Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.02 Spring 2009

Quiz III

There are 24 questions and 10 pages in this quiz booklet. Answer each question according to the instructions given. You have **120 minutes** to answer the questions.

If you find a question ambiguous, be sure to write down any assumptions you make. **Please be neat and legible.** If we can't understand your answer, we can't give you credit!

Use the empty sides of this booklet if you need scratch space. You may also use them for answers, although you shouldn't need to. *If you do use the blank sides for answers, make sure to clearly say so!*

Before you start, please write your name CLEARLY in the space below.

One two-sided "crib sheet" and calculator allowed. No other notes, books, computers, cell phones, PDAs, information appliances, carrier pigeons carrying answer slips, etc.!

Do not write in the boxes below

1-8 (x/31)	9-14 (x/22)	15-17 (x/14)	18-20(x/19)	21-24 (x/14)	Total (x/100)

Name: <u>SOLUTIONS</u>

I Warmup

1. **[2 points]:** To stabilize Aloha, a node should <u>decrease</u> its packet transmission probability on a collision and <u>increase or set to a maximum threshold</u> it on a successful transmission.

2. **[8 points]:** A switch uses time division multiplexing (rather than statistical multiplexing) to share a link between four concurrent connections (A, B, C, and D) whose packets arrive in bursts. The link's data rate is 1 packet per time slot. Assume that the switch runs for a very long time.

   A. The average packet arrival rates of the four connections (A through D), in packets per time slot, are 0.2, 0.2, 0.1, and 0.1 respectively. The average delays observed at the switch (in time slots) are 10, 10, 5, and 5. What are the average queue lengths of the four queues (A through D) at the switch?

   (Answer legibly in the space below.)

   Solution. Use Little's law; the average queue lengths are 2, 2, 0.5, 0.5.

   B. Connection A's packet arrival rate now changes to 0.4 packets per time slot. All the other connections have the same arrival rates and the switch runs unchanged. What are the average queue lengths of the four queues (A through D) now?

   (Answer legibly in the space below.)

   Solution. B, C, D remain unchanged because of the isolation provided by TDM. A's queue grows to infinity because its arrival rate is larger than the service rate of $1/4 = 0.25$ packets per time slot.

3. **[8 points]:** Under some conditions, a distance vector protocol finding minimum cost paths suffers from the "count-to-infinity" problem.

   (Circle True or False for each choice.)

   A. **True / False** The count-to-infinity problem may arise in a distance vector protocol when the network gets disconnected.

   True.

   B. **True / False** The count-to-infinity problem may arise in a distance vector protocol even when the network never gets disconnected.

   False.

   C. **True / False** The "split horizon" technique *always* enables a distance vector protocol to converge without counting to infinity.

   False.

   D. **True / False** The "path vector" enhancement to a distance vector protocol *always* enables the protocol to converge without counting to infinity.

   True.

4. **[3 points]:** Which of these statements is true of layering in networks, as discussed in 6.02?

(Circle True or False for each choice.)

A. **True / False** The transport layer runs only at the communicating end points and not in the switches on the path between the end points.
True.

B. **True / False** The same transport layer protocol can run unchanged over a network path with a variety of different link technologies.
True.

C. **True / False** The lower layers perform error detection on behalf of the higher layers, eliminating the need for higher layers to perform this function.
False.

5. **[4 points]:** The *exponential weighted moving average* in a reliable transport protocol is:

(Circle True or False for each choice.)

A. **True / False** A single-pole low-pass filter to estimate the smoothed round trip time (RTT).
True.

B. **True / False** A low-pass filter with a pole and a zero to estimate the smoothed RTT.
False.

C. **True / False** Not necessary if the RTT is constant.
True.

D. **True / False** Not necessary if the RTT samples are from an unknown Gaussian distribution.
False.

6. **[2 points]:** In lecture we learned that the JPEG encoding for images was a "lossy" encoding. Which step of the JPEG encoding process loses information?

(Answer legibly in the space below.)

Solution. The quantization of the DCT coefficients loses information.

