

**The University of Queensland**  
**School of Information Technology and Electrical Engineering**  
**Semester Two, 2011**

**COMS3200 – Tutorial 5 - Solutions**

**Question 1**

*State the roles of the Data Link layer. Characterise the control information which is added by the Data Link layer to a packet passed from the Network Layer.*

The main roles of the Data Link Layer protocols are: packet framing, error detection, reliability (some protocols) and flow control. A DLL protocol has to add at least the following to the packet: framing, source and destination address (DLL address), sequence number (for reliable protocols) and redundant bits used for error detection (checksum).

**Question 2**

*The following data fragment occurs in the middle of a data stream for which the character-stuffing algorithm is used: A, B, ESC, C, ESC, FLAG, FLAG, D. Assume that FLAG is used for framing and ESC is used to show that a particular FLAG is part of data not a framing byte. What is the output after stuffing?*

After stuffing, we get A B ESC ESC C ESC ESC ESC FLAG ESC FLAG D.

**Question 3**

*If the bit string 01111011110111110 is subjected to bit stuffing, what is the output string?*

In bit stuffing the bit sequence 0111110 is a special "framing marker" and therefore it can't occur in the frame data. To enable the transmission of arbitrary sequences of bits as frame data, the sequence 0111110 must be changed before it is transmitted. All that is done is whenever five consecutive ones are seen, a zero bit is stuffed in after them. This makes sure that six consecutive ones cannot occur.

Bits Sent:           01111011110111110

Bits after Stuffing: 0111101111001111010

**Question 4**

*Can you think of any circumstances under which error correction codes might be preferable to error detection codes?*

- In military situations in order to not disclose the transmitter location.
- In space systems as propagation delay is too long for retransmissions.
- In mobile telephone networks as there is no time for retransmissions.

**Question 5**

*One way of detecting errors is to transmit data as a block of  $n$  rows of  $k$  bits per row and adding parity bits to each row and each column. Will this scheme detect all single errors? Double errors? Tripple errors?*

Parity bits will always detect one bit in error. Any two bits in error will always be detected since each bit will be involved in at least one parity check which is independent of the other bit. Three bits in error will not always be detected if one error bit is in the same row as another error bit and the same column as the remaining error bit. That is, if the bits form a "L"-shape. However, even in this case it can be detected that there is an error in the block and that the block has to be retransmitted.

### Question 6

A cyclic redundancy code has 8 message bits and uses the generator polynomial:

$$x^5 + x^4 + x + 1$$

a) Deduce the redundant checkbits which would be appended to the following message at the sender side: 11100011

b) Verify your answer to a) showing that the receiving side's check of the message extended with the redundant bits succeeds.

a) The generator polynomial  $G(x) = x^5 + x^4 + x + 1 = 110011$  is 6 bits long, therefore we add  $6-1=5$  bits to the end of the message polynomial,  $M(x)$ , to hold the remainder.

$$\begin{array}{r}
 10110110 \\
 \hline
 1110001100000 : 110011 \\
 \hline
 110011 \\
 010111100000 \\
 \hline
 000000 \\
 10111100000 \\
 \hline
 110011 \\
 1110000000 \\
 \hline
 110011 \\
 010110000 \\
 \hline
 000000 \\
 10110000 \\
 \hline
 110011 \\
 1111100 \\
 \hline
 110011 \\
 011010 \\
 \hline
 000000 \\
 11010
 \end{array}$$

The redundant checkbits which would be appended to the message  $M(x)$  are the remainder bits 11010. Therefore, the message that would be sent is 1110001111010.

b) The received message,  $C(x)$ , is valid if the remainder of the division of  $C(x)/G(x)$  is 0.

$$\begin{array}{r}
 10110110 \\
 \hline
 1110001111010 : 110011 \\
 \hline
 110011 \\
 01011111010 \\
 \hline
 000000 \\
 1011111010 \\
 \hline
 110011 \\
 1110011010 \\
 \hline
 110011 \\
 010101010 \\
 \hline
 000000 \\
 10101010 \\
 \hline
 110011 \\
 1100110 \\
 \hline
 110011 \\
 000000 \\
 \hline
 000000 \\
 00000
 \end{array}$$

The remainder is 0, so the checkbits deduced in (a) are correct.

### Question 7

A 3000 km long T1 trunk is used to transmit 64-byte frames using a Go-back-n protocol. If the propagation speed is 6  $\mu\text{sec/km}$ , how many bits should the sequence number be?

