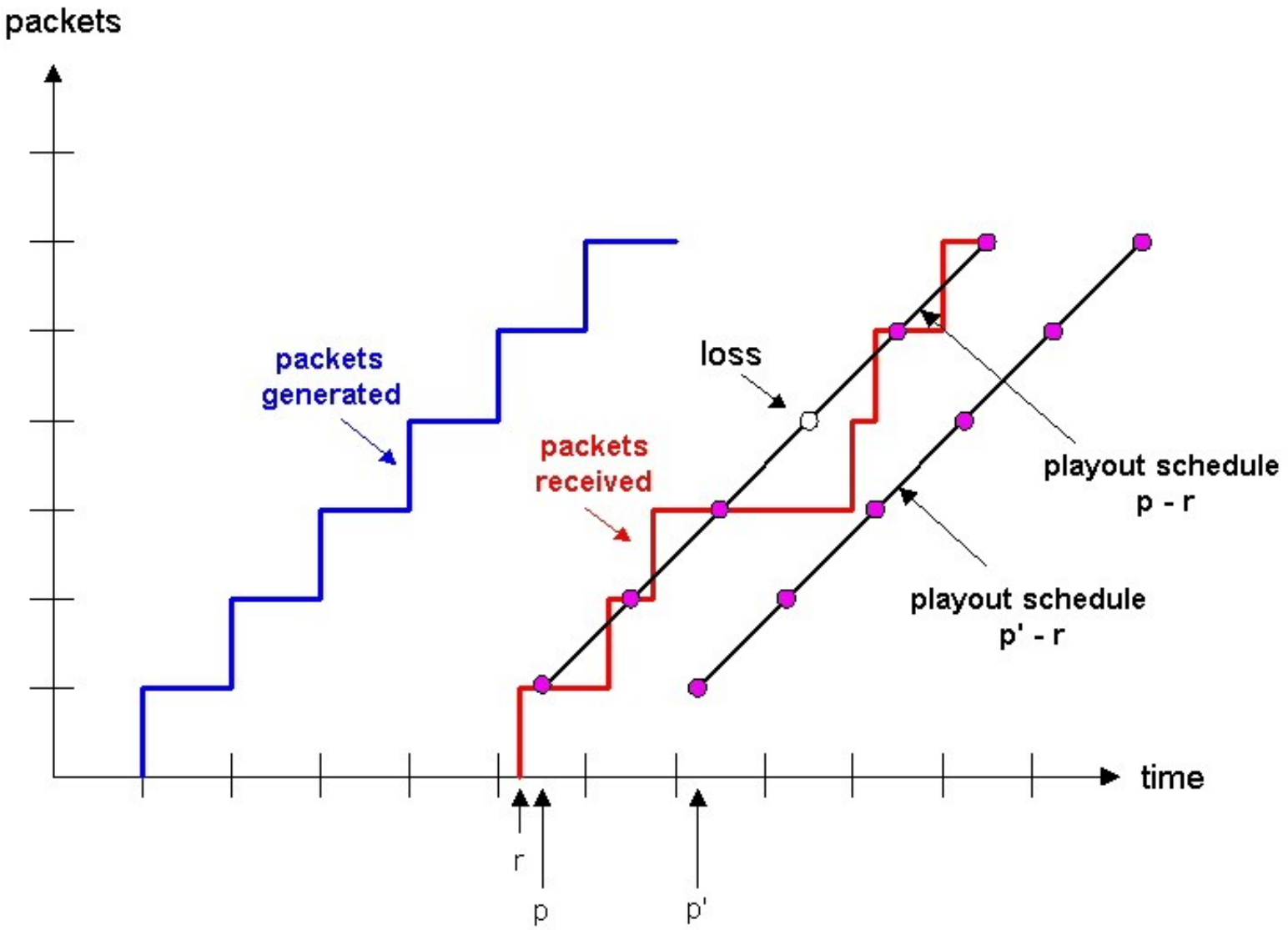
- After 2-5 secs delay, player reads buffer does decompressions and playback
- Since TCP retransmits lost packets, potential to provide better sound quality than UDP
- But, fill rate now fluctuates with time due to TCP congestion control and window flow control
- Can occasionally result in starvation of media

Playout Buffer

- Client buffer
  - Stores data as it arrives from server
  - Play data for user in a continuous fashion
- Playout delay
  - Client typically waits a few seconds to start playing
  - … to allow some data to build up in the buffer
  - … to help tolerate some delays down the road



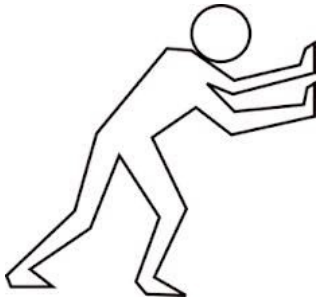
Influence of Playout Delay



More Classifications

Another way to view technologies

Who controls them!!!



Push Based Protocols

Server controls the connection

May or may not adapt to what is happening on the client end

Pull Based Protocols

Client controls the connection and feeding of media

"Adaptive" can be applied to either one

Different Classes Streaming

1. Push Based Protocols – Server Controls

Normal Streaming

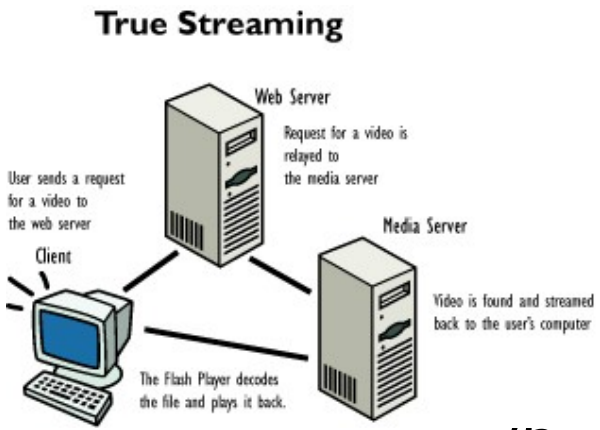
- Connection Established

- Server streams packets to client until session is torn down
- Server maintains connection entire time
- Listens for commands from client via a session control protocol
- RT Streaming Protocol, RT Messaging Protocol examples

Adaptive Streaming

- Client reports back to server

who in turn can decide to switch to higher/lower bitrate stream



Different Classes Streaming

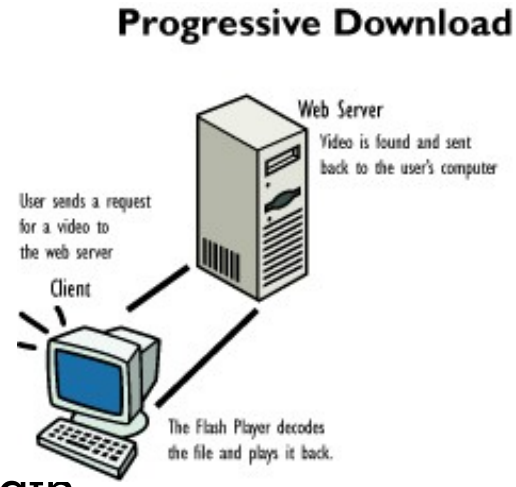
2. Pull Based Protocols

Client Controls

Client requests content

Connection Established

- Server responds to client request
- Depends on client and available bandwidth



Example: Progressive Download

- Once buffer fills to a certain level, client plays media
- If network conditions degrade, rate may fall behind playback rate, buffer underflow may result
- Content divided into short duration media segments
- Encoded at various bit rates, **client picks the rate !!!**

Example: Microsoft Smooth Streaming, Apple Http Live Streaming

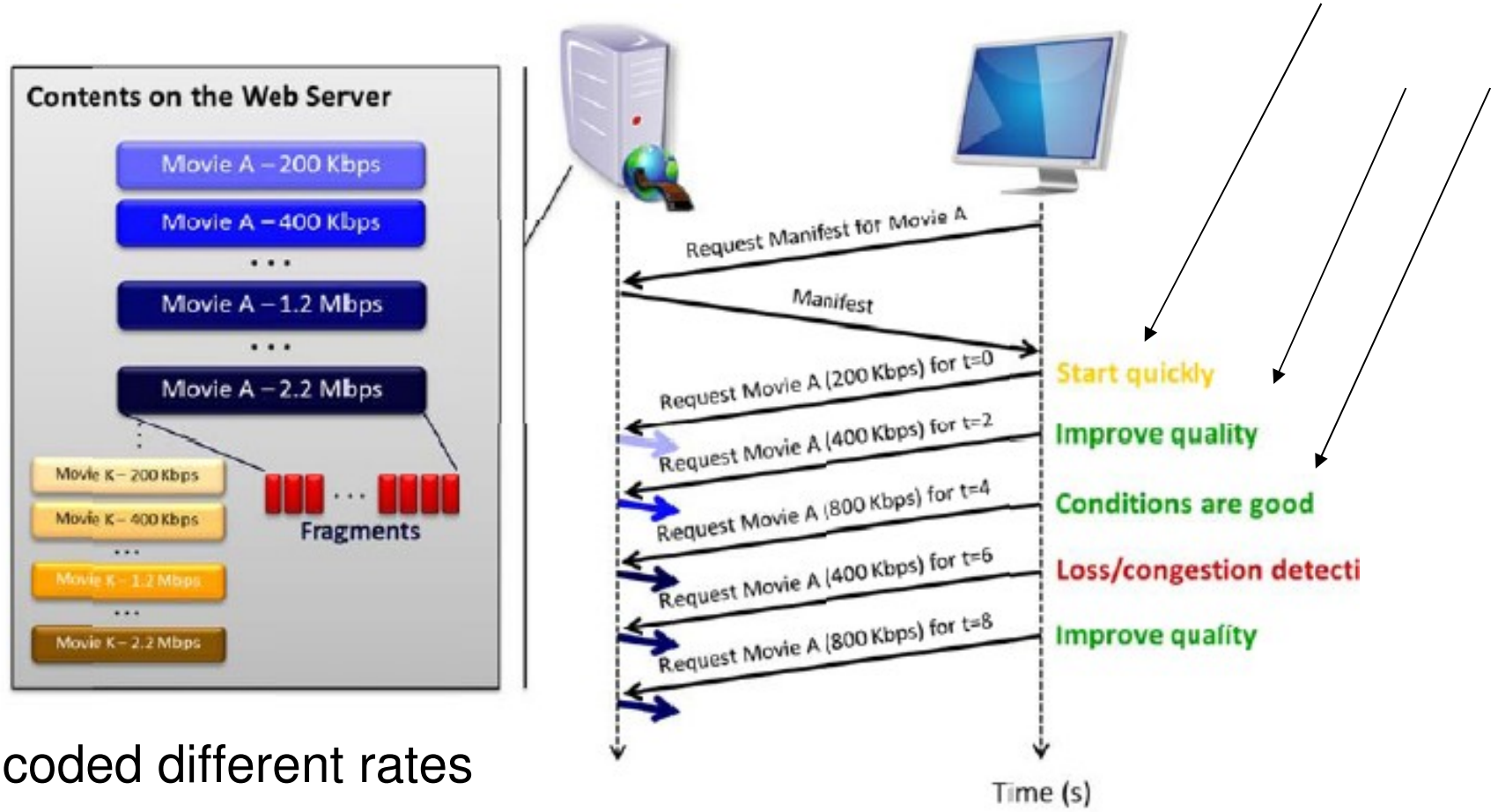
_Can have Adaptive Streaming for Pull based too!

Adaptive Streaming

- Benefits??? Its Adaptive
  - Can change in response to changes in bandwidth and CPU usage
  - Operates transparent to user
    - Switch bitrate behind scenes
  - Multiple bitrate encodings of same source
    - Can accommodate different devices with varying needs for speed

Pull Based Bitrate Adaptation

Bitrate adaptation works with Pull based !!!



Encoded different rates

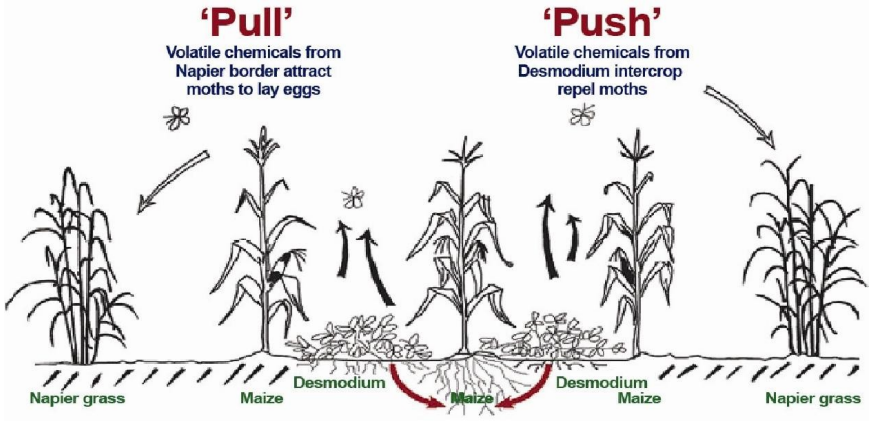
Figure 2: Example bitrate adaptation in pull-based adaptive streaming.

RTSP – Real Time Streaming Protocol

What does it do and why do we need it?

