Name: These R. SOLUTIONS

Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.02 Spring 2010

Quiz III

There are **12 questions** (many with multiple parts) and **11 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, please 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! Please show your work for partial 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 at the top of this page.

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

Do not write in the boxes below

1-2 (x/22)	3 (x/12)	4-6 (x/24)	7-10 (x/22)	11 (x/10)	12 (x/10)	Total (x/100)

I MAC Protocols

1. [3 + 2 + 5 = 10 points]: Ben Bitdiddle is writing a slotted Aloha protocol with contention windows. His goal is to achieve high fairness and come close to the theoretically optimal utilization for slotted Aloha. As in Lab 7, he needs to write the code for two functions: `on_xmit_success` (called whenever a packet transmission succeeds) and `on_collision` (called whenever there is a collision). He writes the following lines of code:

```
def on_xmit_success(self,packet):
    self.cw = 1    # we set the minimum contention window to 1
    self.nextslot = self.network.time + self.cw

def on_collision(self,packet):
    ..................... # incomplete: see below
    self.nextslot = self.network.time + random.random()*self.cw
```

A. What should the incomplete dotted line in `on_collision` be to satisfy Ben's goals? Feel free to introduce additional variables if you need to.

(Explain your answer in the space below.)

Solution: A good approach is to use binary exponential backoff.

```
self.cw = min(2*self.cw, self.cwmax)
```

B. Why does Ben need to multiply by `random.random()` (which generates a random number between 0 and 1) in the second line of `on_collision`?

(Explain your answer in the space below.)

Solution: Otherwise, nodes will collide repeatedly in a deterministic way.

C. Ben would like to try out **quadratic backoff**, a different scheme from part A. Here, the k^{th} contention window value, `self.cw`, is equal to k^2, so the backoffs go as $1, 4, 9, \ldots$. What should the incomplete dotted line in `on_collision` be for quadratic backoff?

(Explain your answer in the space below.)

Solution: We want to mimic quadratic growth. Because $(k+1)^2 = k^2 + 2k + 1$, and we make `self.cw` equal to k, the dotted line is:

```
self.cw = min(self.cw + 2*math.sqrt(self.cw) + 1, self.cwmax)
```

There are other ways to write the same expression, e.g.,

```
self.cw = min((math.sqrt(self.cw) + 1)^2, self.cwmax)
```

2. [3 + 4 + 4 + 1 = 12 points]: Alyssa P. Hacker is designing a MAC protocol for a network used by people who: live on a large island, never sleep, never have guests, and are always on-line. Suppose the island's network has N nodes, and the island dwellers always keep **exactly some four of these nodes backlogged**. The nodes communicate with each other by beaming their data to a satellite in the sky, which in turn broadcasts the data down. If two or more nodes transmit in the same slot, their transmissions collide (the satellite uplink doesn't interfere with the downlink). The nodes on the ground **cannot hear each other**, and each node's packet transmission probability is non-zero. Alyssa uses a slotted protocol with **all packets equal to one slot in length**.

A. For the slotted Aloha protocol with a **fixed** per-node transmission probability, what is the maximum utilization of this network? (Note that there are N nodes in all, of which some four are constantly backlogged.)

(Explain your answer in the space below.)

Solution: We maximize utilization by setting the transmission probability to 1/4, which gives us a utilization of $(1 - 1/4)^3 = 27/64 = 0.42$.

B. In this network, as mentioned above, four of the N nodes are constantly backlogged, but the set of backlogged nodes is not constant. Suppose Alyssa must decide between slotted Aloha with a transmission probability of 1/5 or time division multiple access (TDMA) among the N nodes. For what N does the expected utilization of this slotted Aloha protocol exceed that of TDMA?

(Explain your answer in the space below.)

Solution: TDMA's utilization if $4/N$. The utilization of slotted Aloha is $4 \cdot \frac{1}{5} \cdot (1 - 1/5)^3$. Plugging the numbers, we find that slotted Aloha will have higher utilization when $N \geq 10$.

C. Alyssa implements a stabilization protocol to adapt the node transmission probabilities on collisions and on successful transmissions. She runs an experiment and finds that the measured utilization is 0.5. Ben Bitdiddle asserts that this utilization is too high and that she must have erred in her *measurements*. Explain whether or not it is possible for Alyssa's implementation of stabilization to be consistent with her measured result.

(Explain your answer in the space below.)

Solution: It is possible; the system likely has poor fairness, e.g., some nodes are starved for long periods of time, or there is a significant capture effect.

