Problem 1: Computing TCP's RTT and timeout values

Suppose that TCP's current estimated values for the round trip time (estimatedRTT) and deviation in the RTT (DevRTT) are 200 msec and 34 msec, respectively (see Section 3.5.3 for a discussion of these variables). Suppose that the next three measured values of the RTT are 220, 370, and 250 respectively.

Compute TCP's new value of estimatedRTT, DevRTT, and the TCP timeout value after each of these three measured RTT values is obtained. Use the values of $\alpha = 0.125$ and $\beta = 0.25$.

Problem 2: TCP in action: slow start, congestion avoidance, and fast retransmit.

Consider the figure below, which plots the evolution of TCP's congestion window at the beginning of each time unit (where the unit of time is equal to the RTT); see Figure 3.53 in the text. In the abstract model for this problem, TCP sends a "flight" of packets of size $cwnd$ at the beginning of each time unit. The result of sending that flight of packets is that either (i) all packets are ACKed at the end of the time unit, (ii) there is a timeout for the first packet, or (iii) there is a triple duplicate ACK for the first packet. In this problem, you are asked to reconstruct the sequence of events (ACKs, losses) that resulted in the evolution of TCP's $cwnd$ shown below.
Consider the evolution of TCP's congestion window in the example above and answer the following questions. The initial value of \( \text{cwnd} \) is 1 and the initial value of \( \text{ssthresh} \) (shown as a red +) is 8.

- Give the times at which TCP is in slow start, congestion avoidance and fast recovery at the start of a time slot, when the flight of packets is sent.
- Give the times at which the first packet in the sent flight of packets is lost, and indicate whether that packet loss is detected via timeout, or by triple duplicate ACKs.
- Give the times at which the value of \( \text{ssthresh} \) changes, and give the new value of \( \text{ssthresh} \).

**Problem 3:** Consider a datagram network using 32-bit host addresses. Suppose a router has four links, numbered 0 through 3, and packets are to be forwarded to the link interfaces as follows:

<table>
<thead>
<tr>
<th>Destination Address Range Link</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>11110000 00000000 00000000 00000000 through 11110000 00111111 11111111 11111111</td>
<td>0</td>
</tr>
<tr>
<td>11110000 01000000 00000000 00000000 through 11110000 01000000 11111111 11111111</td>
<td>1</td>
</tr>
<tr>
<td>11110000 01000001 00000000 00000000 through 11110001 01111111 11111111 11111111</td>
<td>2</td>
</tr>
<tr>
<td>otherwise</td>
<td>3</td>
</tr>
</tbody>
</table>

a) Provide a forwarding table that has five entries, uses longest prefix matching, and forwards packets to the correct link interfaces.

b) Describe how your forwarding table determines the appropriate link interface for datagrams with destination addresses:

- 11001000 10010001 01010001 01010101
- 11100001 01000000 11000011 00111100
**Problem 4: Addressing** Consider the topology shown to the right (but ignore the IP addresses shown).

Denote the three subnets with hosts (starting clockwise at 12:00) as Networks A, B, and C. Denote the subnets without hosts as Networks D, E, and F.

a) Assign network addresses to each of these six subnets, with the following constraints: All addresses must be allocated from 214.97.254/23; Subnet A should have enough addresses to support 250 interfaces; Subnet B should have enough addresses to support 120 interfaces; and Subnet C should have enough addresses to support 120 interfaces. Of course, subnets D, E and F should each be able to support two interfaces. For each subnet, the assignment should take the form a.b.c.d/x or a.b.c.d/x – e.f.g.h/y.

b) Using your answer to part (a), provide the forwarding tables (using longest prefix matching) for each of the three routers.

**Problem 5: Dijkstra’s Link State Routing.** Consider the 6-node network shown below, with the given link costs. Using Dijkstra's algorithm, find the least cost path from source node u to all other destinations. Show your work in tabular format, as in Table 4.3 in the text.

**Problem 6: Distance Vector Routing.**

Consider the network fragment shown below. x has only two attached neighbors, w and y. w has a minimum-cost path to destination u (not shown) of 5, and y has a minimum-cost path to u of 6. The complete paths from w and y to u (and between w and y) are not shown. All link costs in the network have strictly positive integer values.
a) Give \( x \)'s distance vector for destinations \( w, y, \) and \( u \).
b) Give a link-cost change for either \( c(x,w) \) or \( c(x,y) \) such that \( x \) will inform its neighbors of a new minimum-cost path to \( u \) as a result of executing the distance-vector algorithm.
c) Give a link-cost change for either \( c(x,w) \) or \( c(x,y) \) such that \( x \) will not inform its neighbors of a new minimum-cost path to \( u \) as a result of executing the distance-vector algorithm.

**Problem 7: BGP and path advertisement.**

Figure 4.42 of the text, is shown to the right.

Consider the path information that reaches stub networks \( W, X, \) and \( Y \). Based on the information available at \( W \) and \( X \), what are their respective views of the network topology? [Hint: you need to think about which networks pass on which path information to neighboring networks]. Justify your answer. The topology view at \( Y \) is shown to the right (lower), so you really only need to answer this question for \( W \) and \( X \).

**Extra Credit Problem [10%].**

Attend the lecture (10%) by Guy Fedorkow of Juniper Networks on “High Scale Routers for Internet Infrastructure” [https://www.cs.umass.edu/speakers/guy-c-fedorkow/2013/nov/8](https://www.cs.umass.edu/speakers/guy-c-fedorkow/2013/nov/8) at 11 am on Friday Nov. 8 in Room 150 Computer Science Building. If you can’t make that, you can view it on line (5%), following the link from the class website. To get this credit, just send an email to kurose@cs.umass.edu AND our TA haopengz@gmail.com indicating whether you attended or watched.