MULTIMEDIA I

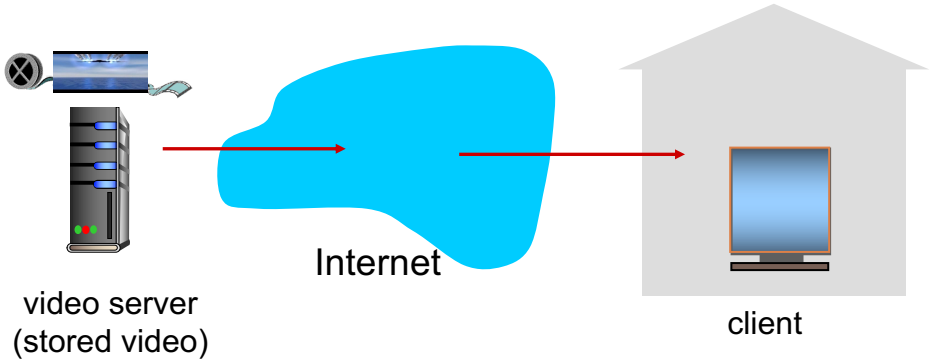
CSC 249

APRIL 26, 2018

Overview

- Multimedia
  - Classes of Applications
  - Services
  - Evolution of protocols
- Streaming from web server
- Content distribution networks
- VoIP
- Real time streaming protocol

Streaming stored video: UDP? HTTP?



Multimedia vs. Other Applications

Classes of multimedia applications:

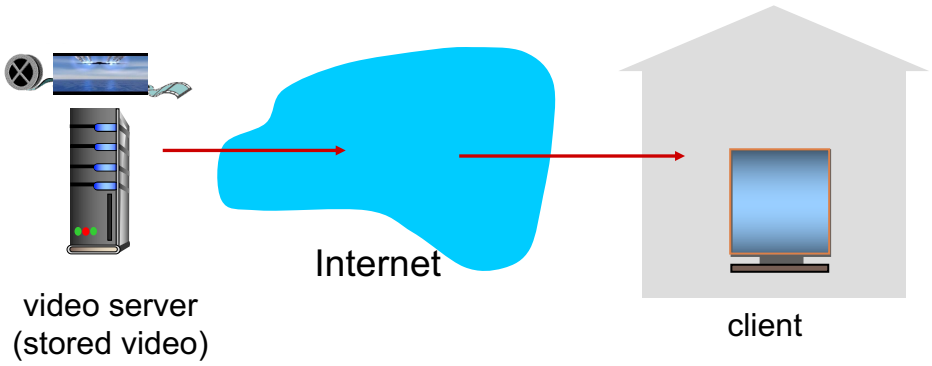
1) Stored streaming
2) Live streaming
3) Interactive, real-time

- Packet Loss?
  - Tolerant?
  - Intolerant?
- Variable delay between packets (jitter)?
  - Tolerant?
  - Intolerant?

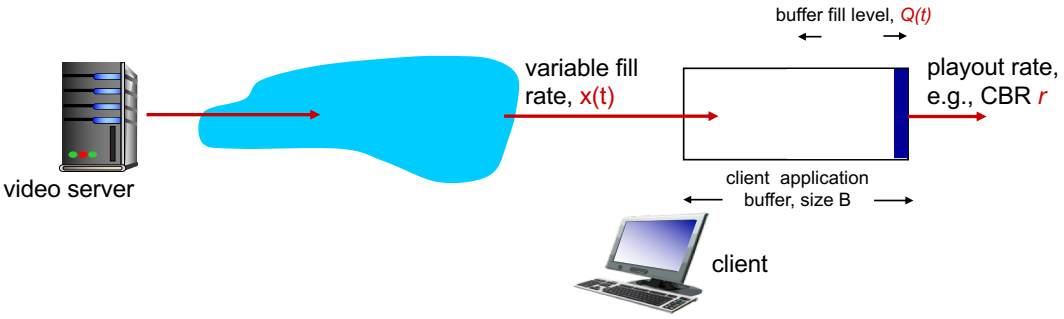
1) Streaming multimedia: UDP

- Best effort, Laissez-faire approach
  - No major changes, no guarantees for delay or loss
  - Works with historical/existing Internet architecture
- Server sends at rate appropriate for client
  - Often: send rate = encoding rate = constant rate
  - Transmission rate can be oblivious to congestion levels
  - Short playout delay (2-5 seconds) to remove jitter
- UDP might not get through firewalls
- Requires RTP – real time transport protocol

Streaming stored video:



