Today’s lecture

• Compression
  – Lossless compression
  – Lossy compression

Why use compression?

• Pros
  – Less data to transfer
  – Application sees better throughput
  – Fewer bytes to store
  – Latency might be better

• Cons
  – Data quality can degrade
  – CPU overhead (harder to compress than to uncompress)
  – Latency might be worse

• Typically done at application or data link layer
Huffman codes

- Core idea: use shorter bit sequences to encode symbols that occur often
  - Symbols can be characters or something else
  - Works for some types of data
- Example: "Ramabhadran"
  - Need 8 symbols in alphabet
  - Equal size encoding
    - $11 \times 3 = 33$ bits
  - Huffman encoding
    - $4 \times 2 + 5 \times 3 + 2 \times 4 = 31$ bits

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>00</td>
</tr>
<tr>
<td>R</td>
<td>010</td>
</tr>
<tr>
<td>m</td>
<td>011</td>
</tr>
<tr>
<td>b</td>
<td>100</td>
</tr>
<tr>
<td>h</td>
<td>101</td>
</tr>
<tr>
<td>d</td>
<td>110</td>
</tr>
<tr>
<td>r</td>
<td>1110</td>
</tr>
<tr>
<td>n</td>
<td>1111</td>
</tr>
</tbody>
</table>

Other methods

- Run length encoding
  - Encode AAABBCD DDD as 3A2B1C4D
  - Works well for faxes
- Dictionary based methods
  - Use codes for words occurring in a dictionary
  - Words have variable lengths (may actually be a phrase)
  - Dictionary needs to be known to both sender and receiver
  - Can be static or dynamic (based on the data to compress)
  - Lempel-Ziv uses dynamic dictionaries
  - Works well for many kinds of data

Today’s lecture

- Compression
  - Lossless compression
  - Lossy compression
Lossy compression overview

- Data reconstructed by the receiver is similar, but not identical to the data at the sender
- Can achieve higher compression ratios
  - User can control quality loss or compression ratio
- Used for images, audio and video
  - JPEG (images)
  - MPEG-2 video
  - MP3 audio
  - Many other formats exist (some proprietary)

JPEG

- Operates on blocks of 8*8 pixels at a time
- DCT transforms data w/o loss
  - Like transforming cartesian to polar coordinates
- Quantization drops small coefficients which represent visually unimportant information
- Encoding (Huffman+RLE)
- Compression factor ~ 30

MPEG – video compression

- Typically MPEG encoding too expensive to do online
- Video is a sequence of frames (e.g. 30/second)
  - JPEG exploits spatial locality within images (frames)
  - MPEG also exploits temporal locality – typically the next frame is somewhat similar (compression factor ~ 100)
- MPEG uses 3 types of frames
  - I frames can be decoded independently
  - P frames depend on the previous I frame
  - B frames depend on the previous and next I or P frame
MPEG frame types

Compressing sound

- Represented as periodic samples
  - Phone quality 8 bit samples every 125 µs (64Kbps)
  - CD quality 16 bit samples 23 µs (stereo 1.41 Mbps)
- MP3
  - Part of the MPEG standard
  - Divides the sound into frequency bands
  - Works on blocks of 64 to 1024 samples
  - Uses DCT, quantization, and encoding
  - Compression factor up to 12