- Users want to turn Internet into one huge remote controlled application, like your TV!
  - RTSP allows this type of functionality

- **Application layer** protocol, **Push Based**
- Allows Users to control display:
  - Rewind, fast forward, pause, resume, repositioning, etc…



Example of Push RTSP

vs.

Example of Pull Windows Smooth Streaming

RTSP - Real Time Streaming Protocol

- What does it do?
  - Allows media player to control transmission of media stream
  - Controls Include
    - Pause/resume, Reposition. Fast Forward
  - RTSP – Out of band protocol
    - What does that mean?
    - Sends separate messages from the data
    - Uses a separate port, 544 – either UDP or TCP
  - Media stream considered “in-band”

RTSP – Real Time Streaming Protocol

What it doesn't do

- Does not define **how** audio/video is encapsulated for streaming over network
- Does not restrict how streamed media is transported
  - Can be transported over UDP or TCP
- Does not specify how media player buffers audio/video

RTSP Protocol

- Example - Request to View Movie Twister
  1. Browser requests description file from Web Server
  2. Gets file(s) plus several media directives for synchronization of media files
  3. Each file reference begins with: rtsp://
  4. States that audio/video streams be played in parallel and with lip sync
  5. Web server sends description file in HTTP response

Metafile Example
Presentation Description File

```
<title>Twister</title>
<session>
        <group language=en lipsync>
                <switch>
                    <track type=audio
                          e="PCMU/8000/1"
                          src = "rtsp://audio.example.com/twister/audio.en/lofi">
                    <track type=audio
                          e="DVI4/16000/2" pt="90 DVI4/8000/1"
                          src="rtsp://audio.example.com/twister/audio.en/hifi">
                </switch>
                <track type="video/jpeg"
                          src="rtsp://video.example.com/twister/video">
        </group>
</session>
```

Plays in parallel lipsync, part of same group

Offers choice of sound quality

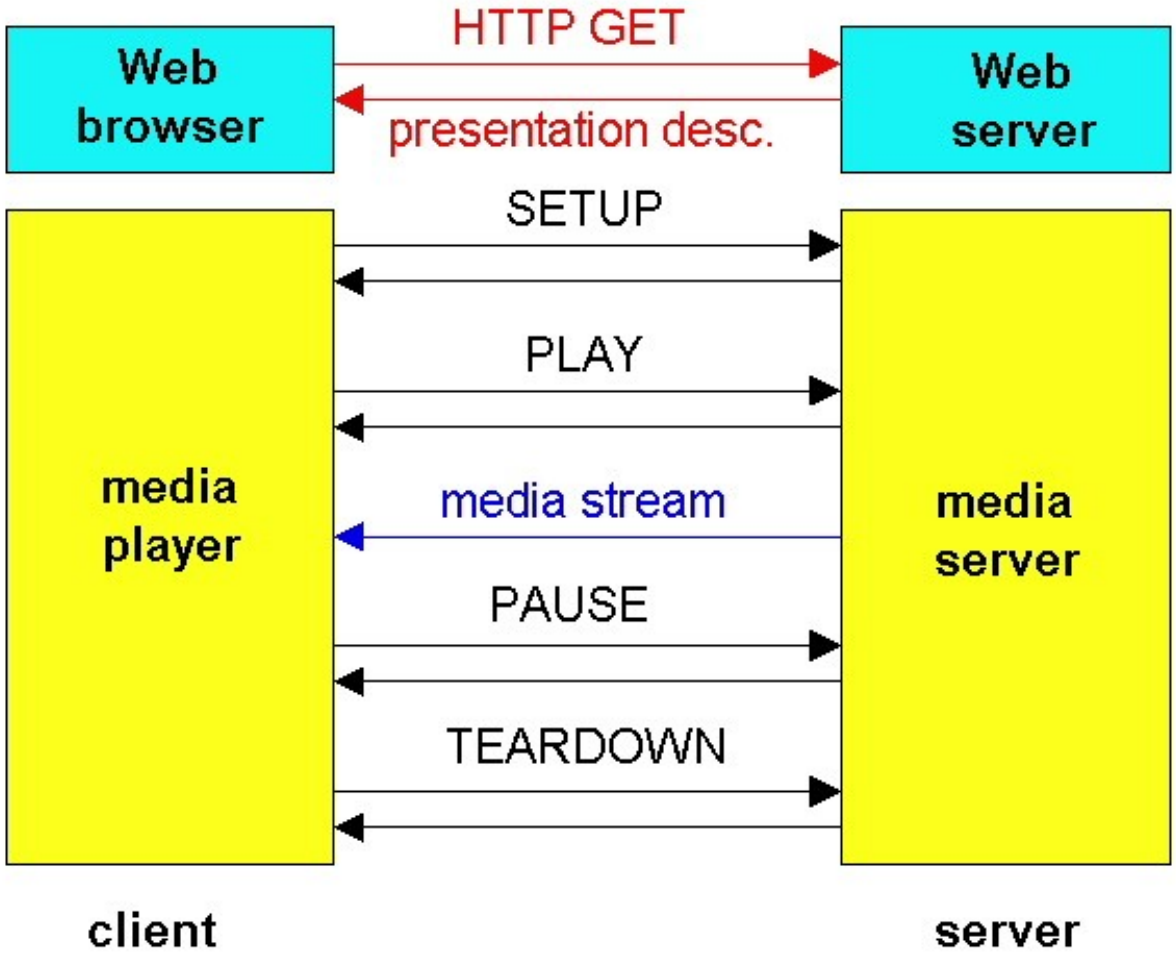
RTSP Protocol

Browser gets message with encapsulated Description file

1. Browser then starts media player based on content type of message
2. Player and server then send each other RTSP messages to coordinate setup, media play and teardown

See next slides …

RTSP Operation



RTSP Exchange Example

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

S: RTSP/1.0 200 1 OK
ID → Session 4231

Client initiates setup request
Includes client port number,
UDP/RTP protocol

C: PLAY rtsp://audio.example.com/twister/audio.en/lofi RTSP/1.0
Session: 4231
Range: npt=0-

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

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

S: 200 3 OK

Pull Based Microsoft Smooth Streaming

- Creates multiple files with different bitrate fragments
  - Modified MP4 file format
- Fragments are about 2 seconds long
- Each file contains fragments at a specific bitrate

Works like this …

1. Client gets a client manifest file
   Describes codecs, video resolution
2. After downloading manifest file
3. Client makes http requests for movie fragments
4. Server locates corresponding MP4 file fragments and sends it

Microsoft Smooth Streaming

Modified structured MP4 file of movie fragments

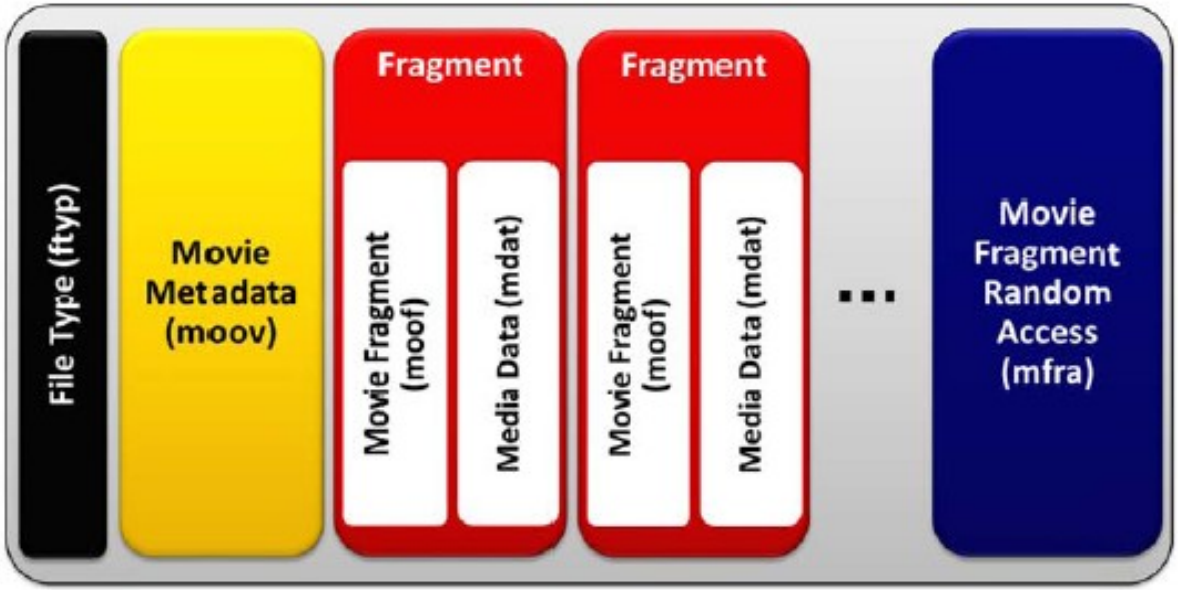


Figure 3: Fragmented MP4 file format (Reproduced from (4)).

Microsoft Smooth Streaming

- HTTP Get requests specify individual fragments
- Each fragment downloaded via unique HTTP request – needs the bitrate and the time offset

```
GET /sample/v_720p.ism/QualityLevels(1500000)/Fragments(video=160577243) HTTP/1.1
```

Figure 4: An example HTTP request for a downloading a fragment.

Comparing Push vs Pull Protocols

- RTSP - Push
- Server sends data to client a in real-time bitrate
  - If encoded at 500 kbps sent at 500 kbps
  - Not flexible in rate
- Server sends enough data to fill client buffer
  - Between 1 and 10 seconds worth
  - If client pauses for 10 minutes, still only sends 5 seconds worth
- Server connected entire time, resource intensive
- Microsoft SS - Pull
  - Client creates file chunks as they are needed
  - File chunks can be cached same as any other HTTP file, take advantage of Content providers, Akamai
  - No dedicated server connection listening, scales better

Summary

- Seems to be a trend of going back to HTTP for content delivery
- Going away from proprietary multimedia protocols like RTMP and RTSP
- This can be seen by Youtube using progressive delivery
- Adaptive delivery is offering greater flexibility in delivering content for both Pull and Push based protocols

References

Nice Overview of Protocols

http://www.cc.gatech.edu/classes/AY2011/cs7270_spring/watching-video-over-Web.pdf

- Microsoft Smooth Streaming

  http://jmvm.vse.cz/wp-content/uploads/2011/05/t_IIS_Smooth_Streaming_Technical_Overview.pdf

