CSE 123: Computer Networks

Homework 1 - Solutions

Out: 10/4, Due: 10/11

Total points: 40

Solutions

1. **Two-dimensional parity**

   Given below is a series of 7 7-bit items of data, with an additional bit each and an extra byte to account for parity.

1	1	1	0	1	1	0	
1	1	0	1	0	1	0	
0	1	1	1	1	1	0	
0	1	1	0	1	0	0	
1	1	0	0	0	1	0	
0	0	1	0	1	0	1	
1	1	0	0	0	0	0	

   a. Fill in the parity bit for each blank, assuming even parity is followed. **(1 point for row parity bits and 1 point for column parity bits)**

      The below table shows the filled in parity bit (in bold) for each blank, assuming even parity is followed.

1	1	1	0	1	1	0	**1**
1	1	0	1	0	1	0	**0**
0	1	1	1	1	1	0	**1**
0	1	1	0	1	0	0	**1**
1	1	0	0	0	1	0	**1**
0	0	1	0	1	0	1	**1**
1	1	0	0	0	0	0	**0**
0	**0**	**0**	**0**	**0**	**0**	**1**	**1**

b. Will two-dimensional parity catch all 2-bit errors? **(1 point for answer and 1 point for explanation)**

Yes, two-dimensional parity catches all 2-bit errors. If the 2 bits that have been altered are not in the same column, then the column parity bits will be incorrect, and if they are not in the same row, the row parity bits will be incorrect. Two positions cannot simultaneously be in the same row and same column; hence the error can be detected.

c. Will two-dimensional parity catch all 3-bit errors? **(1 point for answer and 1 point for explanation)**

Yes, two-dimensional parity catches all 3-bit errors. Assume we use the above example and that we try covering up the error in either the row or column parity bit by flipping a third bit. This will hide one of the errors, but not both. Hence, all 3-bit errors can be caught.

d. Will this parity check catch all 4-bit errors? **(1 point for answer and 1 point for explanation)**

No, two-dimensional parity cannot catch all 4-bit errors. For example, if the first two bits of the first 2 rows are flipped, as shown in red below, the parity bits and the corresponding parity byte remain the same. Hence, this two-dimensional parity does not catch all 4-bit errors.

0	0	1	0	1	1	0	**1**
0	0	1	0	1	0	0	**0**
0	1	1	1	1	1	0	**1**
0	1	1	0	1	0	0	**1**
1	1	0	0	0	1	0	**1**
0	0	1	0	1	0	1	**1**
1	1	0	0	0	0	0	**0**
0	**0**	**0**	**1**	**0**	**0**	**1**	**0**

2. **Maximum Throughput**

Consider a 10Mbps link with a 20-ms round-trip time (RTT). Assume that the sender can send only one frame per RTT, and that the sender has a frame size of 5 KB. Calculate:-

a. The Bandwidth-Delay Product of the channel **(1 point)**

The Bandwidth-Delay product of a channel is defined as the product of the bandwidth of the channel and its associated round-trip time (RTT). In this case, the Bandwidth-Delay product will be

10 Mbps X 20 ms = 200 Kb

b. The maximum throughput the sender can achieve **(1 point for getting the numerator and denominator values right, 1 point for final answer)**

The maximum throughput of the channel is calculated by dividing the maximum amount of data that can be sent every RTT with the value (in ms) of the RTT

$$\frac{5*2^{10}*8\ b}{20\ ms} = 2.048\ Mbps$$

c. The fraction (or percentage) of the capacity of the link that is being used by the sender. **(1 point)**

The fraction of the capacity being used by the sender can be calculated as follows

$$\frac{2.048\ Mbps}{10\ Mbps} = 20.48\%$$

3. The HDLC Protocol

Given our understanding of the HDLC protocol, assume that the following bit stream arrives at the receiver at a particular instant of time:-

0111 1111 1001 1101 1011 0110 0111 1110 1111 0111 1111 0111 1111 0111 0110 1101 1011 1111 0100 0111 1000 0001 1110 1111 1101 1111 1110 0010 1111 0010 1101 0011 1111 1101 1101 1111 0110 0101 0011 1101 1111 1001

