The Analog domain

Top-left corner indicates source of slide:

- From Kurose & Ross. See copyright notice at end of this set.
- From Tanenbaum
- From Moors
Section outline

Signals
- Time and frequency domain
- Sampling & reconstruction of a signal
- Bandwidth and channel capacity

Media that carry content (emphasis on multimedia signals)
- Compression – audio, image, video
  - Lossless and lossy
  - Leads to burstiness
  - Layered coding → prioritisation
  - Adaptive applications
- Characteristics and requirements
  - Dealing with jitter
- Dealing with loss

Media used for transmission (Physical signals)
- Transmission impairments
- Transmission media
- Modulation
- Transmission coding
Jitter control

When packets traverse a network, they will experience delay. This delay may vary with network congestion. Jitter is the measure of delay variation.

Streaming media applications are sensitive to jitter, e.g. if a packet arrives later than normal, then the receiver may have nothing to play out for a short period, leading to audio silence or flickering images.

Ways to address jitter:
- In the network layer: Providing guaranteed “Quality of Service” (limit the possible jitter).
- In the application layer: Buffer received information to add a delay that compensates for the jitter.
Jitter Control

(a) High jitter.  
(b) Low jitter.
Buffering to compensate for jitter

Client-side buffering, playout delay compensates for network-added delay, delay jitter
Buffering to compensate for jitter

The receiver buffers information before playback, with duration of buffering compensating for variable transmission delay.

Packet 4 experiences more propagation delay (3 sec) ⇒ less playback delay (7 sec)

Packet 8 experiences excessive delay → discard it, or extend buffering and create gap in playback
How long should data be buffered?

Receiver needs an estimate of the range of network delays, before playing out packets (including the first – before receiving any samples of interarrival time!)

Larger delays:

- Reduce packet loss due to excessive delay
- Less interactive:
  - Longer to respond to controls, e.g. fast-forward etc
  - Longer delay for interactive media
- Cost more, through requiring larger buffers.

Solution: estimate expected delay/jitter...
Estimating delay/jitter

$t_i = \text{timestamp of packet } i \text{ (from source)}$
$r_i = \text{time when packet } i \text{ is received by destination}$
$d_i = \text{estimate of average network delay after receiving } i^{\text{th}} \text{ packet}$

$$d_i = (1 - u)d_{i-1} + u(r_i - t_i)$$

$u$ is a fixed constant (e.g., $u = 1/8$)

Impact of a particular $r_i-t_i$ measurement diminishes over time.

Also useful to estimate the average deviation of the delay, $v_i$:

$$v_i = (1 - u)v_{i-1} + u |r_i - t_i - d_i|$$

Estimates $d_i$ and $v_i$ are calculated for every received packet, although they are only used at the beginning of a “talk spurt”.

Playout time of packet $i$:

$$p_i = r_i + Kv_i = t_i + d_i + Kv_i$$

Where $K$ is some positive constant

Such “Exponentially Weighted Moving Averages” are widely used in networking TCP performs similar analysis to estimate RTT for retransmission timeouts §6.5.10 Service agreements may be specified in these terms, e.g. T p. 391
Protocols supporting streaming media

2 modes:

**Pull server**: Receiver requests more when needed. Many requests sent to single server.

**Push server**: Receiver requests a flow at a specified rate.
- For stored media, rate can be set to exceed the playout rate ⇒ ensure that buffer never empties, even if receiver’s clock is a bit faster than the source’s.
- If buffer fill > “high-water mark”, receiver sends pause message
- If buffer fill < “low-water mark”, receiver sends resume message
- “high-water mark” > “low-water mark” provides **hysteresis**: reduce rate of switching between modes & hence number of requests.

Protocols used:
- **Real Time Transport Protocol (RTP, RFC 3550)** for conveying timestamps for reconstructing the timing
- **Real Time Transport Control Protocol (RTCP, RFC 3550)**: Feedback to source about network properties (e.g. jitter & bandwidth)
- **Real Time Streaming Protocol (RTSP, RFC 2326)** to play/pause stream.
Outline
Dealing with packet loss

Audio/video often tolerate some loss. Can we use smart coding to increase the tolerance of loss? (May not be able to retransmit due to delay sensitivity.)