T1 data rate is 1.544 Mbps. The go-back-n protocol is a sliding window protocol. To operate efficiently, the protocol must allow the transmitter to keep transmitting until the first acknowledgment arrives. The sequence numbers on the packet need to be big enough to uniquely identify all packets which could be transmitted during this round-trip time.

If we ignore the time taken to transmit the bits in the frame and receive the bits in the acknowledgment, then:

- The time taken for the frame to reach the receiver is 18 msec
- The time taken for the acknowledgment to reach the sender is 18 msec
- The round-trip time is 36 msec

It takes  $\frac{64 \times 8}{1.544 \times 10^6}$  seconds to transmit one frame, therefore:

Number of frames:

$$\frac{36 \times 10^{-3} \times 1.544 \times 10^6}{64 \times 8} = 108$$

108 frames must be uniquely identified by sequence numbers. This requires 7 bits.

### Question 8

Frames of 1000 bits are sent over a 1 Mbps satellite channel. Assume that the propagation delay over a satellite channel is 270 msec. Acknowledgements are always piggybacked onto data frames. The headers are very short. Three bit sequence numbers are used. What is the maximum achievable channel utilisation for a) Stop-and-wait, b) Go-back-n, c) Selective repeat?

Since the headers are very short, we can ignore them for this question. Each frame takes 1 msec to transmit (1000 bits at 1Mbps.)

This channel is a satellite channel and the propagation delay is significant (about  $270+270 = 540$  msec)

(a) Stop-and-wait

This protocol transmits one frame and then waits for a reply before transmitting a new frame. The total time to send one frame is 1 msec, but the total propagation time is the transmission time for the frame and reply frame plus the propagation time:  $1+540+1 = 542$ .

Therefore the efficiency is:  $1 \text{ msec} / 542 \text{ msec} = 0.18\%$

b) Go-back-n

This is a sliding window protocol, and with 3 bits of sequence numbering, we can have 7 outstanding packets, that is 7 packets can be in transit on the satellite channel at a time.

Therefore the efficiency is:  $7 \text{ msec} / 542 \text{ msec} = 1.29\%$

c) Selective-repeat

This is a sliding window protocol, but since it allows packets to be accepted out-of-order only half of the available sequence numbered packets may be sent at any given time, so with 3 bits of

sequence numbering, we can have 4 outstanding packets, that is 4 packets can be in transit on the satellite channel at a time.

Therefore the efficiency is:  $4 \text{ msec} / 542 \text{ msec} = 0.74\%$

Consider these protocols. The stop-and-wait performs poorly since it must incur the propagation delay of the satellite channel in its acknowledgment of each packet. The go-back-n protocol performs much better since it can have more than one packet on the channel before it incurs the penalty of the propagation delay. Clearly, however, 3 bits of sequence numbering is not enough and a larger number of bits (or perhaps an increase in size of the packet) would improve the efficiency of the protocol. The selective-repeat protocol, while being more robust in the face of packet corruption or loss, achieves it by halving the effective number of outstanding packets allowed on the channel.

### Question 9

*In a selective repeat protocol the maximum sequence number is  $2n-1$ . This condition is desirable to make efficient use of header bits. Is it essential? Does the protocol work correctly if the maximum sequence number is 4, for example?*

The protocol requires the number of sequence numbers to be odd to function correctly. Remember that if we perform a modulus  $m$  on a number  $n$  then both  $0$  and  $m$  ( $\text{mod } m$ ) evaluate to zero, also recall that a buffer is kept for each packet by the sender until that packet is acknowledged.

Now consider the conditions for the selection of buffers being half the allowable number of distinct sequence numbers. The original reason for this was to make sure that the sequence numbers for the packets do not overlap when an acknowledgment for any set of packets arrives. However, how are we selecting which buffer to place the data for a packet with a given sequence number? We use a modulus on the number of buffers. To make the protocol work, given  $m$  sequence numbers and  $n$  buffers, we require that any sequence of  $m$  consecutive sequence numbers  $\text{mod } n$  will produce a unique buffer number.

For the example given, there are 5 possible sequence numbers and thus 2 buffers:

<u>SeqNr</u>	<u>SeqNr mod 2</u>
0,1	0,1
1,2	1,0
2,3	0,1
3,4	1,0
4,0	0,0

The last case is incorrect, and if you generalize the argument, the same thing will happen for any implementation where the number of sequence numbers is not odd.