Weynand's book [WW07] covers almost every aspect of video, including compression and many other topics. For a comprehensive description of the early MPEG standards, see Mitchell et al. [MPFL96]. For a description of MP3, see Noll [Nol97].

Finally, we recommend the following live references:

- http://www.w3.org/TR/REC-xml/: The most recent XML specification
- http://mpeg.chiariglione.org/: Home page of the Motion Picture Experts Group, a source for lots of MPEG material

EXERCISES

1. Consider the following C code:

```
#define  MAXSTR 100

struct date {
    char  month[MAXSTR];
    int   day;
    int   year;
};

struct employee {
    char   name[MAXSTR];
    int    ssn;
    struct date *hireday;
    int    salary_history[5];
    int    num_raises;
};

static struct date date0 = {"DECEMBER", 2, 1998};
static struct date date1 = {"JANUARY", 7, 2002};

static struct employee employee0 = {"RICHARD", 4376,
                                  &date0,{80000, 85000,
                                  90000, 0, 0}, 2};
static struct employee employee1 = {"MARY", 4377,
                                  &date1, {90000,
                                  150000, 0, 0, 0}, 1};
```

where num_raises + 1 corresponds to the number of valid entries in array salary_history. Show the on-the-wire representation of employee0 that is generated by XDR.

✓ **2.** Show the on-the-wire representation of employee1 from the previous problem that is generated by XDR.

3. For the data structures given in the previous problem, give the XDR routine that encodes/decodes these structures. If you have XDR available to you, run this routine and measure how long it takes to encode and decode an example instance of structure employee.

4. Using library functions like htonl and Unix's bcopy or Windows' CopyMemory, implement a routine that generates the same on-the-wire representation of the structures given in Exercise 1 as XDR does. If possible, compare the performance of your "by-hand" encoder/decoder with the corresponding XDR routines.

5. Use XDR and htonl to encode a 1000-element array of integers. Measure and compare the performance of each. How do these compare to a simple loop that reads and writes a 1000-element array of integers? Perform the experiment on a computer for which the native byte order is the same as the network byte order, as well as on a computer for which the native byte order and the network byte order are different.

6. Write your own implementation of htonl. Using both your own htonl and (if little-endian hardware is available) the standard library version, run appropriate experiments to determine how much longer it takes to byte-swap integers versus merely copying them.

7. Give the ASN.1 encoding for the following three integers. Note that ASN.1 integers, like those in XDR, are 32 bits in length.

(a) 101

(b) 10,120

(c) 16,909,060

✓ **8.** Give the ASN.1 encoding for the following three integers. Note that ASN.1 integers, like those in XDR, are 32 bits in length.

(a) 15

(b) 29,496,729

(c) 58,993,458

9. Give the big-endian and little-endian representation for the integers from Exercise 7.

✓ 10. Give the big-endian and little-endian representation for the integers from Exercise 8.

11. XDR is used to encode/decode the header for the SunRPC protocol illustrated by Figure 5.18. The XDR version is determined by the RPCVersion field. What potential difficulty does this present? Would it be possible for a new version of XDR to switch to little-endian integer format?

12. The presentation-formatting process is sometimes regarded as an autonomous protocol layer, separate from the application. If this is so, why might including data compression in the presentation layer be a bad idea?

13. Suppose you have a machine with a 36-bit word size. Strings are represented as five packed 7-bit characters per word. What presentation issues on this machine have to be addressed for it to exchange integer and string data with the rest of the world?

14. Using the programming language of your choice that supports user-defined automatic type conversions, define a type netint and supply conversions that enable assignments and equality comparisons between ints and netints. Can a generalization of this approach solve the problem of network argument marshalling?

15. Different architectures have different conventions on bit order as well as byte order—whether the least significant bit of a byte, for example, is bit 0 or bit 7. RFC 791 [Pos81] defines (in its Appendix B) the standard network bit order. Why is bit order then not relevant to presentation formatting?

☆ 16. Let $p \leq 1$ be the fraction of machines in a network that are big-endian; the remaining $1 - p$ fraction are little-endian. Suppose we choose two machines at random and send an int from one to the other. Give the average number of byte-order conversions needed for both big-endian network byte order and receiver-makes-right for $p = 0.1$, $p = 0.5$, and $p = 0.9$. (Hint: The probability that both endpoints are big-endian is p^2; the probability that the two endpoints use different byte orders is $2p(1-p)$.)

17. Describe a representation for XML documents that would be more compact and more efficient to process than XML text.

18. Experiment with a compression utility (e.g., compress, gzip, or pkzip). What compression ratios are you able to achieve? See if you can generate data files for which you can achieve 5:1 or 10:1 compression ratios.

☆ **19.** Suppose a file contains the letters a, b, c, and d. Nominally we require 2 bits per letter to store such a file.

(a) Assume the letter a occurs 50% of the time, b occurs 30% of the time, and c and d each occurs 10% of the time. Give an encoding of each letter as a bit string that provides optimal compression. (Hint: Use a single bit for a.)

(b) What is the percentage of compression you achieve above? (This is the average of the compression percentages achieved for each letter, weighted by the letter's frequency.)