Calculate the total number of:-

a. Correctly received end of frames **(1 point for each correctly identified EOF, no negative penalty otherwise)**

0111 1111 1001 1101 1011 0110 0111 1110 1111 0111 1111 0111 1111 0111
0110 1101 1011 1111 0100 0111 1000 0001 1110 1111 1101 1111 1110 0010
1111 0010 1101 0011 1111 1101 1101 1111 0110 0101 0011 1101 1111 1001

The total number of correctly received end of frames is 4

a. Stuffed 0's **(1 point for each correctly identified stuffed 0, no negative penalty otherwise)**

0111 1111 1001 1101 1011 0110 0111 1110 1111 0111 1111 0111 1111 0111
0110 1101 1011 1111 0100 0111 1000 0001 1110 1111 1101 1111 1110 0010
1111 0010 1101 0011 1111 1101 1101 1111 0110 0101 0011 1101 1111 1001

The total number of stuffed 0s is 1

b. Received errors **(1 point for each correctly identified error, no negative penalty otherwise. Will accept an answer of 4 here too.)**

0111 1111 1001 1101 1011 0110 0111 1110 1111 **0111 1111** **0111 1111** 0111
0110 1101 1011 1111 0100 0111 1000 0001 1110 1111 110**1 1111 11**10 0010
1111 0010 1101 0**011 1111 1**101 1101 1111 0110 0101 0011 1101 1111 1001

The total number of received errors is 5

4. Error-checking using CRC

Assume we have the following bit stream as the sender's message - **1011 0011 0101 0110** - and we want it to encode it with the help of the CRC-8 polynomial.

a. What does the CRC-8 polynomial look like? Write it down using the standard polynomial notation. (Look up Table 2.3 on Page 102 of the text for your reference) **(1 point)**

The CRC-8 polynomial in represented in the standard polynomial notation as $x^8 + x^2 + x^1 + 1$

b. What is the sequence for the CRC-8 polynomial in binary form? **(1 point)**

The binary sequence for CRC-8 is 100000111

c. What does the transmitted bit-sequence look like? Highlight your steps in the calculation. **(1 point for the message being divided, 1 point for the remainder, 1 point for the final transmitted sequence)**

```
100000111 | 10110011010101100000000
            100000111
              110000110
```

```
                    100000111
                     100000011
                     100000111
                           100011000
                           100000111
                               111110000
                               100000111
                                111101110
                                100000111
                                 111010010
                                 100000111
          Remainder   ----->     11010101
```

Hence the transmitted bit sequence is:-

1011 0011 0101 0110 1101 0101

d. Assume the 7th bit in the transmitted sequence is flipped. What is the calculated remainder at the receiver? **(1 point for the message being divided, 1 point for the remainder)**

If the received message (with 7th bit flipped in red) is as follows:-

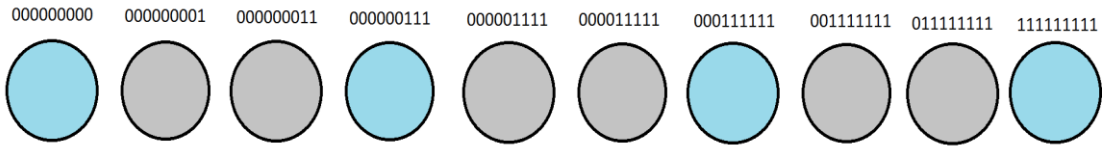
1011 000**0**1 0101 0110 1101 0101

```
   100000111 | 10110001010101011010101
               100000111
                 110010110
                 100000111
                  100100011
                  100000111
                     100100011
                     100000111
                        100100011
                        100000111
                           100100010
                           100000111
                              100101101
                              100000111
     Non-zero remainder   -->    101010
```

The remainder is non-zero (101010), hence the error has been correctly detected

5. Hamming codes

Assume fixed-length bit-strings of length 9 where only some bit-string sequences are allowable in the encoding scheme. Assume that the bit-strings in blue are the allowed codewords and those in gray are the ones that aren't. The following diagram elucidates the assumption.