- IETF Document on RTSP

  http://www.ietf.org/rfc/rfc2326.txt

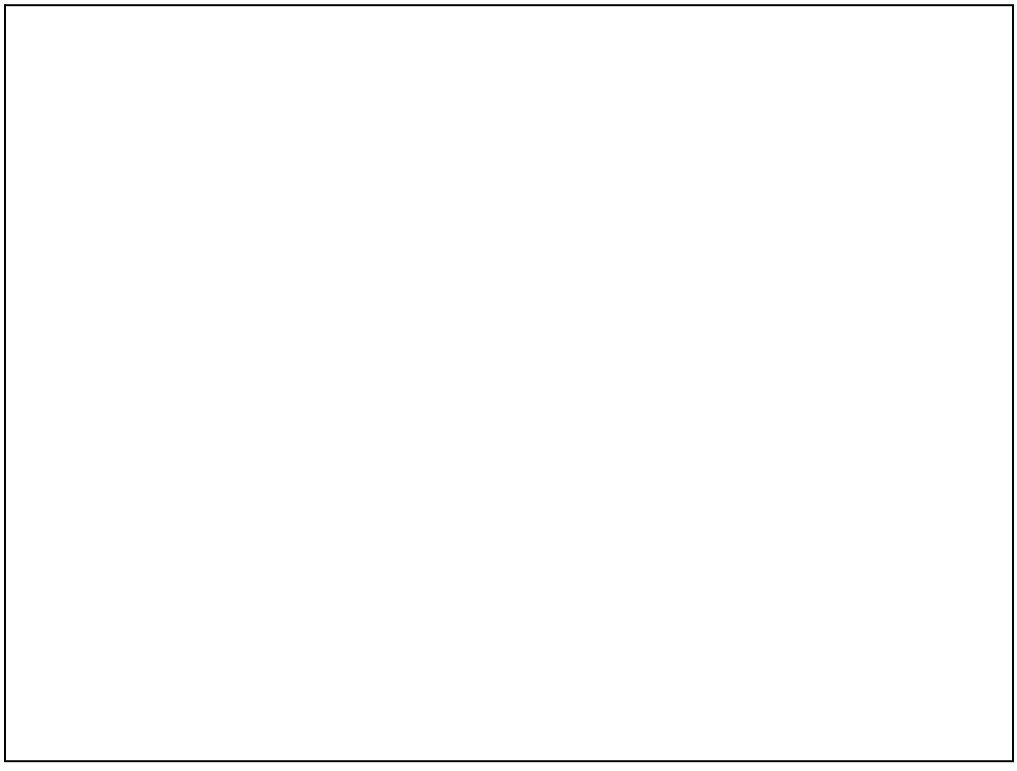
- Wikipedia Page on RTSP

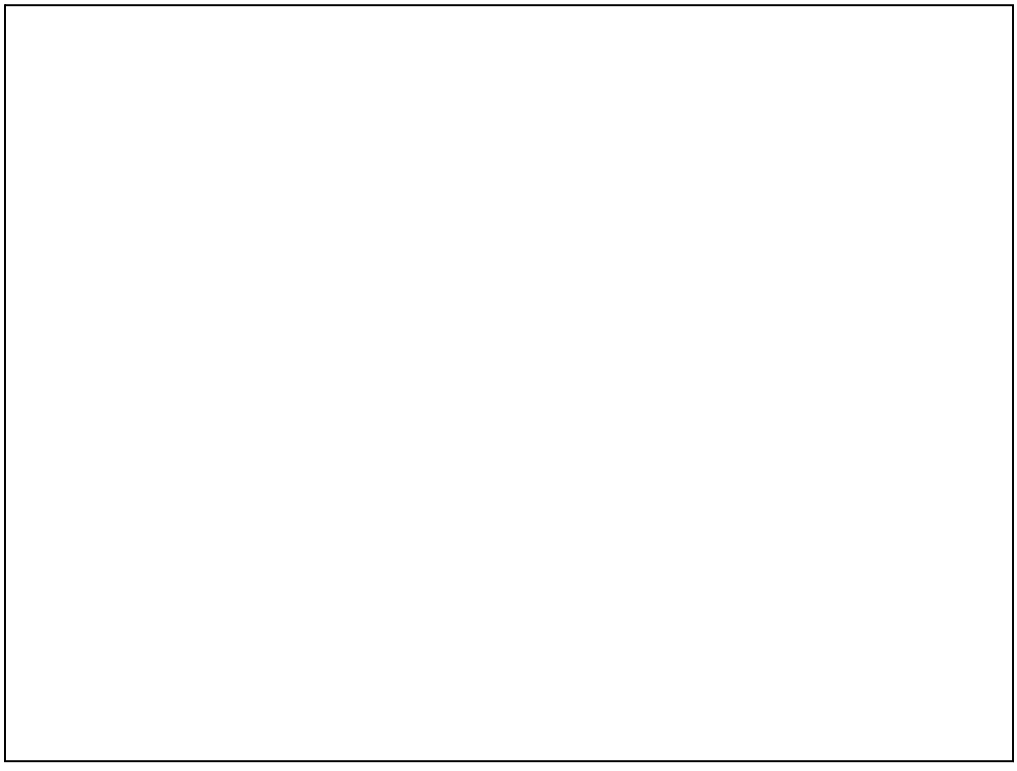
  http://en.wikipedia.org/wiki/Real_Time_Streaming_Protocol

