Jayant Mehra
504 992 428

UCLA CS CS 118

Computer Networks

Midterm Exam

Directions: Write your name on the exam and on every page you submit. Write something for every question. Students who do not write something for everything lose out over students who write down wild guesses. 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	40	28
2	15	15
3	15	9
4	15	10
5	15	14
Total		76

28/40

24

1, Overview, 40 points: Give the most important reason or answer you can think of for each of the following. Each answer should be 1 or 2 lines. Only one reason or answer please! All questions are 4 points apiece.

- **1) Layering and DiffServ:** In the DiffServ proposal, in order to provide different qualities of service to different packets, routers at the edge of a customer premise look at the traffic type of the packet in the TCP headers and set some bits in the routing header (called the TOS bits). Core routers then use the TOS bits in the routing header to give priority to some packets over others. a) The edge routers are committing a layer violation but the core routers are not b) The edge routers are not committing a layer violation but the core routers are c) They both committing a layer violation d) Neither is committing a layer violation

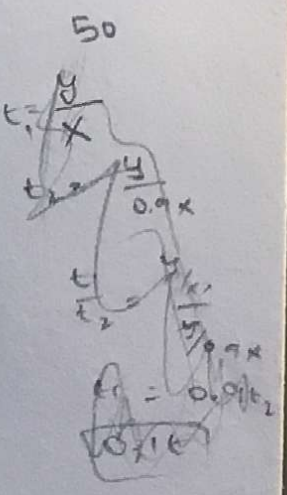
4

- **2) Fourier Analysis:** The basis for Fourier Analysis is that wires are "nice" to sine waves. This means that when a sine wave passes through a wire one basic property of the sine wave stays the same. The property is: a) frequency b) phase c) amplitude d) time it takes to reach a peak.

4

- **3) Media Impairments:** A fiber link suffiers from both chromatic and modal dispersion. For modal dispersion, for the fastest wavelength of light in the laser, the signal that goes down the middle takes 200 nsec while the signal that bounces around takes 250 nsec. For chromatic dispersion, the slowest wavelength of light is 10 % slower than the fastest wavelength. What is the fastest spacing between bits to avoid ISI? a) One every 25 nsec b) Once every 50 nsec c) Once every 75 nsec d) Once every 100 nsec

2



- **4) Latency and throughput:** Does reducing latency always increase throughput? a) Yes because reducing the time for a job always increases the number of jobs per second b) No because latency and throughput are completely unrelated c) Yes because throughput is the reciprocal of latency d) No because one always use a shorter wire of the same transmission speed to reduce propagation delay

4

- **5) Media:** Many wireless keyboards use infrared to connect from the keyboard to the com- puter. What would go wrong (or right) if the keyboard used radio instead of infrared to communicate with the computer? a) The radio is more expensive than infrared b) The radio would be absorbed by obstacles c) Keyboard transmissions would interfere between nearby offices d) The radio would allow the keyboard to be very far from the computer

2

- **6) Preambles and Transitions:** Is 111111110 a good preamble for AMI a) No because there are not enough transitions to synch clock recovery b) No because it is hard to tell where the data starts c) Yes because it guarantees at least one transition d) Yes, because it guarantees several transitions and you can tell where the data starts

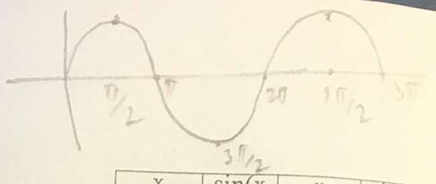
4

- **7) End-to-end argument:** To make sure your data does not get corrupted today we have two mechanisms: each hop does a checksum like CRC 32 and TCP also does an end-to-end checksum. If TCP is doing an end-to-end checksum why is each router doing a CRC as well? a) Because we want to ensure the destination address is correct before forwarding packets to the next hop b) Because corruption on links is so common, it is faster to detect it hop by hop c) Because each router can retransmit packets hop by hop d) Because routers can corrupt packets and the CRC can catch this corruption.

