CSE 232 Section B, Computer Networks; Programming Assignment 3: Simulation of GoBack N protocol

a. Due FRI Nov 8, 2024, midnight
b. To be done in groups of 2
c. The institute plagiarism policy applies,
d. You may program in C, C++ or Python
e. Platform: Linux OS
f. Grading will be done based on submitted material, plus face-to-face interaction with me & TAs,
g. The submission will consist of 3 parts, suitable zipped with a name that contains the roll nos. of both students:
   1. 2 to 4 page document describing the system you have designed (including all assumptions you have made),
   2. the code(s) as separate file(s), and
   3. 5 to 8 slides that you will use to present your work before me and TAs. This must include screenshots from execution of code for the two data link entities.

This programming assignment is to help you gain good understanding of data link protocols by implementing the GoBack N protocol. This protocol will be (or was) discussed in class. You will of course use the experience gained while implementing the simple application "pinger" that uses underlying UDP protocol at the Transport layer. NOTE: you are simulating the data link protocol but using an available Transport layer to send and receive messages in both directions between DL_Entity_1 and DL_Entity_2.

See also the diagram given below.

In order for you to simulate (or implement) the GoBack-N protocol, you will ensure that:

From viewpoint of data link protocol:

1. Packets are generated by both Network_Entity_1 and Network_Entity_2, but are sent /received using the underlying data link layer. Packets will be generated at random time instants, the gap between two successive packets being uniformly distributed between T1 and T2 (you decide value of T1, T2). Once packets are generated, they are added to an outgoing queue. See the discussion of this in Tanenbaum's book.
2. Data link entity, DL_Entity_1 (and similarly DL_Entity_2) picks a packet from the queue whenever it is ready to send, suitably encapsulates it into a frame and sends it as per GoBack-N protocol. Clearly, sequence no. and Ack no. etc. are added as part of encapsulation. At the receiver end DL_Entity_2 (and similarly DL_Entity_1) receives the frame, checks the sequence no. and Ack no. to see if they are in order, and depending upon the checks it deliver the packet contained therein to the Network_Entity_2. It then sends to DL_entity_1 a frame that contains an acknowledgement as per the GoBack N protocol.
3. Use modulo-8 sequence numbering, with a transmit-window size of 7, receiver-window size of 1. That is N=8.

From viewpoint of client-server model that data link entities use:

4. You will create two "datagram" sockets (one on each machine). These will be used to simulate/facilitate movement of frames between the data link entities. The "client" on machine 1 will act as a DL_entity_1, while the "server" on machine 2 will act as DL_entity_2.
5. Force a datagram (that simulates a frame containing a packet & Ack, or simply an Ack) to be dropped with some probability P. You decide what that probability should be so that a few frames are indeed dropped, and are re-transmitted as per the GoBack N protocol.
6. Introduce a certain delay EITHER at the transmitter end or at the receiver end (or both) so as to simulate queuing and propagation delay – what should that be, you decide. BUT, this is random, and