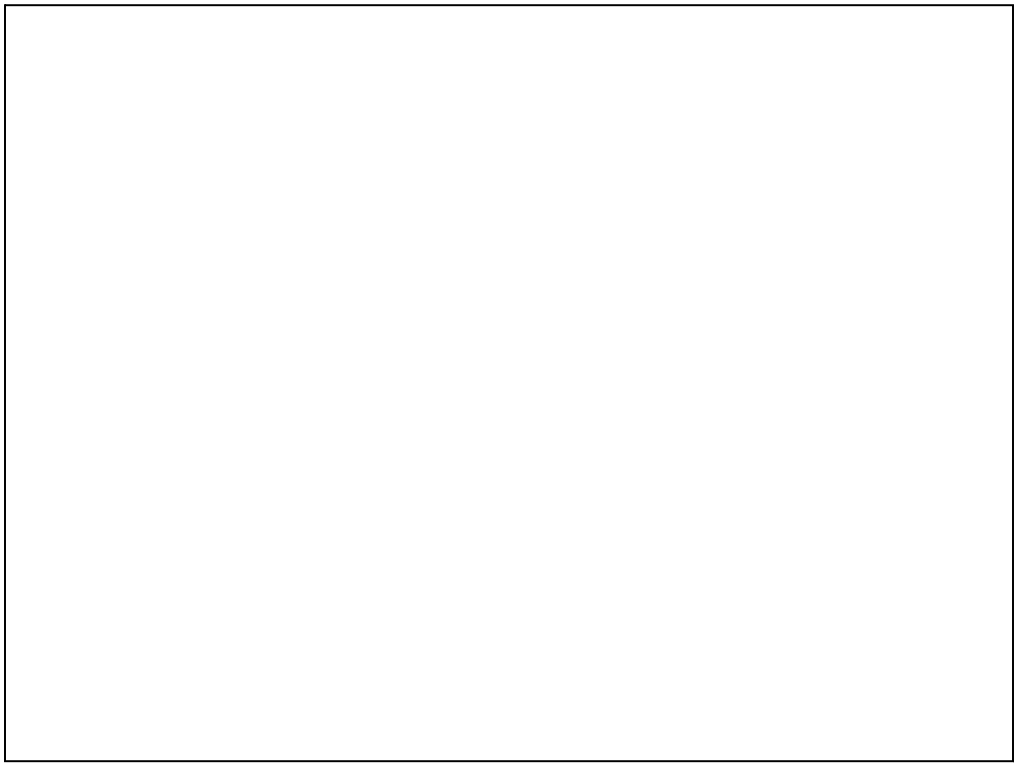
- Page on RTSP by Henning Schulzrinne

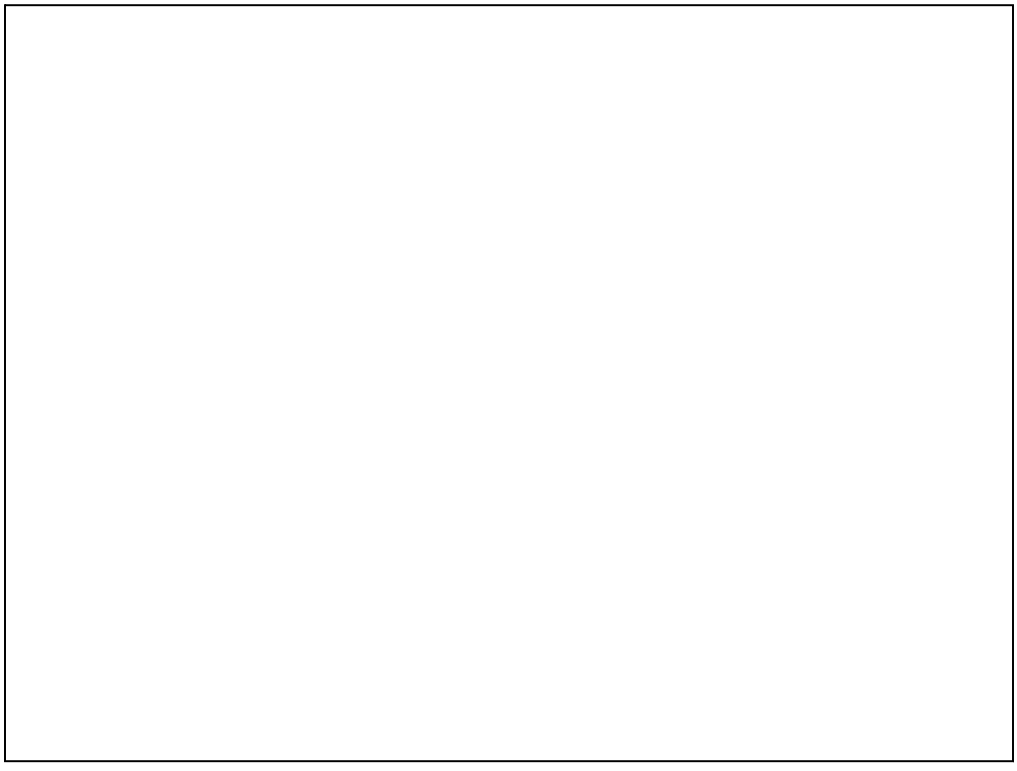
  http://www.cs.columbia.edu/~hgs/rtsp/

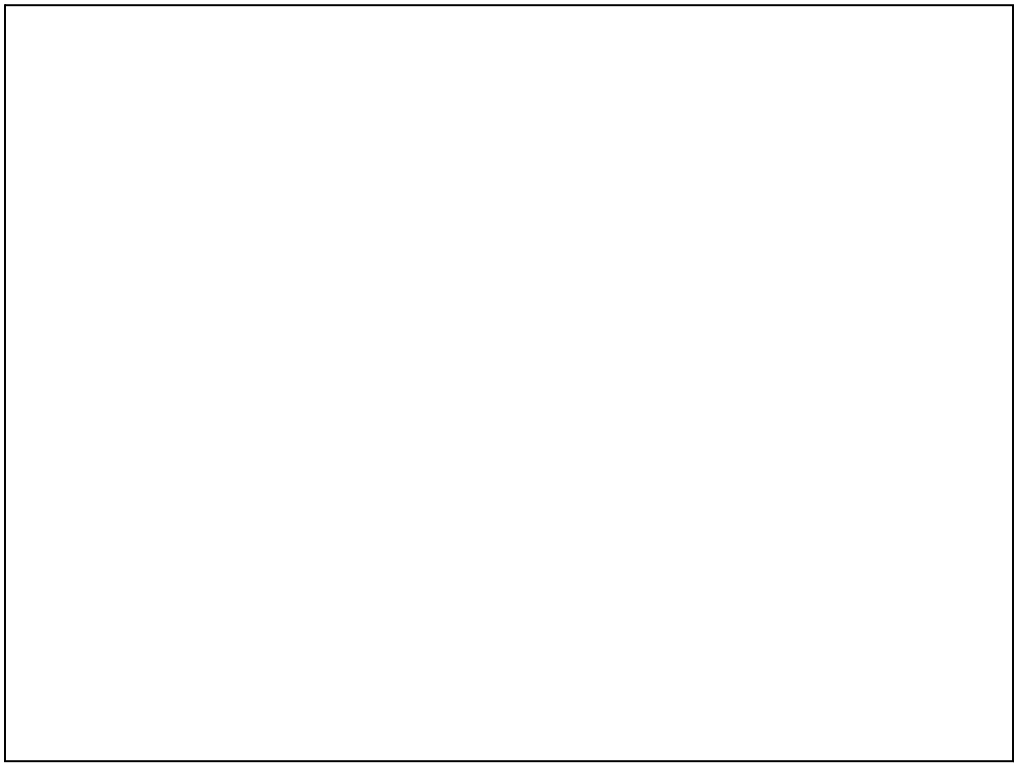
End

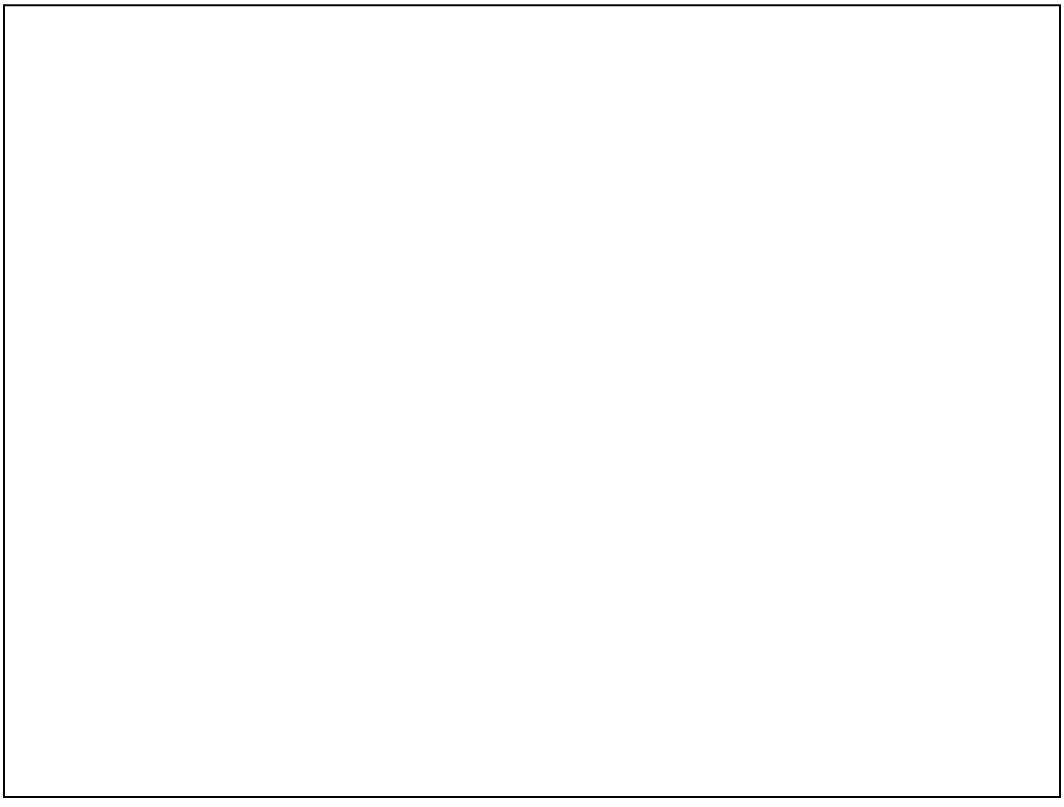


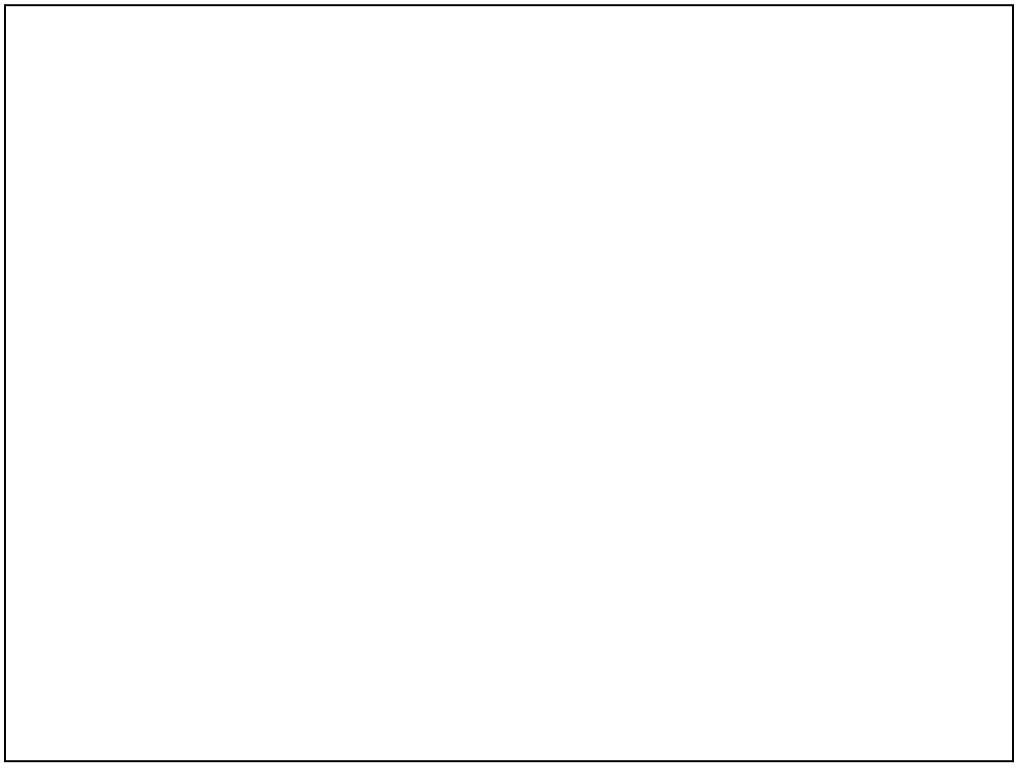


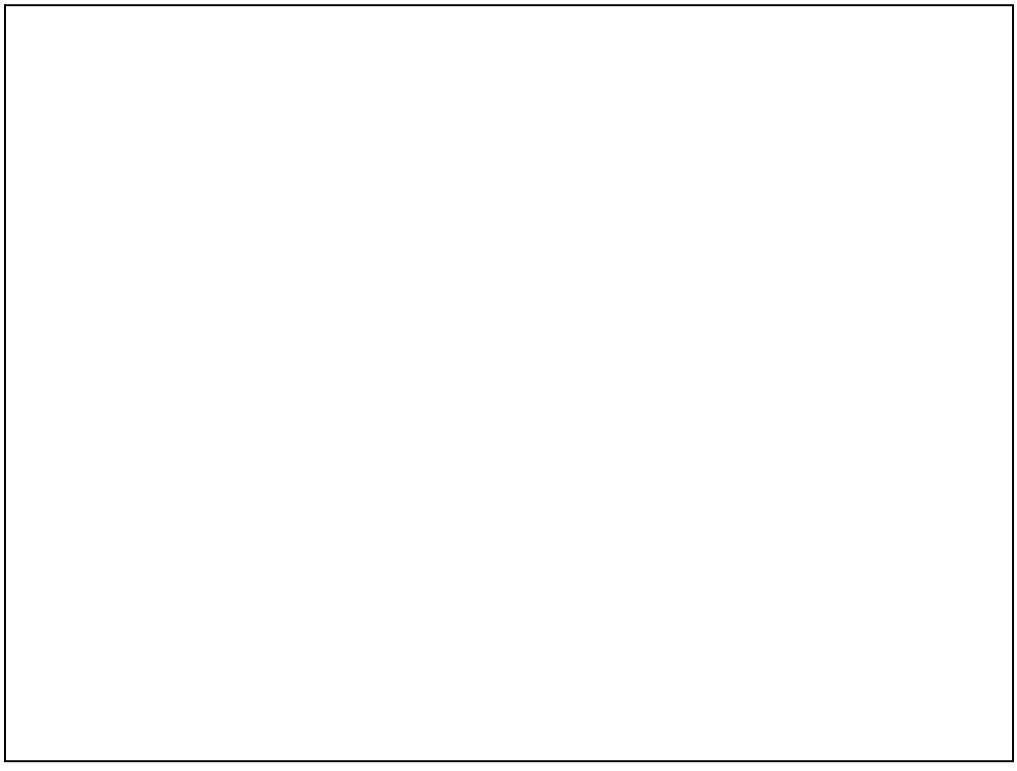


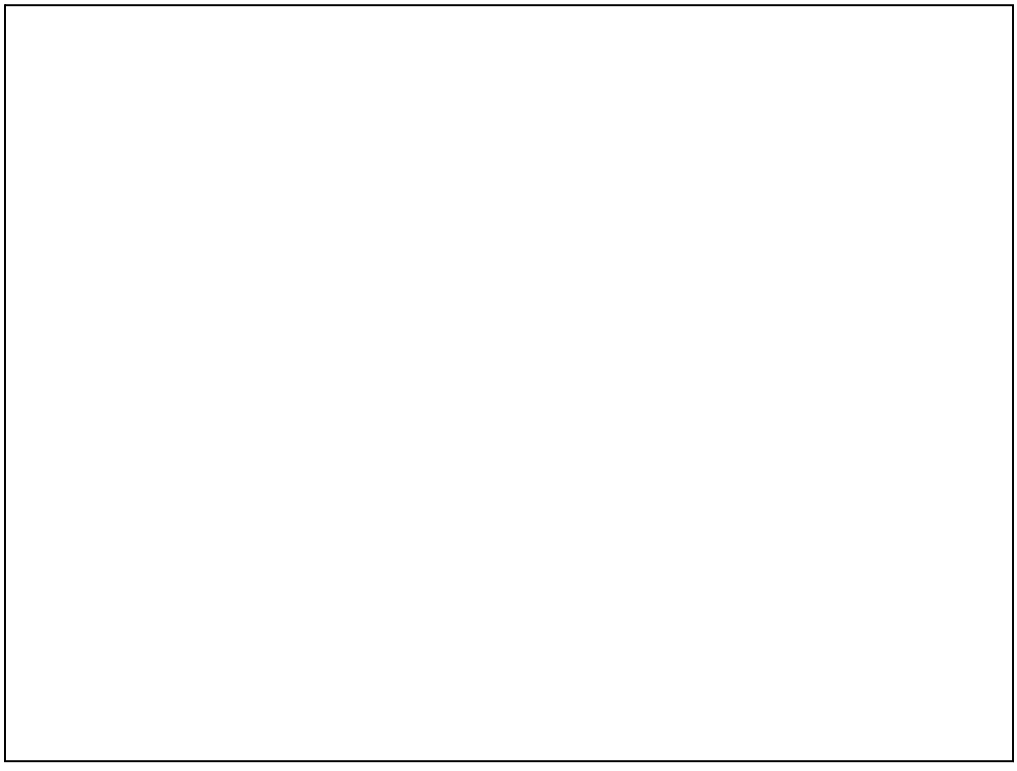


