End-to-End Data

Outline
- Formatting
- Compression

Presentation Formatting

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

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

Taxonomy

- Data types
  - base types (e.g., ints, floats); must convert
  - flat types (e.g., structures, arrays); must pack
  - complex types (e.g., pointers); must linearize

- Conversion Strategy
  - canonical intermediate form
  - receiver-makes-right (an N x N solution)
**Taxonomy (cont)**

- Tagged versus untagged data
  - Type: INT
    - Len: 4
    - Value: 417892

- Stubs
  - Compiled
  - Interpreted

**eXternal Data Representation (XDR)**

- Defined by Sun for use with SunRPC
- C type system (without function pointers)
- Canonical intermediate form
- Untagged (except array length)
- Compiled stubs

```c
#define MAXNAME 256;
#define MAXLIST 100;

struct item {
  int count;
  char name[MAXNAME];
  int list[MAXLIST];
};

bool_t
xdr_item(XDR *xdrs, struct item *ptr)
{
  return(xdr_int(xdrs, &ptr->count) &&
         xdr_string(xdrs, &ptr->name, MAXNAME) &&
         xdr_array(xdrs, &ptr->list, &ptr->count,
                   MAXLIST, sizeof(int), xdr_int));
}
```

**Abstract Syntax Notation One (ASN.1)**

- An ISO standard
- Essentially the C type system
- Canonical intermediate form
- Tagged
- Compiled or interpreted stubs
- BER: Basic Encoding Rules

```
<table>
<thead>
<tr>
<th>tag</th>
<th>length</th>
<th>type</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT</td>
<td>4</td>
<td>byte integer</td>
<td>417892</td>
</tr>
</tbody>
</table>
```
Network Data Representation (NDR)

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

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

XML Schema

```xml
<?xml version="1.0"?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.cs.princeton.edu"
  elementFormDefault="qualified">
  <xs:element name="employee">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="name" type="xs:string"/>
        <xs:element name="title" type="xs:string"/>
        <xs:element name="id" type="xs:string"/>
        <xs:element name="hiredate" type="xs:dateTime"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Markup Languages

- HyperText Markup Language (HTML)
- Extensible Markup Language (XML)

```xml
<?xml version="1.0"?>
<employee>
  <name>Spongebob Squarepants</name>
  <title>Frycook</title>
  <id>0123456</id>
  <hiredate>
    <day>1</day>
    <month>May</month>
    <year>1999</year>
  </hiredate>
</employee>
```

Compression Overview

- Encoding and Compression
  - Huffman codes
- Lossless
  - data received = data sent
  - used for executables, text files, numeric data
- Lossy
  - data received does not match data sent
  - used for images, video, audio
Lossless Algorithms

- Run Length Encoding (RLE)
  - example: AAABBCDDDD encoding as 3A2B1C4D
  - good for scanned text (8-to-1 compression ratio)
  - can increase size for data with variation (e.g., some images)
- Differential Pulse Code Modulation (DPCM)
  - example AAABBCDDDD encoding as A000112333
  - change reference symbol if delta becomes too large
  - 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) 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 sequence of pixels as terms in dictionary
  - can achieve 10-to-1 compression (less common)

Image Compression

- JPEG: Joint Photographic Expert Group (ISO/ITU)
- Lossy still-image compression
- Three phase process
  - process in 8x8 block chunks (macro-block)
  - DCT: transforms signal from spatial domain into and equivalent signal in the frequency domain (loss-less)
  - apply a quantization to the results (lossy)
  - RLE-like encoding (loss-less)

Quantization and Encoding

- Quantization Table
  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

- Encoding Pattern
MPEG

- Motion Picture Expert Group
- Lossy compression of video
- First approximation: JPEG on each frame
- Also remove inter-frame redundancy

MPEG (cont)

- Frame types
  - I frames: intrapicture
  - P frames: predicted picture
  - B frames: bidirectional predicted picture

  ![MPEG Diagram]

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

MPEG (cont)

- B and P frames
  - coordinate for the macroblock in the frame
  - motion vector relative to previous reference frame (B, P)
  - motion vector relative to subsequent reference frame (B)
  - delta for each pixel in the macro block

- Effectiveness
  - typically 90-to-1
  - as high as 150-to-1
  - 30-to-1 for I frames
  - P and B frames get another 3 to 5x

MP3

- CD Quality
  - 44.1 kHz sampling rate
  - $2 \times 44.1 \times 1000 \times 16 = 1.41$ Mbps
  - $49/16 \times 1.41$ Mbps = 4.32 Mbps

- Strategy
  - split into some number of frequency bands
  - divide each subband into a sequence of blocks
  - encode each block using DCT + Quantization + Huffman
  - trick: how many bits assigned to each subband