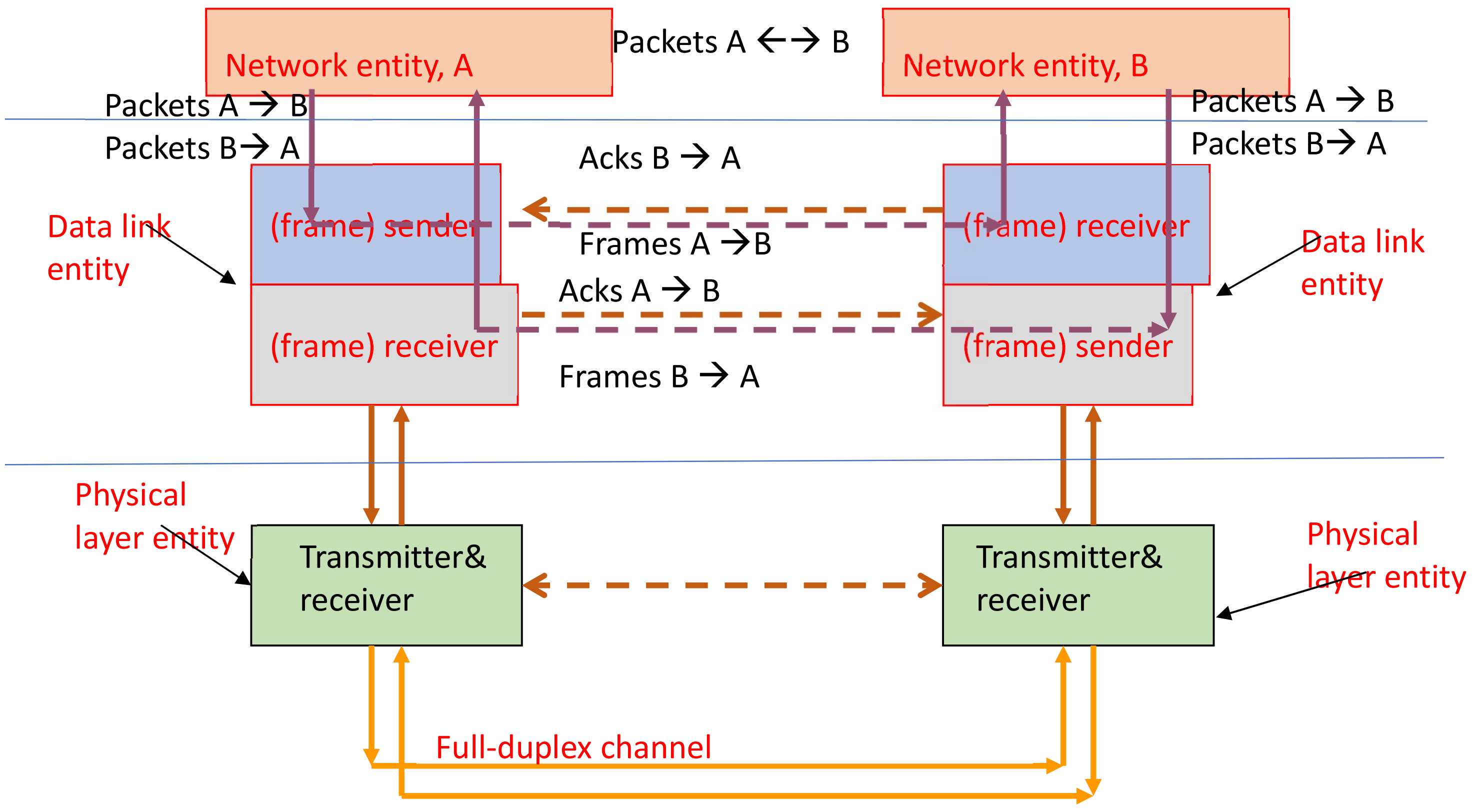
uniformly distributed between T3 and T4. This is over and above the delay experienced by datagrams sent from client to server, or vice-versa.

Questions to be asked and answered by you:

1. How would you simulate the Network entity as a process that produces packets at random time instants? And similarly simulate a network entity that consumes incoming packets that have been added to a queue of incoming packets? Are threads an option?

You are required to:

- **Simulate transfer of 10,000 packets in each direction with assumed values of T1, T2, T3, T4, probability P, and**
  - **a. observe and record the times that correspond to the time when a frame was first sent, and the time when the frame was first received without error and in order at the other end,**
  - **b. compute:**
    - **I. the average delay in delivering packets to the other end, and**
    - **II. average number of times a frame was sent.**
- **Repeat the above for (keep T1 and T2 remain the same):**
  - **I. two significantly different values of drop probability, P, and**
  - **II. two different values of delay, viz. pair of T3 & T4.**


Network entity, A
Packets A ←→ B
Network entity, B
Packets A → B
Packets B→ A
Packets A → B
Packets B→ A
Acks B → A
Data link entity
(frame) sender
(frame) receiver
Frames A →B
Acks A → B
Data link entity
(frame) receiver
(frame) sender
Frames B → A
Physical layer entity
Transmitter& receiver
Transmitter& receiver
Physical layer entity
Full-duplex channel