COMPUTER NETWORKS
CS 45201
CS 55201

CHAPTER 7
Presentation Protocols

P. Farrell and H. Peyravi

Department of Computer Science
Kent State University
Kent, Ohio 44242
farrell@cs.kent.edu
http://www.cs.kent.edu/~ farrell

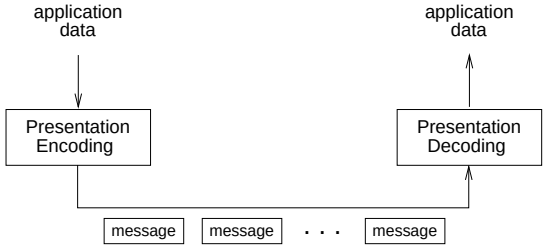
Fall 2001

Contents

Presentation Formatting

Overview

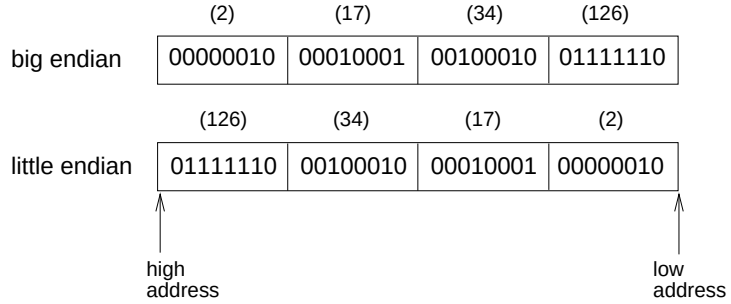
- Marshalling (encoding) application data into messages
- Unmarshalling (decoding) messages into application data



- Data types we consider:
  - integers
  - floating point numbers
  - character strings
  - arrays
  - structures
- Types of data we do not consider (now):
  - images
  - video
  - multimedia documents

Difficulties

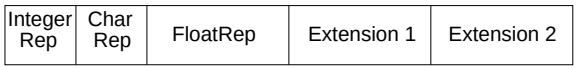
- Representation of base types
  - floating point: IEEE 754 versus non-standard
  - integer: big-endian versus little-endian (e.g., 34,677,374)



- Compiler layout of structures e.g. padding between fields
- Read 7.1.1 on Taxonomy
- XDR (External Data Representation) SunRPC
  - XDR provides canonical intermediate form
  - supports C type system except function pointers
  - uses compiled stub

NDR: Network Data Representation

- Defined by DCE
- Essentially the C type system
- Receiver-makes-right (architecture tag)
- Individual data items untagged
- Compiled stubs from IDL (Interface Definition Language)
- 4-byte architecture definition tag



  - `IntegrRep`
    - 0 = big-endian
    - 1 = little-endian
  - `CharRep`
    - 0 = ASCII
    - 1 = EBCDIC
  - `FloatRep`
    - 0 = IEEE 754
    - 1 = VAX
    - 2 = Cray
    - 3 = IBM

Data Compression

Data must be encoded into a message. Compression is concerned with with removing *redundancy* from that encoding. There are two classes of compression:

- Lossless: ensures that the data recovered from the compression/decompression process is exactly the same as the original data. Commonly used to compress executable code, text files, and numeric data.
- Lossy: does not promise that the data received is exactly the same as the data sent; removes information that it cannot later restore. (Hopefully, no one will notice.) Commonly used to compress digital imagery, including video.

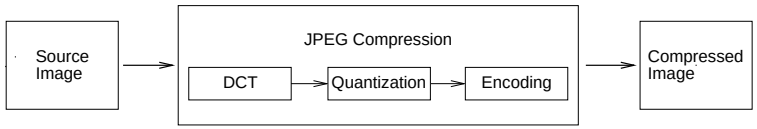
Note: The compression/decompression process takes time. Whether or not you compress data (and how much you compress it) depends on whether you have more cycles (for compression) or bandwidth (for transmission).