Client-side buffering, playout



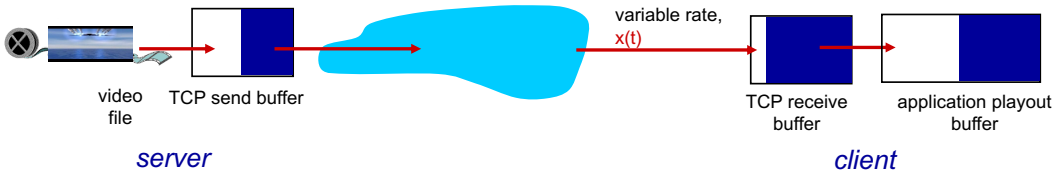
1. Initial fill of buffer until playout begins at t_p
2. Playout begins at t_p,
3. Buffer fill level varies over time as fill rate x(t) varies and playout rate r is constant

2) Differentiated Quality of Service

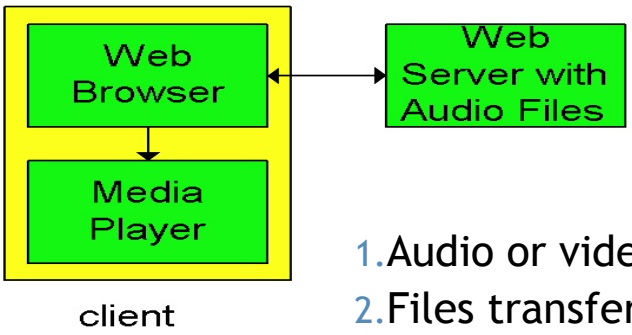
- Differentiated services
  - Implemented via HTTP with evolution toward DASH
- Few changes to Internet infrastructure
- Provide 1st and 2nd class service
- 1st Class
  - Limit the number of 1st class packets
  - These receive priority in router queues
- Net neutrality?

Streaming multimedia: HTTP

- Use of HTTP (TCP) is overtaking use of UDP
  - Fill rate fluctuates due to TCP congestion control, retransmissions (in-order delivery)
  - But… HTTP/TCP passes more easily through firewalls
- Multimedia file retrieved via HTTP GET
  - Sent at maximum possible rate under TCP



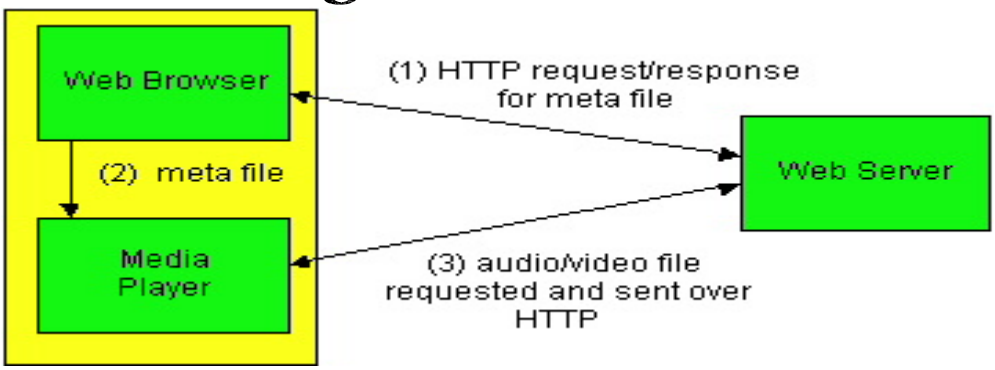
a) Transfer as HTTP Object



1. Audio or video stored in a file
2. Files transferred as HTTP object
   - Received in entirety at client
   - Then passed to player

- audio, video not streamed!
- no, "pipelining," long delays until playout

b) Streaming From Web Server

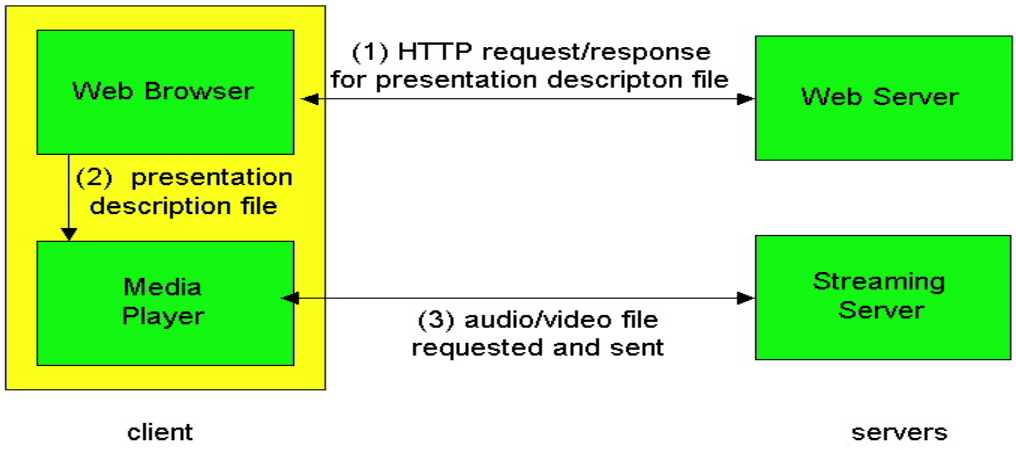


1. Browser requests **metafile**
2. Browser launches media player, passing the metafile
3. Player contacts web server
4. Server streams audio/video to player

