Homework 3

CS 3640: Introduction to Networks and Their Applications
Due: 11:59 pm, Oct 30th (Mon), 2017.

Submission instructions: Submit a PDF file containing your answers and necessary source files containing your code. All files must be submitted on ICON. You are not permitted to share or copy any written materials or code from your peer. Please mention the source and write sufficient comments to show your understanding if you use any code from the Internet.

We recommend Python 3 for the programming part of this homework. However, C/C++ or Java are also acceptable.

Note: You must show all your work to receive *full* credit for an answer. Guessing the right answer will not earn you any point.

Question 1

[8 points] Consider two network entities, A and B, which are connected by a perfect bidirectional channel (i.e., any message sent will be received correctly; the channel will not corrupt, lose, or re-order packets). A and B are to deliver data messages to each other in an alternating manner: First, A must deliver a message to B, then B must deliver a message to A, then A must deliver a message to B and so on. If an entity is in a state where it should not attempt to deliver a message to the other side, and there is an event like `rdt_send(data)` call from above that attempts to pass data down for transmission to the other side, this call from above can simply be ignored with a call to `rdt_unable_to_send(data)`, which informs the higher layer that it is currently not able to send data. [Note: This simplifying assumption is made so you don't have to worry about buffering data.

Draw a FSM specification for this protocol (one FSM for A, and one FSM for B!). Note that you do not have to worry about a reliability mechanism here; the main point of this question is to create a FSM specification that reflects the synchronized behavior of the two entities. You should use the following events and actions that have the same meaning as protocol rdt1.0 in the slide 27 of chapter 3: `rdt_send(data)`, `packet = make_pkt(data)`, `udt_send(packet)`, `rdt_rcv(packet)`, `extract(packet,data)`, `deliver_data(data)`. Make sure your protocol reflects the strict alternation of sending between A and B. Also, make sure to indicate the initial states for A and B in your FSM descriptions.

Solution: Check Figure 1.

