Assignment 3

COL 344/672

Due date: November 11, 2016, 23:55hours (Friday)

Note: *You can work in teams of 3 or less to solve this assignment.* Clarify doubts on Piazza. Upload your socket programming code to Moodle. Mention the entry numbers of all team members in the file name of the zipped file uploaded. Only one team member needs to upload the zipped file.

1. Write two socket programs, sender.c and receiver.c, that together communicate using "Datagram Sockets". The two programs should run on different machines. The sender.c program should take as a command line input the IP address of the machine on which receiver.c is running and the port number on which receiver.c is listening. [In this assignment question, I am assuming that C is the programming language used. However, you are allowed to program in any other language besides C, and use appropriate names for the programs.]

   We will build some features of TCP such as sliding window, congestion control, and packet retransmissions on top of these two programs which use UDP. Assume that maximum segment size MSS=1000 bytes. The "size of data" carried in each packet (that is, the total IP packet size minus IP header, UDP header, and any control fields you insert in the packet) must be less than or equal to this size.

   We will refer to all data transmitted from sender to receiver as a "flow". Assume that the flow consists of 100,000 bytes of data. Although a real TCP flow could consist of data from a file etc., in this assignment we will just transmit some dummy data. Assume that the bytes of the flow are numbered from 0 to 99,999.

   Each packet (datagram) from sender to receiver must contain a control field "sequence number" representing the byte number of the first byte of the packet relative to the first byte in the flow, and another field indicating the "size of data" in the packet. The first packet should have sequence number 0 and size of data equal to MSS.

   The receiver maintains a parameter called cumulative ACK number (x) which is initialised to 0. If this parameter has value x, it essentially means that *all bytes* from 0 to $x-1$ of the flow were received successfully. The receiver, on getting any data packet, replies with a small ACK packet containing only the current cumulative ACK number. For example, on receiving the first packet the receiver sets $x = MSS$ and then sends an ACK. Note that implementing this is a bit tricky because data packets being received are not all of the same size, and some packets may get lost etc. For example, if $x = 10,000$ and a packet with sequence number $11,000$ arrives then the receiver sends back an ACK with cumulative ACK number equal to $10,000$.

   The sender maintains a window parameter W in bytes. The sender is allowed to transmit *at most* W bytes beyond the largest received cumulative ACK number to the receiver. Suppose an ACK containing cumulative ACK of x is received at the sender and $W = 5,900$ bytes. Then the sender is allowed to send 5 packets each of size MSS to the receiver and one packet of size 900 bytes, containing appropriate sequence numbers (x, $x+1000$, $x+2000$, $x+3000$, $x+4000$, $x+5000$ respectively) and size of data fields (1000, 1000, 1000, 1000, 1000, 900 respectively).

The window W is initialised to 1 MSS. Once a cumulative ACK of 100,000 is received, the sender terminates. Every time a packet is sent out, a timer of 1 second corresponding to that packet is started. In case a timer for any packet expires, it is retransmitted with the same fields it contained originally.

If a timer expires for any packet we assume that the packet was lost due to congestion. We hence reduce the window size by setting $W = MSS$ and transmit one packet of size MSS with sequence number equal to the largest ACK number observed so far. All other timers are cancelled

On receiving an ACK packet, the sender increases W by $\lfloor (MSS)^2/W \rfloor$ bytes and cancels the timers for all packets ACKed so far. That is, if cumulative ACK received is x then the timers for all packets which had (sequence number + size of data - 1 < x) are cancelled.

At the sender, print out the values of W, time expired since the start of the experiment, and sequence number, every time a packet is transmitted. At the sender, print out the same quantities whenever an ACK arrives, except that instead of a sequence number it prints the cumulative ACK number.

Now let us simulate packet loss. Write sender.c to take as command line input (for example, with the -l flag) a "loss parameter" which is either 0 or 1. If the value input is 0 then sender.c behaves as described above. If the value is 1, then each packet is dropped at the sender itself with probability 0.05. This means that if the packet is to be dropped, then the sender does not actually transmit the packet to the receiver but performs all other operations, such as starting a timer for the packet etc.

Comment your socket programming code well. Upload to moodle a zip file containing (i) README file describing how to compile and use the programs, (ii) code for sender and (ii) code for receiver.