b) Metafile Example

```
<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>
```

c) Streaming from a Streaming Server



1. Allows for non-HTTP protocol between the server & the media player
2. Uses RTSP (real-time streaming protocol)

3) Guaranteed Quality of Service

- First implementation is "DASH"
  - Dynamic, Adaptive Streaming over HTTP
- Fundamental changes for the Internet
  - Protocols to reserve link bandwidth for entire path, from sender to receiver
  - Modify router queues so reservations can be honored
  - To identify honored packets, applications must be able to label packets as such
  - Network must be able to determine if there is sufficient bandwidth
- Requires new & complex software in hosts & routers

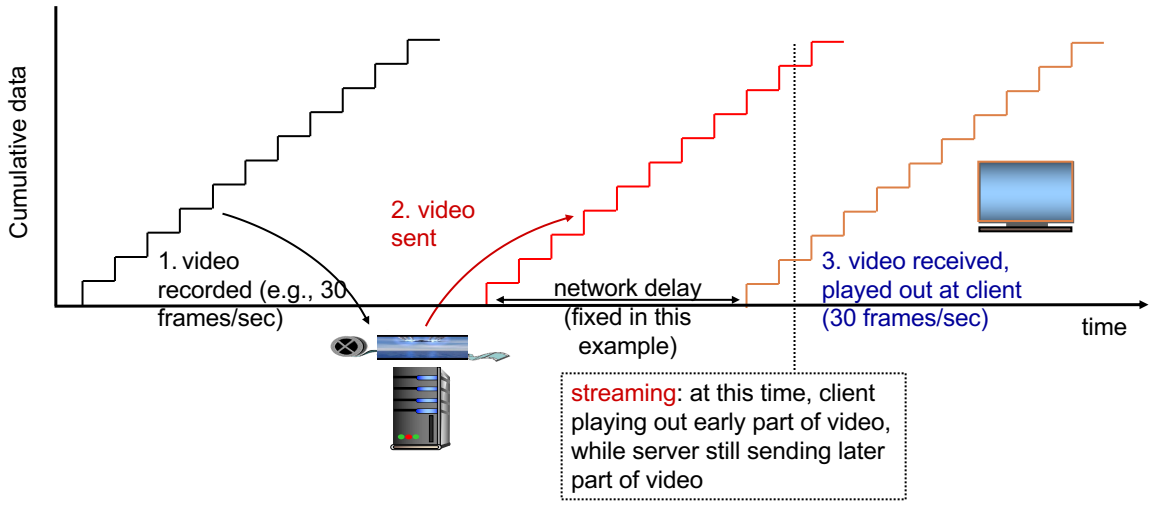
Streaming multimedia: DASH

- DASH: Dynamic, Adaptive Streaming over HTTP
  - Addresses problem of varying bandwidth available to client
- Server:
  - Multiple copies of video are stored and encoded at different rates
  - Server divides video file into multiple chunks
  - Manifest file: provides URLs for the different copies
- Client:
  - Periodically measures server-to-client bandwidth
  - Consulting manifest, requests one chunk of video 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)

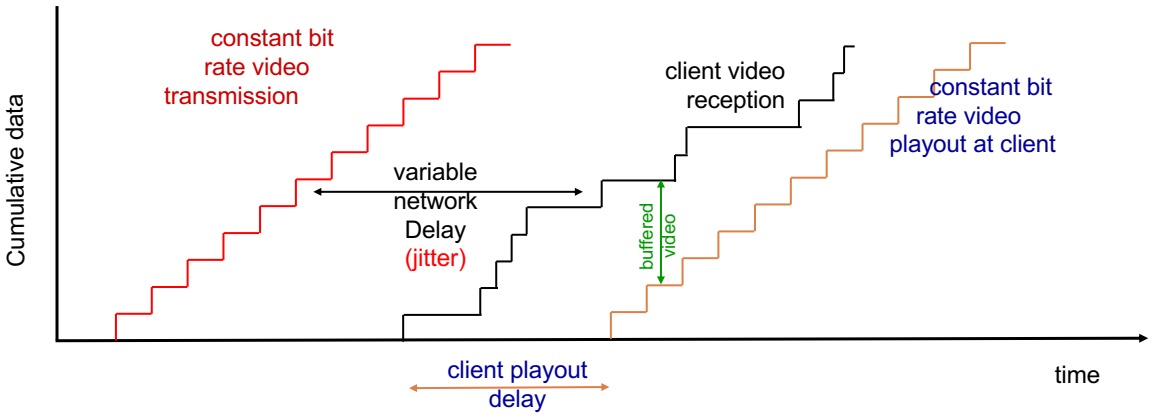
Streaming multimedia: DASH

- "Intelligence" is implemented 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 stored video:

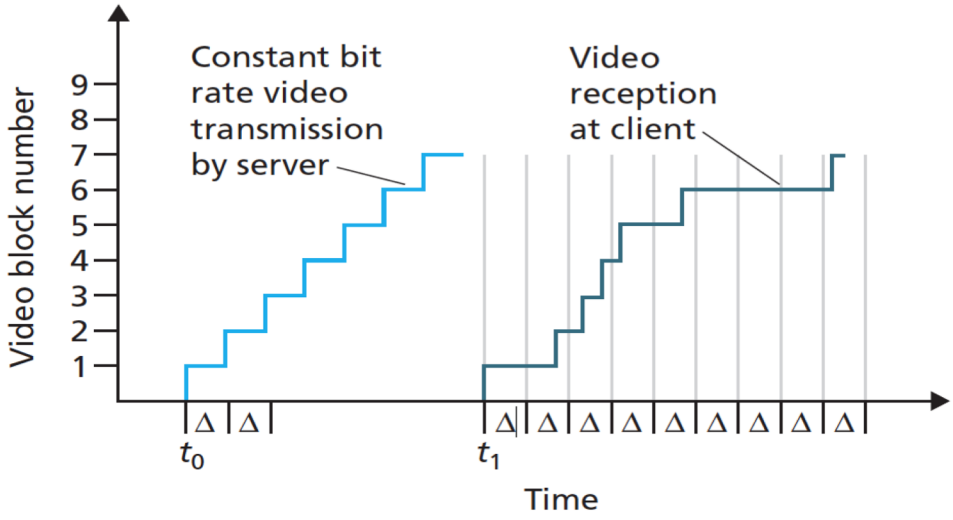


Streaming stored video: with delay



- *client-side buffering and playout delay*: compensate for network-added delay, delay jitter

Discussion Question



Discussion Questions

- Suppose that the client begins playout as soon as the first block arrives at t_1. In the figure, how many blocks of video (including the first block) will have arrived at the client in time for their playout? Discuss.
- Suppose that the client begins playout now at $t_1 + \Delta$. How many blocks of video (including the first block) will have arrived at the client in time for their playout? Discuss.
- In the same scenario as above, what is the largest number of blocks that is ever stored in the client buffer, awaiting playout? Discuss.
- What is the smallest playout delay at the client, such that every video block has arrived in time for its playout?

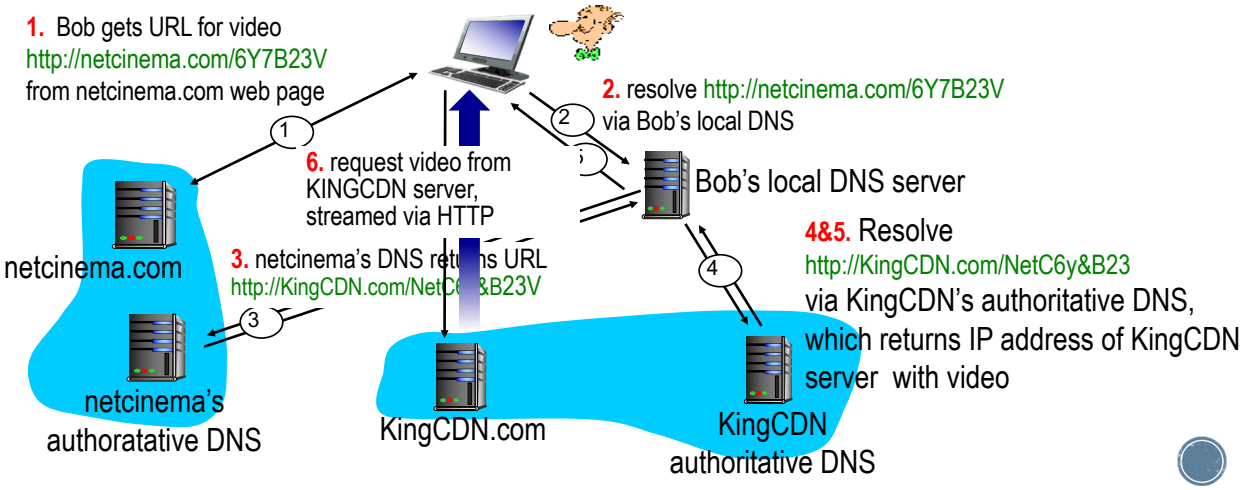
Content distribution networks

- *Challenge:* how to stream content (selected from millions of videos) to hundreds of thousands of simultaneous users?
- Store/serve multiple copies of videos at multiple geographically distributed sites *(CDN)*
  - Use DNS to determine location of client