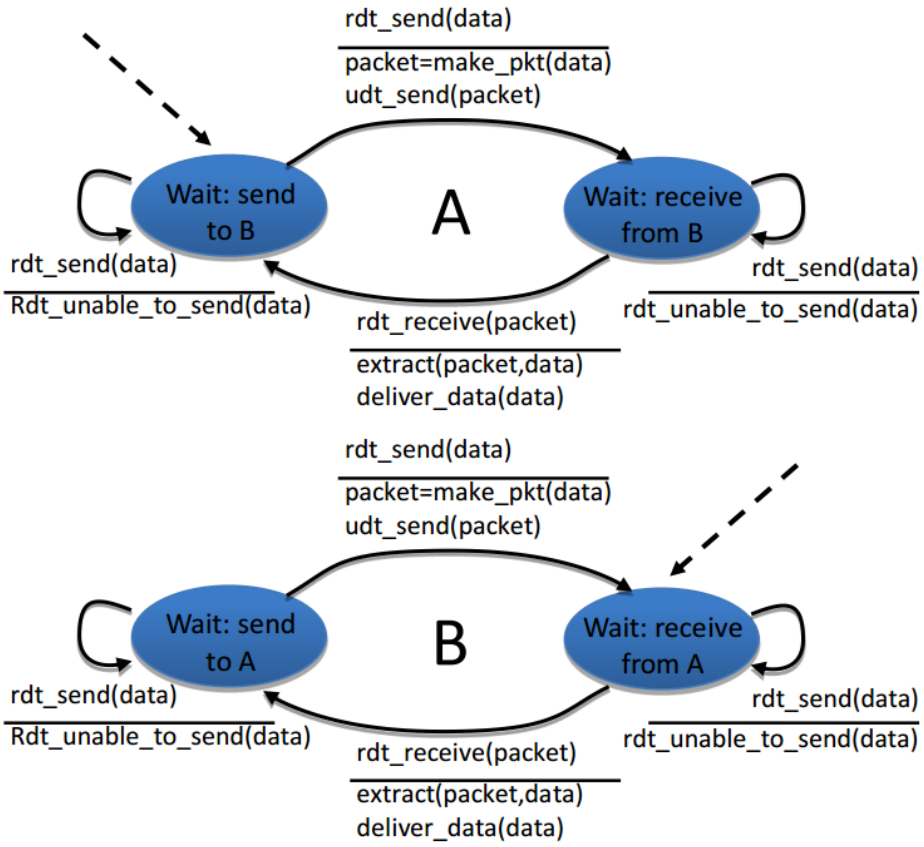

Figure 1: Solution for question 1.

Question 2

[18 points]

(I) Consider the Go-Back-N (GBN) and Selective Repeat (SR) protocols. Suppose the sequence number space is of size k. What is the largest allowable sender window that will avoid the occurrence of problems such as that in slide 58 (or Figure 3.27 in textbook) for each of these protocols? Justify your answer.

Solution: In order to avoid the scenario of Figure 3.27 (of textbook), we want to avoid having the leading edge of the receiver's window (i.e., the one with the "highest" sequence number) wrap around in the sequence number space and overlap with the trailing edge (the one with the "lowest" sequence number in the sender's window). That is, the sequence number space must be large enough to fit the entire receiver window and the entire sender window without this overlap condition. So, we need to determine how large a range of sequence numbers can be covered at any given time by the receiver and sender windows.
Suppose that the lowest-sequence number that the receiver is waiting for is packet m. In this case, it's window is $[m, m + w - 1]$ and it has received (and ACKed) packet m-1 and the w-1 packets before that, where w is the size of the window. If none of those w ACKs have been yet received by the sender, then ACK messages with values of [m-w,m-1] may still be propagating back. If no ACKs with these ACK numbers have been received by the sender, then the sender's window would be $[m - w, m - 1]$. Thus, the lower edge of the sender's window is m-w, and the leading edge of the receivers window is m+w-1. In order for the leading edge of the receiver's window to not overlap with the trailing edge of the sender's window, the sequence number space must thus be big enough to accommodate 2w sequence numbers. That is, the sequence number space must be at least twice as large as the window size, $k \geq 2w$.

(II) What features of GBN and SR are used by TCP? How does TCP get benefited by combining GBN and SR?

Solution: TCP uses cumulative acknowledgement (a feature of GBN) and a receiver buffer (a feature of SR). Cumulative acknowledgement reduces the overhead in sender as it eliminates the necessity of multiple timers. Receiver buffer reduces number of retransmissions as the receiver is likely to deliver a lot of packets from its buffer once any gap is filled.

(III) Why does TCP use three-way handshake? Discuss scenarios when two-way handshake fails.

Solution: TCP is a bi-directional protocol, thus both the sender and the receiver need to agree with each other's initial sequence number. Given that any message may get lost or delayed in channel, a three-way handshake is necessary to ensure that both the sender and the receiver ACKed each other's initial sequence number before transmitting any data.
A failure scenario of two-way handshake is when the ACCEPT message from server gets delayed in channel. Suppose the client has already resent the connection request after a timeout and then received the delayed ACCEPT message from server. Now if the second connection request from client gets delayed in channel, it may end up with a half-open connection in server, when client already finished communication and closed its connection (check slide 83 of Chapter 3).

Question 3

[15 points] In this problem, you will write a program to implement a vertical TCP port scanner. Your task is to implement a client that attempts to connect to a server sequentially using port numbers from a given range. For each port number, use TCP socket to test whether the port is open (i.e. TCP socket connection is successful) or closed (i.e., TCP socket connection is not successful).

Set target IP/hostname to www.uiowa.edu and use ports in the range [0 - 100]. Your program should output the list of open ports.

While running this program, monitor packet traces using Wireshark (www.wireshark.org). Conduct a forensic analysis of the Wireshark packet trace collected during your port scan. Explain the differences you observe for open vs. closed ports.

