Streaming Stored Audio & Video

Streaming stored media:

- Audio/video file is stored in a server
- Users request audio/video file on demand.
- Audio/video is rendered within, say, 10 s after request.
- Interactivity (pause, re-positioning, etc.) is allowed.

Media player:

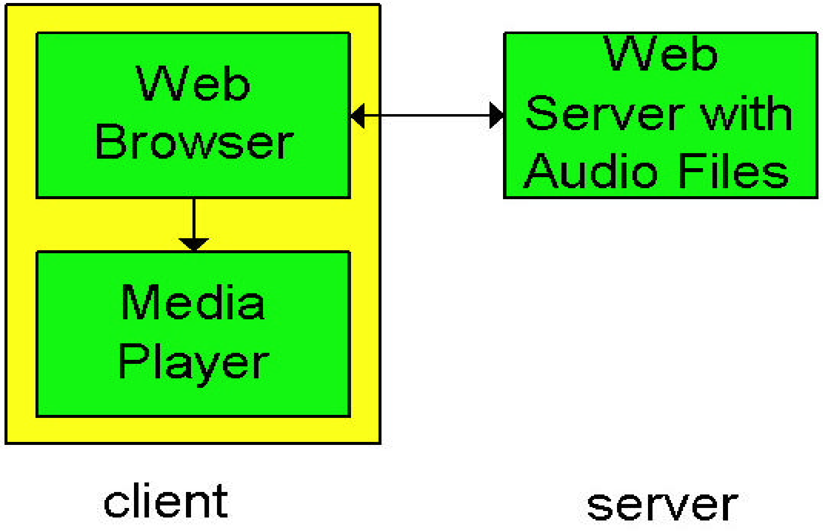
  - removes jitter
  - decompresses
  - error correction
  - graphical user interface with controls for interactivity
- Plug-ins may be used to imbed the media player into the browser window.

Streaming from Web server (1)

- Audio and video files stored in Web servers

naïve approach

- browser requests file with HTTP request message
- Web server sends file in HTTP response message
- content-type header line indicates an audio/video encoding
- browser launches media player, and passes file to media player
- media player renders file

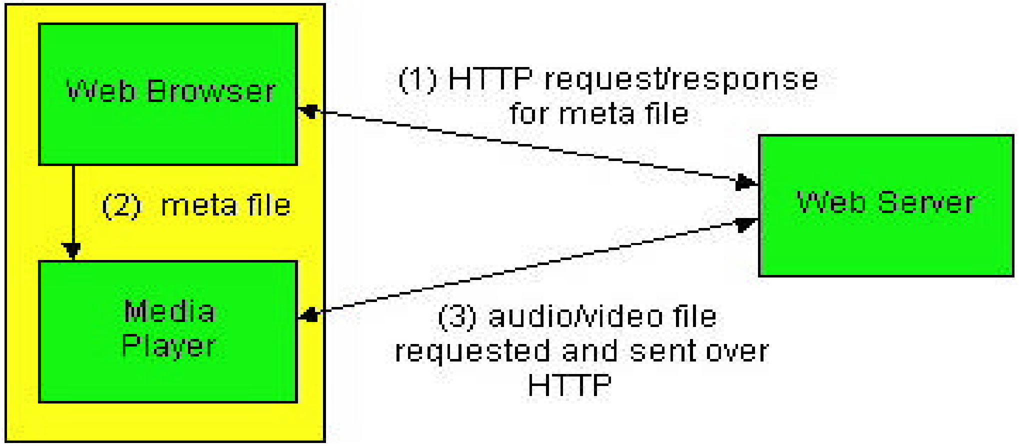


- Major drawback: media player interacts with server through intermediary of a Web browser

Streaming from Web server (2)

Alternative: set up connection between server and player

- Web browser requests and receives a **meta file** (a file describing the object) instead of receiving the file itself;
- Content-type header indicates specific audio/video application
- Browser launches media player and passes it the meta file
- Player sets up a TCP connection with server and sends HTTP request.

