CHAPTER 28

PRACTICE SET

Questions

Q28-1. In dictionary coding, the number of iterations in the loop is one less than the number of characters in the message. In this case, the number of iterations is 59.

Q28-3. In a Huffman tree, the number of leaves is the same as the number of symbols. Therefore, we have 20 leaves in the tree.

Q28-5. The encoding table can be the following:

$A \rightarrow 00$ $\quad$ $B \rightarrow 01$ $\quad$ $C \rightarrow 10$ $\quad$ $D \rightarrow 11$

Huffman coding cannot help here because we could have encoded each character as two bits. Huffman coding will help if the occurrence probabilities of the characters in the messages are different and we can send fewer bits for characters with high probability and more bits for characters with low probability.

Q28-7. ADM is a variation of DM in which the value of Δ may change from one step to another.

Q28-9. The answer is given for each case for both PCM and DM.

a. PCM: 4 (because $2^4 = 16$) and DM: 1

b. PCM: 5 (because $2^5 = 32$) and DM: 1

c. PCM: 6 (because $2^6 = 64$) and DM: 1

Q28-11. In DM, only the immediate previous y_n value is used in the calculation of e_n. In DPCM, weighted averages of several previous y_n values are used in the calculation of e_n.

Q28-13. The answer is no. The value of the T matrix entries, for each value of N, is fixed. The receiver either has the matrix or can generate it.

Q28-15. The value of Q(m, n) becomes larger and larger when we move toward the higher frequencies in the matrix, from M(0, 0) to M(N–1, N–1) diagonally. This is done to reduce the effect of higher frequencies. Human eyes cannot recognize the higher frequencies when their amplitude (strength) is outside of an upper and a lower threshold. The values in the Q(m, n) matrix are selected based on this property. Dividing the amplitude of higher frequencies by a larger number makes these amplitudes smaller and smaller, finally moving them toward zero values and elimination (better compression).

Q28-17. The whole idea is to change the real numbers (values with fractions) to integer values to reduce the number of transmitted bits. The real value 14.125 or $(1110.001)_2$ needs seven bits to send; the integer value 14 or $(1110)_2$ needs only four bits to send. The number of bits to send tremendously increases if we want to use a high level of precision.

Q28-19. The answer is no. Multimedia data encoded in JPEG cannot be recovered using GIF and vice versa. If two entities want to communicate, they must use the same encoding and decoding scheme.

Q28-21. In the second approach, the whole multimedia file is downloaded using HTTP as the application-layer protocol. In the third approach, HTTP is only responsible for downloading the metafile; the actual multimedia file is normally downloaded using a specific application program that may use the service of UDP/RTP.

Q28-23. In live audio/video, the multimedia data is being sent on the Internet no matter whether we use it or not. In real-time interactive audio/video, two parties start exchanging multimedia when they both agree to do so.

Q28-25. The answer is yes. The protocol defines a server as a process running at the application level. The two servers can be on the same machine or different machines.

Q28-27. The answer is negative. The sequence numbers or timestamps are needed to glue the chunks together in time. When all chunks can be carried in one packet, they arrive all together and in the order they are inserted into the packet. There is no need for sequence numbers or timestamps.

Q28-29. The sequence numbers in RTP and TCP play almost the same role. They check to see if a packet is missing. However, there are some differences in the nature and applications of the sequence numbers in these two protocols.

a. In TCP, sequence numbers are byte-oriented. The sequence number defines the number of the first byte in each segment. This means that the sequence number can jump from one segment to another if the segment carries more than one byte. In RTP, sequence numbers are packet oriented: the sequence number defines the packet. Two consecutive packets sent by the source have consecutive sequence numbers.

b. The applications of the sequence numbers in these two protocols are somewhat different. The sequence numbers in TCP are related to acknowledgment numbers, and, if the sequence numbers attached to bytes in a packet are not acknowledged, the bytes are resent. In RTP, the sequence numbers are still used to indicate whether an RTP packet is lost. However, there is no retransmission. The receiver needs to know this fact to somehow find a remedy for it.

Q28-31. If there is only one source sending the multimedia application and audio and video are mixed by the application program, we have only one synchronization source (the synchronization source and contributor source is the same). This means that the value of the contributor count is 0 and there is no CSRC field. Without the extension header and CSRC fields, the size of the RTP packet is only 12 bytes.

Q28-33. The encoding applied on a chunk of data can be different from the encoding applied on another chunk of data, although this practice is not very common.