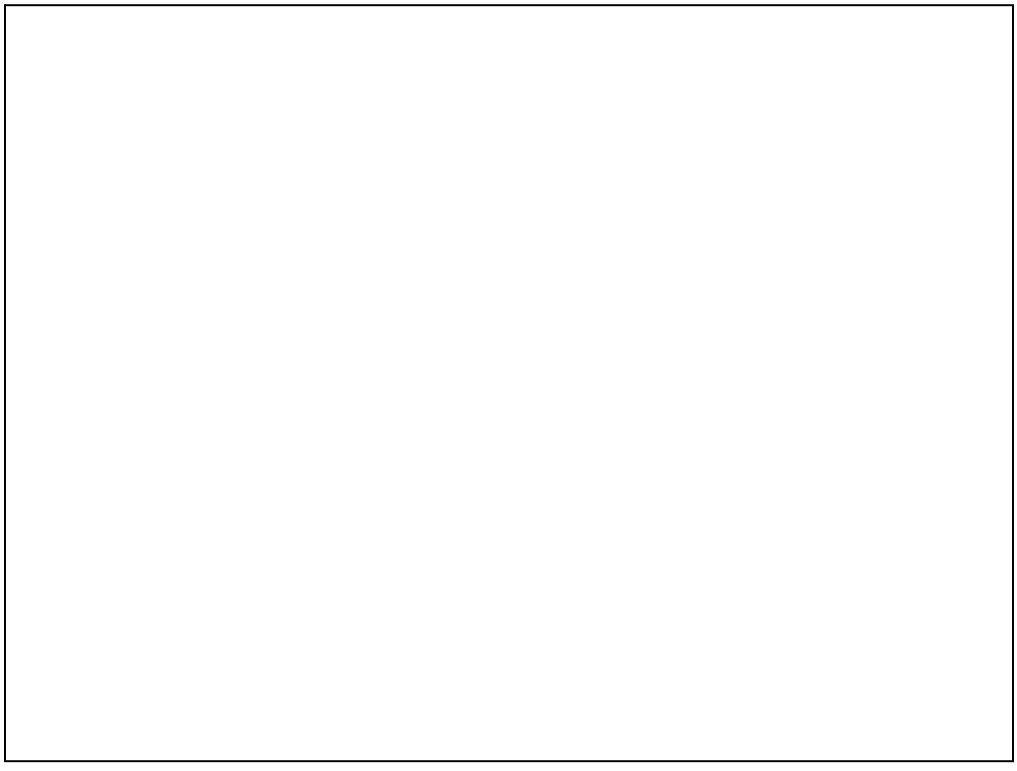


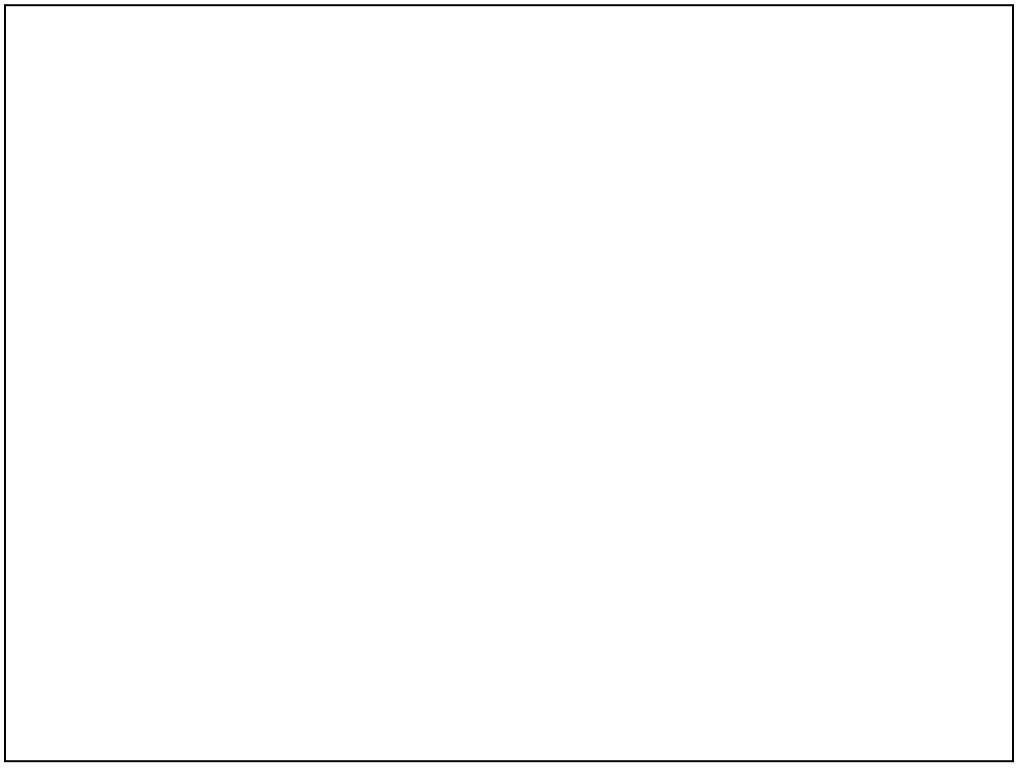


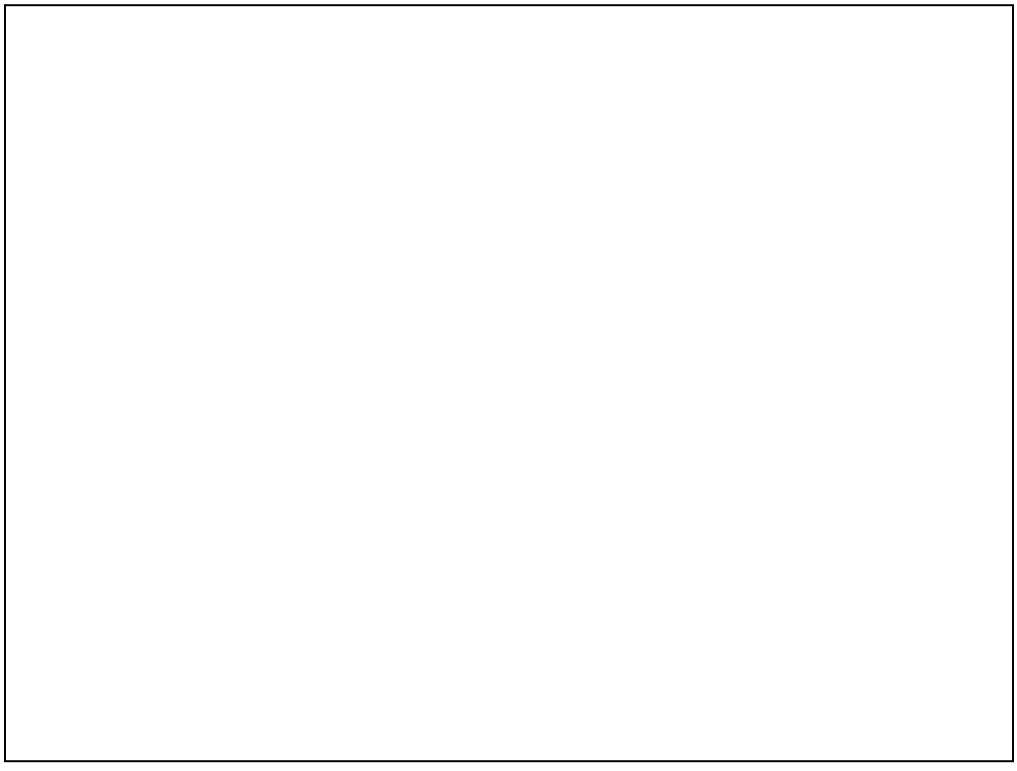


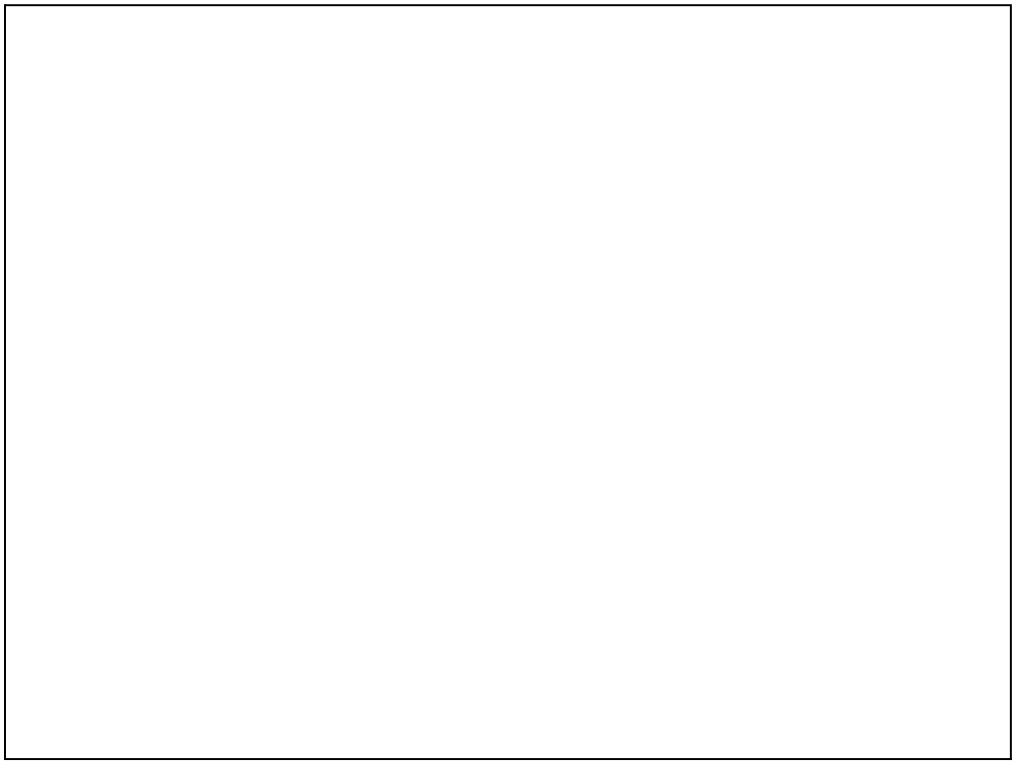




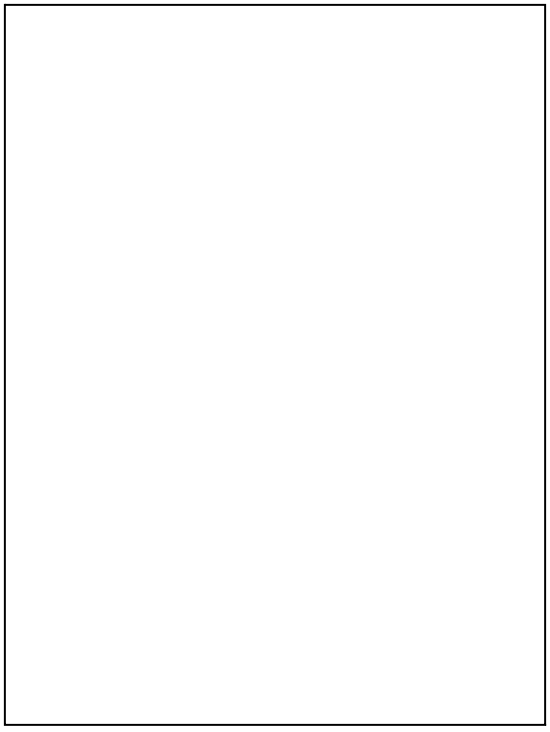


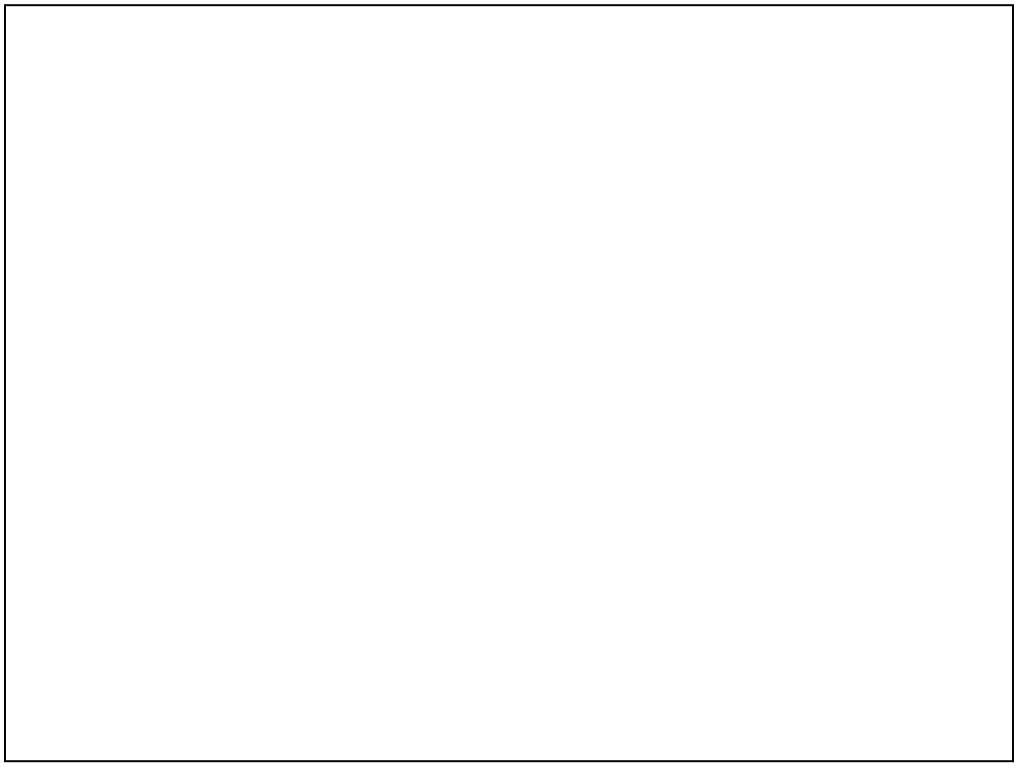


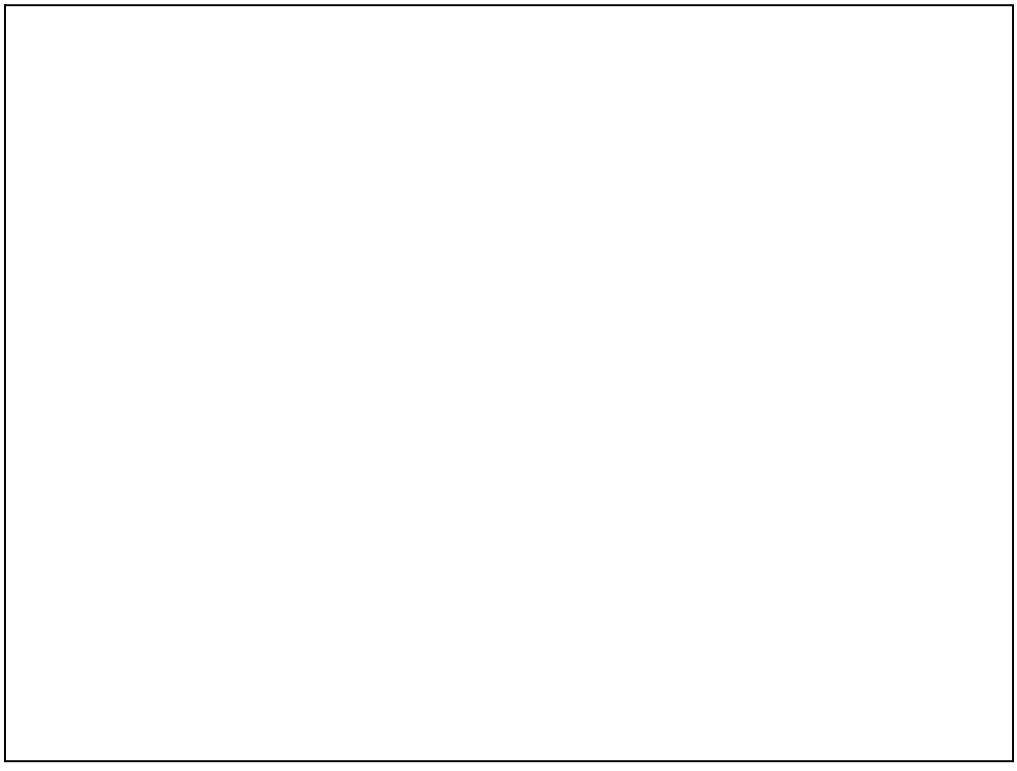


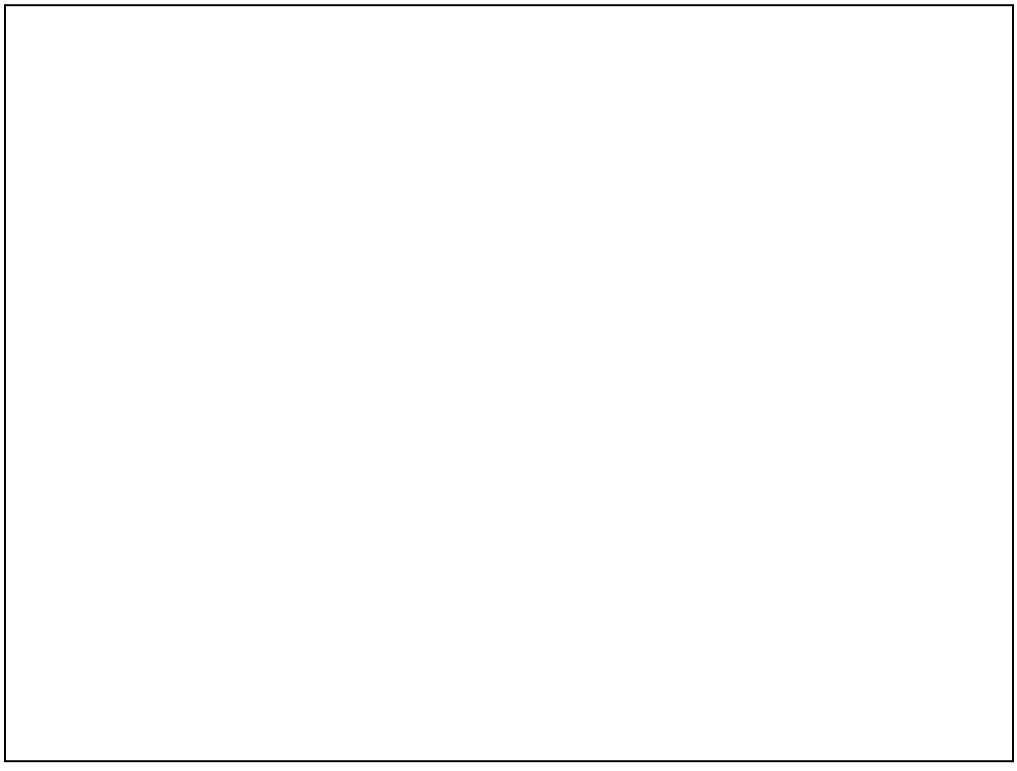