D. Ben now suggests that Alyssa should make packet sizes much bigger than a slot and use carrier sensing to improve utilization. The nodes can't hear each other. Will this approach increase the utilization of this network?

(Explain your answer in the space below.)

Solution: Carrier sense makes no sense here because nodes can't hear one another.

II Tweet-and-wait

3. [4 + 8 = 12 points]: Lem E. Tweetit is designing a new protocol for Tweeter, a Twitter rip-off. All tweets in Tweeter are 1000 bytes in length. Each tweet sent by a client and received by the Tweeter server is immediately acknowledged by the server; if the client does not receive an ACK within a timeout, it re-sends the tweets, and repeats this process until it gets an ACK.

Sir Tweetsalot uses a device whose data transmission rate is 100 Kbytes/s, which you can assume is the bottleneck rate between his client and the server. The round-trip propagation time between his client and the server is 10 milliseconds. Assume that there is no queueing on any link between client and server and that the processing time along the path is 0. You may also assume that the ACKs are very small in size, so consume neglible bandwidth and transmission time (of course, they still need to propagate from server to client). Do not ignore the transmission time of a tweet.

A. What is the smallest value of the timeout, in **milliseconds**, that will avoid spurious retransmissions?

(Explain your answer in the space below.)

Solution: The round-trip propagation time is 10 milliseconds. The transmission time for a tweet is (1000 bytes / 100 kbytes/s) = 10 milliseconds. The ACKs have negligible transmission time, so the RTT is $10 + 10 = 20$ milliseconds.

B. Suppose that the timeout is set to 90 milliseconds. Unfortunately, the probability that a given client transmission gets an ACK is only 75%. What is the **utilization** of the network?

(Explain your answer in the space below.)

Solution. The expected time for a tweet to get an ACK is equal to

$$\text{RTT} + \frac{1-p}{p} \cdot \text{Timeout},$$

where p is the probability of a given tweet receiving an ACK and RTT the round-trip time. Plugging in the numbers, we find that the expected time is 50 milliseconds. Hence, the rate at which Sir Tweetsalot can send successful tweets is 1000/50 = 20 tweets/s. The capacity of the network is 100 kbytes/s, and if that were used at 100% utilization for tweets, we would be able to send 100 tweets/s. Therefore, the **utilization is 20%.**

Note: Some students may set the RTT to 10 ms. For that, they should lose 2 points in Part A, but we should grade part B as if 10 ms is correct. In that case, the answer would be **25%**. In general, we should grade part B independently even if they mess up Part (A). One problematic case may arise if they end up with an RTT more than 90 ms from Part (A), and end up thinking there are spurious retransmissions. One can only hope that at that point they realize that this weirdness is a huge hint that they erred in Part (A)!

III Sliding Windows

4. [10 points]: Consider the sliding window protocol described in lecture and implemented in Lab 9 this term. The receiver sends "ACK k" when it receives a packet with sequence number k. Denote the window size by W. The sender's packets start with sequence number 1. Which of the following is true of a correct implementation of this protocol over a best-effort network?

(Circle True or False for each choice.)

A. True / False Any new (i.e., previously unsent) packet with sequence number **greater than** W is sent by the sender if, and only if, a new (i.e., previously unseen) ACK arrives.

True.

B. True / False The sender will never send more than one packet between the receipt of one ACK and the next.

False. The sender could time-out and retransmit.

C. True / False The receiver can discard any new, out-of-order packet it receives after sending an ACK for it.

False. The sender thinks the receiver has delivered this packet to the application.

D. True / False Suppose that no packets or ACKs are lost and no packets are ever retransmitted. Then ACKs will arrive at the sender in non-decreasing order.

False. Packets or ACKs could get reordered in the network.

E. True / False The sender should retransmit any packet for which it receives a duplicate ACK (i.e., an ACK it has received earlier).

False. Duplicate ACKs can be ignored by the sender.

5. [6 points]: In his haste in writing the code for the exponential weighted moving average (EWMA) to estimate the smoothed round-trip time, `srtt`, Ben Bitdiddle writes

```
srtt = alpha * r + alpha * srtt
```

where `r` is the round-trip time (RTT) sample, and $0 <$ `alpha` < 1.

For what values of `alpha` does this buggy EWMA over-estimate the intended `srtt`? You may answer this question assuming any convenient non-zero sequence of RTT samples, `r`.

(Explain your answer in the space below.)

Solution: This buggy EWMA over-estimates the true `srtt` when $\alpha > 0.5$. The true `srtt` is equal to `alpha * r + (1-alpha) * srtt`. `alpha * r + alpha * srtt > alpha * r + (1-alpha) * srtt` $\Rightarrow$ `alpha > 0.5`.