Q28-35. Each packet needs to carry an SSRC and a CSRC identifier. The SSRC identifier is the same for audio and video packets, but the CSRC identifier is different for each type of packet. In other words, we have two CSRCs, but only one SSRC during the session.

Q28-37. RTCP sends messages to control the flow and quality of data and allows the receiver to send feedback to the source. TCP already has this mechanism, so it doesn't need this protocol.

Q28-39. The caller and the callee each play the roles of the client and the server. In other words, SIP can be considered as a peer-to-peer application program instead of a client-server program. Both parties use the registered port number 5060.

Q28-41. This needs to be defined during the call setup, when they exchange SIP messages to establish the call. Each party can define the port number at which desires to receive the RTP packets.

Q28-43. The wired or wireless environment defines the link at which communication occurs (data-link layer). The RTP/RTCP and SIP are application-layer protocols. Like any application-layer message, they are eventually carried in a data-link layer (wired or wireless).

Q28-45. Bob can register the two addresses with the SIP proxy server. When a telephone call arrives, the proxy server can *fork* the call to both destinations.

Q28-47. Yes, H.323 can be used for both audio and video.

Problems

P28-1. We replace each run of the same character with a count and the symbol as shown below.

3A6C1B4C5D4A3B

The compression rate is 26/14 or almost 1.86.

P28-3. Let us show how the dictionary for each case is made using the knowledge about the previous case.

a. Dictionary is 0 $\rightarrow$ A; the code is 0.

b. Dictionary is 0 $\rightarrow$ A and 1 $\rightarrow$ AA; the code is 00.

c. Dictionary is 0 $\rightarrow$ A and 1 $\rightarrow$ AA; the code is 01.

d. Dictionary is 0 $\rightarrow$ A, 1 $\rightarrow$ AA, and 2 $\rightarrow$ AAA; the code is 010.

e. Dictionary is 0 $\rightarrow$ A, 1 $\rightarrow$ AA, and 2 $\rightarrow$ AAA; the code is 011.

f. Dictionary is 0 $\rightarrow$ A, 1 $\rightarrow$ AA, and 2 $\rightarrow$ AAA; the code is 012.

P28-5. We follow the procedure in Table 28.2. The loop is shown inside the frame with a thick border. We need to initialize the dictionary before reading the code. The message is "**AACCCBCCDDAB**".

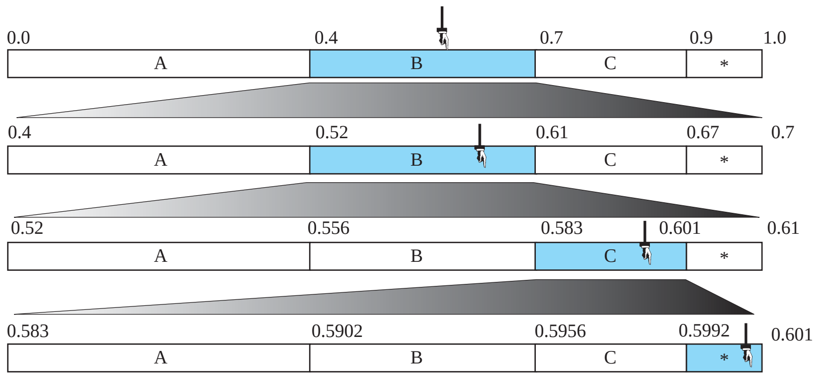
	PreC	S	C	S + first char	Dictionary		Message
					0	A	
					1	B	
					2	C	
					3	D	
			0				A
if	0	A	0	AA	4	AA	A
if	0	A	2	AC	5	AC	C
else	2	C	6	CC	6	CC	CC
if	6	CC	1	CCB	7	CCB	B
if	1	B	6	BC	8	BC	CC
if	6	CC	3	CCD	9	CCD	D
if	3	D	3	DD	10	DD	D
if	3	D	0	DA	11	DA	A
if	0	A	1	AB	12	AB	B

P28-7. We need to read the code, bit by bit, until we get a sequence of bits that can be decoded using the table:

0	0	1	1	0	1	1	0	0	1	1	1	1	0	1	1	1	1	1	1	0	1	0
A	**A**	–	–	**C**	–	–	**C**	**A**	–	–	**D**	–	**B**	–	–	**D**	–	–	**D**	**A**	–	**B**

The decoded message is "AACCADBDDAB".