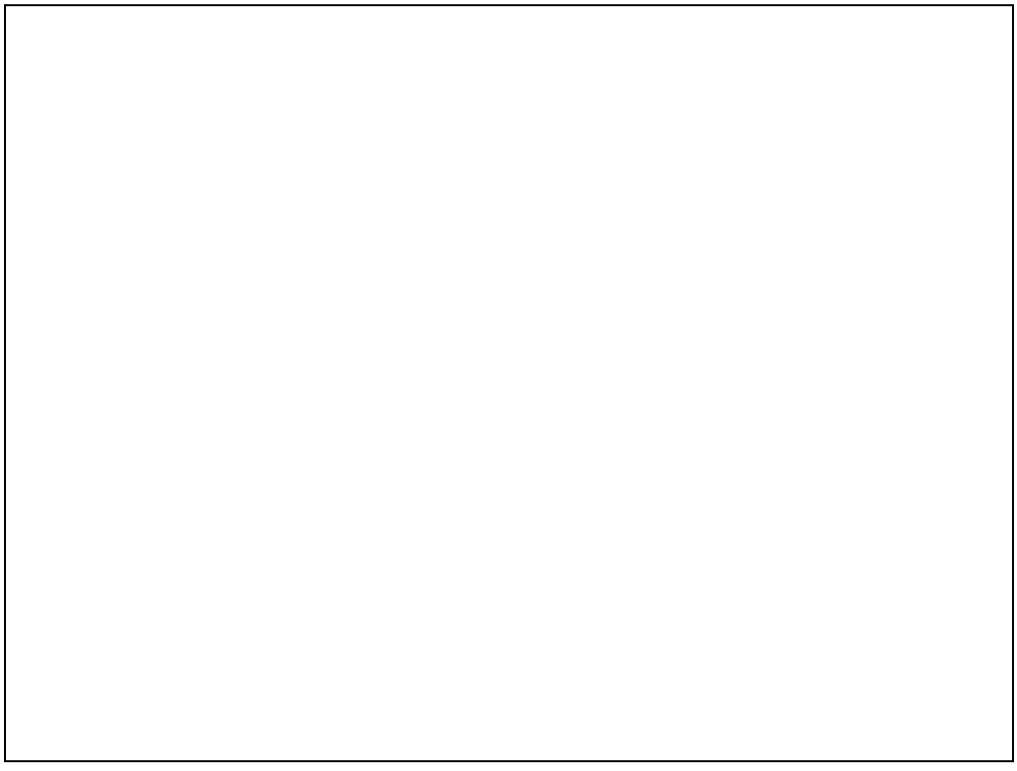
Details Streaming Stored Content

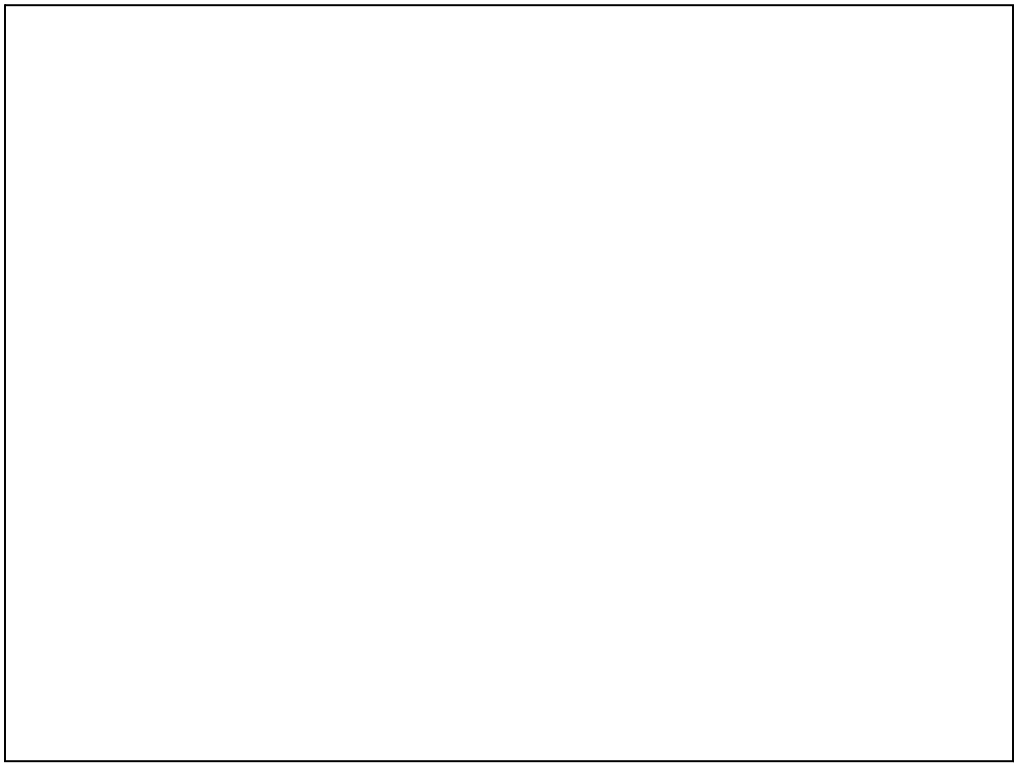


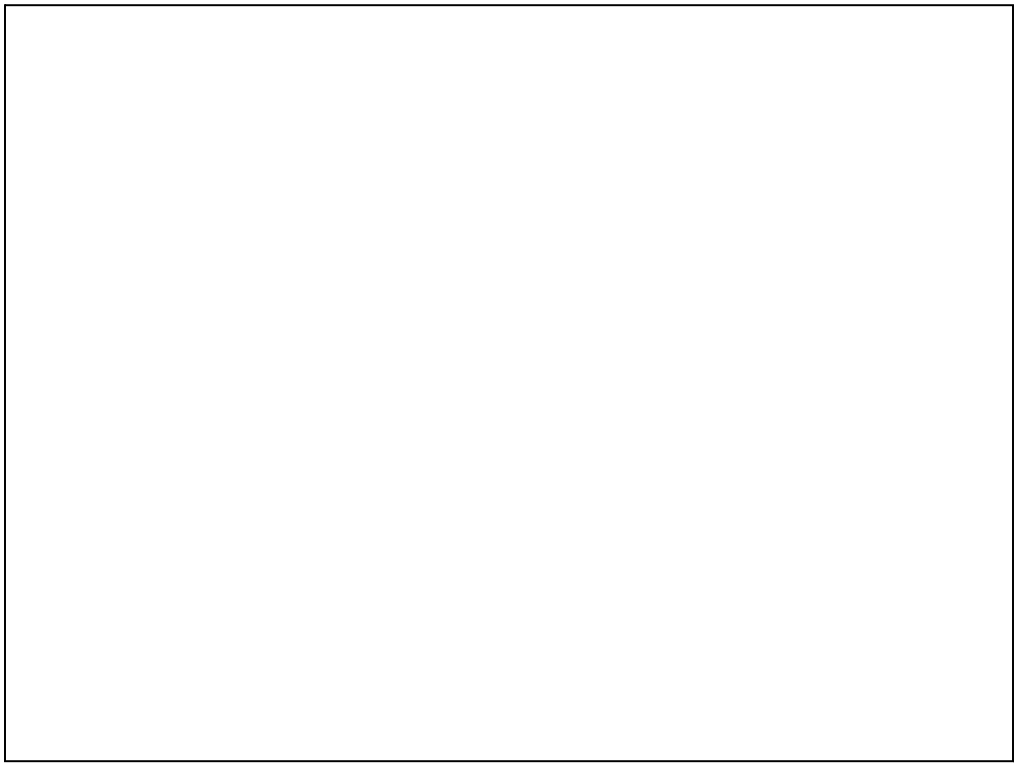


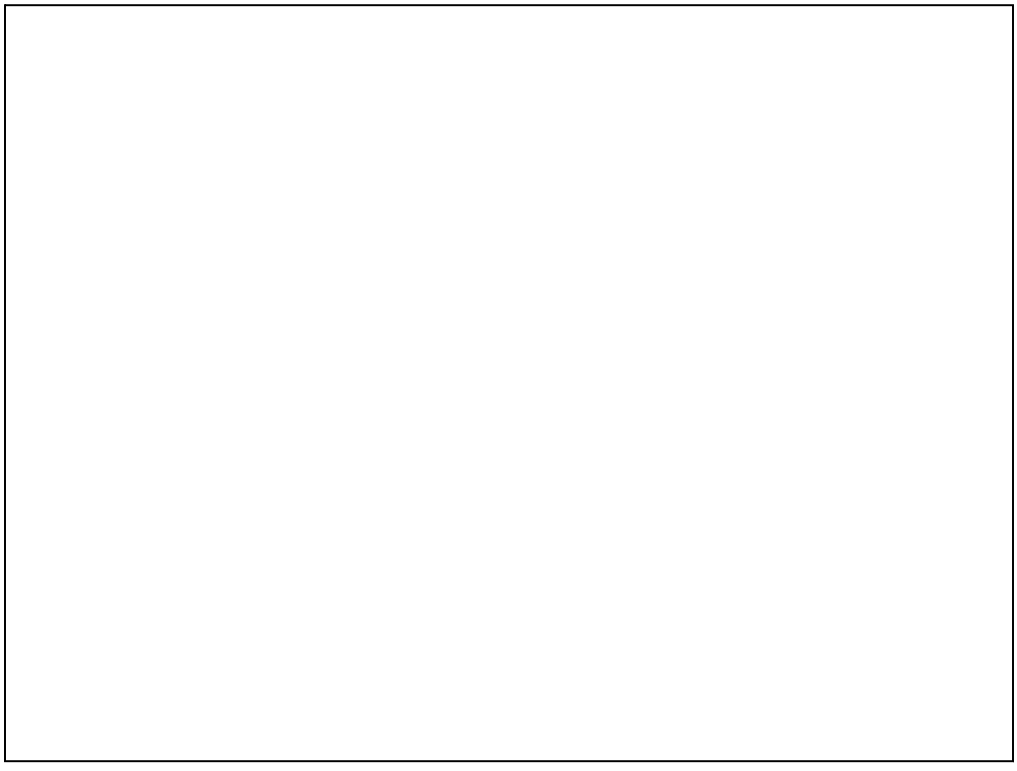


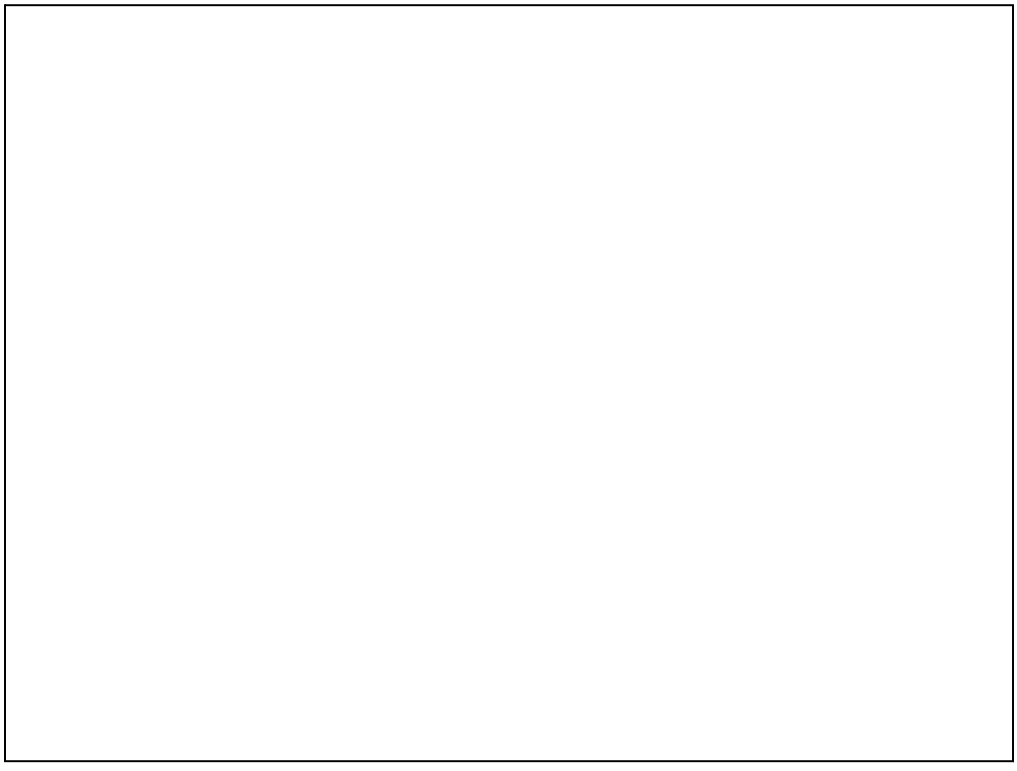


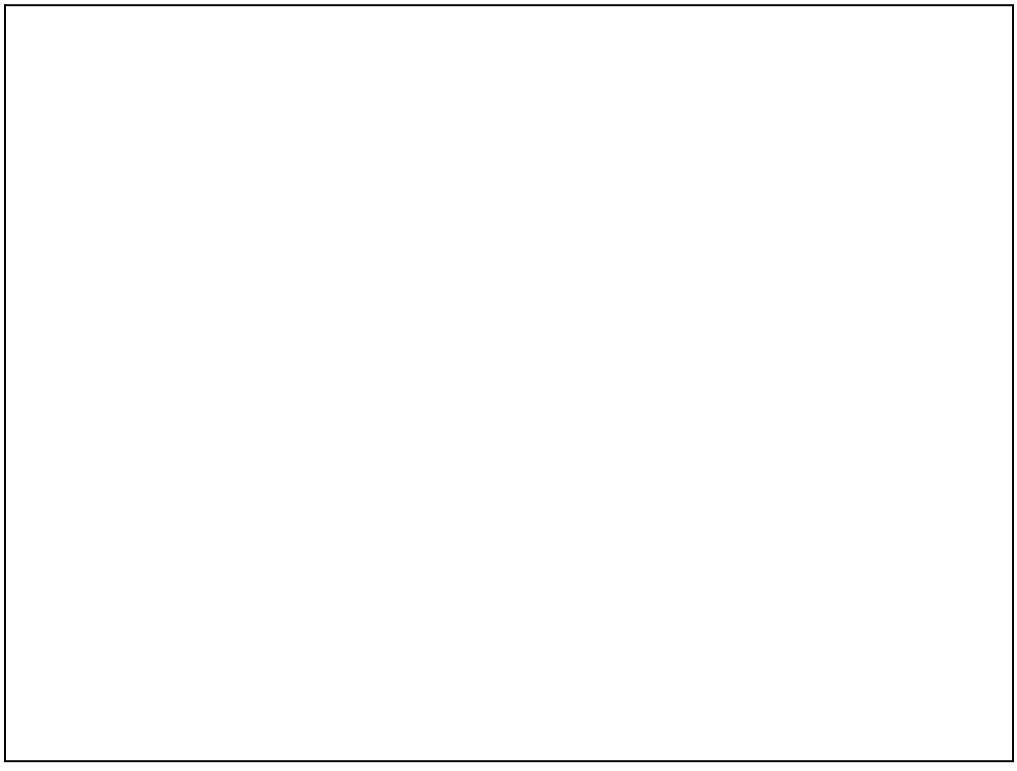












More Classifications

Another way to view technologies

Who controls them!!!

