Department of Computer Science and Information Systems, BITS Pilani
First Semester 2019-2020
Advanced Computer Networks (CS G525)
ASSIGNMENT #1

Date of posting: 15-08-2019	**Submission Deadline: 21-08-2019(Mid-Night)**
Maximum Marks: 7	**Mode: Individual**

Problem Statement: Implement a modified Go-Back-N ARQ protocol (described below) using UDP sockets.

Learning Objectives: This assignment problem will help you to tune up your programming skills. Particularly, you are going to learn TCP/UDP socket programming concepts using UNIX libraries.

Programming Language to be used: C/C++ (gcc compiler)

Deliverables/Submission: Upload a Zip of three files: client.c, server.c and readme.txt on NALANDA

Modified Go-Back-N ARQ Protocol Description:

The **modified Go-Back-N ARQ** would work simply as an extended stop-and-wait ARQ protocol, where instead of transmitting a single packet and waiting for its acknowledgment, i.e., ACK, the sender would transmit certain number of packets equal to the size of window (*WS*), and then, will wait for *WS* number of ACKs. On receipt of *WS* ACKs against the transmitted packets in a window, the sender can transmit another *WS* packets. You can refer the timing diagram shown in Fig. 1 to understand the working of **modified Go-Back-N ARQ** protocol. A base pointer (Base) keeps track of the packet number of the next packet to be transmitted in the subsequent rounds.

What needs to be done?

You have to implement a client and a server program using DATAGRAM sockets. The client sends a file (e.g., infile.txt) to the server using the **modified Go-Back-N ARQ** protocol. The client breaks the file into equal-sized packets (pkt_size) and transmit to the server. The packets received at server are first copied into a buffer and then written to a file. (Note: It is your choice to either write one packet at a time or all the packets at once to the file.). The packets would reach immediately without fail (you can assume no packet loss...!!!) to the server as both the programs (server and client) would either run on the same host machine or in same LAN. Therefore, you can consider to introduce

a dummy time delay at server before sending ACK packet corresponding to a DATA packet to get the output in the desired format.

Packet Format: You have to use a uniform packet format for data packets and ACK packets. Typical fields in a packet structure can be are; ***pkt_type, pkt_length, is_last_pkt, payload, pkt seq_no*** . (Note: you can add more fields based on your implementation.)

The client program should take the **server IP address, Server port number**, **file_name**, **pkt_size**, and **WS** as command line arguments. The client and server should display the output in the format shown in Fig. 2. The client and server must be able to run on different hosts as well as on the same host.

Implementation Guidelines:

i) Handle errors/exceptions and program termination in a graceful manner.

ii) Put comments in your code at appropriate places for better readability.

iii) Create a readme file (readme.txt) which contains key implementation details (e.g., packet format, packetization process, file writing process etc.) and limitations of your implementation.

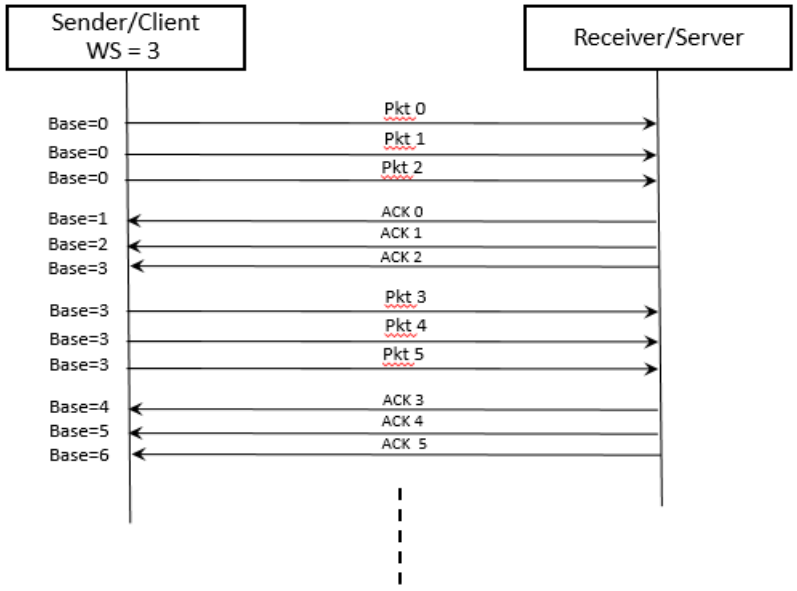

Fig.1

Client	Server
----SEND PACKET 0	---- RECEIVE PACKET 0 length 500 LAST PKT N
----SEND PACKET 1	---- SEND ACK 0
----SEND PACKET 2	---- RECEIVE PACKET 1 length 500 LAST PKT N
----RECEIVE ACK 0	---- SEND ACK 1
----RECEIVE ACK 1	---- RECEIVE PACKET 2 length 500 LAST PKT N
----RECEIVE ACK 2	---- SEND ACK 2
----SEND PACKET 3	---- RECEIVE PACKET 3 length 500 LAST PKT N
----SEND PACKET 4	---- SEND ACK 3
----SEND PACKET 5	---- RECEIVE PACKET 4 length 500 LAST PKT N
----RECEIVE ACK 3	---- SEND ACK 4
----RECEIVE ACK 4	---- RECEIVE PACKET 5 length 500 LAST PKT N
----RECEIVE ACK 5	---- SEND ACK 5
----SEND PACKET 6	---- RECEIVE PACKET 6 length 500 LAST PKT N
----SEND PACKET 7	---- SEND ACK 6
----SEND PACKET 8	---- RECEIVE PACKET 7 length 500 LAST PKT N
----RECEIVE ACK 6	---- SEND ACK 7
----RECEIVE ACK 7	---- RECEIVE PACKET 8 length 300 LAST PKT Y
----RECEIVE ACK 8	---- SEND ACK 8
----CLOSING CONNECTION	---- CLOSING CONNECTION

Fig.2

---x---x---x---