3 solutions:
- Forward Error Correction
- Redundant lower resolution stream
- Interleaving
1. Forward Error Correction

simple scheme

- for every group of $n$ chunks create a redundant chunk by exclusive OR-ing the $n$ original chunks
- send out $n+1$ chunks, increasing the bandwidth by factor $1/n$.
- can reconstruct the original $n$ chunks if there is at most one lost chunk from the $n+1$ chunks

Playout delay needs to be fixed to the time to receive all $n+1$ packets

Tradeoff: increasing $n$:
- less bandwidth waste
- longer playout delay
- higher probability that 2 or more chunks will be lost
2. Redundant lower resolution stream

“piggyback lower quality stream”
Send lower resolution audio stream as the redundant information e.g., nominal stream PCM at 64 kbps and redundant stream GSM at 13 kbps.

- Whenever there is non-consecutive loss, the receiver can conceal the loss.
- Can also append (n-1)st and (n-2)nd low-bit rate chunk

Both techniques 1 & 2 require additional bandwidth.
3. Interleaving & interpolation

Transmitting in order different from en/decoding order converts burst transmission loss into distributed en/decode loss. Loss of a packet reduces the temporal resolution rather than creating a gap in time. Can interpolate between received samples.

Chunks are broken up into smaller units, e.g. 4ms units

Packet contains small units from different chunks

If packet is lost, still have most of every chunk
No redundancy overhead, but adds to playout delay

Figure from Kurose and Ross. See also T Fig. 7-60
Outline
Pixelation (Pixelation)

Divide image into a matrix of picture elements (pixels)

Typical resolutions:
- Normal PAL/SECAM TV: 576 rows x 720 columns
- Computer monitors: 1024x768, 1280x960, 1600x1200, ...
Coding of pixels

Monochrome images: one signal: luminance (brightness)
Colour images: multiple colour signals. Coding depends on platform:

- Computer bitmapped displays:
  - Separate Red-Green-Blue signals, each with own luminance (e.g. 8b each).
  - 24b/pixel ⇒ 16M colours > number distinguishable by human eye
- Television (& JPEG/MPEG compression): 3 signals:
  - Luminance (Y) - for B&W backward compatibility
  - Chrominance (colour) signals (I, Q) named “hue & saturation” or “tint & color”
    e.g. NTSC from Red-Green-Blue:
    \[
    \begin{align*}
    Y &= 0.30R + 0.59G + 0.11B \\
    I &= 0.60R - 0.28G - 0.32B \\
    Q &= 0.21R - 0.52G + 0.31B
    \end{align*}
    \]

Eye is more responsive to brightness than colour ⇒ can encode colour with lower spatial resolution.
Redundancy in images

Still images exhibit **spatial redundancy**: value of a pixel is likely similar to that of adjacent pixels. Run-length & other encoding.

Moving images also exhibit **temporal redundancy**: most of one frame is identical to the preceding or following frame(s), with limited exceptions.
JPEG

Joint Photographic Experts Group (JPEG)
ISO standard IS 10918

• Often yields compression ratio of 20:1 or better
• Roughly symmetric: encoding takes as long as decoding

4 modes and many options (typical standard!) ⇒ we’ll examine simplified lossy sequential mode.

David Taubman of EE&T is one of the primary designers of the JPEG2000 standard.
JPEG operation (in lossy sequential mode)

A. Block preparation
- Convert RGB to luminance and chrominance
- Reduce resolution of chrominance, e.g. average 2x2 blocks of pixels
- Divide data into 8x8 blocks

Fig. 7-71 and 7-72
JPEG operation (in lossy sequential mode)

B. Discrete Cosine Transform: DCT(0,0) is the average over the block. Other elements:
- “tell how much spectral power is present at each spatial frequency”
- decay rapidly with distance from origin

\[ B(k_1, k_2) = \sum_{i=0}^{N_1-1} \sum_{j=0}^{N_2-1} 4 \cdot A(i,j) \cdot \cos \left( \frac{\pi \cdot k_1}{2 \cdot N_1} (2i + 1) \right) \cdot \cos \left( \frac{\pi \cdot k_2}{2 \cdot N_2} (2j + 1) \right) \]

N1, N2 = dimension of block; k1, k2 (Fx, Fy) = DCT indices
A(I,j) intensity of pixel @ row I, column j

Fig. 7-71 and 7-73
JPEG operation (in lossy sequential mode)

C. Quantization: Divide each DCT coefficient by a certain weighting
Each application supplies own weightings (sent at beginning of image?)