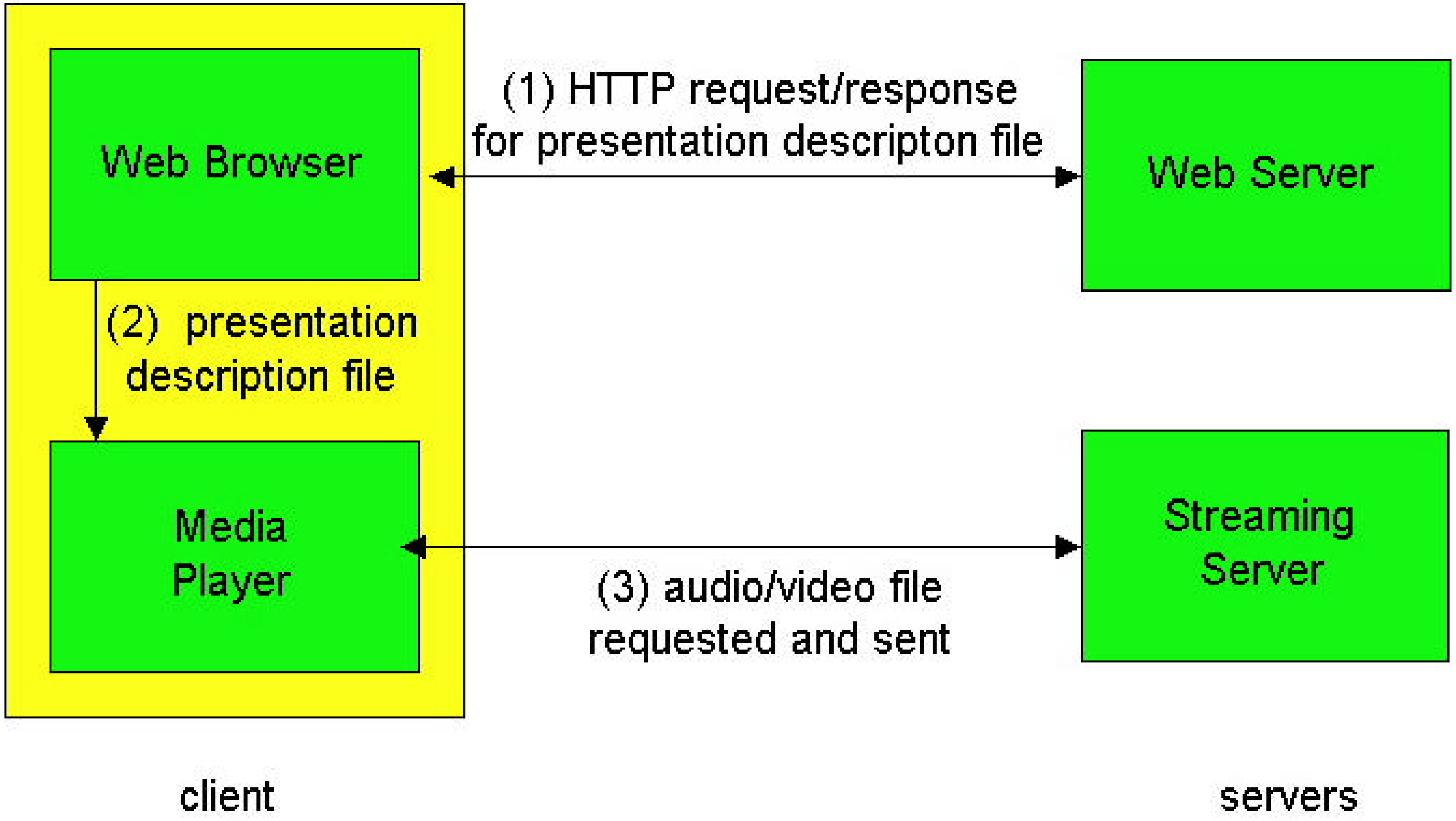

Web Browser
(1) HTTP request/response for meta file
Web Server
(2) meta file
Media Player
(3) audio/video file requested and sent over HTTP

Some concerns:

- Media player communicates over HTTP, which is not designed with pause, ff, rwnd commands
- May want to stream over UDP