Question 4

[50 points] This question is designed to help students delve into the principles of reliable data transfer. To get you started, you are given a simulation of the Stop-and-Wait (rdt 3.0) protocol in `Stop_and_Wait_sim.py`. Notice that the simulation parameters are defined as global variables at the beginning of the code.

You are also given an image file named `Hawkeyes.JPG` (Go Hawks!). You must put the image file in the same directory where you put the simulation code. The simulation contains a sender, a receiver and a channel. As you can expect, the channel connects the sender with the receiver. The sender in the simulation reads the image file, splits the file data into segments, creates packet for each segment, and forwards packets to the channel to send to the receiver. The receiver receives packets from channel, extracts segments from packets, delivers segments to the application layer, and forwards ACKs to the channel. The channel then delivers the ACKs to the sender.

To make the channel realistic, we simulate loss and delay of packets/ACKs in the channel. To this end, the channel drops some packets/ACKs with a certain probability. The channel also delays to deliver some packets/ACKs with a certain probability. You can control the loss and delay probabilities by changing the values assigned to the global variables `lossProb` and `delayProb`, respectively.

The receiver reassembles the segments received and reconstructs the image file. If all the segments were delivered to the application layer in the correct order, the reconstructed image would be an exact copy of the original image. Check both the images to check the correctness of the protocol. The simulation also reports if a segment in receiver does not match with the corresponding segment in sender. Check the simulation code carefully to understand how this simulation is done. Then complete the following tasks.

Task 1. [6 points] Run the simulation for `delayProb = 0`, `lossProb = 20`, and `seqNSpaceSize = 2`. You should observe that the reconstructed file is an exact copy of the original file. Note that setting the size of the sequence number space to 2 essentially

makes the protocol a bit-altering (0 or 1 as sequence/ACK numbers) protocol. Now set `delayProb = 30`, and run the simulation again. You will most likely find that the image reconstructed by the receiver is distorted. In this case, the simulation will report which segments in receiver do not match with the corresponding segments in sender. Examine the generated log file to find out what went wrong. Is it a bug in the simulation code, or a limitation of rdt3.0? Justify your answer with a timing diagram. The timing diagram must show the sender on left and receiver on right. Also, the diagram must show all the packets/ACKs with their sequence numbers, and any loss or delay of packets/ACKs.

Solution: Check Figure 2.

