# CS640: Introduction to Computer Networks

Aditya Akella

Lecture 21 -
Multimedia Networking

## Application Classes

- Typically sensitive to delay, but can tolerate packet loss (would cause minor glitches that can be concealed)
- Data contains audio and video content ("continuous media"), three classes of applications:
  - Streaming stored content
  - Unidirectional Real-Time
  - Interactive Real-Time

2

## Application Classes (more)

- Streaming stored content
  - Clients request audio/video files from servers and pipeline reception over the network and display
    - Interactive: user can control operation (similar to VCR: pause, resume, fast forward, rewind, etc.)
  - Streaming → start playing before all content arrives
  - Continuous playout: some delivery constraints

3

## Application Classes (more)

- Unidirectional Real-Time:
  - similar to existing TV and radio stations, but delivery on the network
  - Non-interactive, just listen/view
- Interactive Real-Time:
  - Phone conversation or video conference
  - More stringent delay requirement than Streaming and Unidirectional because of real-time nature
  - Video: < 150 msec acceptable
  - Audio: < 150 msec good, <400 msec acceptable

4

## Streaming Applications

- Important and growing application
  - Due to reduction of storage costs, increase in high speed net access from homes and enhancements to caching
- Audio/Video file is segmented and sent over either TCP or UDP
- Public segmentation protocol: Real-Time Protocol (RTP)
- User Interaction: Real-time Streaming protocol (RTSP)

5

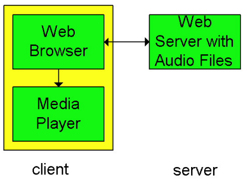
## Streaming

- Helper Application: displays content, which is typically requested via a Web browser; e.g. RealPlayer; typical functions:
  - Decompression
  - Jitter removal
  - Error correction: use redundant packets to be used for reconstruction of original stream
  - GUI for user control

6

## Streaming From Web Servers

- Audio: in files sent as HTTP objects
- Video (interleaved audio and images in one file, or two separate files and client synchronizes the display) sent as HTTP object(s)
- A simple architecture is to have the Browser request the object(s) and after their reception pass them to the player for display
  - No pipelining



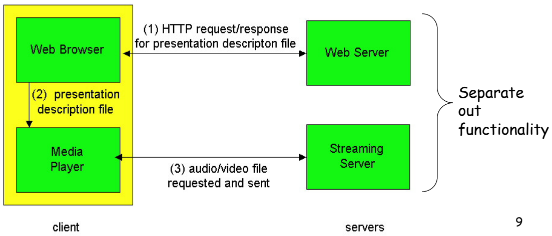

## Streaming From Web Server

- Alternative: set up connection between server and player, then download
- Web browser requests and receives a Meta File (a file describing the object) instead of receiving the file itself;
- Browser launches the appropriate Player and passes it the Meta File;
- Player sets up a TCP connection with Web Server and downloads the file

8

## Using a Streaming Server

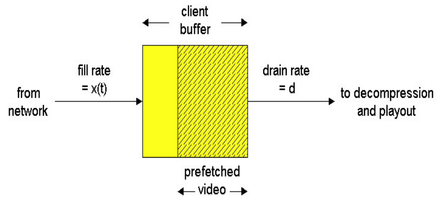
- This gets us around HTTP, allows use of UDP vs. TCP and the application layer protocol can be better tailored to Streaming; many enhancements options are possible …




9

## Options When Using a Streaming Server

- UDP: Server sends at a rate (Compression and Transmission) appropriate for client; to reduce jitter, Player buffers initially for 2-5 seconds, then starts display
- Use TCP, and sender sends at maximum possible rate under TCP; retransmit when error is encountered; Player uses a much large buffer to smooth delivery rate of TCP




10

## 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)
- As before, meta file is communicated to web browser which then launches the Player;
  - Meta file contains "presentation description file" which has information on the multi-media content