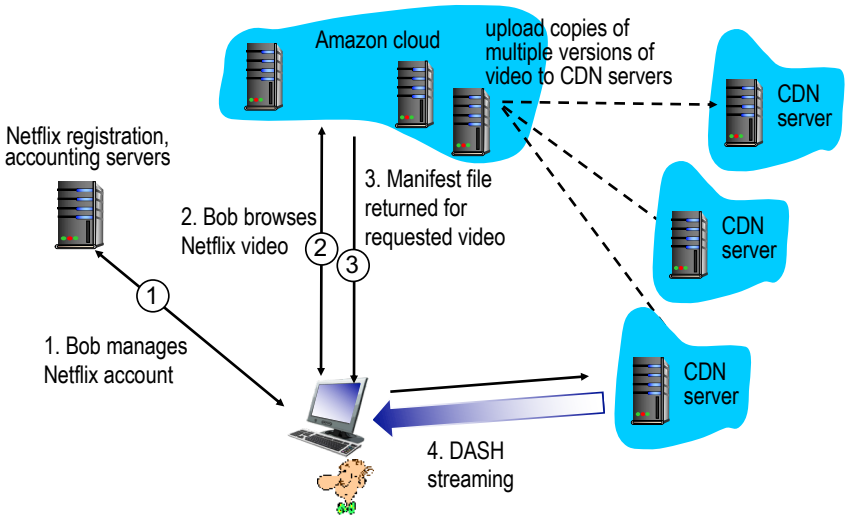
CDN content access: a closer look

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

- Video is stored in CDN at http://KingCDN.com/NetC6y&B23V
- Netcinema is the company contracting with KingCDN for distribution



Case study: Netflix



User Control of Streaming Media: RTSP

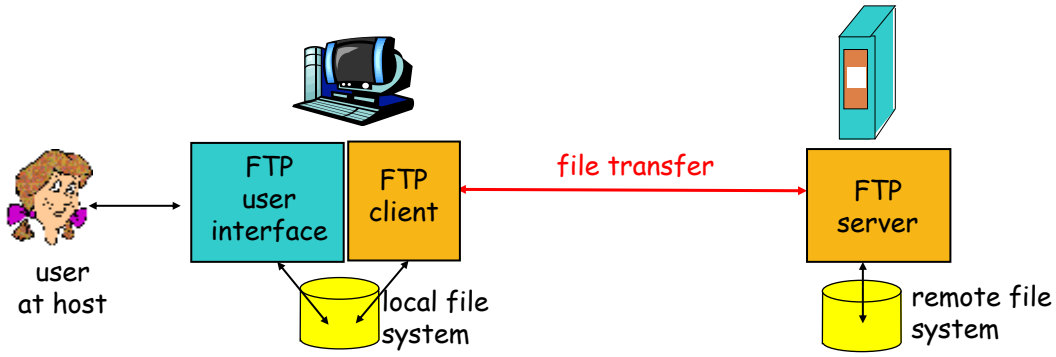
HTTP

- Was not designed for multimedia content
- No commands for fast forward, etc.

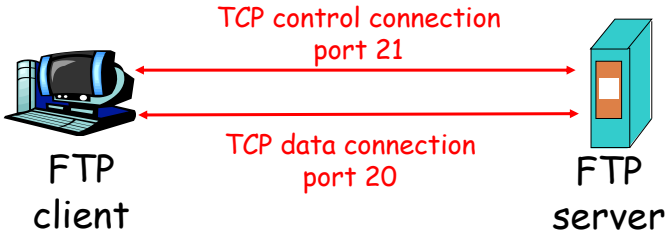
RTSP

- Real-time streaming protocol
- Client-server application layer protocol
- User control
  - rewind, fast forward, pause, resume, repositioning, etc…

Chapter 2 FTP: The file transfer protocol



FTP: separate control & data connections



The Server:

- Listens on **port 21** for an incoming connection request
- Performs file transfer over TCP data connection via **port 20**

RTSP: out of band control

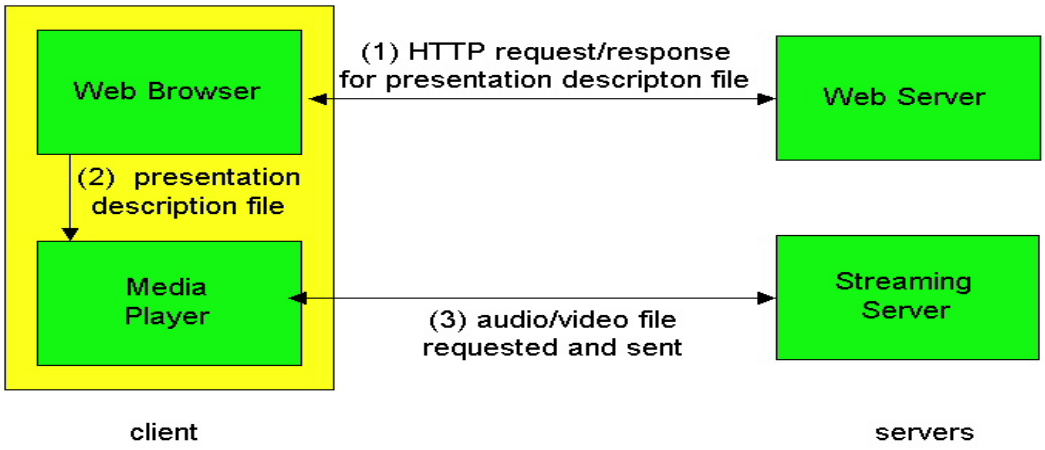
RTSP uses "out-of-band" message channel:

- RTSP control messages use different port numbers than media stream
- The actual media stream is considered "in-band"

FTP uses "out-of-band" control channel:

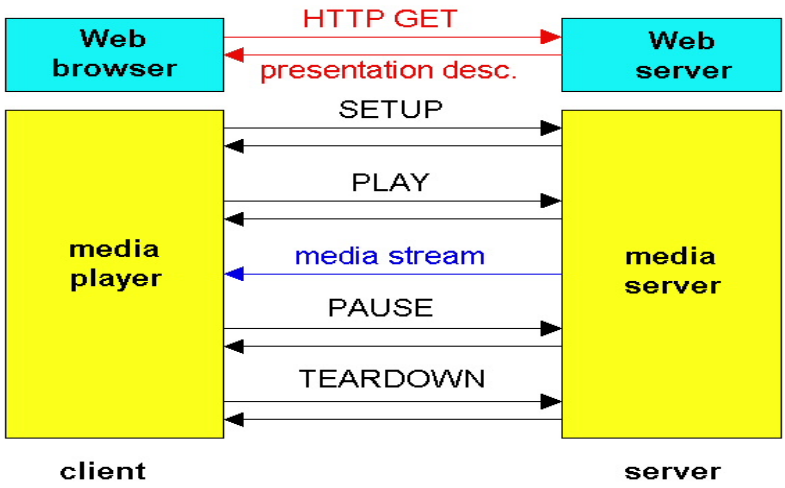
- Control info (directory changes, file deletion, rename) is sent over one TCP connection
- File transfer occurs over a second TCP connection.
- "Out-of-band" & "in-band" channels use different port numbers

3) Streaming from a Streaming Server



1. Allows for non-HTTP protocol between the server & the media player
2. Uses RTSP

RTSP Operation



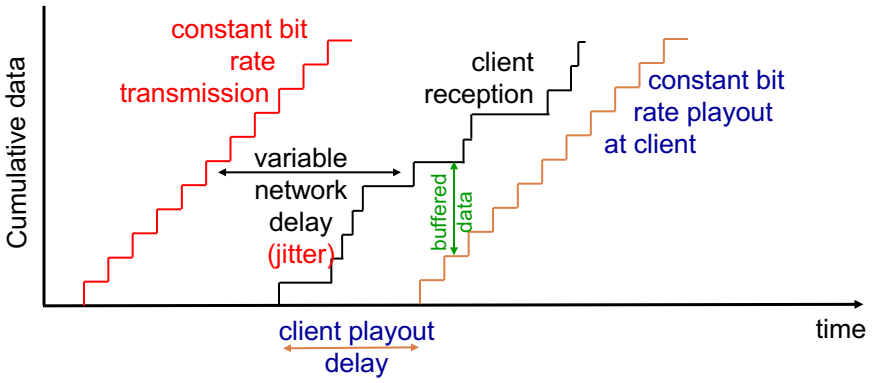
VoIP Characteristics

- Speaker's audio: alternating "talk spurts," silent periods.
  - 64 kbps during "talk spurt"
  - Packets generated only during talk spurts
  - Example: 20 msec chunks at 8 Kbytes/sec = 160 bytes of data for each voice data chunk created
- Application-layer header added to each chunk
- Chunk+header encapsulated into UDP or TCP segment
- Application sends transport segment into socket every 20 msec during a talk spurt

VoIP: packet loss, delay

- *Network loss:* IP datagram lost due to network congestion (router buffer overflow)
- *Delay loss:* IP datagram arrives too late for playout at receiver
  - delays: processing, queueing in network; end-system (sender, receiver) delays
  - typical maximum tolerable delay: 400 ms
- *Loss tolerance:* depending on voice encoding, loss concealment, packet loss rates between 1% and 10% can be tolerated