7. **[2 points]:** The human genome consists of approximately 10^9 codons where each codon can be thought of as one of 21 "symbols" coding for one of the twenty amino acids or serving as a "stop" symbol. Give an expression for the number of bits of information in the genome assuming that each codon occurs independently at random with equal probability.

(Answer legibly in the space below.)

Solution. Each codon contains $\log_2 21$ bits of information, so the total is $10^9 \log_2 21 \approx 4.39 \times 10^9$.

8. **[2 points]:** You are trying to determine the registration number on Alice's Belize license plate. License plates in Belize have four characters, each either a digit or an upper-case letter, and are selected at random. Alice tells you that her license plate contains only digits. How much information has Alice given you about her license plate? You can give your answer in the form an expression.
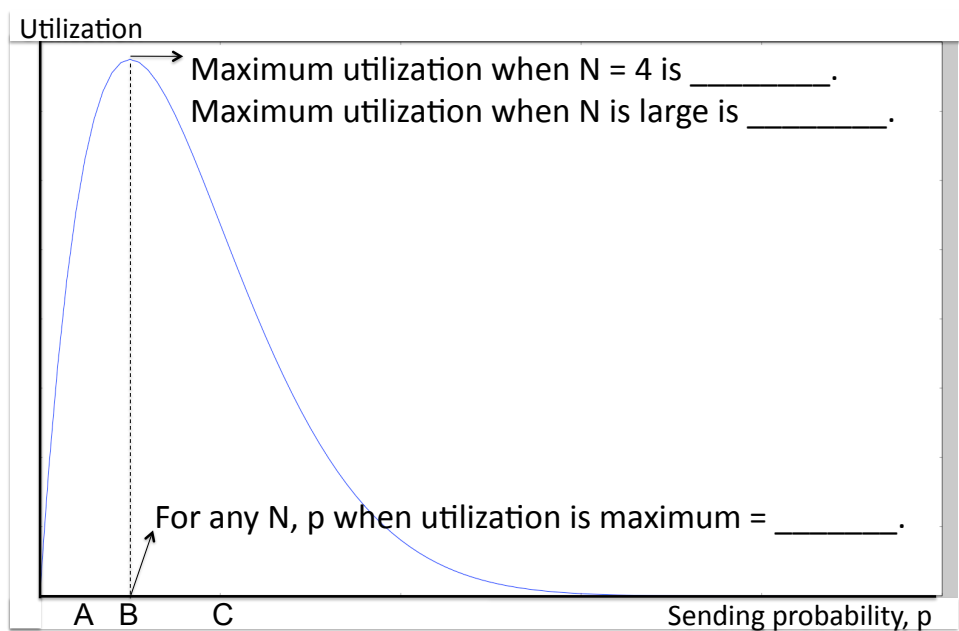
(Answer legibly in the space below.)

Solution. There are 36^4 *a priori* choices of license plate numbers, but what Alice tells us narrows it down to one of 10^4 choices. Therefore, Alice has given us $\log_2(36^4/10^4) = 4\log_2 3.6 \approx 7.4$ bits of information.

II   Aloha

Ben Bitdiddle sets up a shared medium wireless network with one access point and N client nodes. *Unless mentioned otherwise,* assume that the N clients are backlogged and that the access point has no packets to send. Each of the N nodes wants to send its packets to the access point. The network uses slotted Aloha with each packet fitting exactly in one slot. Recall that each backlogged node in Aloha sends a packet with some probability p.

9.   [4 points]: Suppose each node uses the same, fixed value of p. If two or more client nodes transmit in the same slot, a collision occurs at the access point and both packets are lost. The graph below shows the utilization of this protocol as a function of p with N backlogged nodes. Label the points shown in the graph (fill in the *three* blanks).

Utilization

Maximum utilization when N = 4 is _____.
Maximum utilization when N is large is _____.

For any N, p when utilization is maximum = _____.

A B   C   Sending probability, p

Solution. From top to bottom, the answers are $Np(1-p)^{N-1} = 27/64$, $1/e$, and $1/N$.

10.   [2 points]: The graph above shows three values of p: A, B, and C. Rank them in *decreasing* order of collision probability.

Solution. C, B, A.

