Assignment 3, CS641, Due Date: Sunday, Oct 28 2018, 5pm

Zip all your code as well as report as a single file and upload to Moodle.

In this assignment you must built a simplified version of TCP as an application over UDP. Here are the basic features the protocol must have.

SENDER SIDE DETAILS

- The sender maintains a variable `Window` which maintains the window size in packets. All packets transmitted are UDP packets with payload (excluding headers) size 1000 bytes. `Window` is initialized to 1. The sender can have `floor(Window)` number of packets in-flight, that is, those many packets can be transmitted beyond the largest ACK number received. Note that `floor(x)` is the largest interger less than or equal to `x`.
- It maintains an array `status`. Note that `status[k]` should be 0 if packet number k has not been transmitted, 1 if the packet has been transmitted, and 2 if the packet has been ACKed.
- Every packet contains a field representing the packet number. Packets are numbered starting from 0.
- It maintains an array `sentTime` containing the most recent time-stamp at which various packets were sent out. Every time packet number k is re-transmitted, this array should be updated.
- A variable called `timeout`. This variable contains the time in milliseconds after which a packet can be declared lost. It can be initialized to 2000 milliseconds and then can be updated based on measured RTTs using any algorithm of your choice.
- Runs a periodic timer which times-out every 20 milliseconds. When this timer times-out then the sender checks to find out if any packets with packet number in range $[ackNum + 1, \ldots, ackNum + floor(Window)]$ have timed-out, that is if time exceeding `timeout` has elapsed since the packet was sent out. If any packet has timed out, then `Window` is reset to 1 and that packet is retransmitted. Note that $ackNum$ is the largest ACK number seen so far.
- When a cumulative ACK arrives from the receiver, the sender updates the `status` array. Every time an ACK arrives, `Window` is incremented by `1/Window`.
- The sender program accepts a user specified input which gives a probability of "packet loss" to be simulated. Call this input `lossProb`. For example, if `lossProb=0.01` then the sender simulates a packet drop for every packet with that probability. It throws a random variable for every packet to decide if the packet is to be deemed "lost" or not. If the packet is deemed to be lost, then it does not transmit the packet but still updates `status` like as if it has transmitted the packet.
- The sender maintains log files recording the time-stamp for events such as packet sending, loss detection events, ack reception events etc.

- The sender sends only 10000 packets out and then quits when all are ACKed.
- The sender can have any other variables required for implementing the above protocol.

RECEIVER SIDE DETAILS

- The receiver maintains an array `recvStatus` indicating which packets have been received.
- On receiving packet number k, it updates `recvStatus[k]`. It then send a small packet as an ACK to the sender. The ACK contains m, where m is the cumulative ACK number, which represents the largest packet number such that all packets with number less than or equal to it have been received.
- The receiver logs time-stamps of packet reception events, the ACK numbers sent to the sender etc.

Write a report which contains plots of `Window` vs. time for different specified values of `lossProb`. Also include corresponding plots of "packet number" and "ACK number" versus time from logs collected at the sender. Include any interesting obeservations you make.