Delay jitter



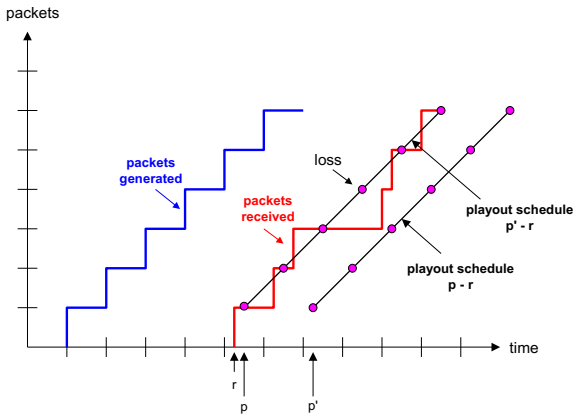
- End-to-end delays of two consecutive packets: difference can be more or less than 20 msec (transmission time difference)

VoIP: fixed playout delay

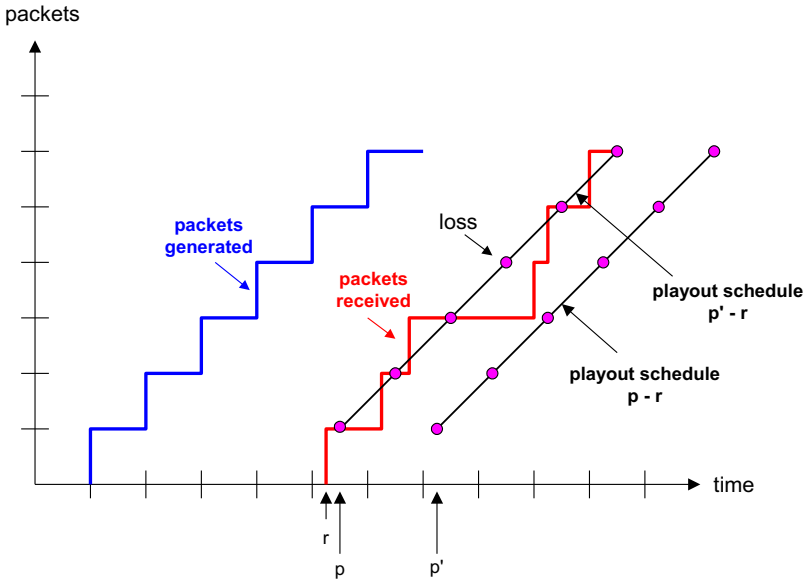
- Receiver attempts to playout each chunk exactly q msecs after a chunk is generated.
  - Chunk has time stamp t: play out chunk at $t+q$
  - Chunk arrives after $t+q$: data arrives too late for playout: data "lost"
  - A delay of q < 150 msec is not detectable by the human ear
  - A delay of q > 400 msec becomes in tolerable for an interactive conversation
- Tradeoff in choosing q:
  - *Large q:* less packet loss
  - *Small q:* better interactive experience

VoIP: fixed playout delay

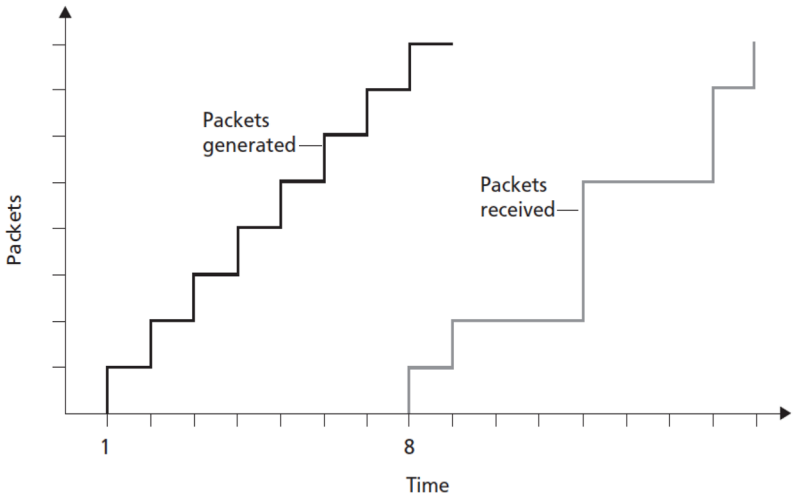
- Sender generates packets every 20 msec during talk spurt.
- First packet received at time r
- First playout schedule: begins at p
- Second playout schedule: begins at p'



VoIP: fixed playout delay



Discussion Question



VoIP: recovery from packet loss (1)

Challenge: recover from packet loss given small tolerable delay between original transmission and playout

