CS224M HW-3 (Socket Programming)

Due date: 12th October 2022

Objective: Build ARQ on top of UDP Sockets

Write two "C" socket programs, sender.c and receiver.c, that communicate together using "Datagram Sockets (UDP)". You can learn socket programming basics from Beej's Guide to Network Programming to implement this assignment. You can also see simple examples of socket programs in your textbook Peterson and Davie, Sec 1.4. You may program in C++ and in that case use "cpp" extensions for your program.

The sender and receiver must both be on the same machine. The programs should take command line arguments as shown below.

```
sender.c <SenderPort> <ReceiverPort> <RetransmissionTimer> <NoOfPacketsToBeSent>

receiver.c <ReceiverPort> <SenderPort> <PacketDropProbability>
```

The loopback device is a special, virtual network interface that your computer uses to communicate with itself. Since both sender and receiver are on your machine, they use this interface to communicate. We want to emulate network delays between your sender and receiver and so will use the `tc` command to add network delays to the loopback interface as follows. We are assuming that the loopback interface is "lo", but you should use ifconfig to get the right loopback for your machine.

```
sudo tc qdisc add dev lo root netem delay <Delay_in_miliseconds>
```

There is a retransmission timer used by the sender. This can be implemented using the time.h library or sleep system calls, or any other method. Timer value given will be in seconds. Please note that the retransmission timer should be at least twice the emulated network delay.

You can check out this lecture to get a brief understanding of ARQ. You can further refer to this chapter to get an outline of the Stop-and-Wait algorithm explained below.

1. **Sender** : Sender gets a command line argument for the number of packets to be sent, say P. Each packet generated by the sender must contain a Sequence number which identifies the packet. The Sequence number must start with 1 for the very first packet sent and then get incremented for each subsequent packet. The sender sends the packet to the receiver and starts a **retransmission timer**. Let the sequence number in this packet be x.

   (a) If the retransmission timer expires and there is no acknowledgment from the receiver then the sender will retransmit the same packet.

   (b) If the sender receives an acknowledgment before the retransmission timer expires and the acknowledgement contains x+1 as its sequence number then the sender transmits the packet with sequence number x+1.
   (c) If there is an acknowledgment from the receiver containing a sequence number other than x+1 then the sender has to ignore the acknowledgement.

   Packet sent is a string. It should be in the format mentioned below.
   Packet Format : "Packet:<sequenceno>"
   Note: The above packet format is at the Application Layer. You must write data to the socket in this format. Headers at lower layers are taken care of by the OS, the NIC card etc.

2. **Receiver** : Receiver responds to the packet received from sender with an acknowledgement that it has received the packet. The steps followed by the receiver (each time it receives a packet) are:
   (a) Receive the packet and check if the packet with correct sequence number is received (the receiver expects the very first packet to have sequence number 1).
   (b) If the sequence no. is incorrect, an acknowledgment is sent with the expected packet sequence number (which is x+1, where x is the sequence no. of the last packet received from the sender).
   (c) If the sequence no. is correct, the following steps are done:
      i. A uniformly distributed random variable is generated between 0 and 1. If this number is less than the PacketDropProbability then no acknowledgement is generated. The receiver will wait for the sender to retransmit the packet with the same Sequence number.
      ii. Otherwise, an acknowledgement will be sent to the sender which contains the sequence no. of the packet that the receiver is expecting next (format mentioned below).

   The acknowledgment generated by the receiver must contain the Sequence number that the receiver expects from sender. Suppose the packet received has sequence number x, then the acknowledgement (if it is generated) should contain sequence number x+1.
   Acknowledgment sent is a string. It should be in the format mentioned below.
   Acknowledgment Format:"Acknowledgment:<sequenceno>"

The sender should print to the terminal every packet it sends and also when a packet's retransmission timer expires. Similarly, the receiver should print to the terminal every acknowledgement that it sends and also when it drops a packet. The sender and the receiver must also write these to separate text files. The text files must be named sender.txt and receiver.txt respectively.

NOTE: You can use code from external sources for socket creation and binding. If you do decide to use external code, you must mention the source in the code comments. The rest of the code MUST BE YOUR OWN. Additionally, in total, less than 30% of the total code should be external code.

Submission Instructions:

1. The assignment should be done in a group consisting of a maximum of **two** students.

2. The code should follow programming etiquette with appropriate comments.
3. Add a **README** file which includes a description of the code and gives detailed steps to compile and run the code.
4. Zip all your code and the README as a single file with the name **rollno1-rollno2.tar.gz** and upload it to Moodle. Only one group member should upload the file.