Now, calculate the following:-

a. What is the hamming distance of this encoding scheme? **(1 point)**

The hamming distance of the above scheme is 3, since the minimum number of bit flips between any two codewords is 3.

b. Is the encoding scheme efficient? Why or why not? **(1 point for the answer and 1 point for the explanation)**

No, the above encoding scheme isn't efficient because some codewords are further apart than three from each other. For example in this instance, the distance between the first and last codeword is 9.

c. How many bit flips can be detected? **(1 point)**

The value of d is related to the hamming distance as follows

$$\text{Hamming Distance} = 2d + 1$$

$$\text{Hence, } d = 1$$

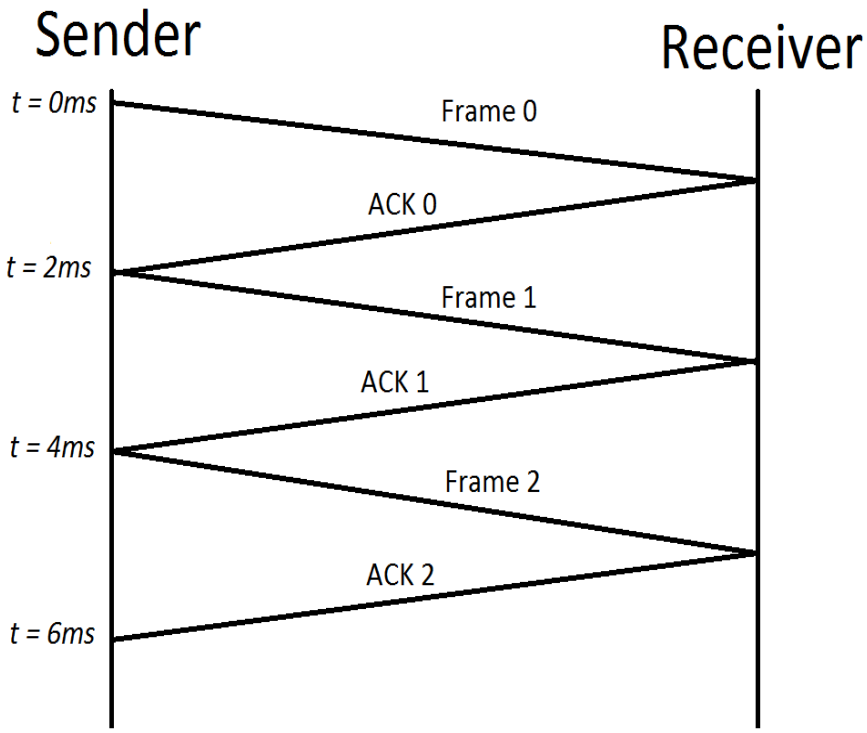
The number of bit flips that can be detected is given by 2d, which is 2 bits in this case.

d. How many bit flips can be corrected? **(1 point)**

The number of bit flips that can be corrected is given by d, which is 1 bit in this case

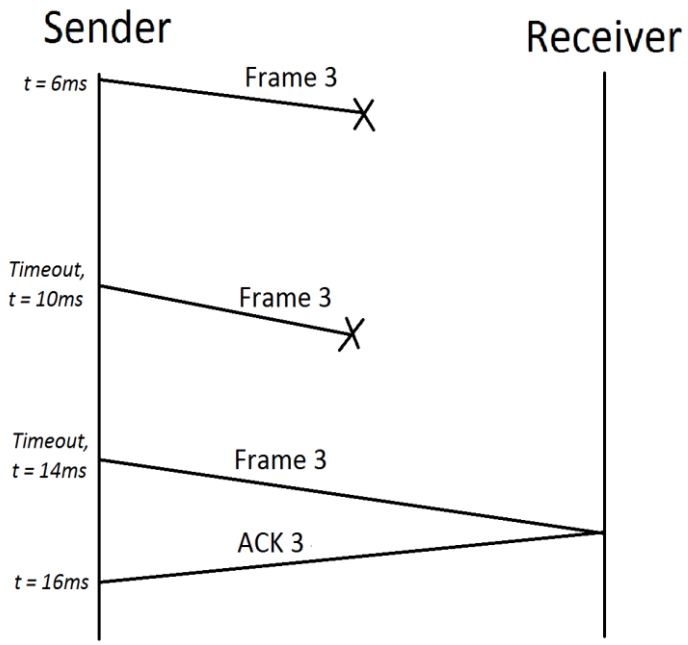
6. The Automatic Repeat Request (ARQ) Protocol