- **8) Data Link Sublayering:** Why framing should be performed before error detection at a receiver, rather than vice–versa. a) So that errors in the framing characters can be caught by the CRC b) So that we can know where the destination address and other headers are in the bit stream c) So we can know which protocol we are using before doing error detection d) So the receiver can know where the checksum is

- **9) Data Link restarting:** Consider the following scheme where a sender restarts after a crash and sending 10 frames. When the sender comes up and starts sending new frames with sequence number 0, the receiver sends an Ack back for frame 10 (which is what it expects). Suppose the sender "jumps" to 10 and starts sending. Will this jumping method guarantee that all frames sent after the sender crash are received for other cases as well? a) No because this means the sender has lost frames 0 through 9 b) No, because if the sender crashes after sending one frame, it could be fooled when it comes up and sends a new frame after a crash c) Yes because we are only guaranteeing frames sent after the last crash d) Yes, because the receiver will keep sending acknowledgements unconditionally

- **10) Ethernet collisions:** Ethernet guarantees that a) All stations detect collisions at the same time b) Only senders detect collisions c) that e) If any frame does collide at some point in the wire, then all stations will eventually learn that the collision occurred d) that the CRC will reliably detect collisions

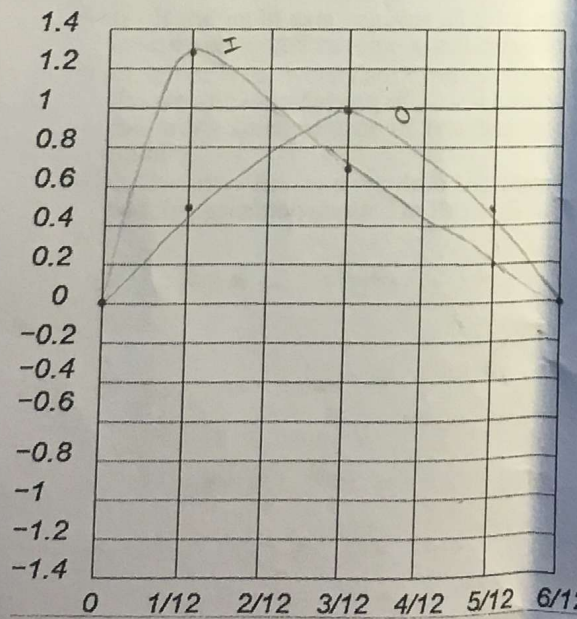


x	sin(x	x	sin(x	x	sin(x)	x	sin(x)
π*0/6	0.000	π *3/6	1.000	π *6/6	0.000	π *9/6	-1.000
π *1/6	0.500	π *4/6	0.866	π *7/6	-0.500	π *10/6	-0.866
π *2/6	0.866	π *5/6	0.500	π *8/6	-0.866	π *11/6	-0.500

2. Fourier Analysis: An input signal' s Fourier analysis has already been done for you. The signal is $I = \sin(2\pi t) + 1/2 \sin(4\pi t) + 1/3 \sin(6\pi t)$. We want to show that just three sine waves comes close to approximating a "nice" shape. To avoid the use of calculators, we have a table above which has the value of sine waves at various angles (in fractions of π).

- (7 points) Using the sine table above , first compute and plot the input sine wave at the 5 moments t = 0, 1/12, 1/4, 5/12, 1/2. Please draw the wave approximately on the ruled figure (which has amplitude spacing in units 0.2, and time spaced in units of 1/12 of a second.

- (8 points) Assume the input is passed through a channel whose frequency response is such that all waves of frequency 1 Hz or less are passed through unscaled (and with no phase shift) while all waves of frequency greater than 1 Hz are scaled down to 0. Compute the amplitude of the output at the same 5 times and draw it on the same figure as the input. What is the final shape? (if you have no time, say what the output wave looks like).



Input(0)= 0
Input(1/12)= 1.266
Input(3/12)= 0.667
Input(5/12)= 0.4
Input(6/12)= 0

Output(0)= 0
Output(1/12)= 0.5
Output(3/12)= 1
Output(5/12)= 0.5
Output(6/12)= 0

3. Framing, 20 points: Bit stuffing can be expensive to implement if framing is done in software because it can cause the stuffed data not to be aligned on 8 bit boundaries. Thus many software framing techniques used on dialup lines do byte stuffing. Byte stuffing will guarantee that the stuffed data length is a multiple of 8. But Byte stuffing has a worst case overhead of 50% as you have seen in the slides. Alyssa P. Hacker has come up with what she calls the most efficient byte stuffing technique ever, much more efficient than even HLDC bit stuffing or PPP