Lossless Compression Algorithms

- Run Length Encoding (RLE)
  - example: AAABBCDDDD encoded as 3A2B1C4D
  - good for scanned text (8-to-1 compression ratio) Faxes
  - can increase size for data with variation (e.g., some images)
- Differential Pulse Code Modulation (DPCM)
  - example: AAABBCDDDD encoded as A0001123333
  - reference can be changed
  - works better than RLE for many digital images (1.5-to-1)
- Dictionary-Based Methods
  - build dictionary of common terms (variable length strings)
  - transmit index into dictionary for each term
  - Lempel-Ziv (LZ) compression is the best-known example
  - commonly achieve 2-to-1 ratio on text
  - variation of LZ used to compress GIF images
    - first reduce 24-bit color to 8-bit color
    - treat common sequences of pixels as terms in dictionary (LZ)
    - not uncommon to achieve 10-to-1 compression ($\times 3$)

Image Compression

- JPEG: Joint Photographic Expert Group (ISO/ITU)
- Lossy still image compression
- Three phase process



  - process in 8×8 block chunks (macroblock)
  - greyscale: each pixel is a single value
  - color: each pixel is three values (YUV)
  - DCT: transforms signal from spatial domain into an equivalent signal in the frequency domain (loss-less)
    DCT: Discrete Cosine Transform
  - apply a quantization to the results (lossy)
  - RLE-like encoding (loss-less)
- DCT
  - DCT(0,0) is DC part i.e. average of 64 input pixels
  - DCT(i,j), as i,j increase get higher frequencies (finer detail)
  - higher frequencies less important

Quantization Phase

$$\texttt{QuantizedValue}(i,j) = \texttt{IntegerRound}(DCT(i,j)/\texttt{Quantum}(i,j))$$

$$\texttt{IntegerRound}(x) = \begin{cases} \lfloor x + 0.5 \rfloor & \text{if } x \geq 0 \\ \lfloor x - 0.5 \rfloor & \text{if } x < 0 \end{cases}$$

Decompression is then simply defined as

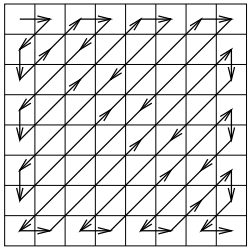
$$DCT(i,j) = \texttt{QuantizedValue}(i,j) \times \texttt{Quantum}(i,j)$$

$$\texttt{Quantum} = \begin{bmatrix} 3 & 5 & 7 & 9 & 11 & 13 & 15 & 17 \\ 5 & 7 & 9 & 11 & 13 & 15 & 17 & 19 \\ 7 & 9 & 11 & 13 & 15 & 17 & 19 & 21 \\ 9 & 11 & 13 & 15 & 17 & 19 & 21 & 23 \\ 11 & 13 & 15 & 17 & 19 & 21 & 23 & 25 \\ 13 & 15 & 17 & 19 & 21 & 23 & 25 & 27 \\ 15 & 17 & 19 & 21 & 23 & 25 & 27 & 29 \\ 17 & 19 & 21 & 23 & 25 & 27 & 29 & 31 \end{bmatrix}$$

Example of JPEG quantization table

- As i,j grow values grow, so higher frequencies (finer details) more likely to be lost

Encoding Phase

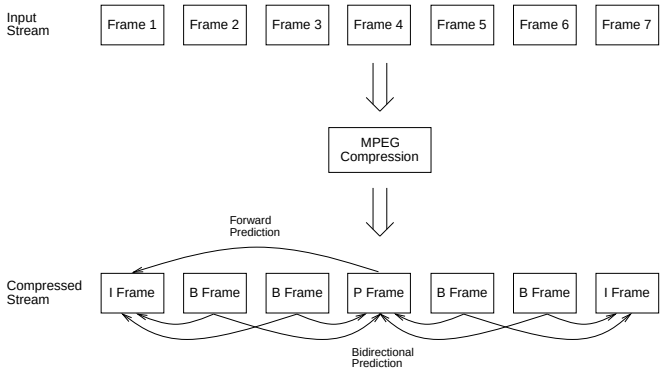


Encode along lines using

- difference from previous DC for DC
- Huffman for non-zero
- RLE for 0's

Video Compression

- MPEG: Motion Picture Expert Group
- Lossy compression of video
- First approximation: JPEG on each frame
- Added compression: remove inter-frame redundancy
- Frame types
  - I frames: intrapicture (self-contained)
  - P frames: predicted picture
  - B frames: bidirectional predicted picture



- Example sequence transmitted as I P B B I B B

- I frames 16×16 macroblocks. For YUV color representation
  - U, V downsampled to 8×8
- B and P frames. Each macroblock has:
  - coordinate for the macroblock in the frame
  - motion vector relative to previous reference frame
  - motion vector relative to subsequent reference frame (B only)
  - delta for each pixel in the macro block
    - delta encoding sends the difference between two frames.
  - Issue remaining
    - make macroblocks similar in following frames
    - motion estimation
- Effectiveness
  - typically 90-to-1
  - as high as 150-to-1
  - 30-to-1 for I frames
  - P & B frames get another 3 to 5$\times$