Push Based Protocols

Server controls the connection

May or may not adapt to what is happening on the client end

Pull Based Protocols

Client controls the connection and feeding of media

"Adaptive" can be applied to either one

25

Different Classes Streaming

1. Push Based Protocols – Server Controls

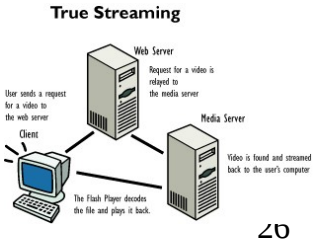
Normal Streaming

- Connection Established

- Server streams packets to client until session is torn down
- Server maintains connection entire time
- Listens for commands from client via a session control protocol
- RT Streaming Protocol, RT Messaging Protocol examples

Adaptive Streaming

- Client reports back to server who in turn can decide to switch to higher/lower bitrate stream



Different Classes Streaming

2. Pull Based Protocols

Client Controls

Client requests content

Connection Established

- Server responds to client request
- Depends on client and available bandwidth

Example: Progressive Download

- Once buffer fills to a certain level, client plays media
- If network conditions degrade, rate may fall behind playback rate, buffer underflow may result
- Content divided into short duration media segments
- Encoded at various bit rates, **client picks the rate !!!**

Example: Microsoft Smooth Streaming, Apple Http Live Streaming

_Can have Adaptive Streaming for Pull based too!

27

Adaptive Streaming

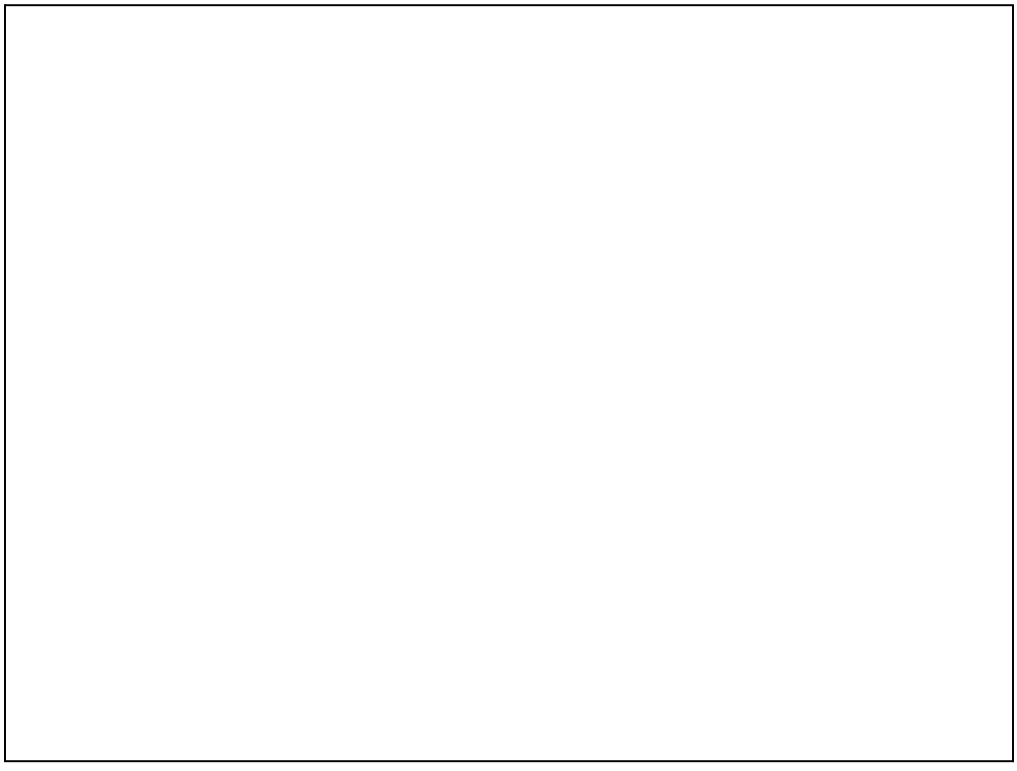
- Benefits??? Its Adaptive
  - Can change in response to changes in bandwidth and CPU usage
  - Operates transparent to user
    - Switch bitrate behind scenes
  - Multiple bitrate encodings of same source
    - Can accommodate different devices with varying needs for speed

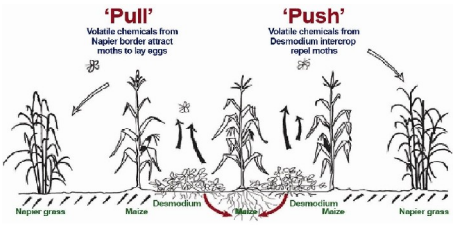
Pull Based Bitrate Adaptation

Bitrate adaptation works with Pull based !!!

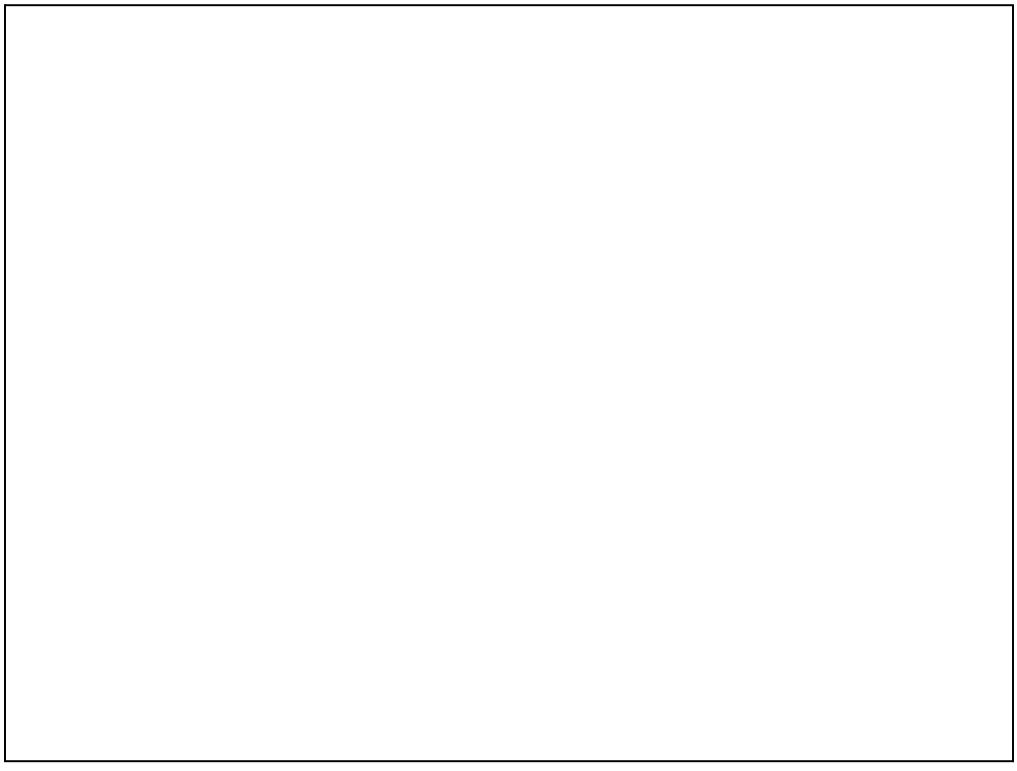
Encoded different rates

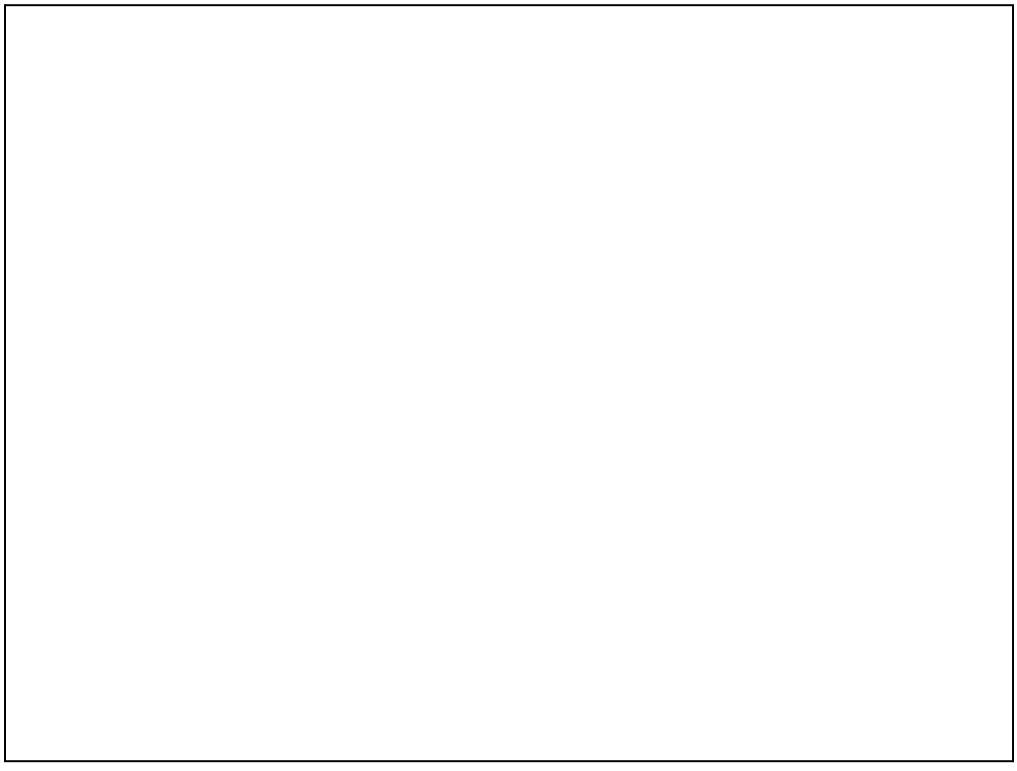
Figure 2: Example bitrate adaptation in pull-based adaptive streaming.

