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.

If you choose to work as a team of TWO include each team member's name on each item included in the assignment (Report, Code, Readme, etc.).

Project Description:

In this project you will be required to implement two sliding window protocols – Go-Back-N (GBN) and Selective Repeat (SR) - using an unreliable channel (i.e. UDP in our case). In order to implement these protocols you will need to be able to verify the contents of a transmitted package. Since UDP is an unreliable channel you will need to implement a checksum and append it to the front of the message before it's sent. The receiver will then need to use the checksum to verify the contents of the message. As expected, this project requires two Programs/ processes - sender process and receiver process.

Checksum and Message Header:

For this project you should use the Internet checksum to verify the contents of messages. This checksum is 16-bits long and works by summing up the message contents and then inverting the calculated sum. Its important to note that because the size of the sum is only 16-bits it's important to add any values that would be carried out back into the sum.

A sequence number will also need to be included in the message in order to detect out of order packets. It's important to include the sequence number when you are calculating the checksum.

Sender Program:

Your sender program should take the 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`

Using the parameters passed, the sender will communicate with the receiver process using UDP. During the session, the sender will display the following information for each segment sent and received:

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:

Your sender program should take the following command line arguments: input file (so you can determine the type of protocol to use) and the port number the receiver will run on.
For example, `Myreceiver inputfile 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 event and action taken appropriately. The cases are:

1. Bit error/Checksum error: Just before sending the segment you will intentionally alter/change some bits or bytes so your checksum will not work at the receiver. We want see the receiver detect the error with the packet. Assume that this bit error/checksum error has a probability of 0.1 to occur (i.e. roughly 10 segments out of 100 may have such issue).
2. Lost Packet: In this case we want see the response from receiver for a lost packet. The receiver will treat a particular packet as a lost packet even though it is actually arrived perfectly. Assume the 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 will 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 seconds is good for testing).
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 easier to read can also be used.

Tips

- Work on getting the checksum working correctly and test it thoroughly.
- It can be hard to test the sender and receiver, so test the individual parts as you build them.
- Each message will need to have a sequence number, make sure to include this in the message contents when you calculate the checksum for each message.
- Make sure you understand how your programming language handle's UDP sockets.

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.
- Ensure that your name (and your partner's name) is included on all submitted materials.
- Direct your questions to the TA