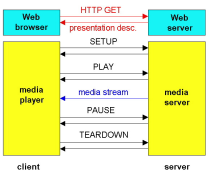
11

## Presentation Description Example

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

12

# RTSP Operation




- C: SETUP rtsp://audio.example.com/xena/audio RTSP/1.0
  Transport: rtp/udp; compression; port=3056; mode=PLAY
- 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)
- C: PAUSE rtsp://audio.example.com/xena/audio.en/lofi RTSP/1.0
  Session: 4231
  Range: npt=37
- C: TEARDOWN rtsp://audio.example.com/xena/audio.en/lofi RTSP/1.0
  Session: 4231
- S: 200 3 OK

13

# Real-Time Protocol (RTP)

- Provides standard packet format for real-time application
- Application-level; Typically runs over UDP
- Specifies header fields for identifying payload type, detecting packet loss, accounting for jitter etc.
- Payload Type: 7 bits, providing 128 possible different types of encoding; eg PCM, MPEG2 video, etc.


RTP Header

14

# Real-Time Protocol (RTP)

- Timestamp: 32 bytes; gives the sampling instant of the first audio/video byte in the packet; used to remove jitter introduced by the network
- Sequence Number: 16 bits; used to detect packet loss


RTP Header

15

## Real-Time (Phone) Over IP's Best-Effort

- Internet phone applications generate packets during talk spurts
- Bit rate is 8 Kbytes/s, and every 20 msec, the sender forms a packet of ~160 Bytes
- The coded voice information is encapsulated into a UDP packet and sent out
- Packets may be arbitrarily delayed or lost
  - When to play back a chunk?
  - What to do with a missing chunk?

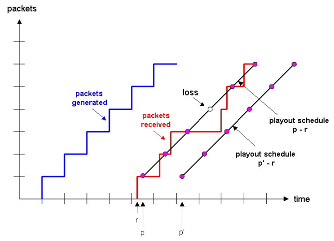
16

## Removing Jitter

- Decision on when to play out a chunk affected by network "jitter"
  - Variation in queueing delays of chunks
- One option: ignore jitter and play chunks as and when they arrive
  - Can become highly unintelligible, quickly
- But jitter can be handled using:
  - sequence numbers
  - time stamps
  - delaying playout

17

## Fixed Playout Delay




- Trade-off between lost packets and large delays
- Can make play-out even better with "adaptive play-out"

18

## Recovery From Packet Loss

- Loss interpreted in a broad sense: packet never arrives or arrives later than its scheduled playout time
- Since retransmission is inappropriate for Real Time applications, FEC or Interleaving are used to reduce loss impact and improve quality
- FEC is Forward Error Correction
  - Simplest FEC scheme adds a redundant chunk made up of exclusive OR of a group of n chunks
    - Can reconstruct if at most one lost chunk
  - Redundancy is 1/n, bad for small n
  - Also, play out delay is higher

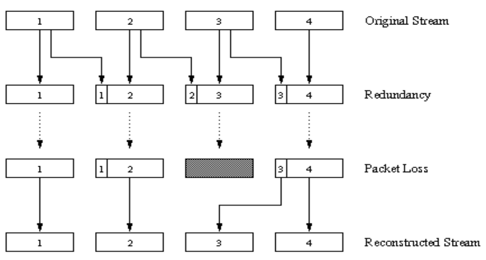
19

## Another FEC Mechanism

- Send a low resolution audio stream as redundant information
- Upon loss, playout available redundant chunk
  - Albeit a lower quality one
- With one redundant low quality chunk per chunk, scheme can recover from single packet losses

20

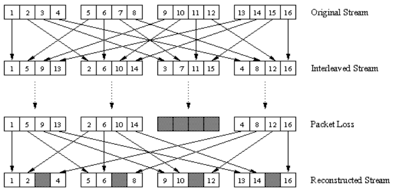
## Piggybacking Lower Quality Stream




21

# Interleaving

- Divide 20 msec of audio data into smaller units of 5 msec each and interleave
- Upon loss, have a set of partially filled chunks
- Has no redundancy, but can cause delay in playout beyond Real Time requirements




22