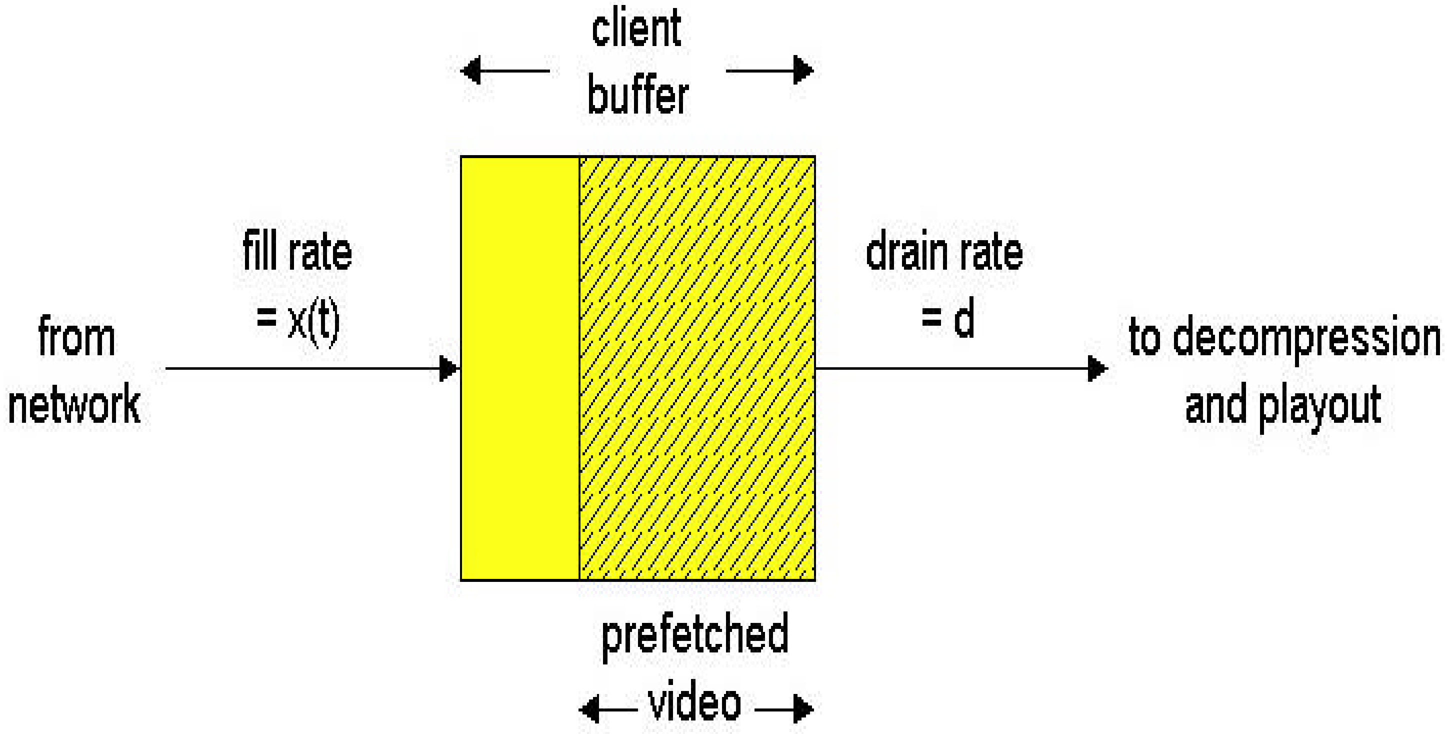
Streaming from a streaming server

- This architecture allows for non-HTTP protocol between server and media player
- Can also use UDP instead of TCP.


Web Browser
(1) HTTP request/response for presentation descripton file
Web Server
(2) presentation description file
Media Player
(3) audio/video file requested and sent
Streaming Server
client
servers

Options when using a streaming server

- Send at constant rate over UDP. To mitigate the effects of jitter, buffer and delay playback for 1-10 s. Transmit rate = d, the encoded rate. Fill rate x(t) equals d except when there is loss.
- Use TCP, and send at maximum possible rate under TCP; TCP retransmits when error is encountered; x(t) now fluctuates, and can become much larger than d. Player can use a much large buffer to smooth delivery rate of TCP.



