Name: ______________________________

DEPARTMENT OF EECS
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.02 Spring 2012

Quiz III

May 22, 2012

"×" your section	Section	Time	Recitation Instructor	TA
☐	1	10-11	Vincent Chan	Jared Monnin
☐	2	11-12	Vincent Chan	Anirudh Sivaraman
☐	3	12-1	Sidhant Misra	Sungwon Chung
☐	4	1-2	Sidhant Misra	Omid Aryan / Nathan Lachenmyer
☐	5	2-3	Katrina LaCurts	Sunghyun Park
☐	6	3-4	Katrina LaCurts	Muyiwa Ogunnika

Please read and follow these instructions:

0. Please write your name in the space above and × your section.
1. There are **14 questions** (some with multiple parts) and **11 pages** in this quiz booklet.
2. Answer each question according to the instructions given, within **120 minutes**.
3. **Please answer legibly. Explain your answers, especially when we ask you to.** If you find a question ambiguous, write down your assumptions. Show your work for partial credit.
4. Use the empty sides of this booklet if you need scratch space. *If you use the blank sides for answers, make sure to say so!*

Two two-sided "crib sheets" and a calculator allowed. No other aids.

Do not write in the boxes below

1-3(x/20)	**4-6 (x/35)**	**7-11 (x/29)**	**12-14 (x/16)**	**Total(x/100)**

I MAC

1. [4 points]: We studied two ways of stabilizing Aloha: one in which a node's probability of transmission changes in response to collisions and successes, and another in which each node uses a varying contention window. Most practical systems use contention windows. Give one reason why they do that. **Explain your answer.**

Solution: With contention windows, nodes are *guaranteed* to attempt a transmission within a bounded amount of time, specified by the current value of the contention window. This property does not hold when nodes use a probability of sending. The result is that one obtains better fairness across the different nodes, especially over short time scales.

2. [6 points]: Ben Bitdiddle runs the slotted Aloha protocol with stabilization. Each packet is one time slot long. At the beginning of time slot T, node i has a probability of transmission equal to p_i, $1 \leq i \leq N$, where N is the number of backlogged nodes. The increase/decrease rules for p_i are doubling/halving, with $p_{\text{min}} \leq p_i \leq p_{\text{max}}$, as described in 6.02 this term.

Ben notes that exactly two nodes, j and k, transmit in time slot T. After thinking about what happens to these two packets (don't ask us!), derive an expression for the probability that **exactly one node** (out of the N backlogged nodes) will transmit successfully in time slot $T+1$. **Explain your answer.**

Solution: $\displaystyle\sum_{i=1}^{N} q_i \Big(\prod_{j=1, j\neq i}^{N} (1-q_j) \Big)$ where $q_i = \begin{cases} p_i & : i \neq j,k \\ \max\{\frac{p_i}{2}, p_{\text{min}}\} & : i = j,k \end{cases}$

For each node, we consider the probability that it sends successfully, q_i, and that none of the other nodes try to send, $\prod_{j=1,j\neq i}^{N}(1-q_j)$. The probability of sending has decreased for j and k, since they just collided, so we make a special case for their probabilities, so $q_j = \max\{\frac{p_j}{2}, p_{\text{min}}\}$ and $q_k = \max\{\frac{p_k}{2}, p_{\text{min}}\}$.

Note that this formula is a fairly straightforward extension of the $Np(1-p)^{N-1}$ formula we know and love (ha!), to allow each node to have a different probability of attempting a transmission.

3. [10 points]: Tim D. Vider thinks Time Division Multiple Access (TDMA) is the best thing since sliced bread ("if equal slices are good for bread, then equal slices of time must be good for the MAC too", he says). Each packet is one time slot long.

However, in Tim's network with N nodes, the **offered load is not uniform** across the different nodes. The **rate** at which node i generates new packets to transmit is $r_i = \frac{1}{2^i}$ packets per time slot ($1 \leq i \leq N$). That is, in each time slot, the **application** on node i produces a packet to send over the network with probability r_i.