- Suppose the framing character is the zero byte (which we denote by O). We want to code the data to remove all occurences of 0. Alyssa's idea is as follows. We divide the data into blocks of up to 254 bytes such that: EITHER each block is less than 254 bytes and ends with a 0
- OR the block is 254 bytes in length and does not contain a 0 byte.
- We then encode each block by placing the length of the block at the start, followed by the bytes in the block, but without the trailing 0 byte, if any. Notice that we can get rid of the zero byte because the length of each block tells us where to place the zero byte during decoding. The exception is if the length is 254 in which case the block DOES NOT contain a trailing 0 and we use a length field of 255 to encode this case. (We cannot use 254 because that is reserved for a block of 253 bytes followed by a trailing 0). This last exception allows us to encode long sequences that do not contain a 0 byte. As an example if the input is XY00L0, where X,Y, L are arbitrary bytes, the first block is XY0, the second block is 0, the third block is L0. The first block encoded becomes 3 X Y, the second block encoded becomes 1, the third block encoded becomes 2 L.
- a) The problem does not specify how one encodes the data if the last block is less than 254 bytes in length and does not end in a 0. Obviously, one could try to pad the last block. What is the minimal amount of padding necessary to make sure we can use Alyssa's rules and yet decode correctly. (2 points)
- b.) If the input data contains all 0's, what is the length of the coded data? ~~In general, how much longer can the coded data be than the original input data?~~ (3 points)
- c). What is the worst case overhead for a very large packet of length L >> 254? What is the worst case overhead of your scheme for a very small packet L << 254? Overhead is the worst case number of non-data bytes divided by the number of data bytes. (5 points)
- d. Describe the receiver decoding algorithm briefly in words incorporating whatever padding scheme you used in Part a. (5 points)

(a) 254 - length , if L = 255 −0.5

(b) If the input is 0000... all the encoded blocks become 1 1 1 .... Thus, the length of each coded block = 1. Thus, the length is the same. ✓ 2.5

(c) The worst case could be when there are no 0s. L >> 254

something like → The framing 0 byte.

overhead = $\frac{1 + (\frac{L}{254})}{L}$ → $\frac{L}{254}$ is the # of blocks. Say ~~each~~ no block does not has 0s. Thus, we add 255 to each block which is 1 extra byte for each block within the pack.

2.5 ✓

L << 254

overhead = $\frac{1 + \text{padding} + 1}{L}$ (framing 0 byte; extra bytes; 1 → 255)

1.5 ✓

$= \frac{2 + \text{padding}}{L}$ (from part a) Padding = 1

8 64 32 16 8 4 2 0
1 1 1 1 1 1 1 1

2 0
1 1

(3)

(d) For each block the first two bytes indicate the length of the block. The decoder reads those bytes say it L

(d) Each frame starts with a 0 byte. while processing the blocks within the frame, we read the first one two bytes to get the length of the block (say L). If L < 255, we add 0 at pos L & start processing the next block. If L = 255, we consume those bytes & then just start processing the next block.

2

To handle padding, when you read 255 but then read a bunch of 0s you ignore them until you hit the first non-zero byte.

- **4, CRCs, 20 points:** Consider a CRC generator $x^2 + 1$ and a message 110.
- (3 points) a) First compute the message together with the CRC. Show your working so we can give you partial credit.
- (1 point) b) Suppose we wish to consider all error patterns in which the 5th bit is corrupted. What term must all such error polynomials start with?
- (5 points) c) We wish to find all errors such that Bit 5 is corrupted but the error is not detected by the CRC. As in the HWs, the key is to use multiplication. Write down all polynomials the generator must be multiplied with to ensure that the 5th bit is corrupted and yet the result is undetectable. Note that unlike the case of a burst in your HW, we do not need the last term in the error polynomial to be 1 since we only want Bit 5 to be corrupted.
- (5 points) d) Multiply each of these polynomials by the generator to find all possible undetectable error patterns with the 5th bit corrupted. Remember that when you multiply you use mod 2 addition unlike ordinary algebra. For example, $x^2 + 1$ times $x^2 + 1$ is $x^4 + 2x^2 + 1$ in ordinary algebra but is $x^4 + 1$ in Mod 2 arithmetic because 2 mod 2 = 0.
- (1 point) e) How many of these undetectable error patterns that corrupt Bit 5 have a total of 3 bit errors?

