IERG3310 Lab1 Report_GBN (Bonus)

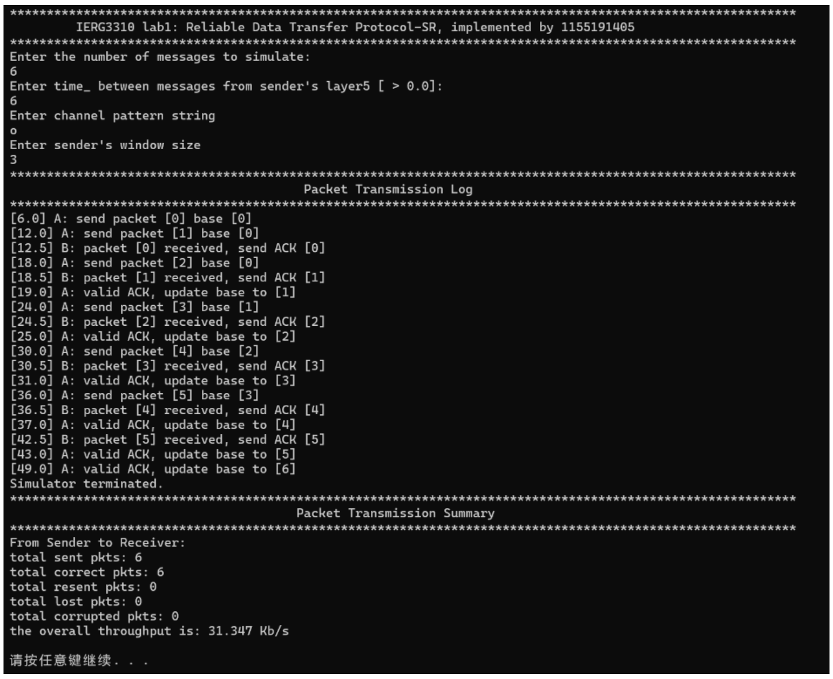
SU Caiyi 1155191405

- ComputeChecksum
  - This function calculates a checksum value for each outgoing packet on the sender's side. It sums up all bytes in the payload, plus the sequence and acknowledgment numbers; any overflow is folded back in a wraparound style. The final step is to take the bitwise complement of the sum, producing a 16-bit checksum stored in the packet structure.
  - Purpose:
    - Allows both sender and receiver to detect bit errors that might occur during transmission.
- CheckCorrupted
  - This function runs on the receiving side (or whenever a packet arrives) to verify the packet's integrity. It recomputes the checksum by summing all packet fields (payload, sequence number, acknowledgment number, and the packet's own checksum). If the result meets a certain expected pattern (specifically, matching the checksum rules), the packet is considered uncorrupted; otherwise, it is flagged as corrupted.
  - Purpose:
    - Helps the receiver (and, in some cases, the sender) determine if the packet data has been altered or damaged in transit.
- A_output
  - When the sender (entity A) gets data from the application layer, A_output attempts to send it to the receiver if the send window is not full. If the send window is full, it stores (buffers) the message until space is available. On a successful send, A_output sets up the packet, calculates its checksum, places it in the sliding window buffer, and calls the lower-layer send routine. If the packet sequence number just sent corresponds to the base of the window, it starts a timer for that packet.
  - Purpose:
    - Implements the main "send" logic of Go-Back-N.
    - Enforces flow control with a sliding window, buffering data if the window is already full.
- A_input
  - A_input handles acknowledgments (ACKs) returning from the receiver. First, it checks if the ACK packet is corrupted. If it is valid, it inspects the ACK number to see if it advances the sender's base. If the ACK is greater than or equal to the base, A_input moves the base to one past the ACK number and stops the relevant timer. If there are still unacknowledged packets between the new base and the next sequence number, it restarts the timer. A_input also attempts to send any buffered messages if now there is space in the window.
  - Purpose:

        - Performs the cumulative ACK mechanism and window sliding for Go-Back-N.
        - Ensures that once ACKs arrive, the sender can release buffer capacity for new messages.

- A_packet_time_rinterrupt
    - If a packet's timer expires, A_packet_time_rinterrupt triggers the Go-Back-N retransmission of all unacknowledged packets in the window. It does not simply resend the single timed-out packet; it goes back to the earliest unacknowledged sequence number (the base) and retransmits every packet up to the sender's current next sequence number minus one. It then restarts the timer for the earliest unacknowledged packet.
    - Purpose:
        - Enforces the classic Go-Back-N rule: when a single timeout occurs, the sender resends every unacknowledged packet starting from the base.
- B_input
    - On the receiver side (entity B), B_input checks if the incoming data packet is corrupted. If it is, the receiver discards it but still sends the last known valid ACK. If the packet is not corrupted and its sequence number matches the next expected sequence, B_input delivers the payload to the application layer, increments the expectation counter, and sends an ACK to the sender. Otherwise, if it is not the exact expected sequence number, the receiver discards the packet and sends a duplicate ACK for the last correctly received sequence number.
    - Purpose:
        - Implements in-order delivery of data.
        - Ensures that only the exactly expected sequence number is accepted in Go-Back-N.