A. (4 points) Tim runs an experiment with TDMA for a large number of time slots. At the end of the experiment, how many nodes (as a function of N) will have a substantial backlog of packets (i.e., queues that are growing with time)? **Explain your answer.**

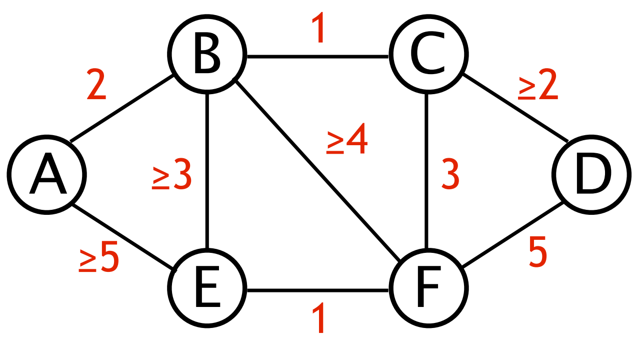
Solution: For the queue to grow at a node, the node needs to generate packets faster than they are removed from the queue. Nodes are generating packets at a rate $\frac{1}{2^i}$, and packets are removed from the queue at rate $\frac{1}{N}$, since we're using TDMA. Hence, the queue of each node i that satisfies $\frac{1}{2^i} > \frac{1}{N}$ will grow unbounded. The number of such nodes is $\lceil \log_2 N \rceil - 1$. (We don't care about the student getting the floor or ceiling or -1 perfectly right!)

B. (6 points) Let $N = 20$. Calculate the **utilization** of this non-uniform workload running over TDMA. A possibly useful formula: if $|\alpha| < 1$, $\sum_{i=k}^{N} \alpha^i = \alpha^k \cdot \frac{(1-\alpha^{N-k+1})}{(1-\alpha)} \approx \frac{\alpha^k}{1-\alpha}$ when $N - k$ is large. **Explain your answer.**

Solution: We break this into two cases: nodes that always have a backlog of packets, and nodes that don't. The nodes that always have a backlog will always send data when it's their turn. The number of nodes with this property is the answer to Part A, i.e., $\lfloor \log_2 N \rfloor$. Call this number k. The rest of the nodes send data whenever they are backlogged. The probability that any given time slot in which one of these nodes (i) gets its turn to send is given by r_i.

Putting these two facts together, the utilization is $k/N + \frac{N \sum_{i=k+1}^{N} r_i}{N}$. Sticking in $N = 20$, for which $k = 4$, we find that the utilization is $4/20 + \sum_5^{20} 1/2^i \approx 0.26$.

II Go Ahead, Make My Route



Jack Ripper uses a minimum-cost **distance-vector routing protocol** in the network above. Each link cost (not shown) is a **positive integer** and is the same in each direction of the link. Jack sets "infinity" to 32 in the protocol. After all routes have converged (breaking ties arbitrarily), F's routing table is as follows:

Destination	Cost	Route
A	6	link $\langle FC \rangle$
B	4	link $\langle FC \rangle$
C	3	link $\langle FC \rangle$
D	5	link $\langle FD \rangle$
E	1	link $\langle FE \rangle$

4. [17 points]: Using the information provided, answer these questions:

A. (2 points) Fill in the two missing blanks in the table above. Solution: See the table above.

B. (9 points) For **each link** in the picture, write the link's cost in the box near the link. Each cost is either a positive integer or an expression of the form "$< c, \leq c, \geq c$, or $> c$", for some integer c. Solution: See the figure above.

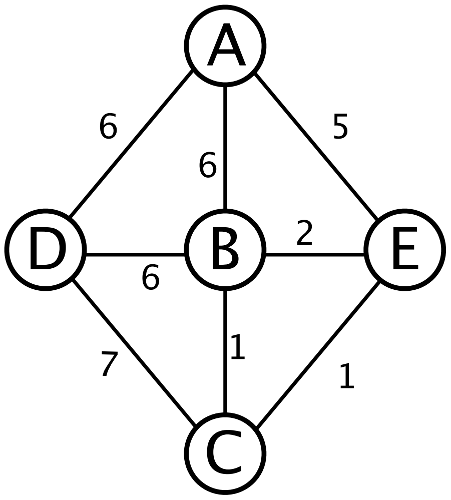
C. (2 points) Suppose link $\langle FE \rangle$ fails, but there are no other changes. When the protocol converges, what will F's **route** (not path) to E be? (If there is no route, say "no route".) **Explain.**

Solution: Link $\langle FC \rangle$ **or** link $\langle FB \rangle$. Since ties are broken arbitrarily, $\langle FB \rangle$ could be taken if its cost was exactly 4. $\langle FD \rangle$ will never be taken as it doesn't provide a lower-cost path to C.