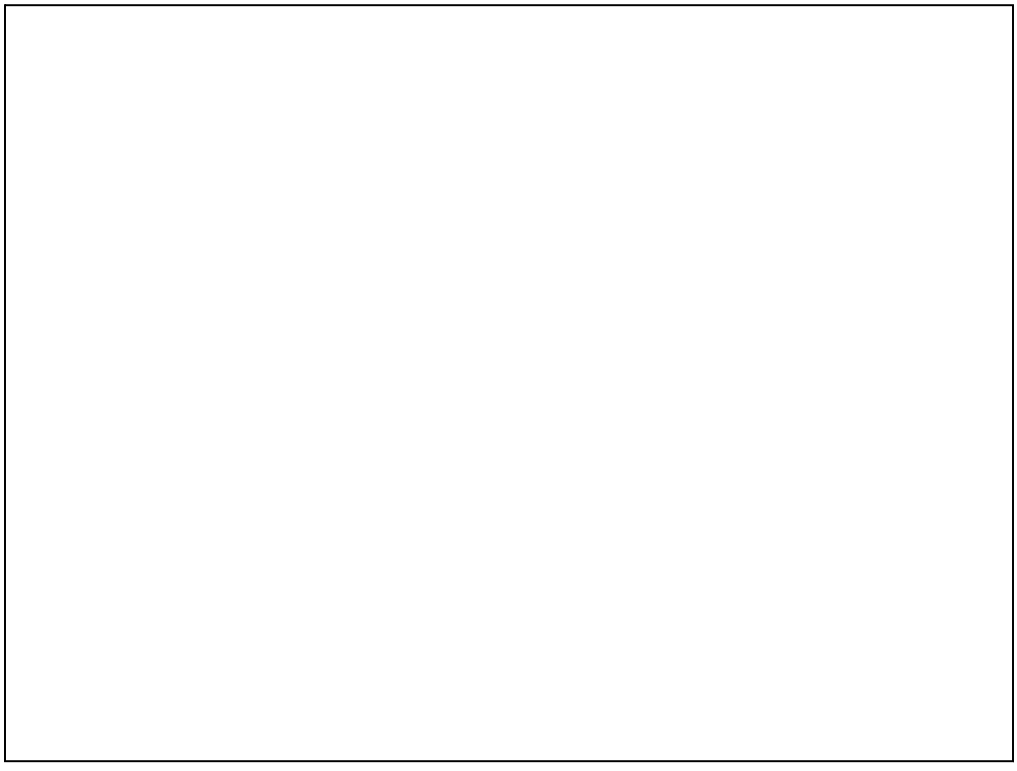


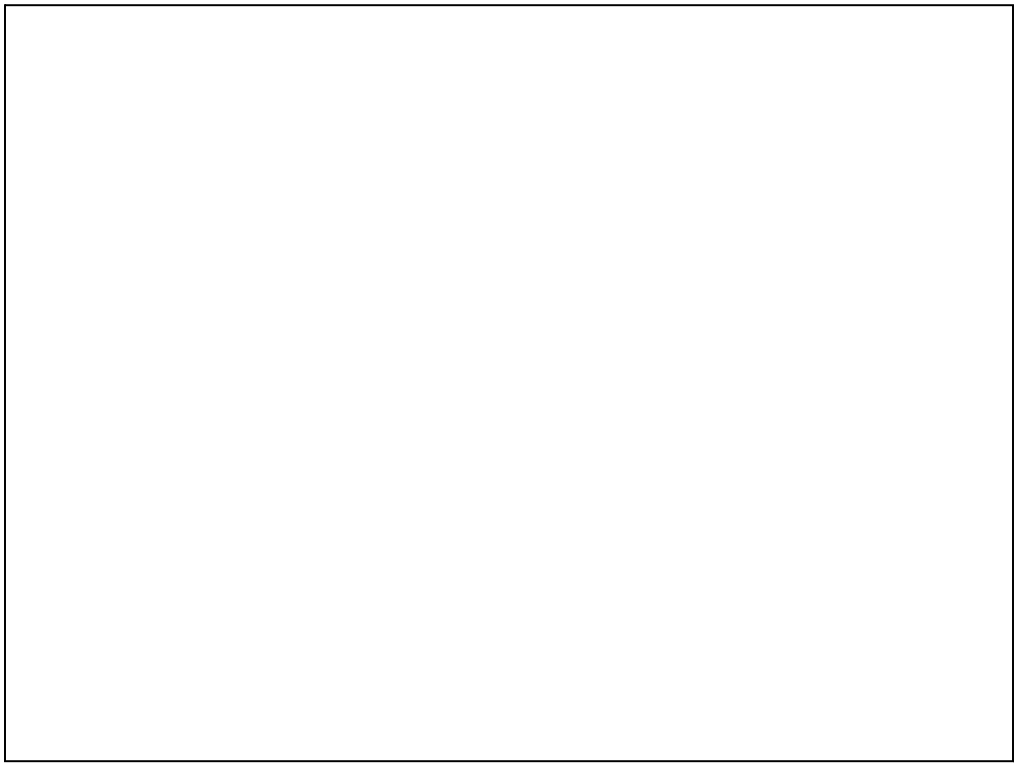


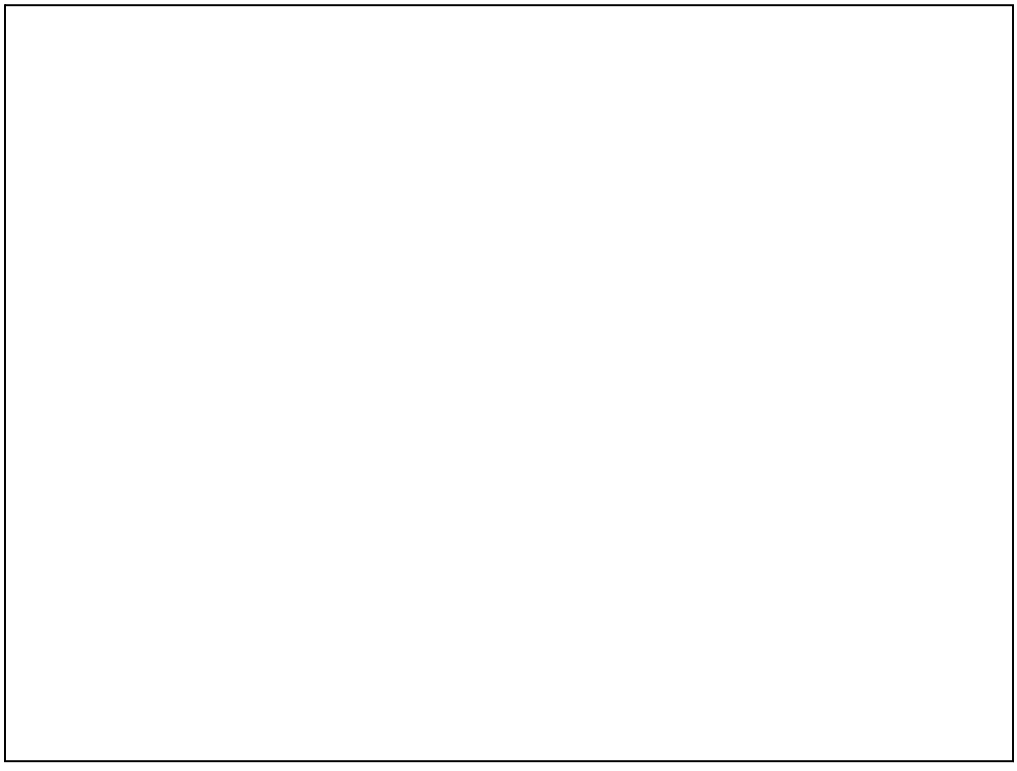
Example of Push RTSP
vs.
Example of Pull Windows Smooth Streaming

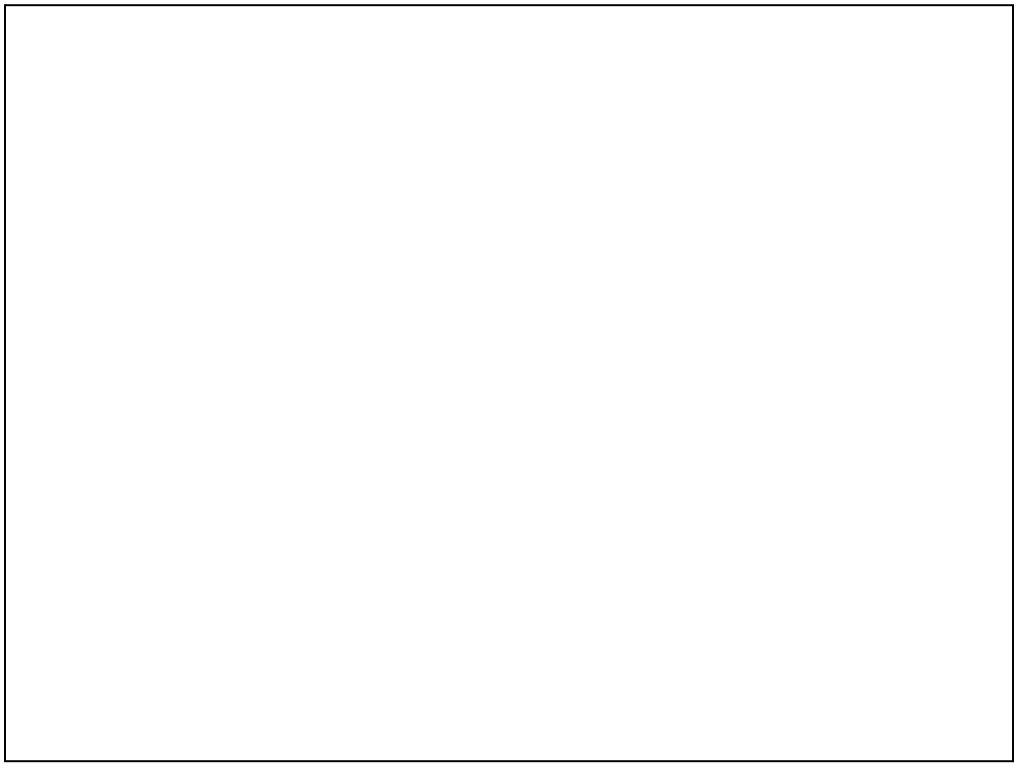


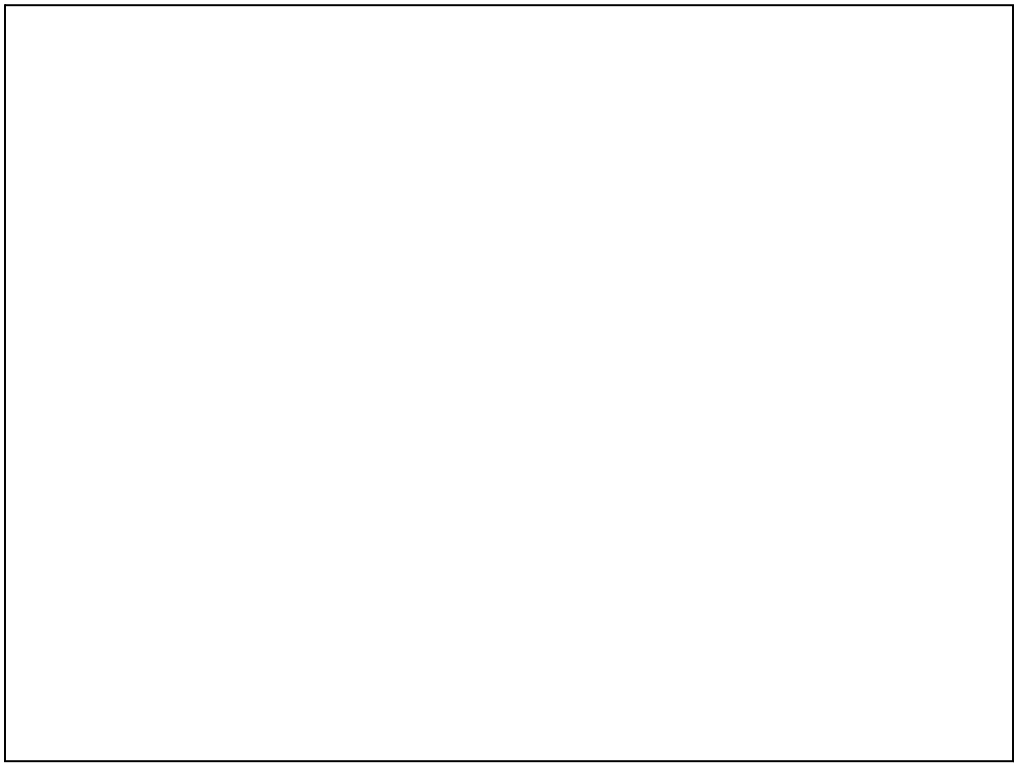












Pull Based Microsoft Smooth Streaming

- Creates multiple files with different bitrate fragments
  - Modified MP4 file format
- Fragments are about 2 seconds long
- Each file contains fragments at a specific bitrate

Works like this …

1. Client gets a client manifest file
   Describes codecs, video resolution
2. After downloading manifest file
3. Client makes http requests for movie fragments
4. Server locates corresponding MP4 file fragments and sends it

Microsoft Smooth Streaming

Modified structured MP4 file of movie fragments

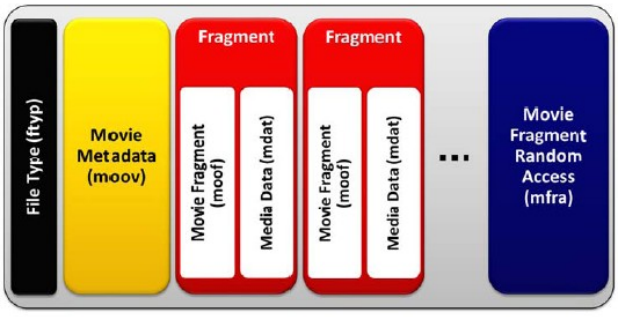


Figure 3: Fragmented MP4 file format (Reproduced from (4)).

Microsoft Smooth Streaming

- HTTP Get requests specify individual fragments
- Each fragment downloaded via unique HTTP request – needs the bitrate and the time offset

```
GET /sample/v_720p.ism/QualityLevels(1500000)/Fragments(video=160577243) HTTP/1.1
```

Figure 4: An example HTTP request for a downloading a fragment.

Comparing Push vs Pull Protocols

- RTSP - Push
- Server sends data to client a in real-time bitrate
  - If encoded at 500 kbps sent at 500 kbps
  - Not flexible in rate
- Server sends enough data to fill client buffer
  - Between 1 and 10 seconds worth
  - If client pauses for 10 minutes, still only sends 5 seconds worth
- Server connected entire time, resource intensive
- Microsoft SS - Pull
  - Client creates file chunks as they are needed
  - File chunks can be cached same as any other HTTP file, take advantage of Content providers, Akamai
  - No dedicated server connection listening, scales better