(c) Repeat this, assuming a and b each occurs 40% of the time, c occurs 15% of the time, and d occurs 5% of the time.

☆ **20.** Suppose we have a compression function c, which takes a bit string s to a compressed string $c(s)$.

(a) Show that for any integer N there must be a string s of length N for which $\text{length}(c(s)) \geq N$; that is, no effective compression is done.

(b) Compress some already compressed files (try compressing with the same utility several times in sequence). What happens to the file size?

(c) Given a compression function c as in (a), give a function c' such that for all bit strings $s, \text{length}(c'(s)) \leq \min(\text{length}(c(s)), \text{length}(s)) + 1$; that is, in the worst case, compression with c' expands the size by only 1 bit.

21. Give an algorithm for run length encoding that requires only a single byte to represent nonrepeated symbols.

22. Write a program to construct a dictionary of all "words," defined to be runs of consecutive nonwhitespace, in a given text file. We might then compress the file (ignoring the loss of whitespace information) by representing each word as an index in the dictionary. Retrieve the file rfc791.txt containing [Pos81], and run your program on it. Give the size of the compressed file assuming first that each word is encoded with 12 bits (this should be sufficient), and then that the 128 most common words are encoded with 8 bits and the rest with 13 bits. Assume that the dictionary itself can be stored by using, for each word, $\mathrm{length}(\mathrm{word}) + 1$ bytes.

☆ **23.** The one-dimensional discrete cosine transform is similar to the two-dimensional transform, except that we drop the second variable (j or y) and the second cosine factor. We also drop, from the inverse DCT only, the leading $1/\sqrt{2}N$ coefficient. Implement this and its inverse for $N = 8$ (a spreadsheet will do, although a language supporting matrices might be better) and answer the following:

(a) If the input data is $\langle 1, 2, 3, 5, 5, 3, 2, 1\rangle$, which DCT coefficients are near 0?

(b) If the data is $\langle 1, 2, 3, 4, 5, 6, 7, 8\rangle$, how many DCT coefficients must we keep so that after the inverse DCT the values are all within 1% of their original values? 10%? Assume dropped DCT coefficients are replaced with 0s.

(c) Let s_i, for $1 \leq i \leq 8$, be the input sequence consisting of a 1 in position i and 0 in position $j, j \neq i$. Suppose we apply the DCT to s_i, zero the last three coefficients, and then apply the inverse DCT. Which $i, 1 \leq i \leq 8$, results in the smallest error in the ith place in the result? The largest error?

24. Compare the size of an all-white image in JPEG format with a typical photographic image of the same dimensions. At what stage or stages of the JPEG compression process does the white image become smaller than the photographic image?

For the next three exercises, the utilities cjpeg and djpeg may be useful and can be obtained from http://www.ijg.org/. Other JPEG conversion utilities can also be used. For manual creation and examination of graphics files, the pgm portable grayscale format is recommended; see the Unix pgm(5)/ppm(5) man pages.

25. Create a grayscale image consisting of an 8×8 grid with a vertical black line in the first column. Compress into JPEG format and decompress. How far off are the resultant bytes at the default quality setting? How would you describe the inaccuracies introduced, visually? What quality setting is sufficient to recover the file exactly?

26. Create an 8×8 grayscale image consisting of a 64-character ASCII text string. Use lowercase letters only, with no whitespace or punctuation. Compress into JPEG format and decompress. How recognizable is the result, as text? Why might adding whitespace make things worse? With the quality setting at 100, would this be a plausible way of compressing text?

27. Write a program that implements forward and backward DCT, using floating-point arithmetic. Run the program on a sample grayscale image. Since DCT is lossless, the image output by the program should match the input. Now modify your program so that it zeroes some of the higher-frequency components, and see how the output image is affected. How is this different from what JPEG does?

28. Express DCT(0,0) in terms of the average of the $pixel(x, y)$s.

29. Think about what functions might reasonably be expected from a video standard: fast-forward, editing capabilities, random access, and so on. (See the paper by Le Gall, "MPEG: A video compression standard for multimedia applications," given in this chapter's Further Reading list, for more ideas.) Explain MPEG's design in terms of these features.

30. Suppose you want to implement fast-forward and reverse for MPEG streams. What problems do you run into if you limit your mechanism to displaying I frames only? If you don't, then to display a given frame in the fast-forward sequence, what is the largest number of frames in the original sequence you may have to decode?

31. Use mpeg_play to play an MPEG-encoded video. Experiment with options, particularly -nob and -nop, which are used to omit the B and P frames, respectively, from the stream. What are the visible effects of omitting these frames?

32. The mpeg_stat program can be used to display statistics for video streams. Use it to determine, for several streams:

(a) Number and sequence of I, B, and P frames.

(b) Average compression rate for the entire video.

(c) Average compression rate for each type of frame.

33. Suppose we have a video of two white points moving toward each other at a uniform rate against a black background. We encode it via MPEG. In one I frame the two points are 100 pixels apart; in the next I frame they have merged. The final point of merger happens to lie at the center of a 16×16 macroblock.

(a) Describe how you might optimally encode the Y component of the intervening B (or P) frames.

(b) Now suppose the points are in color and that the color changes slowly as the points move. Describe what the encoding of the U and V values might look like.