Assume that two computers are communicating using the stop-and-wait ARQ protocol. Assume that the RTT for the communication channel is 2ms, and that the timeout is twice the RTT. Also assume that both parties use sequence numbers on data and ACK frames. If the sender has to send 8 frames to the receiver, draw the sequence of steps involved if the first 3 frames are sent without incident (as shown in the figure) and the following takes place:-



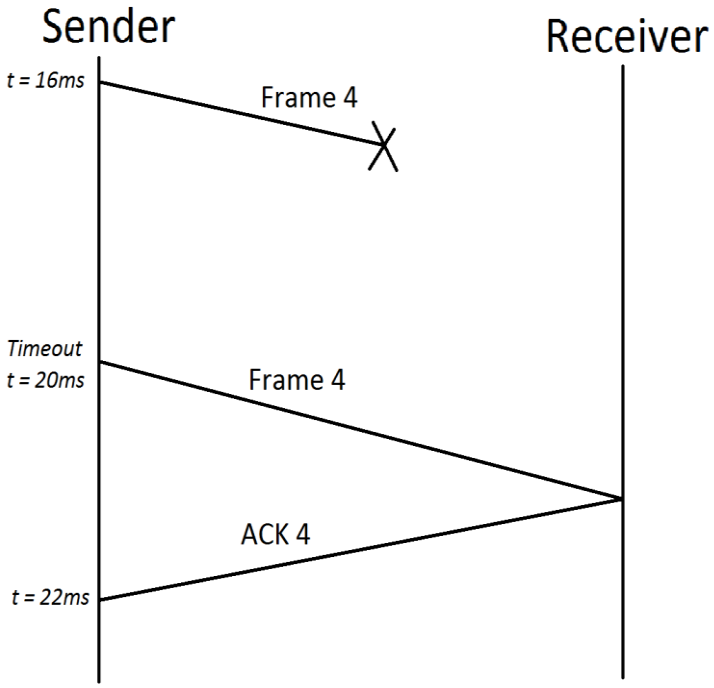
a. The 4th frame is lost during the first and second transmission, but is acknowledged without incident after the third transmission. **(1 point)**

The following diagram represents the above scenario



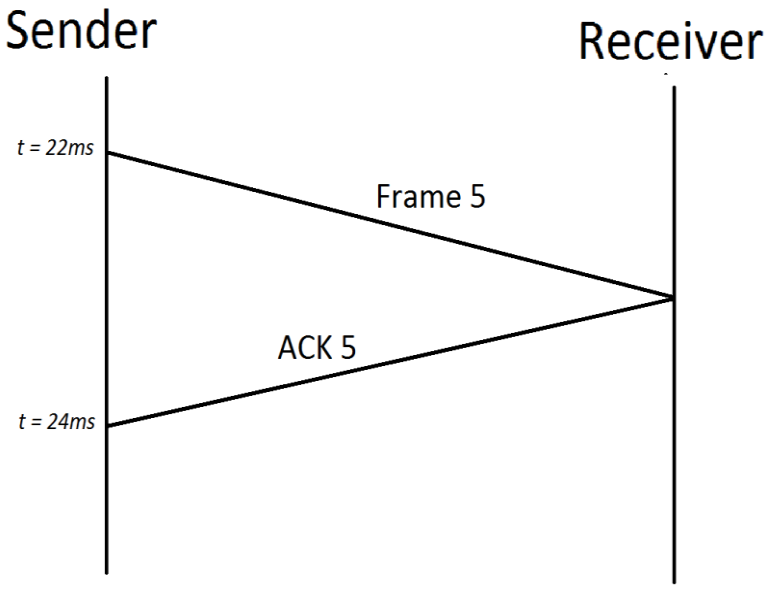
b. The 5th frame is lost during the first transmission, but acknowledged without incident after the second transmission. **(1 point)**

