UCSD CSE CS 123 Computer Networks

Sample Midterm Exam Solutions

Directions: Write your name on the exam. Write something for every question. You will get some points if you attempt a solution but nothing for a blank sheet of paper. Write something down, even wild guesses. Problems take long to read but can be answered concisely.

Question	Maximum	Score
1	30	
2	20	
3	15	
4	20	
5	15	

1, Overview, 30 points: Give the most important reason you can think of for each of the following. Each answer should be 1 line. Only one reason please!

- **Layering and Interfaces:** Many of today's routers implement firewalls. Thus the routers at the boundaries of DEC networks will allow email from the outside world but not other applications such as Telnet. Why is Peter Protocol unhappy about firewalls?

  `Because looking at email headers in routers is a layer violation`

- **Clock Recovery:** Why eye patterns are often used by real communication engineers in a lab?

  `Because they provide a quick visual check of the quality of the link, in particular the amount of intersymbol interference, and the sampling margin.`

- **Shannon Limit:** Hugh Hopeful is working with a transmitter that can only send at two amplitude levels. Hugh assumes that he can never send faster than the Nyquist rate. Why is Hugh wrong?

  `Because there are other ways to send more than 1 bit per symbol such as using multiple phases or frequencies.`

- **Media:** Why lasers and single-mode fibre, though more expensive, are sometimes used to send light over a fibre link?

  `Because they avoid chromatic and modal dispersion which reduces ISI which in turn allows sending at higher bit rates.`

- **Coding, Preambles and Transitions:** Why 11111111 is a reasonable preamble for AMI coding but not for Ethernet.

  Because in AMI the 1 levels alternate at clock boundaries allowing the receiver to determine bit boundaries, but in Manchester 11111111 and 00000000 are identical except for a 90 degree phase shift.

- **CRCs and Hamming Distance:** Why frames, after adding CRC-32, have a Hamming Distance of at least 4 for reasonable frame sizes. (this is easier than it may appear).

  Because we argued that CRC can catch all odd errors (as it divides $x+1$) and it can catch even errors (by making x^k+1 not divisible by CRC) and so it catches up to 3 random errors; hence its Hamming Distance must be at least one larger or 4.

- **Data Link:** Why undetected error rates on Data Links should be several orders of magnitude lower than the lost frame rate.

  Because undetected errors at each hop can lead to router errors and (worse) even undetected errors at the receiver if there is no end-to-end checksum.

- **Data Link:** Why it is sometimes not worth doing error recovery at the Data Link.

  Because it is expensive to buffer packets for retransmission at routers and to send acks on each hop (which costs extra bandwidth).

- **LANs and multiplexing:** Why most LANs do statistical multiplexing instead of strict multiplexing.

  Because data traffic is bursty and the number of active users is tyoically smaller than the total number of users; thus stat muxing divides bandwidth among active users, giving each active user typically a larger share.

- **LANs and wide area networks:** Why its reasonable to have high bandwidths for Local Area Networks and lower bandwidths for wide area networks.

  Because traffic exhibits locality: there is more traffic local to a LAn than outside a LAN.

2. Data Link Protocols on Half-Duplex Links, 20 points: So far in all our Data Link protocols we have assumed the links to be full-duplex so data and acks can be sent at the same time. In this problem, we examine the problems of running Data Links over half-duplex links.

Consider a satellite link where the one-way propagation delay between sender and receiver is $P = 250$ msec. Assume that the link is half-duplex; data can flow only in one direction at a time. The sender and receiver share the link using Time Division Multiplexing: the sender is allowed to send for T_s time and then the receiver sends for T_r time, and so on. We wish to implement a reliable data link protocol over this link. Assume that sender sends data in frames of size F bits and the receiver sends acks of size A bits. The speed of the link is B bits per second.

We start by implementing an alternating bit protocol between sender and receiver. Let $T_s = P + F/B$ (i.e., sender gets to send for long enough to send a frame and have it be received by the receiver) and $T_r = P + A/B$ (i.e., receiver gets to send for long enough to send an ack and have it be received by the sender)

- What is a natural value of the sender timeout for retransmissions?

  No real need for a timeout at sender; at each opportunity to transmit the sender retransmits if it hasn't got an ack for all outstanding frames. In some sense, the sender timeout is $T_s + T_r$.

- What is the efficiency of this system in terms of link usage (ignore link errors)?

  Only send useful data frames for time F/B in a total time of $F/B+A/B+2P$. Thus efficiency is equal to $\frac{F/B}{F/B+A/B+2P}$

- To improve the efficiency, we change the implementation to a sliding window system. If the sender window size is limited to X frames, how would you set T_s and T_r to get maximum link usage efficiency. What is the resulting link efficiency (ignore link errors)?

  T_s should become $X \cdot F/B+P$ and T_r remains unchanged. Thus efficiency becomes $\frac{F \cdot X/B}{F \cdot X/B+A/B+2P}$ which becomes arbitrarily close to 1 as X increases.

- After using the system we notice that link errors are frequent and so we change to a selective reject protocol. How would you modify the information returned in an ack to improve the efficiency of the selective reject system? Explain how the sender would use this information.

  Acks should carry the sequence numbers of all frames correctly received in previous sender transmission time slot. Sender should retransmit the incorrectly received frames in next frame plus any other new frames that can fit. There is no need for a retransmit timer.

3. CRCs and Error Polynomials, 15 points: Consider the CRC generator polnomial $x + 1$ used to generate CRC checksums just as we studied in class.

- How many bits do we have in the checksum? 1 bit
- Consider a message 101. What is the checksum value? Show your division 0
- Consider a message 111. What is the checksum value? Show your division? 1
- Prove that this CRC polynomial catches all odd bit errors. *same argument as in notes*
- You have seen this CRC called by a different name. What is it? Parity

4. HDLC Framing, 15 points: Hugh Hopeful has invented a new flag for HDLC (Hopeful Data Link Control) protocol. He uses the flag 01010101.

- In order to prevent data bits from being confused with flags, the sender stuffs a one after receiving a sequence 010101. Does this work? Justify your answer with a short proof or counterexample.

  Does not work. Suppose data is 1101 and the stuffed frame is thus 01010101 1101 01010101. After the reciver receives the data bits 11, it can think the 01 of the data plus the first 6 bits of the flag forms a flag, thus outputting 11 as the data, a mistake.

- To reduce the overhead, Hugh tries to stuff a zero after receiving 0101010. Will this work? Justify your answer with a short proof or counterexample.

  Does not work. Same example as above.