Today’s Lecture

- End-to-End Application Issues
- Coding
  - Pthreads / etc.
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

- 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
#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

\[
\text{tag, length, value} = \begin{array}{ccc}
\text{type} & \text{length} & \text{value} \\
\hline
\text{INT} & 4 & 4 \text{-byte integer} \\
\end{array}
\]

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

\[
\text{IntegerRep} = \begin{cases}
0 & \text{big-endian} \\
1 & \text{little-endian}
\end{cases}
\]

\[
\text{CharRep} = \begin{cases}
0 & \text{ASCII} \\
1 & \text{EBCDIC}
\end{cases}
\]

\[
\text{FloatRep} = \begin{cases}
0 & \text{IEEE 754} \\
1 & \text{VAX} \\
2 & \text{Cray} \\
3 & \text{IBM}
\end{cases}
\]
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>
```

XML Schema

```xml
<?xml version="1.0"?>
<xsd:schema xmlns="http://www.cs.princeton.edu/XMLSchema"
            targetNamespace="http://www.cs.princeton.edu"
            xmlns="http://www.cs.princeton.edu"
            elementFormDefault="qualified">
  <xsd:element name="employee">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="name" type="xsd:string"/>
        <xsd:element name="title" type="xsd:string"/>
        <xsd:element name="id" type="xsd:string"/>
        <xsd:element name="hiredate">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="day" type="xsd:integer"/>
              <xsd:element name="month" type="xsd:string"/>
              <xsd:element name="year" type="xsd:integer"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

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 A0001123333
  – 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

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5</td>
<td>7</td>
<td>9</td>
<td>11</td>
<td>13</td>
<td>15</td>
<td>17</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>9</td>
<td>11</td>
<td>13</td>
<td>15</td>
<td>17</td>
<td>19</td>
</tr>
<tr>
<td>7</td>
<td>9</td>
<td>11</td>
<td>13</td>
<td>15</td>
<td>17</td>
<td>19</td>
<td>21</td>
</tr>
<tr>
<td>9</td>
<td>11</td>
<td>13</td>
<td>15</td>
<td>17</td>
<td>19</td>
<td>21</td>
<td>23</td>
</tr>
<tr>
<td>11</td>
<td>13</td>
<td>15</td>
<td>17</td>
<td>19</td>
<td>21</td>
<td>23</td>
<td>25</td>
</tr>
<tr>
<td>13</td>
<td>15</td>
<td>17</td>
<td>19</td>
<td>21</td>
<td>23</td>
<td>25</td>
<td>27</td>
</tr>
<tr>
<td>15</td>
<td>17</td>
<td>19</td>
<td>21</td>
<td>23</td>
<td>25</td>
<td>27</td>
<td>29</td>
</tr>
</tbody>
</table>

- 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

- Example sequence transmitted as I P 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

Coding

- Form into small groups
  - Twist: Must sit by people who you did not work with on the project

Build a simple UDP chat server
Write Threaded Code

• UDP Server
  – Step 1
    • Listen on port X (pass in via parameter)
    • Read message / display on screen
  – Step 2
    • Keep track of incoming IP / ports
    • Design
      – Local vs. global
      – struct vs. class

Write Main Function

• main function
  – Step 1
    • Start up the thread
    • Loop until input is QUIT!
  – Step 2
    • Send the typed text to all known other clients
  – Step 3
    • Allow adding of IP via syntax
      – ADDCLIENT 129.74.20.40 8908
  – Step 4
    • Dump current client list via DUMPSTATUS