P28-9. The code 100110011 is interpreted as $(0.100110011)_2$ or (0.599609375) in decimal. We start with the half-open interval [0, 1) to see where this number fits. The following shows how we narrow the intervals to find the location of the received code. The result is "BBC*"; the actual message is "BBC".



P28-11. We first calculate the value of q_n from the code. If the code is bit 1, the value of q_n is 1; otherwise, it is −1. The value of the y_n can then be calculated as $y_n = y_{n-1} + q_n \times \Delta$. Note that the value of y_0 is not shown in the table, but is given in the problem as $y_0 = 8$.

n	1	2	3	4	5	6	7	8	9	10	11
C_n	1	0	0	1	0	1	1	0	0	1	1
q_n	1	−1	−1	1	−1	1	1	−1	−1	1	1
y_n	14	8	2	8	2	8	14	8	2	8	14

P28-13. We first calculate the value of q_n from the code. If the code is bit 1, the value of q_n is 1; otherwise, it is −1. The value of M_n can then be calculated depending on the change in q_n. We can then calculate the value of Δ_n as defined in the problem. The value of y_n can then be calculated using $y_n = y_{n-1} + q_n \times \Delta_n$. Note that the value of y_0 is not shown in the table, but is given in the problem as $y_0 = 20$.

n	1	2	3	4	5	6	7	8	9	10	11
C_n	1	1	1	0	0	1	1	0	0	1	1
q_n	1	1	1	−1	−1	1	1	−1	−1	1	1

M_n	1	1.5	2.25	1.13	1.70	0.85	1.28	0.64	0.96	0.48	0.72
Δ_n	4	6	13.5	15.3	26	22.1	28.3	18.1	17.4	8.4	6
y_n	24	30	43.5	28.2	2.2	24.3	52.6	34.5	17.1	25.5	31.5

P28-15. The answer is yes. We have $T(m, n) = C(m, n) \cos(x)$. The value of $\cos(x)$ is always between -1 and 1. We only need to prove that $C(m, n) \leq 1$, which can be done using two cases:

a. If $m = 0 \rightarrow C(m, n) = (1/N)^{1/2} \leq 1$ because $N \geq 1$.

b. If $m > 0 \rightarrow C(m, n) = (2/N)^{1/2} \leq 1$ because $N \geq 2$.

P28-17. The following shows the four matrices. The first one ($N = 1$) is actually a number (scalar value). Note that we have rounded the values to two decimal digits.

$$\begin{bmatrix} 1.00 \end{bmatrix} \qquad \begin{bmatrix} 0.71 & 0.71 \\ 0.71 & -0.71 \end{bmatrix} \qquad \begin{bmatrix} 0.50 & 0.50 & 0.50 & 0.50 \\ 0.65 & 0.27 & -0.27 & -0.65 \\ 0.50 & -0.50 & -0.50 & 0.50 \\ 0.27 & -0.65 & 0.65 & -0.27 \end{bmatrix}$$

$N = 1$ $\qquad$ $N = 2$ $\qquad$ $N = 4$

$$\begin{bmatrix} 0.35 & 0.35 & 0.35 & 0.35 & 0.35 & 0.35 & 0.35 & 0.35 \\ 0.49 & 0.42 & 0.28 & 0.10 & -0.10 & -0.28 & -0.42 & -0.49 \\ 0.46 & 0.19 & -0.19 & -0.46 & -0.46 & -0.19 & 0.19 & 0.46 \\ 0.42 & -0.10 & -0.49 & -0.28 & 0.28 & 0.49 & 0.10 & -0.42 \\ 0.35 & -0.35 & -0.35 & 0.35 & 0.35 & -0.35 & -0.35 & 0.35 \\ 0.28 & -0.49 & 0.10 & 0.42 & -0.42 & -0.10 & 0.49 & -0.28 \\ 0.19 & -0.46 & 0.46 & -0.19 & -0.19 & 0.46 & -0.46 & 0.19 \\ 0.10 & -0.28 & 0.42 & -0.49 & 0.49 & -0.42 & 0.28 & -0.10 \end{bmatrix}$$

$N = 8$

P28-19. The palette should have the combination of $2 \times 2 \times 2 = 8$ rows, as shown below with the bit index.