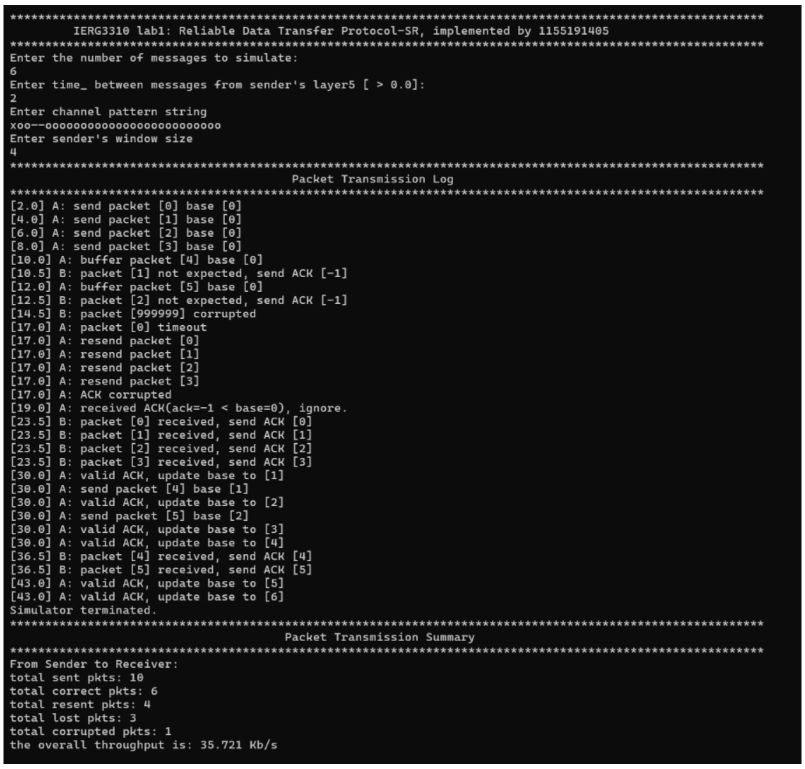
In practice, **GBN** might be preferred for lower-loss networks or where implementation simplicity is key, while **SR** is often favored in more error-prone environments or when bandwidth efficiency is crucial and extra complexity is acceptable.



```
***********************************************************************************************************
         IERG3310 lab1: Reliable Data Transfer Protocol-SR, implemented by 1155191405
***********************************************************************************************************
Enter the number of messages to simulate:
6
Enter time_ between messages from sender's layer5 [ > 0.0]:
6
Enter channel pattern string
o
Enter sender's window size
3
***********************************************************************************************************
                                         Packet Transmission Log
***********************************************************************************************************
[6.0] A: send packet [0] base [0]
[12.0] A: send packet [1] base [0]
[12.5] B: packet [0] received, send ACK [0]
[18.0] A: send packet [2] base [0]
[18.5] B: packet [1] received, send ACK [1]
[19.0] A: valid ACK, update base to [1]
[24.0] A: send packet [3] base [1]
[24.5] B: packet [2] received, send ACK [2]
[25.0] A: valid ACK, update base to [2]
[30.0] A: send packet [4] base [2]
[30.5] B: packet [3] received, send ACK [3]
[31.0] A: valid ACK, update base to [3]
[36.0] A: send packet [5] base [3]
[36.5] B: packet [4] received, send ACK [4]
[37.0] A: valid ACK, update base to [4]
[42.5] B: packet [5] received, send ACK [5]
[43.0] A: valid ACK, update base to [5]
[49.0] A: valid ACK, update base to [6]
Simulator terminated.
***********************************************************************************************************
                                        Packet Transmission Summary
***********************************************************************************************************
From Sender to Receiver:
total sent pkts: 6
total correct pkts: 6
total resent pkts: 0
total lost pkts: 0
total corrupted pkts: 0
the overall throughput is: 31.347 Kb/s

请按任意键继续. . .
```

Sample1



```
***********************************************************************************************************
         IERG3310 lab1: Reliable Data Transfer Protocol-SR, implemented by 1155191405
***********************************************************************************************************
Enter the number of messages to simulate:
6
Enter time_ between messages from sender's layer5 [ > 0.0]:
2
Enter channel pattern string
xoo--ooooooooooooooooooooooo
Enter sender's window size
4
***********************************************************************************************************
                                         Packet Transmission Log
***********************************************************************************************************
[2.0] A: send packet [0] base [0]
[4.0] A: send packet [1] base [0]
[6.0] A: send packet [2] base [0]
[8.0] A: send packet [3] base [0]
[10.0] A: buffer packet [4] base [0]
[10.5] B: packet [1] not expected, send ACK [-1]
[12.0] A: buffer packet [5] base [0]
[12.5] B: packet [2] not expected, send ACK [-1]
[14.5] B: packet [999999] corrupted
[17.0] A: packet [0] timeout
[17.0] A: resend packet [0]
[17.0] A: resend packet [1]
[17.0] A: resend packet [2]
[17.0] A: resend packet [3]
[17.0] A: ACK corrupted
[19.0] A: received ACK(ack=-1 < base=0), ignore.
[23.5] B: packet [0] received, send ACK [0]
[23.5] B: packet [1] received, send ACK [1]
[23.5] B: packet [2] received, send ACK [2]
[23.5] B: packet [3] received, send ACK [3]
[30.0] A: valid ACK, update base to [1]
[30.0] A: send packet [4] base [1]
[30.0] A: valid ACK, update base to [2]
[30.0] A: send packet [5] base [2]
[30.0] A: valid ACK, update base to [3]
[30.0] A: valid ACK, update base to [4]
[36.5] B: packet [4] received, send ACK [4]
[36.5] B: packet [5] received, send ACK [5]
[43.0] A: valid ACK, update base to [5]
[43.0] A: valid ACK, update base to [6]
Simulator terminated.
***********************************************************************************************************
                                        Packet Transmission Summary
***********************************************************************************************************
From Sender to Receiver:
total sent pkts: 10
total correct pkts: 6
total resent pkts: 4
total lost pkts: 3
total corrupted pkts: 1
the overall throughput is: 35.721 Kb/s
```

Sample2