Now assume that the access point is also backlogged and each of its packets is destined for some client. As before, any two or more nodes (including the access point) sending in the same slot causes a collision. Ben sets the transmission probability, p, of each client node to $1/N$ and sets the transmission probability of the access point to a value p_a.

11. [5 points]: What is the utilization of the network in terms of N and p_a? (Note that the client node transmission probability $p = 1/N$.)

(Answer legibly in the space below.)

Solution. We want the probability of exactly one transmission succeeding in a slot. Either exactly one client should succeed, or only the access point should. The resulting utilization is

$$(1 - p_a)N(1/N)(1 - 1/N)^{N-1} + p_a(1 - 1/N)^N.$$

Interestingly, for large N, this quantity is $1/e$, regardless of what p_a we choose.

12. [3 points]: Suppose N is large. What value of p_a ensures that the aggregate throughput of packets received successfully by the N clients is the same as the throughput of the packets received successfully by the access point? Explain your answer.

(Answer legibly in the space below.)

Solution. For large N, the utilization of the access point is the limit when $N \to \infty$ of the second term in the expression above, which works out to $p_a(1/e)$. The aggregate utilization of the clients is the first term, $(1 - p_a)(1/e)$. The two are equal when $p_a = 1/2$.

From here on, only the client nodes are backlogged—the access point has no packets to send. Each client node sends with probability p (don't assume it is $1/N$).

Ben Bitdiddle comes up with a cool improvement to the receiver at the access point. If exactly one node transmits, then the receiver works as usual and is able to correctly decode the packet. If exactly two nodes transmit, he uses a method to *cancel the interference* caused by each packet on the other, and is (quite remarkably) able to decode both packets correctly.

13. [4 points]: What is the probability, P_2, of *exactly* two of the N nodes transmitting in a slot? Note that we want the probability of *any two* nodes sending in a given slot.

(Answer legibly in the space below.)

Solution. We can choose 2 of N nodes in $N(N-1)/2$ ways. The probability of two *given* nodes sending and every other node not sending is $p^2(1-p)^{N-2}$. So the answer is $\frac{N(N-1)}{2}p^2(1-p)^{N-2}$.

14. [4 points]: What is the utilization of slotted Aloha with Ben's receiver modification? Write your answer in terms of N, p, and P_2, where P_2 is defined in the problem above.

(Answer legibly in the space below.)

Solution. We want exactly one node to transmit, or exactly two nodes to transmit. In the latter case, we have two successes in one slot. Hence, the utilization is $Np(1-p)^{N-1} + 2P_2$.

III   Loss-minimizing routing

Ben Bitdiddle has set up a multi-hop wireless network in which he would like to find paths with high probability of packet delivery between any two nodes. His network runs a distance vector protocol similar to what you developed in Lab 9. In Ben's distance vector (BDV) protocol, each node maintains a *metric* to every destination that it knows about in the network. The metric is the node's estimate of the packet success probability along the path between the node and the destination. The packet success probability along a link or path is defined as 1 minus the packet loss probability along the corresponding link or path.

Each node uses the periodic HELLO messages sent by each of its neighbors to estimate the packet loss probability of the link from each neighbor. You may assume that the link loss probabilities are symmetric; i.e., the loss probability of the link from node A to node B is the same as from B to A. Each link L maintains its loss probability in the variable L.lossprob and $0 < L$.lossprob < 1.

15.   [8   points]: The key pieces of the Python code for each node's integrate() function in BDV is given below. It has **three missing blanks**. Please fill them in so that the protocol will eventually converge without routing loops to the correct metric at each node. The variables are the same as in Lab 9: self.routes is the dictionary of routing entries (mapping destinations to links), self.getlink(fromnode) returns the link connecting the node self to the node fromnode, and the integrate procedure runs whenever the node receives an advertisement (adv) from node fromnode. As in Lab 9, adv is a list of (destination, metric) tuples. In the code below, self.metric is a dictionary storing the node's current estimate of the routing metric (i.e., the packet success probability) for each known destination.

```
# Process an advertisement from a neighboring node in BDV
def integrate(self, fromnode, adv):
  L = self.getlink(fromnode)
  for (dest, metric) in adv:
    my_metric = _____     # fill in the blank
    if (dest not in self.routes
        or self.metric[dest] _____ my_metric          # fill in the blank
        or _____): # fill in the blank
      self.routes[dest] = L
      self.metric[dest] = my_metric

   # rest of integrate() not shown
```