D. (4 points) Now suppose links $\langle BC \rangle$ and $\langle BF \rangle$ **also** fail soon after link $\langle FE \rangle$ fails. There are no packet losses. In the **worst case**, C and F enter a "count-to-infinity" phase. How many distinct route advertisements (with different costs) must C hear from F, before C determines that it does not have any valid route to node A? **Explain.**

Solution: 6. After the failures, the count-to-infinity scenario unfolds as follows. F will advertise $(A, 6)$ to C. C will advertise $(A, 9)$ to F ($9 =$ cost of F's path to A plus the cost of $\langle CF \rangle$). F then updates its cost and advertises $(A, 12)$, etc. In total, C needs to hear 6 advertisements from F before its cost to A reaches "infinity". The advertisements are $(A, 6), (A, 12), (A, 18), (A, 24), (A, 30), (A, 32)$. An answer of 5 is equally acceptable and correct, because one could assume that the first advertisement in the count-to-infinity phase is $(A, 12)$.

Alyssa P. Hacker runs the **link-state routing protocol** in the network shown below. Each node runs Dijkstra's algorithm to compute minimum-cost routes to all other destinations, breaking ties arbitrarily.



5. [8 points]: Answer the following questions, **explaining each answer**.

A. (3 points) In what order does C add destinations to its routing table in its execution of Dijkstra's algorithm? Give all possible answers.

Solution: Routes will be added in non-decreasing order of the cost of the min-cost path to nodes. Hence, B, E, A, D or E, B, A, D.

B. (3 points) Suppose the cost of link $\langle CB \rangle$ increases. What is the largest value it can increase to, before **forcing** a change to any of the routes in the network? (On a tie, the old route remains.)

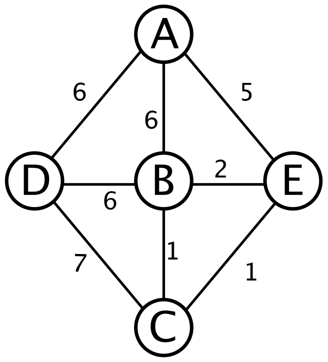
Solution: If the route to D is $\langle CD \rangle$, the cost of $\langle CB \rangle$ can increase to 3. If it increases further, the route to B will certainly change.

If the route to D is $\langle CB \rangle$, then the cost of $\langle CB \rangle$ cannot change at all before forcing a change (namely, a change in the route to D).

C. (2 points) Assume no link-state advertisement (LSA) packets are lost on any link. When C generates a new LSA, how many copies of that LSA end up getting flooded in total over all the links of this network, using the link-state flooding protocol described in 6.02?

Solution: Two over each link, so 16 total.

Picture reproduced here for convenience.



The links in Alyssa's network are unreliable; on each link, any packet sent over the link is delivered with some probability, p, to the other end of the link, independent of all other events $(0 < p < 1)$. Suppose links $\langle CE \rangle$ and $\langle BD \rangle$ fail.

6. [10 points]: Answer the following questions, **explaining each answer**.

A. (2 points) How do C and E discover that the link has failed? How does the method work?

Solution: C and E use the `HELLO` protocol, where `HELLO` messages are sent periodically between neighbors, and a link is considered to have failed if a `HELLO` message from the other end has not been heard in $k \times$ `HELLO_INTERVAL`, for some (small) positive integer k. The question was ambiguous because it did not say which link (we had intended it to be link $\langle CE \rangle$. C and E would detect the failure of link $\langle BD \rangle$ from an LSA originated by either B or D.

B. (5 points) Over this unreliable network, link state advertisements (LSAs) are lost according to the probabilities mentioned above. Owing to a bug in its software, E **does not originate any LSA of its own or flood them**, but all other nodes (except E) work correctly. Calculate the probability that A learns that link $\langle CE \rangle$ has failed from the **first** LSA that originates from C after C discovers that link $\langle CE \rangle$ has failed. **Note that link $\langle BD \rangle$ has also failed.**

Solution: The probability that A receives the first LSA from C is the probability that the LSA makes it over either path $C - D - A$ or $C - B - A$ (since $\langle BD \rangle$ has failed, and since E isn't forwarding LSAs, those are the only paths available). The probability that one of those paths failed is $1 - p^2$ ($1-$ the probability of success), so the probability that both fail is $(1 - p^2)^2$. The probability that at least one path succeeds, then, is $1 - (1 - p^2)^2$. Simplified, that's $2p^2 - p^4 = p^2(2 - p^2)$.

C. (3 points) Suppose only link $\langle CE \rangle$ had failed, **but not** $\langle BD \rangle$, which like the other surviving links can delivery packets successfully with probability p. Now, would the answer to part **B** above **increase**, **decrease**, or **remain the same**? Why? (No math necessary.)

Solution: Increase. It would add two more potential paths for the LSA to take: $CDBA$ and $CBDA$.