6. [2 + 4 + 2 = 8 points]: A sender S and receiver R communicate reliably over a series of links using a sliding window protocol with some window size, W packets. The path between S and R has one bottleneck link (i.e., one link whose rate bounds the throughput that can be achieved), whose data rate is C packets/second. When the window size is W, the queue at the bottleneck link is always **full**, with Q data packets in it. The round trip time (RTT) of the connection between S and R during this data transfer with window size W is T seconds. There are no packet or ACK losses in this case, and there are no other connections sharing this path.

A. Write an expression for W in terms of the other parameters specified above.

(Explain your answer in the space below.)

Solution: $W = C \cdot T$. Note that some students may interpret T as the RTT without any queueing. That's wrong, but we ought to still give them credit as long as they have consistently made this mistake in all the parts. With this interpretation, $W = CT + Q$.

B. We would like to reduce the window size from W and still achieve high utilization. What is the minimum window size, W_{min}, which will achieve 100% utilization of the bottleneck link? Express your answer as a function of C, T, and Q.

(Explain your answer in the space below.)

Solution: Clearly, $W = W_{min} + Q$, where W_{min} is the smallest window size that gives 100% utilization. A smaller window than that would keep the network idle some fraction of the time. Hence, $W_{min} = C \cdot T - Q$.

With the flawed interpretation of T, $W_{min} = CT$.

C. Now suppose the sender starts with a window size set to W_{min}. If all these packets get acknowledged and no packet losses occur in the window, the sender increases the window size by 1. The sender keeps increasing the window size in this fashion until it reaches a window size that causes a packet loss to occur. What is the smallest window size at which the sender observes a packet loss caused by the bottleneck queue overflowing? Assume that no ACKs are lost.

(Explain your answer in the space below.)

Solution: Packets start getting dropped when the window size is $W + 1$, i.e., when it is equal to $CT + 1$.

With the flawed interpretation of T, the window size at which packets start being dropped is $W + 1 = CT + Q + 1$.

IV Routing and the Network Layer

7. [3 points]: Consider a network running the link-state routing protocol as described in lecture and Lab 8. How many copies of any given LSA are received by a given node in the network?

Solution: When there are no packet losses, it is equal to the number of neighbors of the node.

8. [6 points]: In implementing Dijkstra's algorithm in the link-state routing protocol at node u, Louis Reasoner first sets the route for each directly connected node v to be the link connecting u to v. Louis then implements the rest of the algorithm correctly, aiming to produce minimum-cost routes, but does not change the routes to the directly connected nodes. In this network, u has at least two directly connected nodes, and there is more than one path between any two nodes. Assume that all link costs are non-negative. Which of the following statements is true of u's routing table?

(Circle True or False for each choice.)

A. True / False There are topologies and link costs where the **majority** of the routes to other nodes will be **incorrect**.

Solution: True. For example, all the neighbors but one could have very high cost, and all the other links have low cost, so all the routes could in fact be just one link.

B. True / False There are topologies and link costs where **no** routing table entry (other than from u to itself) will be **correct**.

Solution: False. The lowest-cost neighbor's route will be the direct link, of course!

C. True / False There are topologies and link costs where **all** routing table entry (other than from u to itself) will be **correct**.

Solution: True. A trivial example is when all the links have equal cost.

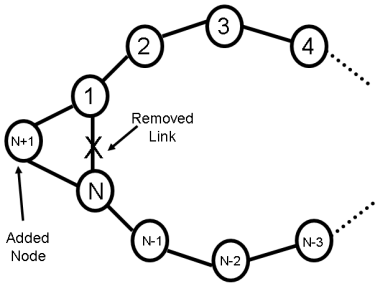
9. [2 points]: Give one difference between what a switch does in a packet-switched network and a circuit-switched network.

(Explain your answer in the space below.)

Solution: Switches in a circuit-switched network participate in a connection set-up/teardown protocol, but not in a packet-switched network.

Switches in a packet-switched network look-up the destination address of a packet during forwarding, but not in a circuit-switched network.

10. [5 + 6 = 11 points]: A network with N nodes and N bidirectional links is connected in a ring, and N is an even number. The network runs a distance-vector protocol in which the advertisement step at each node runs when the local time is $T * i$ seconds and the integration step runs when the local time is $T * i + \frac{T}{2}$ seconds, $(i = 1, 2, \ldots)$. Each advertisement takes time δ to reach a neighbor. Each node has a separate clock and **time is not synchronized** between the different nodes. Suppose that



at some time t after the routing has converged, node $N + 1$ is inserted into the ring, as shown in the figure above. Assume that there are no other changes in the network topology and no packet losses. Also assume that nodes 1 and N update their routing tables at time t to include node $N + 1$, and then rely on their next scheduled advertisements to propagate this new information.