- Send enough bits to allow recovery without retransmission

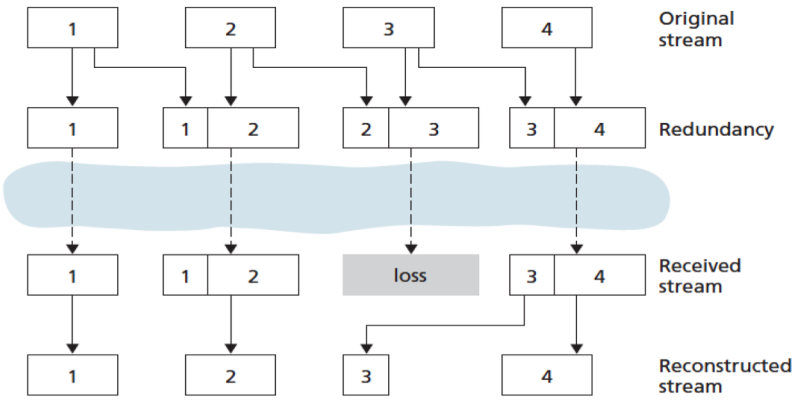
Simple Forward Error Correction (FEC)

- For every group of *n* media chunks, create a redundant chunk by exclusive OR-ing the *n* original chunks
- Send *n+1* chunks, increasing bandwidth by factor *1/n*
- Can reconstruct the original *n* chunks
  - If at most there is **one** lost chunk from the *n+1* chunks sent
  - Yet we incur playout delay while waiting to reconstruct the lost chunk

VoiP: recovery from packet loss (2)

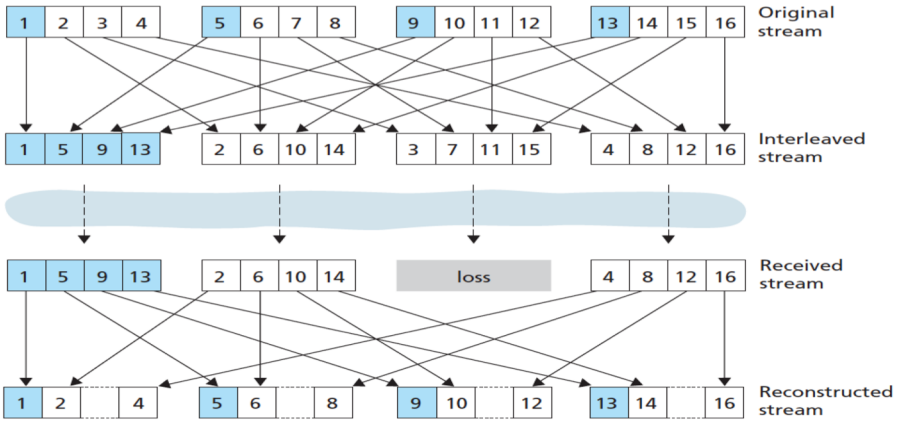
another FEC scheme:

- Piggyback lower quality stream
- Send lower resolution audio stream as redundant information
- *e.g.*, nominal stream at 64 kbps and redundant stream at 13 kbps



- Non-consecutive loss: receiver can conceal loss
- Generalization: can also append (n-1)st and (n-2)nd low-bit rate chunk

VoiP: recovery from packet loss (3)



interleaving to conceal loss:

- Audio chunks divided into smaller units, *e.g.* four 5 msec units per 20 msec audio chunk
- Packet contains small units from different chunks
- If packet lost, still have *most* of every original chunk
- No redundancy overhead, but increases playout delay

Discussion Question

For the 2 redundant FEC schemes (next slide):

a) How much additional bandwidth does each scheme require? How much playback delay does each scheme add?

b) How do the two schemes perform if the first packet is lost in every group of five packets? Which scheme will have better audio quality?

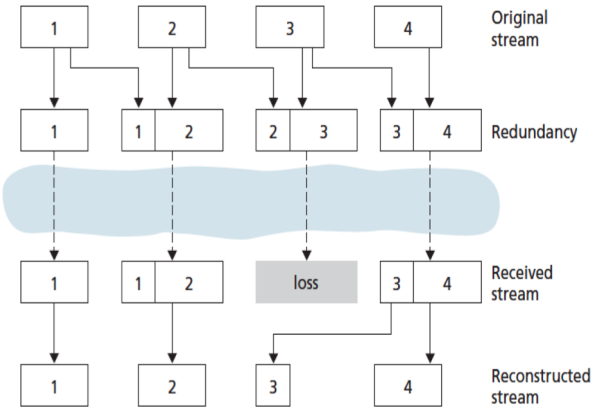
c) How do the two schemes perform if the first packet is lost in every group of two packets? Which scheme will have better audio quality?

Discussion Question

1) Simple Forward Error Correction

- For every group of *n* chunks, create a redundant chunk by exclusive OR-ing *n* original chunks
- Send *n+1* chunks, increasing bandwidth by factor *1/n*
- Can reconstruct the original *n* chunks if at most there is one lost chunk from the *n+1* chunks sent, with playout delay

2) Piggyback low quality audio



Summary

- Evolution: UDP → HTTP → DASH
- Streaming stored video
- Three scenarios for transferring
  - As HTTP object
  - From web server
  - From streaming server
- Compare RTSP to FTP
- Content distribution networks
- Voice Over IP