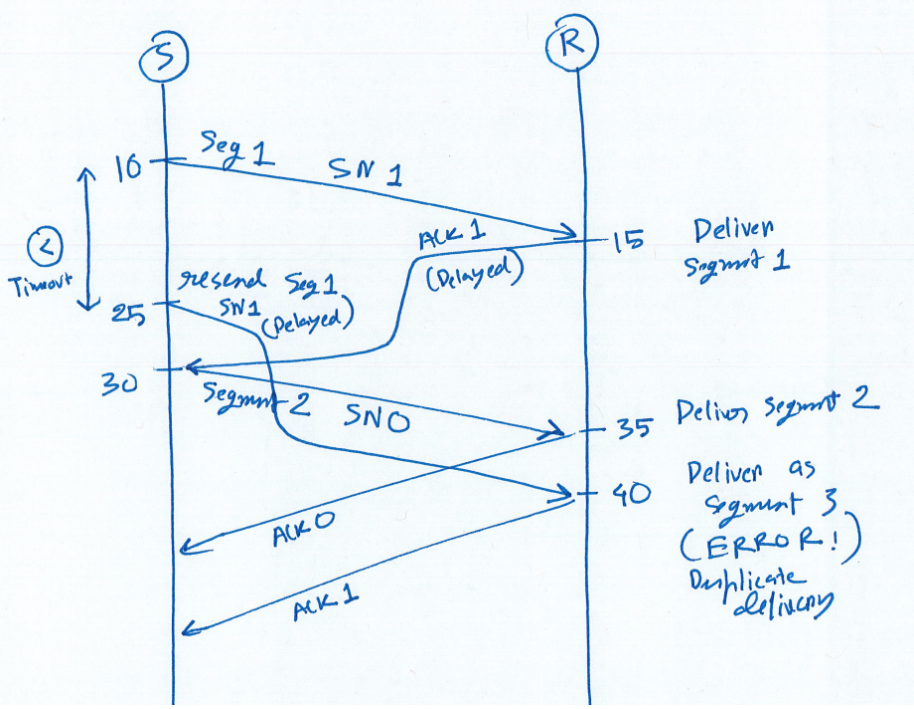

Figure 2: Timing diagram for Question 4 Task 1.

Task 2. [2 points] Increment the size of the sequence number space to 3, 4, 5 etc. Does this solve the problem found in task 1? What is the lowest value of the size of the sequence number space that solves the problem? Explain why this number solves the problem. Correct the simulation according to your findings.

Solution: Figure 3 shows a situation with 3 as the size of the sequence number space. Segment 31 is dropped by the channel and never retransmitted by the sender as the sender received a delayed ACK for a previous packet. Clearly, for the given environment (i.e. RTT, delay etc.), this can be avoided if the size of SN space is at least 4.

Task 3. [18 points] Extend the simulation to simulate the Go-Back-N (GBN) protocol.

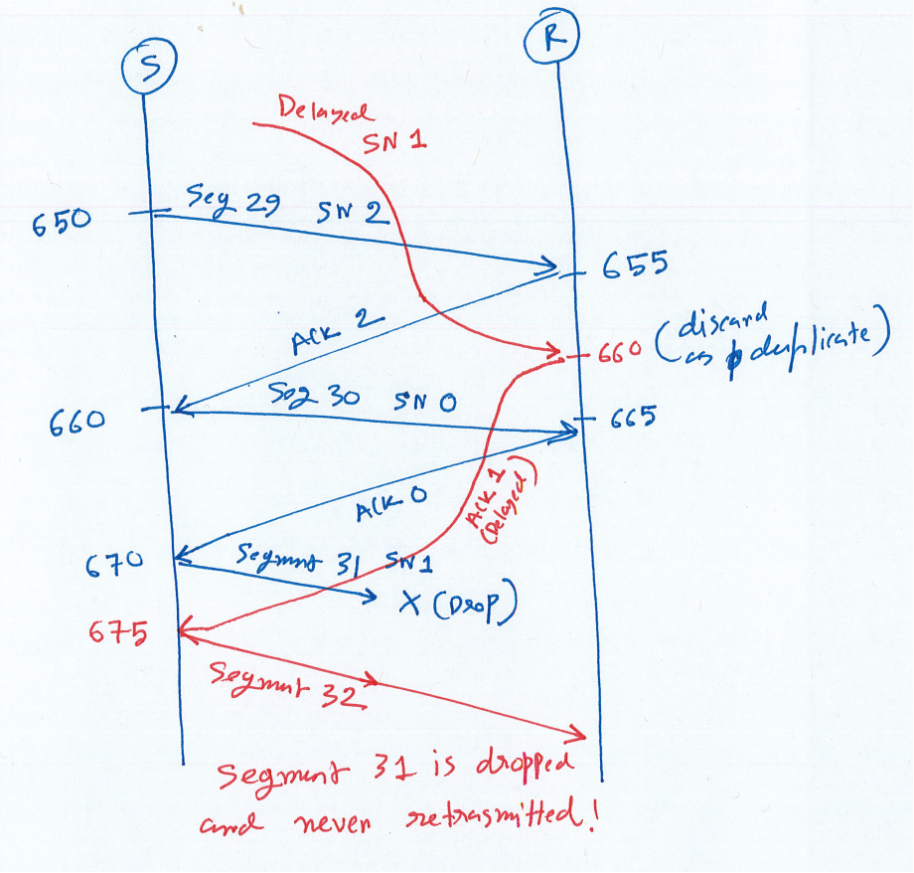

Figure 3: Timing diagram for Question 4 Task 2.

