IIT Goa CS 212 (Computer Networks LAB)
Instructor: Dr. Neha Karanjkar

LAB 4: Sliding Window Protocols

Introduction:

The goal of this assignment is to:

- Implement the Go-back-N and Selective Repeat protocols for reliable data transfer by translating their FSM descriptions into Python code
- Perform simulations to validate the implementation
- Use simulations to gain insights on how the performance of these protocols varies with various parameters.

Submission Instructions:

- This is a **team** assignment that can be done in teams of at-most 2 persons. Sharing of code or reports across teams or copying code from the Internet or other sources will be considered pagiarism and will incur zero marks for the entire LAB component in the course and reporting of the cases to SSAC.

- This is a **take-home** assignment. The final report and code for all the questions have to be submitted on Google classrooms.

- **Format for final submission:**

  - Only one person in the team is required to submit the solution. The other team member may submit a comment stating the name of their team mate who has submitted on their behalf. The report must contain the names of all members.
  - You need to upload into the Google form:
    - A typeset report in pdf format. The file should be named **<name>_Lab4_report.pdf**, where name is of the person submitting the solutions as it appears on google classroom. The quality of the report carries significant weightage. Your report should be complete, correct, clear and concise. In the report, you can paste **images/screenshots/plots** where appropriate.

- A Python file named **<name>_Protocol_SR.py** containing your implementation of the Selective Repeat protocol as asked in question 2. The code should contain comments where appropriate to explain your logic.
- Do **not** include any other files in your submission (such as the template files provided by the instructor). Your implementation of the protocols must adhere to the interface provided in the template. It should be possible for anyone to run the simulation with the existing template, by only adding your protocol files to the existing template folder and changing the name in the "import <protocol-file>" line in the testbench.

Go-Back-N Protocol

1. **[10 marks]** You are provided with a Template that consists of Python code for simulating the Go-Back-N Protocol. Go through the code carefully and try to understand the behavior of each block in the Template.
   - **a)** Check if the implementation of the rdt_Sender and rdt_Receiver classes matches the *Finite State Machine* (FSM) description given in the textbook[1].
   - **b)** List at-least one major difference you observe between the Python implementation for the rdt sender/receiver classes and the FSM description in the textbook.
   - **c)** Simulate the system by running the command

     ```
     $ python3 Testbench.py
     ```

     Check if the simulation runs without any errors for the case where Pc (probability of packet corruption) and Pl (probability of packet loss) are non zero for both the DATA and ACK channels.

2. **[10 marks]** Assuming the parameter values for the DATA channel and the ACK channel are identical, set the parameter values (in file Testbench.py) as follows:
   - Pc=0.2, Pl=0.2 , channel propagation delay = 2 seconds
   - Packet_length for DATA packets = 1000 bits, packet length for ACK packets = 10 bits
   - Transmission rate = 1000 bits per second
   - Window size(N)=10, Range of sequence numbers(K)=16

1 Section 3.4 in Computer Networking: A Top-down Approach, by Kurose and Ross (6th or 7th Edition)

Modify the file Testbench.py so that a single simulation is run exactly until the receiving application receives the first 1000 messages.

a) Report the time at which the receiving application receives the first 1000 messages, averaged across five such simulation runs.

b) Measure and report channel utilization on the sender side (for the DATA channel) averaged across five simulation runs.

c) Measure and report (across five simulation runs) the fraction of the packets sent by the rdt_Sender that are simply retransmissions of previously sent packets.

3. **[10 marks]** We wish to study the effect of channel parameters (such as Pc and Pl) and the protocol parameters (such as the Window size N) on the performance of this protocol. It is reasonable to believe that as Pc and Pl increase, more and more re-transmissions will occur that will increase the channel utilization.

   Answer the following questions keeping all parameter values the same as those listed in question 2 (changing *only* those parameter values as asked) and assuming that the parameter values for the DATA and ACK channels are identical.

   a) Let T be the simulation time at which the receiving application receives the first 1000 messages. Draw a plot to show how T varies with Pl (as Pl ranges from 0.1 to 0.9).

   b) Let x be the percentage of packets sent by the rdt_sender that are just retransmissions of previously sent packets. Draw a plot to show how x varies with Pl (as Pl ranges from 0.1 to 0.9, for a simulation performed until the receiving application receives the first 1000 messages).

   c) Let U be the channel utilization on the sender-side. Draw a plot to show how U varies with Pl (as Pl ranges from 0.1 to 0.9, for a simulation performed until the receiving application receives the first 1000 messages).

4. **[10 marks]** As an engineer, suppose you are tasked with deploying such a protocol in a real world scenario where Pc/Pl values for the channel have been estimated beforehand. The following information is known:

   Channel transmission rate is 1000 bits per second, Pc=0.5, Pl=0.5, propagation delay =2, Packet_length for DATA packets = 1000 bits, packet length for ACK packets = 10 bits.

You need to tune your protocol parameters (Window size N, K and timeout_value) for achieveing the best performance.

a) How would you define/quantify “best performance” in this case?

b) What would happen if the window size N is too small or too large? What limits its value?

c) List the values for N, K and timeout that you would choose for achieving the best performance, and explain your reasoning behind this choice.

Selective Repeat Protocol

5. **[10 marks]** Describe the Sender-side and Receiver-side logic for the Selective Repeat protocol as a Finite State Machine (in a format similar to the FSM description of the Go-Back-N protocol).

6. **[10 marks]** Keeping the rest of the template as it is, modify only the rdt_Sender and rdt_Receiver classes to implement the Selective Repeat Protocol, and save this as file Protocol_SR.py.
   Debug and validate your implementation by running a long enough simulation (say, until the receiving application receives 1000 messages). Your simulation should run with no errors for each of the following cases:
   a) Pl=0, Pc=0 for both DATA and ACK channels
   b) Pl=0, Pc=0.5 for both DATA and ACK channels
   c) Pl=0.5, Pc=0.5 for both DATA and ACK channels
   d) Various values of the Window size N in the sender and the receiver.
   Submit the file Protocol_SR.py and paste a screenshot or partial simulation log for one of the simulations.

7. **[10 marks]** For the same values of channel parameters (pc, pl, transmission rates etc), does the SR protocol show a better performance as compared to the GBN protocol? Describe your observations.

8. **[10 marks]** For the SR protocol to work correctly, the value of K (representing the range of sequence numbers) must be at-least twice as large as N (the Window size). That is, N<=K/2. Try setting N=K and check if you get any errors during simulation. Explain why setting N<=K/2 would avoid these errors.