Homework 2

Due: Oct 4, 10:30am in the classroom

1. Is it possible for an application to enjoy reliable data transfer even when the application runs over UDP? If so, How? Yes, but the application layer must implement the error checking
2. Suppose a process in Host C has a UDP socket with port number 6789. Suppose both Host A and Host B each send a UDP segment to Host C with destination port number 7780. Will both of these segments be directed to the same socket at Host C? If so, how will the process at Host C know that these two segments originated from two different hosts? The socket will not receive either of the UDP segments.
3. Why were sequence numbers introduced into our rdt protocols? To know if a packet contains new information or is a repeat.
4. Why were timers introduced into our rdt protocols? To deal with packet loss
5. Suppose that the roundtrip delay between sender and receiver is constant and known to the sender. Would a timer still be necessary in protocol rdt3.0, assuming that packets can be lost? Explain. Yes, a packet still need to be sent after the amount of time has passed with no response.
6. Consider the Go-Back-N protocol with a window size of 5.
   * 1. Describe what happens if 5 packets are sent, and the first packet is lost before any packets reach their destination.
        1. No Packets are accepted and the timer will expire at the sender, triggering the resend of the packets
     2. Describe what happens if 5 packets are sent, the first packet reaches its destination, and the first acknowledgement is lost.
        1. More packets will be sent, and if received, their acks will be sent
     3. Describe what happens if a 6th packet is sent before the first acknowledgement is received.
        1. With a window size 5, this wouldn’t happen
7. Repeat the previous question with the Selective Repeat protocol. How are Selective Repeat and Go-Back-N different?
   1. The first packet will be resent
   2. The first packet will be resent
   3. This would not happen with a window of 5
8. True or False. If false, explain.
   * 1. Host A is sending Host B a large file over a TCP connection. Assume Host B has no data to send Host A. Host B will not send acknowledgements to Host A because Host B cannot piggyback the acknowledgements on data.
        1. False, B will send acks
     2. The size of the TCP rwnd never changes throughout the duration of the connection.
        1. False; receiver window gets bigger as more packets are received successfully
     3. Suppose Host A is sending Host B a large file over a TCP connection. The number of unacknowledged bytes that A sends cannot exceed the size of the receive buffer.
        1. True;
     4. Suppose Host A is sending a large file to Host B over a TCP connection. If the sequence number for a segment of this connection is *m*, then the sequence number for the subsequent segment will be *m* + 1.
        1. False; the segment number is the count of bytes in the current segment plus the previous
     5. The TCP segment has a field in its header for rwnd.
        1. True
     6. Suppose that the last SampleRTT in a TCP connection is equal to 1 second. The current value of TimeoutInterval for the connection will be ≥ 1 second.
        1. False. Next estimated = alpha \* last estimated + (1-alpha)\*last actual.
     7. Suppose Host A sends one segment with sequence number 38 and 4 bytes of data over a TCP connection to Host B. In this same segment the acknowledgement number is 38.
        1. False; the ack # has nothing to do with the current sequence #
9. Suppose Host A sends two TCP segments back to back to Host B over a TCP connection. The first segment has a sequence number 71; the second has a sequence number 131.
   1. How much data is in the first segment? 60 bytes
   2. Suppose that the first segment is lost but the second segment arrives at B. In the acknowledgement that Host B sends to Host A, what will be the acknowledgement number? 71
10. Suppose Client A initiates a Telnet session with Server S. At About the same time, Client B also initiates a Telnet session with Server S. Provide possible source and destination port numbers for the segments sent from:
    1. A to S: src: 5000 dest: 23
    2. B to S: src: 5000 dest: 23
    3. S to A: src: 23 dest: 5000
    4. S to B : src 23 dest: 5000
    5. If A and B are different hosts, is it possible that the source port number in the segments from A to S is the same as that from B to S? Yes
    6. How about if they are the same host? No
11. Suppose Host A, Host B, and Server S have respective IP addresses A, B, and S. There is a process on Host A that has a segment flow to Server S with source port 8000 and destination port 81. Furthermore, there are two processes on Host B with two corresponding segment flows to Server S with source ports 9005 and 9006 and destination port 81. What are the source and destination ports of the response segment flows from Server S to the three processes? Additionally, what are the source and destination IP addresses in the network-layer datagrams carrying the transport-layer segments.
    1. S->A: src-81, dest-8000, SrcIP-S, DestIP-A; S->B: src-81, dest-9005, SrcIP-S, DestIP-B; S->B: src-81, dest-9006, SrcIP-S, DestIP-B;
12. Using the rdt3.0 protocol, draw message traces where:
    1. The 3rd of 5 total data packets is garbled.
    2. The 3rd of 5 total acknowledgements is garbled.
13. Consider rdt3.0. Draw a diagram showing that, if a network reorders messages between a sender and receiver, the alternating bit protocol fails. Clearly indicate sequence numbers associated with packets and acknowledgements, and point out where the protocol fails.
14. Consider the Go-Back-N and Selective Repeat protocols. Suppose that the sequence number space is size *k*. What can go wrong if our sender window becomes too large? What is the largest allowable sender window to avoid these problems?
    1. We could send more information than the receive buffer could hold, resulting in loss of data. The largest sender window is k/2
15. Host A and B are communicating over a TCP connection, and Host B has already received from A all bytes up through byte 120. Suppose Host A then sends two segments to Host B back-to-back. The first and second segments contain 20 and 50 bytes of data, respectively. In the first segment, the sequence number is 121, the source port number is 9001, and the destination port number is 80. Host B sends an acknowledgement whenever it receives a segment from Host A.
    1. In the second segment sent from Host A to B, what are the sequence number, source port number, and destination port number? 141, 80, 9001 respectively
    2. If the first segment arrives before the second segment, in the acknowledgement of the first arriving segment, what is the acknowledgement number, the source port number, and the destination port number? 142, 80, 9001 respectively
    3. If the second segment arrives before the first segment, in the acknowledgement of the first arriving segment, what is the acknowledgement number? 121
    4. Suppose the two segments sent by A arrive in order at B. The first acknowledgement is lost and the second acknowledgement arrives after the first timeout interval. Draw a timing diagram, showing these segments and all other segments and acknowledgements sent. Provide sequence numbers and the number of bytes of data for each segment. Provide the acknowledgement number for each acknowledgement.