Real Time Streaming Protocol: RTSP

HTTP

- Designers of HTTP had fixed media in mind: HTML, images, applets, etc.
- HTTP does not target stored continuous media (i.e., audio, video, SMIL presentations, etc.)

RTSP: RFC 2326

- Client-server application layer protocol.
- For user to control display: rewind, fast forward, pause, resume, repositioning, etc…

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; it can be transported over UDP or TCP
- does not specify how the media player buffers audio/video

RealNetworks

- Server and player use RTSP to send control info to each other

RTSP: out of band control

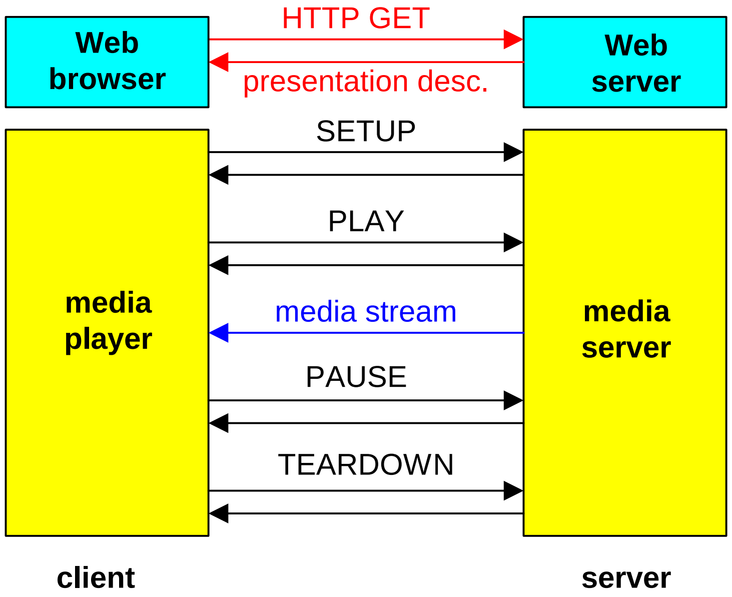
FTP uses an "out-of-band" control channel:

- A file is transferred over one channel.
- Control information (directory changes, file deletion, file renaming, etc.) is sent over a separate TCP connection.
- The "out-of-band" and "in-band" channels use different port numbers.

RTSP messages are also sent out-of-band:

- The RTSP control messages use different port numbers than the media stream, and are therefore sent out-of-band.
- The media stream, whose packet structure is not defined by RTSP, is considered "in-band".
- If the RTSP messages were to use the same port numbers as the media stream, then RTSP messages would be said to be "interleaved" with the media stream.

RTSP initiates and controls delivery



- Client obtains a description of the multimedia presentation, which can consist of several media streams.
- The browser invokes media player (helper application) based on the content type of the presentation description.
- Presentation description includes references to media streams, using the URL method rtsp://
- Player sends RTSP SETUP request; server sends RTSP SETUP response.
- Player sends RTSP PLAY request; server sends RTSP PLAY response.
- Media server pumps media stream.
- Player sends RTSP PAUSE request; server sends RTSP PAUSE response.
- Player sends RTSP TEARDOWN request; server sends RTSP TEARDOWN response.

Meta file 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>
```

RTSP session

- Each RTSP has a session identifier, which is chosen by the server.
- The client initiates the session with the SETUP request, and the server responds to the request with an identifier.
- The client repeats the session identifier for each request, until the client closes the session with the TEARDOWN request.
- RTSP port number is 554.
- RTSP can be sent over UDP or TCP. Each RTSP message can be sent over a separate TCP connection.

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
Session 4231

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

RTSP: streaming caching

- Caching of RTSP response messages makes little sense.
- But desirable to cache media streams closer to client.
- Much of HTTP/1.1 cache control has been adopted by RTSP.
  - Cache control headers can be put in RTSP SETUP requests and responses:
    - If-modified-since: , Expires: , Via: , Cache-Control:
- Proxy cache may hold only segments of a given media stream.
  - Proxy cache may start serving a client from its local cache, and then have to connect to origin server and fill missing material, hopefully without introducing gaps at client.
- When origin server is sending a stream through client, and stream passes through a proxy, proxy can use TCP to obtain the stream; but proxy still sends RTSP control messages to origin server.