Solution. In order of blanks, the answers are:

```
(1 - L.lossprob)*metric
< (<= is also fine since we said that lossprob is strictly > 0)
self.routes[dest] == L
```

Ben wants to try out a link-state protocol now. During the flooding step, each node sends out a link-state advertisement comprising its address, an incrementing sequence number, and a list of tuples of the form (`neighbor, lossprob`), where the `lossprob` is the estimated loss probability to the `neighbor`.

16. [2 points]: Why does the link-state advertisement include a sequence number?
(Answer legibly in the space below.)

Solution. To enable a node to determine whether the advertisement is new or not; only new information should be integrated into the routing table. (This information is also used to decide whether to rebroadcast the advertisement, since we want to rebroadcast an advertisement only once per link.)

Ben would like to reuse, without modification, his implementation of Dijkstra's shortest paths algorithm from Lab 9, which takes a map in which the links have non-negative costs and produces a path that minimizes the <u>sum of the costs</u> of the links on the path to each destination.

17. [4 points]: Ben has to transform the `lossprob` information from the LSA to produce link costs so that he can use his Dijkstra implementation without any changes. Which of these transformations will accomplish this goal?
(Circle the BEST answer)

    A. Use `lossprob` as the link cost.

    B. Use $\frac{-1}{\log(1-\texttt{lossprob})}$ as the link cost.

    C. Use $\log \frac{1}{1-\texttt{lossprob}}$ as the link cost.

    D. Use $\log(1 - \texttt{lossprob})$ as the link cost.

Solution. The correct choice is **C**. The reason is that maximizing the product of link success probabilities is the same as maximizing the sum of the logs of these probabilities, and that is the same as minimizing the sum of the logs of the reciprocals of these probabilities. **A** is correct only when `lossprob` $<< 1$, which isn't always the case. **D** is plausible because of the log term, but is *negative*, so Dijkstra's doesn't work on a network with negative costs with negative-cost loops. **B** is plausible for the same reason, but is a *decreasing* function of `lossprob`, and so can't be right.

IV   Reliable data delivery

18. [4 points]: Consider a best-effort network with variable delays and losses. Here, Louis Reasoner suggests that the receiver does not need to send the sequence number in the ACK in a correctly implemented stop-and-wait protocol, where the sender sends packet $k + 1$ *only after* the ACK for packet k is received. Explain whether he is correct or not.

(Answer legibly in the space below.)

Solution. (Not surprisingly,) Louis is wrong. Imagine that the sender sends packet k and then retransmits k. However, the original transmission and the retransmission get through to the receiver. The receiver sends an ACK for k when it gets the original transmission, and in response the sender sends packet $k + 1$. Now, when the sender gets an ACK, it cannot tell whether the ACK was for packet k (the retransmission), or for packet $k + 1$!

19. [8 points]: The 802.11 (WiFi) link-layer uses a stop-and-wait protocol to improve link reliability. The protocol works as follows:

   A. The sender transmits packet $k + 1$ to the receiver as soon as it receives an ACK for the packet k.
   B. After the receiver gets the entire packet, it computes a checksum (CRC). The processing time to compute the CRC is T_p and you may assume that it does not depend on the packet size.
   C. If the CRC is correct, the receiver sends a link-layer ACK to the sender. The ACK has negligible size and reaches the sender instantaneously.

The sender and receiver are near each other, so you can ignore the propagation delay. The bit rate is $R = 54$ Megabits/s, the smallest packet size is 540 bits, and the largest packet size is 5,400 bits.

What is the maximum processing time T_p that ensures that the protocol will achieve a throughput of <u>at least 50%</u> of the bit rate of the link in the absence of packet and ACK losses, <u>for any packet size</u>?

(Answer legibly in the space below.)