	Red	Blue	Green	
000	00000000	00000000	00000000	R = 0, B = 0, G = 0
001	00000000	00000000	00000100	R = 0, B = 0, G = 4
010	00000000	00000101	00000000	R = 0, B = 5, G = 0
011	00000000	00000101	00000100	R = 0, B = 5, G = 4
100	00000111	00000000	00000000	R = 7, B = 0, G = 0
101	00000111	00000000	00000100	R = 7, B = 0, G = 4
110	00000111	00000101	00000000	R = 7, B = 5, G = 0
111	00000111	00000101	00000100	R = 7, B = 5, G = 4

a. For each color in the palette we can send three bits (the index of the row).

b. The bit index for the red color is 100, for the blue color is 010, for the green color is 001, for the black color is 000, for the white color is 111, and for the magenta is 110.

P28-21. The answer follows:

a. At time 00:00:17

first packet:	10 units arrived, 9 units played, 1 unit in buffer
second packet:	2 units arrived, none played, 3 units in buffer

b. At time 00:00:20

first packet:	none left
second packet:	5 units arrived, 2 units played, 3 units in buffer

c. At time 00:00:25

first packet:	none left
second packet:	10 units arrived, 7 units played, 3 units in buffer

d. At time 00:00:30

first packet:	none left
second packet:	none left
third packet:	3 units arrived, 2 units played, 1 unit in buffer

P28-23. The first eight hexadecimal digits are in binary:

1000 0110 **0**000 0011 0010 0001 0011 0010

a. The first two bits define the version, which means the version is 2.

b. The third bit is the P bit, which means there is no padding.

c. The fourth bit is the X bit, which means there is no extension header.

d. The next four bits define the number of contributors, which is 6 in this case.

e. The seven bits after the color bit (M) (with the value 3 in decimal) define the payload type, which means GSM audio.

f. Since we have six contributors, the total size of the packet header is 12 bytes (first part of the header) plus 6×4 bytes for CSRCs, for a total of 36 bytes.

P28-25. We can mention several reasons.

a. Live or real-time streaming of multimedia requires that the data be sent in chunks. A chunk of data is a sequence of bits that have been encoded together and need to be decoded together; each bit may depend on the rest of the bits. Although TCP collects eight bits in a byte and transfers each byte as an atomic unit, the size of the byte is not enough to be a chunk of multimedia data (even a single pixel in an image may need more than eight bits). A chunk cannot be associated with several bytes because TCP may split bytes, making a chunk into different segments, which means a chunk of multimedia data may be in one segment and the other parts in other segments. What is needed in multimedia is that a chunk of data be carried in a single packet. Although TCP may use push and urgent flags to artificially create boundaries for each segment, it is not guaranteed that a TCP implementation will honor these flags all of the time. On the other hand, a UDP user datagram is a packet with clear-cut boundaries. A UDP packet, when combined with RTP/RCTP packets, can carry a single chunk of multimedia payload, and the size of the UDP datagram is large enough to hold any chunk of multimedia data as defined so far.

b. TCP is not an appropriate protocol for multicasting and multi-party communication because it is a connection-oriented protocol in which a connection involves only two parties. UDP, on the other hand, is a connection-less protocol. Multicasting and multi-party communication can be achieved using UDP. Using the same multicast address, a UDP sender can send multicast packets to many receivers; several UDP senders can use the same multicast addresses to create multi-party communication.

c. TCP delivers data bytes to the application program in order. This means that if a segment is missing, the out-of-order bytes need to wait. This is not appropriate for live or real-time interactive multimedia streaming, as discussed in the text. This problem does not exist in UDP. Each UDP packet is independent, and the chunk of data carried in a UDP datagram can be delivered to the application program as it arrives.

d. Since multimedia chunks can use different encoding methods, we need a field to define the type of payload. A TCP segment lacks this field. A UDP datagram can allow encapsulation of the RTP packet, which includes this field.

e. For a live or real-time multimedia streaming, there is a need for synchronization using timestamps that define the temporal relationship between each chunk and the other chunks. None of the three protocols, TCP, UDP, or SCTP, provide timestamping. UDP, however, allows the encapsulation of the RTP packets, which includes timestamping. Since an SCTP packet can carry many data chunks in a single packet, timestamping may not be needed if the stream is short.

f. Congestion control in multimedia communication is a big issue because of the volume of data exchanged. Although TCP has a congestion control mechanism, it is not suitable for multimedia communication, because it is based on segments, not chunks. Congestion control in UDP/RTP is more convenient for multimedia communication.