The following diagram represents the above scenario



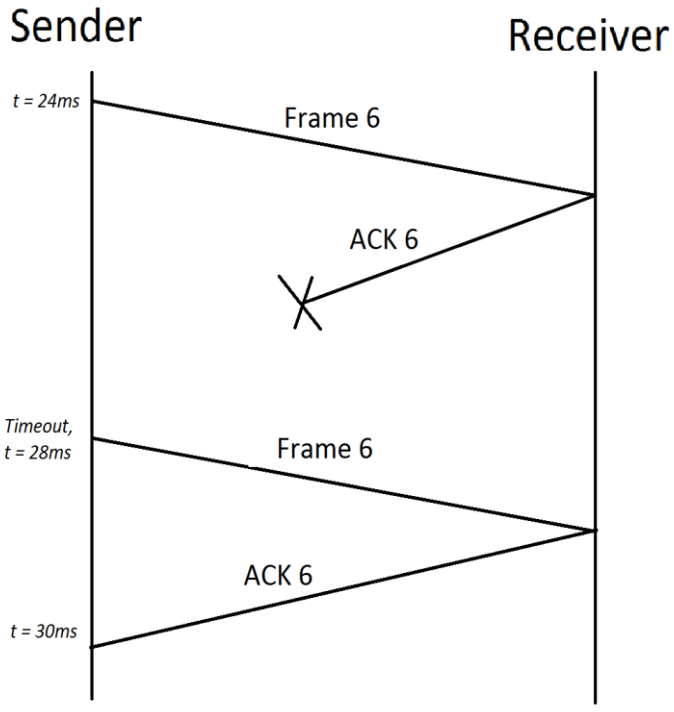
c. The 6th frame is sent and acknowledged without incident. **(1 point)**

The following diagram represents the above scenario



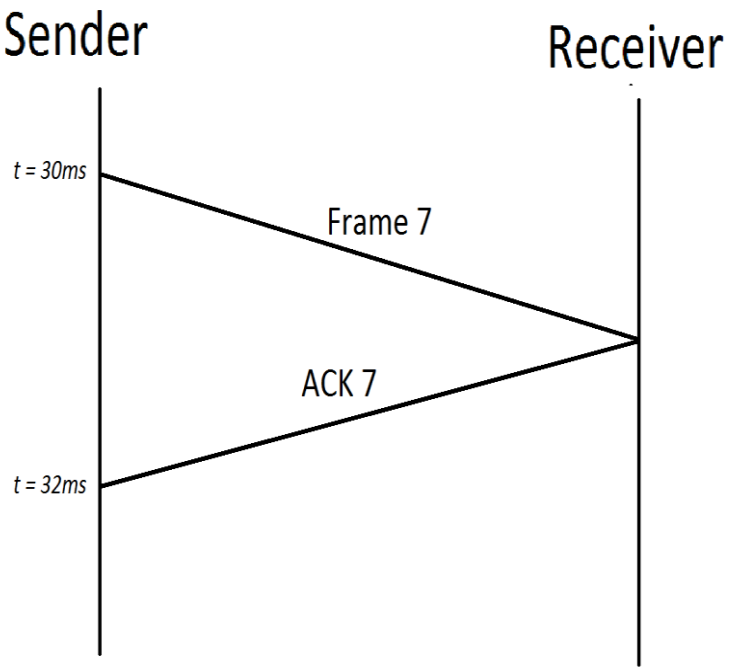
d. The 7th frame is sent without incident but the acknowledgment is lost the first time; sent without incident during the second transmission. **(1 point)**

The following diagram represents the above scenario



e. The 8th frame is sent and acknowledged without incident. **(1 point)**

The following diagram represents the above scenario



Also, how much time (in *ms*) did it take for the above 8 frames to be transferred and acknowledged successfully? **(1 point)**

From the above diagrams, it can be seen that the total time it took for the frames to be transferred acknowledged successfully is 32 ms.