1. Host A and B are directly connected with a 100 Mbps link. There is one TCP connection between the two hosts, and Host A is sending to Host B an enormous file over this connection. Host A can send its application data into its TCP socket at a rate as high as 120 Mbps but Host B can read out of its TCP receive buffer at a maximum rate of 50 Mbps. Describe the effect of TCP flow control.
   1. Max flow rate is limited by the slowest portion, 50Mbps
2. Compare Go-Back-N, Selective Repeat, and TCP (no delayed ACK). Assume that the timeout values for all three protocols are sufficiently long such that 5 consecutive data segments and their corresponding ACKs can be received (if not lost by the channel) by the Host B and Host A respectively. Suppose Host A sends 5 data segments to Host B, and the 3rd segment sent from A is lost. In the end, all 5 data segments have been correctly received by Host B.
   1. How many segments has Host A sent in total and how many ACKs has Host B sent in total? What are their sequence numbers? Answer this for all three protocols. GBN{sent: 8, acks: 7, seq: 1, 2, 3, 4, 5, 3, 4, 5} SR{ sent: 6, acks: 5, seq: 1, 2, 3, 4, 5, 3}
   2. If the timeout values for all three protocols are much longer than 5 RTT, then which protocol successfully delivers all five data segments in the shortest time interval? Selective Repeat
3. Below is the complete code for a UDP ping server written in Python. Your task is to write the corresponding UDP ping client in Python. Your client will send a simple ping message to a server, receive a corresponding pong message back from the server, and determine the delay between when the client sent the ping message and received the pong message. The client code will be similar to the server code, so study the server code first.

The client code should do the following.

For 10 ping messages:

* 1. Send the ping message using UDP to the server.
  2. Wait up to one second for a response.

(2) If the server responds:

* + - 1. Print the response message from the server
      2. Calculate and print the RTT in seconds.

(4) Else

(a) Print “Request timed out”

# UDPPingerServer.py

# We will need the following module to generate randomized lost packets import random from socket import \*

# Create a UDP socket

# Notice the use of SOCK\_DGRAM for UDP packets serverSocket = socket(AF\_INET, SOCK\_DGRAM)

# Assign IP address and port number to socket serverSocket.bind(('', 12000))

while True:

# Generate random number in the range of 0 to 10 rand = random.randint(0, 10)

# Receive the client packet along with the address it is coming from message, address = serverSocket.recvfrom(1024) # Capitalize the message from the client message = message.upper()

# If rand is less is than 4, we consider the packet lost and do not respond if rand < 4: continue

# Otherwise, the server responds

serverSocket.sendto(message, address)

import time

from socket import \*

for pings in range(10):

clientSocket = socket(AF\_INET, SOCK\_DGRAM)

clientSocket.settimeout(1)

message = 'test'

addr = ("127.0.0.1", 12000)

start = time.time()

clientSocket.sendto(message, addr)

try:

data, server = clientSocket.recvfrom(1024)

end = time.time()

elapsed = end - start

print '%s %d %d' % (data, pings, elapsed)

except timeout:

print 'REQUEST TIMED OUT'