(a) $G = x^2+1 = 101$ -3 X

$r = 3$

$2^{r-1} \cdot M = 11000$

(b) x^4 ✓

(c) x^2, x^2+x, x^2+x+1, x^2+1 ✓

(d) $x^2(x^2+1) = x^4+x^2$

$(x^2+x)(x^2+1) = x^4 + x^2 + x^3 + x = x^4+x^3+x^2+x$

$(x^2+x+1)(x^2+1) = x^4 + x^2 + x^3 + x + x^2 + 1$ -1

$= x^4 + x^3 + 1$

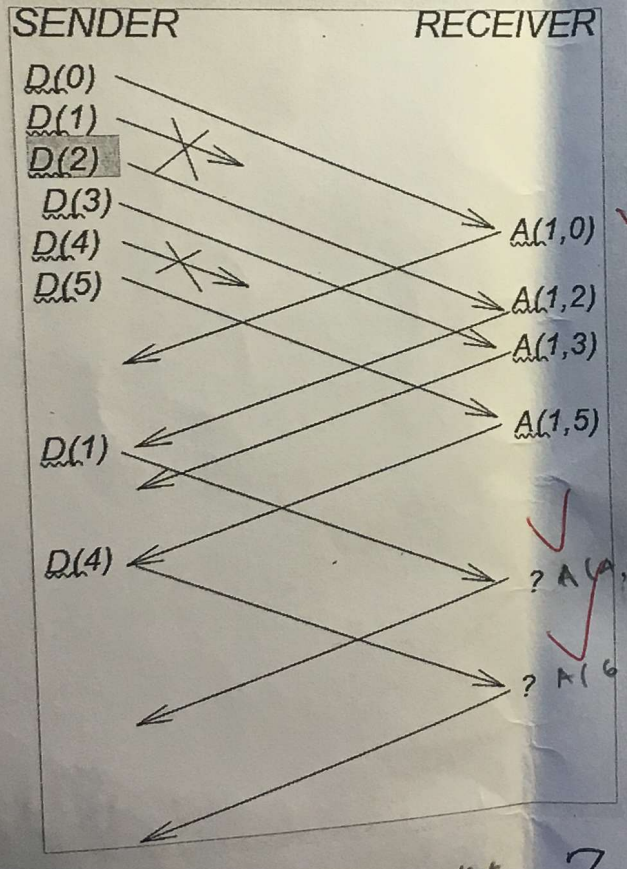
$(x^2+1)(x^2+1) = x^4 + x^2 + x^2 + 1 = x^4 + 1$

(e) x^4+x^3+1 has a 3 bit error

X -1

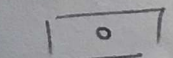
5. SMART Retransmission Strategy, 20 points: The SMART retransmission strategy is a retransmission strategy that tries to do as well as selective reject without the overhead of adding a bitmap (or a list of sequence numbers). The idea is that the receiver sends the same cumulative ack as before but also sends one extra number

- the number of the reeived frame that caused the ack to be sent. To see how this works, imagine the sender sends frames (see figure below) with sequence numbers 0, 1, 2, 3, 4, 5 but frame 1 and 4 get lost. On receiving 0, the receiver will send back Ack (1,0); on receiving 2, the receiver will send back Ack (1, 2); on receiving 3, the receiver sends back (1, 3); and on receiving 5, the receiver sends back (1, 5).
- When the sender receives Ack(1,2) why does the sender know enough to retransmit D(1)? (2 points)
- When sender receivers Ack (1,3) why does the sender know enough not to retransmit D(2)? What information does the sender need to maintain to not take any action at this point? (5 points)
- Write down on the picture (in place of the question number) the acks that are sent after receiver gets D(1) and D(4). (4 points)
- When the ack to the retransmitted D(1) arrives, how does the sender know not to retransmit D(4)? (4 points)
- Why is this scheme better than the fast retransmit scheme (done by TCP) where 3 duplicate acks trigger a retransmission. (4 points)

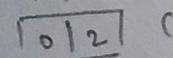


(a) The sender was expecting (3,2) but receives a (1,2) which indicates that frame 1 hasn't been rec. The reason being R won't incremented by 2 if rec the Ack for 2 - not for 1.

(b) At the pt. it rec. Ack(1,3), it has already retransmitted D(1). Moreover it has already rec. Acks for 0 & 2.
The sender can maintain a list of all acks ~~rec~~ it receives. That is when it ~~what~~ rec'ins A(1,0)

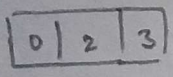


at A(1,2)



(but transmits 1)

at A(1,3)



(but does not transmit

The sender can also keep track of what it sent last. Therefore, (0,2) + (1) would mean that at Ack(1,3) nothing needs to be retransmitted.

(c) The sender keeps track of the last frame sent when it rec a ack for 1, it adds it to the list. It sees that (0,1,2,3,5) + (4) means that (acks rec) (last sent) it does not need to retransmit (4) again.

7.

d) This is better than the [illegible] ack. transmission scheme because latency is lower since we don't have to wait for [illegible] ack. Eg A(1,0) triggers the retransmission of D[illegible].

-1