Note that we used 10 simulation ticks for the RTT, and 15 simulation ticks for the timeout period in our simulation of the rdt3.0 protocol. Keep these values unchanged for your simulation of GBN. Given that only one packet can be transmitted by the sender in a simulation tick, what should the ideal window size? Implement your simulation with the window size you think is ideal. Don't forget to change the size of the sequence number space accordingly.

Task 4. [18 points] Extend the simulation to simulate the Selective Repeat (SR) protocol. Use the same RTT, timeout period, and window size that you used for simulating GBN.

Task 5. [6 points] Plot total simulation times associated with all three protocols (i.e. your corrected rdt3.0, GBN and SR) for the following values:

1. 0%, 10%, 20% and 30% packet loss probabilities with a fixed packet delay probability of 10%.

2. 0%, 10%, 20% and 30% packet delay probabilities with a fixed packet loss probability of 10%.

3. 64, 256, 512, and 1024 bytes MSS.

Note that, the image reconstructed by the receiver should be an exact copy of the original image for all simulation runs.

Finally, give a brief summary of the simulation results. Explain which protocol works better for which situations according to the simulation results.

Submission: You should submit three separate source files for three protocols. For rdt3.0, you need to submit the corrected simulation (with your modified value for the size of the sequence number space). You should also submit three plots and a summary of the simulation results.

Question 5

[9 points] Suppose an organization has the following subnets:

Subnet	Number of Hosts
Development Department	145
Marketing Department	75
Admin Department	50
Accounting Department	12
WAN link 1	2
WAN link 2	2
WAN link 3	2
WAN link 4	2

Suppose the network address for the major network (i.e. the entire organization) is 172.68.0.0/x. Our goal for this problem is to use VLSM to minimize the number wasted IP addresses. Keeping this goal in mind, answer the following questions.

1. What should be the value of x? **Solution:** 23

2. What is the subnet mask for the major network? **Solution:** 255.255.254.0

3. What is the broadcast address for the major network? **Solution:** 172.68.1.255

4. Give subnet address, subnet mask, broadcast address, and address range of the valid hosts for *each* of the subnets in the organization.

   Solution:

   Development Department:

Subnet address: 172.68.0.0/24
Subnet mask: 255.255.255.0
Broadcast address: 172.68.0.255
Address range: 172.68.0.1 to 172.68.0.254

Marketing Department:
Subnet address: 172.68.1.0/25
Subnet mask: 255.255.255.128
Broadcast address: 172.68.1.127
Address range: 172.68.1.1 to 172.68.1.126

Admin Department:
Subnet address: 172.68.1.128/26
Subnet mask: 255.255.255.192
Broadcast address: 172.68.1.191
Address range: 172.68.1.129 to 172.68.0.190

Accounting Department:
Subnet address: 172.68.1.192/28
Subnet mask: 255.255.255.240
Broadcast address: 172.68.1.207
Address range: 172.68.1.193 to 172.68.1.206

WAN link 1:
Subnet address: 172.68.1.208/30
Subnet mask: 255.255.255.252
Broadcast address: 172.68.1.211
Address range: 172.68.1.209 to 172.68.1.210

WAN link 2:
Subnet address: 172.68.1.212/30
Subnet mask: 255.255.255.252
Broadcast address: 172.68.1.215
Address range: 172.68.1.213 to 172.68.1.214

WAN link 3:
Subnet address: 172.68.1.216/30
Subnet mask: 255.255.255.252
Broadcast address: 172.68.1.219
Address range: 172.68.1.217 to 172.68.1.218

WAN link 4:
Subnet address: 172.68.1.220/30
Subnet mask: 255.255.255.252
Broadcast address: 172.68.1.223
Address range: 172.68.1.221 to 172.68.1.222

5. How many IP addresses are available for further subnetting?
   Solution: 254-223 = 31