Implementation and Simulation of Go-Back-N and Selective Repeat Protocols

All tangible works (e.g. code and report) handed in must be original. Duplicate or very similar assignments will receive a failing grade, ZERO. Also, the defined actions will be taken according to the rules of the department, college, and university.

You can work alone or as a team of TWO.

Project Description:

In this project you are required to implement sliding window protocols – Go-Back-N (GBN) and Selective Repeat (SR) - using an unreliable channel (i.e. UDP in our case). You also need to compute checksum of the segment and place it at the beginning of the segment that also needs to be parsed at the other end accordingly. As expected, this project requires two Programs/ processes - sender process and receiver process.

Sender Program:

Your sender program should take following command line arguments: input file (please see the input section for the fields to be included in input file), port number on which receiver is connected and number of packets you want to send. For example, `Mysender inputfile portNum 1000`

In response to the above command, sender can communicate with the receiver process using UDP. During the session, the sender will display the following information for each segment send and receive:

1. The sequence number of the segment sent to the receiver and timer information.
   For example: `Sending` S_n`; Timer started`
1. Display the acknowledgement number received from the receiver
   For example: `Received ACK: ACK_NUM`
2. When the timer for a particular sequence number expires, it should resend segments according to the protocol you are implementing. Display relevant information appropriately.
   For example: Timer expired; `Resending SegX SegY …; Timer started`

Receiver Program:

The receiver program/process should take the command line argument that specifies the port number on which the receiver will execute.

```
Myreceiver portNum
```

1. Create a socket with the specified port number and it will wait for client.
2. Display the sequence numbers that are received.
   `Received Segment` S_n
3. Send the acknowledgement according to protocol
   `ACK Sent: ACK_NUM`

Simulation:

Sender and receiver programs must include some faulty network behaviors in order to test the implementation and observe the robustness of the protocol. You must display information for each events and actions taken appropriately. The cases are:

1. Bit error/Checksum error: It means just before sending the segment you intentionally alter/change some bits or bytes. So your checksum will not work at the receiver. We want see the response from the receiver. Assume bit error/checksum error has a probability of 0.1 (i.e. roughly 10 segments out of 100 may have such issue).
2. Lost Packet: In this case, receiver will treat a particular packet as a lost packet even though it is actually arrived perfectly. We want see the response from receiver for a lost packet. Assume lost packet has a probability of 0.1 (i.e. roughly 10 segments out of 100 may have such issue).

3. Lost ACK: Same as lost packet but happens at sender. We want see the response from sender for a lost ACK. Assume lost ACK has a probability of 0.05 (i.e. roughly 5 ACKs out of 100 will have such issue).

Input Format:

The application will be a command line interface (CLI), and we will need to pass an input file which would contain the following inputs:

1. Protocol Name: GBN or SR
2. `m N`; where `m` = no of bits used in sequence numbers, `N` = Window size
3. Timeout period in micro second/millisecond. Use your judgement to set the value during simulation.
4. Size of the segment in bytes

Example 1:

```
GBN
4 15
10000000
500
```

Example 2:

```
SR
4 8
10000000
500
```

Output Format:

Print statements simulating the behavior of the protocols implemented as mentioned above. Any format that you think will be better for comprehension can be used.

Submission Details:

Submission instructions and notes:

- Any programming languages can be used (preferred C/C++, Java, C#, Python).
- Submit a report covering the different aspects of project components.
- Submit error free code with instructions so that TA can run and test your program.
- Direct questions to TA