A. What is the minimum time before every node in the network has a route to node $N + 1$?

(Explain your answer in the space below.)

Solution: The key insight to observe is that each introduces a delay of at least $T/2$ because it takes that long between the integration and advertisement steps. Given this fact, the answer would be

$$(\frac{N}{2} - 1) \cdot (\frac{T}{2} + \delta).$$

However, there is a small "fence-post error" in this argument. As stated in the problem, the nodes labeled 1 and N update their routing tables at time t to include node $N + 1$. In the best case, these two nodes could both immediately send out advertisements, and nodes 2 and $N - 1$ could run their integration steps immediately after receiving these advertisements. Because of that, the delay of $T/2$ only starts applying to the other nodes in the network. Hence, the answer is

$$(\frac{N}{2} - 2) \cdot \frac{T}{2} + (\frac{N}{2} - 1) \cdot \delta.$$

B. What is the maximum time before every node in the network has a route to node $N + 1$?

(Explain your answer in the space below.)

The key insight is that it takes in the worst case $T + T/2$ seconds per hop because each node may get the information about the new node **just after** it completes the previous integration step. So it has to wait T for the next integration, and then another $T/2$ to advertise.

The correct answer is

$$(\frac{N}{2} - 1) \cdot (\frac{3T}{2} + \delta).$$

Note: Be lenient on fence-post errors!

V Huffman Codes

11. [1 + 5 + 1 + 3 = 10 points]: Consider messages made up entirely of vowels (A, E, I, O, U). Here's a table of probabilities for each of the vowels:

l	p_l	$\log_2(1/p_l)$	$p_l \log_2(1/p_l)$
A	0.22	2.18	0.48
E	0.34	1.55	0.53
I	0.17	2.57	0.43
O	0.19	2.40	0.46
U	0.08	3.64	0.29
Totals	1.00	12.34	2.19

A. Give an expression for the number of bits of information you receive when learning that a particular vowel is either I or U.

(Explain your answer in the space below.)

Solution: $\log_2(1/(.17 + .08)) = log_2(4) = 2$ bits.

B. Using Huffman's algorithm, construct a variable-length code assuming that each vowel is encoded individually. Please draw a diagram of the Huffman tree and give the encoding for each of the vowels.

(Explain your answer in the space below.)

Encoding for A: _____

Encoding for E: _____

Encoding for I: _____

Encoding for O: _____

Encoding for U: _____

Solution: O/A/E have encodings of length 2, I/U have encodings of length 3.

C. Using your code from part (B) above, give an expression for the expected length in bits of an encoded message transmitting 100 vowels.

(Explain your answer in the space below.)

Solution: 100[(.25)(3) + (.75)(2)] = 100[.75 + 1.5] = 225 bits.

D. Ben Bitdiddle spends all night working on a more complicated encoding algorithm and sends you email claiming that using his code the expected length in bits of an encoded message transmitting 100 vowels is 197 bits. Would you pay good money for his implementation?

(Explain your answer in the space below.)

Solution: Ben is wrong, his implementation probably has a bug! The entropy is 2.19 bits...

VI LZW Compression

Here's the pseudo-code for the LZW adaptive variable-length decoder as given in the lecture notes:

```
initialize TABLE[0 to 255] = code for individual bytes
CODE = read next code from encoder
STRING = TABLE[CODE]
output STRING

while there are still codes to receive:
    CODE = read next code from encoder
    if TABLE[CODE] is not defined:
        ENTRY = STRING + STRING[0]
    else:
        ENTRY = TABLE[CODE]
    output ENTRY
    add STRING+ENTRY[0] to TABLE
    STRING = ENTRY
```

Suppose that this decoder has received the following five codes from the LZW encoder (these are the first five codes from a longer compression run):

```
97 -- index of 'a' in the translation table
98 -- index of 'b' in the translation table
257 -- index of second addition to the translation table
256 -- index of first addition to the translation table
258 -- index of third addition to in the translation table
```

12. [10 points]: After it has finished processing the fifth code, what are the entries in the translation table and what is the cumulative output of the decoder?

table[256]: ab

table[257]: bb

table[258]: bba

table[259]: abb

cumulative output from decoder: a b bb ab bba

FIN