III Seek-Transmit, by Gloria Mundi

Gloria Mundi, a recent alum, has just started an open source project she calls *Seek-Transmit* to help users find movie clips (legally, of course!) over a best-effort network. Her system has two parts:

1. **Seek:** The client running on the user's device sends a short search string, called a *seek*, to a seek server, which responds with a list of zero or more *transmit servers* that have the desired content.

   Seek uses the 6.02 stop-and-wait protocol. The ACKs are replaced by the server's response to a seek; if the client does not get a response within a timeout period, T seconds, it re-tries the seek. It keeps retrying every T seconds until it gets a response. There are no spurious retries.

2. **Transmit:** The client contacts one of the transmit servers to stream the content. Transmit uses the 6.02 sliding window protocol for reliable delivery. There are no spurious retransmissions.

Gloria's goal is for each client to obtain a response to the seek as quickly as possible, and then for the transmit phase to achieve as high a throughput as possible.

Each packet includes in its headers a **source** field and a **hop limit** field.

7. [4 points]: Answer the following questions about these packet header fields:

A. (3 points) What is the purpose of the **hop limit** field? How does it achieve this purpose?

Solution: The **hop limit** field prevents packets from getting stuck in routing loops (which is bad for the packet, obviously, but also congests the network). The field is decremented at every switch, and when it reaches zero, the packet is dropped.

B. (1 point) State one use of the **source** field in Gloria's Seek-Transmit system.

Solution: The **source** field has at least two uses: (1) for the seek server to know where to send its response (back to the source), and (2) for the client in the Transmit protocol to know where to send ACKs.

Gloria deploys multiple seek servers around the world. The video clips are also on a number of transmit servers around the world. She writes some monitoring software to help select the best seek and transmit servers, from among the set of possible servers. The monitoring software measures the following and provides the results to the client:

- R_i, the average round-trip time (RTT), in seconds, along the **network path** to the i^{th} server, **excluding** the server's processing delay, but **including** the transmission delay of a packet and its response.
- D_j, the average delay (in seconds) to process a seek request at the j^{th} seek server.
- p_i, the probability that the client successfully gets a response from the i^{th} server to a packet it sends. (The packet/response is a seek/seek response for a seek server, and data/ACK for a transmit server.)
- B_i, the rate of the bottleneck link for transfers between the i^{th} server and client, in bytes per second.

8. [5 points]: Recall that the goal is for the client to select **one** seek server that will respond quickest (among the set of choices) to the client. Recall also that the seek protocol is stop-and-wait, with a fixed timeout value of T seconds. For each seek server i, the client computes some function, f_i (in terms of the parameters defined above), and then minimizes or maximizes it, to accomplish the desired goal. Write down f_i, state whether it should be maximized or minimized, and **explain your answer**.

Solution: f_i should be the **expected time to get a response** from a particular seek server. For the i^{th} server,

$$f_i = (R_i + D_i) + \frac{1-p_i}{p_i}T.$$

This function should be **minimized**.

9. [5 points]: Recall that Gloria uses the 6.02 sliding window protocol to deliver video from a transmit server to the client. Assume that the window size for the data transfer is picked to achieve the maximum possible throughput of the transfer, subject to the parameters of the network defined above. Assume also that each transmit server produces data extremely fast, so the server itself is not the bottleneck. The client's goal is to pick **one** transmit server that will give it the highest video transfer throughput. For each transmit server i, the client computes some function, g_i (in terms of the parameters defined above), and then minimizes or maximizes it, to accomplish the desired goal. Write down g_i, state whether it should be maximized or minimized, and **explain your answer**.

Solution: The function g_i represents the expected throughput from transmit server i. For the i^{th} server, this is $B_i p_i$. It is *not* simply B_i, because a path with large B_i may not be the one with highest throughput if the loss rate is also high. We want to **maximize** g_i.

10. [5 points]: Annette Werker implements an exponentially weighted moving average (EWMA) method to estimate the smoothed RTT (srtt) and the RTT deviation (rttdev) of the network path between a server and client. These estimators have the form

$$\text{srtt} \leftarrow \alpha r + (1-\alpha)\text{srtt} \quad \text{and} \quad \text{rttdev} \leftarrow \beta(|r - \text{srtt}|) + (1-\beta)\text{rttdev}.$$

Fill in the blanks (if choices are provided, pick the best one):

A. The variable r in the above expressions is the most recent RTT sample.

B. These EWMA estimators help the sender (sender / receiver / switch / MAC) to set its timeout (window size / timeout / transmit probability / queue size).