DCT Coefficients  \( \div \)  Quantization table  \( = \)  Quantized coefficients

<table>
<thead>
<tr>
<th>150</th>
<th>80</th>
<th>40</th>
<th>14</th>
<th>4</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>92</td>
<td>75</td>
<td>36</td>
<td>10</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>52</td>
<td>38</td>
<td>26</td>
<td>8</td>
<td>7</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>8</td>
<td>6</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>16</td>
<td>32</td>
<td>64</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>16</td>
<td>32</td>
<td>64</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>8</td>
<td>16</td>
<td>32</td>
<td>64</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>16</td>
<td>32</td>
<td>64</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>32</td>
<td>64</td>
</tr>
<tr>
<td>32</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>64</td>
</tr>
<tr>
<td>64</td>
<td>64</td>
<td>64</td>
<td>64</td>
<td>64</td>
<td>64</td>
<td>64</td>
<td>64</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>150</th>
<th>80</th>
<th>20</th>
<th>4</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>92</td>
<td>75</td>
<td>18</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>26</td>
<td>19</td>
<td>13</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

T Fig. 7-71 and 7-74
JPEG operation (in lossy sequential mode)

D. Differential quantization: Record only the difference between adjacent coefficients (not shown in this example)

E. Run length encoding: Encode strings of same value as (length, value), rather than (value, value, ... value)

F. Statistical (Huffman) output encoding: Encode more common values using shorter codes.

T Fig. 7-71 and 7-75
Quality of image display

Flicker: Image briefly disappears.
- Images decay faster on older retinas.
- Can appear on *still* and moving pictures
- Rates:
  - Movies (projecting full image each frame): 24 frames/sec
  - Screens (refreshing one row/field at a time): 50 fields/sec

Smoothness / jitter: If successive independent frames are spaced too far apart, motion will appear jerky.
- Can appear only on moving pictures.
- 25 frames/sec OK.
MPEG video coding

Motion Picture Experts Group (MPEG) of the ISO
Compress both audio & video (since movies contain sound).

Popular “MP3” music recordings use MPEG-1 audio layer 3

MPEG-1:
- ISO International Standard 11172
- 352 x 240
- used in CD-ROMs
- 1.2Mb/s (compression of 40:1)

MPEG-2:
- ISO International Standard 13818
- low res and 1440 x 1152 and 1920 x 1080
- basis for DVDs
- 4-8 Mb/s

MPEG-3:
- Intended for HDTV, but merged into MPEG-2 when shown that MPEG-2 could do this.

MPEG-4:
- often used in the Internet (used to record these lectures)
- < 1 Mb/s
MPEG frames

I (Intracoded): Self-contained still images, encoded using a variant of JPEG. Included once or twice per second, providing synchronisation points, e.g.:
- For receivers entering mid-way through multicast transmission.
- In case preceding frames were received in error.
- To avoid need for high-speed computation for fast-forward.

P (Predictive): Difference from last frame, e.g. identify moving body on still background.
- Detection of differences requires complicated processing ⇒ asymmetrical.

B (Bidirectional): Interpolation between last and next e.g. useful when an object is temporarily obscured (last & next are same, B frame is different)

D (DC-coded): Block averages (lower resolution used for fast forward). Not supported by MPEG-2.
Key points about video

**High bandwidth** – Mb/s
   Can adapt to some extent: Reduce resolution (DVD → TV) to reduce bandwidth

**Reasonably long between samples** (e.g. 40ms c.f. speech 125μs)

**Bursty**: Little to transmit when successive frames are similar.

**Unpredictable mean rate**: Depends on content:
   e.g. Rock video (frequent change of scene) vs test pattern
   ⇒ Difficult to precisely predict requirements during connection setup.

**Delay sensitive**:  
- For stored video, jitter increases size of playback buffer.  
- For interactive video, delays impede interaction.

It may be true that “Video killed the radio star”, but video can also kill networks!