Solution. Because T_p is independent of packet size, and smaller packets have a lower transmission time over the link, what matters for this question is the processing time for the smallest packet. The maximum throughput of the stop-and-wait protocol is 1 packet every round-trip time (RTT), which in our case is the sum of the transmission time and T_p. The transmission time for a 540 bit packet at 54 Megabits/s is 10 microseconds. Hence, if T_p^* is the maximum allowable processing time, we have:

$$540 \text{ bits}/(10 \text{ microseconds} + T_p^*) = 27 \text{ Megabits/s},$$

giving us $T_p^* = 10$ microseconds.

20. [7 points]: Consider a sliding window protocol between a sender and a receiver. The receiver should deliver packets reliably and in order to its application.

The sender correctly maintains the following state variables:
   `unacked_pkts` – the buffer of unacknowledged packets
   `first_unacked` – the lowest unacked sequence number (<u>undefined</u> if all packets have been acked)
   `last_unacked` – the highest unacked sequence number (<u>undefined</u> if all packets have been acked)
   `last_sent` – the highest sequence number sent so far (whether acknowledged or not)

If the receiver gets a packet that is strictly larger than the next one in sequence, it adds the packet to a buffer if not already present. We want to ensure that the size of this buffer of packets awaiting delivery *never exceeds* a value $W \geq 0$. Write down the check(s) that the sender should perform before sending a new packet in terms of the variables mentioned above that ensure the desired property.

(Answer legibly in the space below.)

Solution. The largest sequence number that a receiver could have *possibly* received is `last_sent`. The size of the receiver buffer can become as large as `last_sent - first_unacked`. One might think that we need to add 1 to this quantity, but observe that the only reason any packets get added to the buffer is when some packet is lost (i.e., at least one of the packets in the sender's unacked buffer must have been lost).

We also need to handle the case when all the packets sent by the sender have been acknowledged—clearly, in this case, the sender should be able to send data.

Hence, if the sender sends a new packet only if

```
if len(unacked_packets) == 0 or last_sent - first_unacked < W
```

the desired requirement is satisfied.

V Source coding

21. [6 points]: Consider a Huffman decoding tree for messages made up by randomly choosing one of 10 symbols (each symbol occurs with a non-zero probability). Let ℓ be the length of the encoding for the symbol least likely to occur.

A. What is the minimum possible value for ℓ? Give one example sequence of probabilities for the symbols that produces this minimum.

(Answer legibly in the space below.)

Solution. 4. All probabilities are equal to 1/10.

B. What is the maximum possible value for ℓ? Give one example sequence of probabilities for the symbols that produces this maximum.

(Answer legibly in the space below.)

Solution. 9. Probability of symbol i (for $1 \le i \le 8$) is $1/2^i$, and the last two symbols (9 and 10) have equal probability of $1/2^9$.

Alyssa P. Hacker's experiment consists of flipping a biased coin 100 times and encoding the resulting H/T sequence into a message. Each coin flip is independent of the others. The coin has a probability of landing heads, $P(H)$ of 0.75. Alyssa has chosen a Huffman code for encoding pairs of results (HH, HT, TH, TT), so the message is constructed by concatenating the codes for 50 pairs.

22. [4 points]: Give an encoding for each of the four possible pairs that would have resulted from running Huffman's algorithm to produce the decoding tree.

Code for HH: 0

Code for HT: 10

Code for TH: 110

Code for TT: 111

23. [2 points]: Using your code for the previous question, what is the average length of an encoded message obtained from 100 coin flips?

(Answer legibly in the space below.)

Solution. 50*(1*(9/16) + 2*(3/16) + 3*(3/16) + 3*(1/16)) = 1350/16 = 84.375 bits.

24. [2 points]: Give an expression for the minimum number of bits required to encode the results for a single flip of this biased coin whose $P(H) = 0.75$.

(Answer legibly in the space below.)

Solution. It is the entropy of the distribution, which is

$$3/4 \log_2(4/3) + 1/4 \log_2 4 = 0.81 \text{ bits.}$$

One might also interpret the question as asking for the number of bits for one coin flip, which would be 1. The question should really have asked for the expected value of the minimum number of bits required.