This is an open-book examination. You may consult the text, your notes, or any other inanimate source of information while completing this examination. You should stop all work on the exam at most three hours after you begin.

I have tried to provide space for all of your answers in the exam booklet. If you prefer, you are certainly free to write your answers on separate sheets. In either case, please do scrap work on separate sheets of paper and then present your final answers as clearly as possible.

1. (35 pts.) Assume that the graph shown below represents the interconnections between the 5 routers of a switched network and that the numbers labeling the edges of the graph represent the delays packets experience crossing the corresponding links. (As usual, we are making the unrealistic assumption that the delays on a given link are identical for traffic traversing the link in both directions.)

(a) Show the process through which the distance vector algorithm (Bellman-Ford) would determine the minimum path lengths between the nodes of this graph by filling in the tables below (I have tried to provide a few extra tables in case you make mistakes). Because we are assuming that each links cost is the same in both directions, the table will always be symmetric around the diagonal. As a result, you can save a little time by only filling in the top right (or bottom left) half of the table.
(b) Note how many steps were required from the initial state (with lots of $\infty$'s in the table) until the point where the distance vector algorithm converged on the set of minimal route lengths for the graph given in part (a). Can you find a set of edge weights that will make the algorithm converge in fewer steps when applied to a graph with this structure? If so, show how you would modify the edge weights by marking up the copy of the graph shown below and briefly explain why this change would reduce the number of steps required. If not, explain why the algorithm cannot converge in fewer steps than you used in part (a).
(c) The simulation of the distance vector you performed in one of our homework questions and in part (a) does not accurately reflect two aspects of the way the algorithm is used in practice:

- The distance vector algorithm does not just compute the lengths of shortest paths. It also determines the first step of each of these shortest paths.
- The distance vector algorithm does not always start from the initial state. If the delay associated with some link in the graph changes after the algorithm has converged, the entries in the existing table are used as the initial state. The iterative process you hand-simulated for part (a) is then applied using the updated link delays until a new steady state is reached.

For this problem, I would like you to consider how many steps might be required before the algorithm converges to a new steady state when an edge delay changes.

Consider the graph and edge weights shown below.

The table below shows both the path lengths and the first hops that would be computed by applying the Bellman-Ford algorithm to the delays shown in the graph until it converged. For example, the entry $5_D$ in the Bth row of the Eth column indicates that the best path from E to B starts with D and has total delay of 5.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>$7_D$</td>
<td>$1_A$</td>
<td>$3_C$</td>
<td>$3_C$</td>
</tr>
<tr>
<td>B</td>
<td>$7_C$</td>
<td>0</td>
<td>$6_D$</td>
<td>$4_B$</td>
<td>$5_D$</td>
</tr>
<tr>
<td>C</td>
<td>$1_C$</td>
<td>$6_D$</td>
<td>0</td>
<td>$2_C$</td>
<td>$2_C$</td>
</tr>
<tr>
<td>D</td>
<td>$3_C$</td>
<td>$4_D$</td>
<td>$2_D$</td>
<td>0</td>
<td>$1_D$</td>
</tr>
<tr>
<td>E</td>
<td>$3_C$</td>
<td>$5_D$</td>
<td>$2_E$</td>
<td>$1_E$</td>
<td>0</td>
</tr>
</tbody>
</table>

Modify the edge weight associated with one edge in the graph in such a way that the number of steps required before the distance vector algorithm converges to a new steady state is as large as possible. Indicate which edge weight you would modify by writing in the new value on the diagram of the graph shown above. Show the first few steps of the process in the tables provide on the next page. (By “few” I mean enough to answer parts (d) and (e).) To save you a little time, I have filled in the “Step 1” table with the values from the table shown above. For this problem, however, you will discover that you cannot save time by only filling in half of the table. While the table (well at least the path lengths) will be symmetric once the algorithm converges, it will not be symmetric during the intermediate steps.
2. (30 pts.) Consider the message M = 100011001010.

(a) Calculate the CRC error-detection bits for M when G = x^4 + 1. Show the complete message that would be transmitted.

(b) Show an example of a damaged version of the message you would send given the results of part (a) that would not be detected as an error by the CRC. Clearly indicate which bits are damaged and briefly justify your answer.

(c) The Internet Checksum is designed to work by breaking the message to be checked into 16 bit words and then processing those words in a specific way. Since our M, 100011001010, is not even 16 bits long, applying the Internet Checksum algorithm to M would be a bit boring. It should, however, be easy to imagine other forms of checksum that are identical to the Internet Checksum except that the original message is first broken up into units of some size.
other than 16. For example, one could divide the message into 4 bit units and then apply the checksum process. Show how to compute such a checksum for M. Show the message that would be transmitted in this case (assuming the checksum follows the data bits even though it doesn’t in an IP packet).

(d) Show an example of a damaged version of the message you would send given the results of part (c) that would not be detected as an error. Clearly indicate which bits are damaged and briefly justify your answer.

3. (35 pts.) During their discussion of sliding window protocols, Peterson and Davie provide some guidance on the selection of values for SWS, the sending window size, and RWS, the receiver window size. In particular, they state that “SWS is easy to compute for a given delay-bandwidth product” and suggest that RWS = 1 and RWS = SWS are the most common settings for RWS.

(a) Let D be the propagation time of the link, let P be the packet size in bits, and let R be the transmission rate in bits per second. Give a formula for the utilization of the link as a function of SWS assuming no packets are lost or delayed. What value of SWS maximizes utilization without wasting buffer space?

(b) Now, consider a scenario in which any single packet or acknowledgment may be lost with probability $p_0$. Let $p$ be the probability that either a packet or its acknowledgment is lost. Give a formula for $p$ as a function of $p_0$. Explain your formula clearly and briefly. (Note: You can continue to the next subpart of this problem even if you cannot find a formula for $p$.)

(c) Assuming the probability that a given packet is not acknowledged is $p$, let $S$ denote the expected number of consecutive packets that will be sent and acknowledged before either a packet or its acknowledgment is lost (i.e. before a timeout might occur). Give as simple a formula for $S$ as possible. (You may use $p$ in this formula rather than $p_0$ even if you did not answer the preceding question.) Explain your formula clearly and briefly.

(d) Finally, give a formula for the average utilization of a link with propagation time $D$, packet size $P$, transmission rate $R$, and a probability $p_0$ that either a packet or an acknowledgment will be lost. Assume that SWS is large enough that the sender is always free to send as long as no timeout has occurred and that RWS = 1. Assume that the timeout interval is set as tightly as possible. Assume that the receiver never uses NAKs or selective ACKs. You may use $S$ or $p$ in this formula rather than $p_0$ even if you did not answer the two preceding questions. Explain your formula clearly and briefly.

(e) In what way(s) would using a RWS value larger than 1 improve the performance of the sliding window protocol? To answer this question, provide one or more specific scenarios in which a RWS greater than 1 would improve performance. Make it clear what aspect of performance you are discussing: utilization or delay.