C. The larger the value of α and β, the lower (greater / lower) the contribution of any past samples to the srtt and rttdev estimates.

D. The srtt and rttdev estimates are updated when an ACK is received from the receiver (what event?).

11. [10 points]: Annette Werker conducts tests between a transmit server and a client using the sliding window protocol. There is no other traffic on the path and no packet loss. She finds that:

- With a window size $W_1 = 50$ packets, the throughput is 200 packets per second.
- With a window size $W_2 = 100$ packets, the throughput is 250 packets per second.

Annette finds that even this small amount of information allows her to calculate several things, assuming there is only one bottleneck link. Calculate the following, **explaining your answers**:

A. (2 points) The minimum round-trip time between the client and server.
Solution: Using W_1, the minimum RTT is $\frac{50}{200}$ s $= 250$ ms.

B. (3 points) The average **queueing** delay at the bottleneck when the window size is 100 packets.
Solution: Using W_2, the RTT is $\frac{100}{250} = 0.4$ s $= 400$ ms. Hence, the queueing delay is $400 - 250 = 150$ ms, when the window size is 100 packets.

C. (5 points) The average queue size when the window size is 100 packets.
Solution: Using Little's Law, N, the average queue size is $\lambda \times D = 250 \times 0.15 = 37.5$ packets.

Gloria realizes that packets that arrive too late at the receiver cause the video to stall, and that the application does not need *all* the packets to play properly. Missing packets are acceptable, but the packets must be delivered to the application in **increasing sequence order**. She **removes all retransmissions from the protocol**, so now the sender just sends data packets with incrementing sequence numbers at some constant rate. Packets may show up in arbitrary order, be delayed arbitrarily, and be duplicated in the network.

Gloria would like to modify the receiver transport layer to apply the following rules:

R1. Deliver packets to the application only in strictly increasing sequence order (perhaps skipping some).

R2. Every t milliseconds, deliver exactly one packet to the application. This packet should be the one with the smallest sequence number in the receiver's buffer that is strictly greater than the one previously delivered to the application. You may assume that such a packet always exists in the receiver's buffer.

12. [3 points]: The receiver transport protocol receives the following packets at the times shown:

Time (ms):	0,	8,	9,	13,	14,	19,	44,	48,	56
Packet:	5,	3,	1,	2,	10,	8,	5,	7,	4

Let $t = 10$ ms. The receiver transport delivers the first packet to the video application at time 10 ms. Write the time and packet sequence at which various packets are delivered to the application.

Time (ms):	10,	20,	30,	40,	50,	60,	70
Packet:	1,	2,	3,	5,	7,	8,	10

13. [12 points]: Write Python code (don't worry about perfect syntax!) to implement Gloria's receiver transport protocol according to the rules above. Write two functions: `receive` and `deliver2app` (next page). The specifications of these functions are given below. Name variables descriptively, or define them. Use the blank side if you need more space, but tell us if you do so.

```
'''
receive() is called whenever a data packet arrives. Maintain
self.rcvbuf, a buffer of packets that haven't yet been delivered
to the app. Delete packets that will never be delivered to the app
from rcvbuf. You may use standard Python library modules and functions,
e.g., foo.sort() if you wish to sort list foo in ascending order.
'''
def receive(self, pkt):
    # Notation: pkt.seq is the sequence number of pkt

    # Assume first valid seq number is 1.
    # Initialize last_delivered to 0 in class definition.
    # No need to send an ACK because there are no retransmissions!
    if pkt.seq > last_delivered and pkt not in self.rcvbuf:
        self.rcvbuf.append(pkt)
        self.rcvbuf.sort()  # in-place sort
```

```
'''
Assume that deliver2app is called every t seconds by the reciver
transport protocol's main loop. Call app_receive(pkt) to deliver
a pkt from self.rcvbuf to the app, following rules R1 and R2.
'''
def deliver2app(self):

    pkt = self.rcvbuf.pop(0)
    app_receive(pkt)
    last_delivered = pkt.seq
```

14. [1 points]: This elaborate set up with Seek-Transmit and Gloria Mundi has nothing to do with networking but everything to do with classical education. What does *sic transit gloria mundi* mean? **(Circle ALL that apply.)**

A. Gloria fell sick on Monday while traveling.

B. Glory is fleeting, but obscurity is forever.

C. Thus passes the glory of the world.

D. You *#@$*@!#s. The "T" in MIT is "technology", not "torture". This truly *is* the last straw; I'm reporting you to the new President (the complaints to the Dean didn't work the last time).

Solution: C.

FIN

Have a great summer!