

Overview

The two C programs, client.c and server.c, simulate the Go-Back-N (GBN) protocol to provide reliable data transfer over the unreliable UDP protocol. The simulation includes random packet loss and corruption to test the protocol's resilience.

Client.c - The Sender

The sender's goal is to transmit a set number of packets and ensure they are all acknowledged.

- **Sliding Window:** It sends a burst of packets up to a defined WINDOW_SIZE without waiting for individual ACKs.
 - **Packet Loss Simulation:** It has a chance to randomly "drop" packets instead of sending them to simulate network loss.
 - **Single Timer:** It uses one timer for the entire window of unacknowledged packets. It waits for a cumulative ACK using the select() function.
 - **Receiving ACKs:** When a valid cumulative ACK arrives, it slides the window forward, allowing new packets to be sent.
 - **Timeout & Retransmission:** If the timer expires before an ACK is received (a timeout), the sender retransmits all packets in the current window. This is the core "Go-Back-N" behavior.
-

Server.c - The Receiver

The receiver's role is to accept packets in the correct order and acknowledge them.

- **Packet Expectation:** It maintains a single state variable, expectedSeq, for the sequence number of the next packet it expects to receive.
 - **Corruption Simulation:** It has a chance to randomly "corrupt" and discard incoming packets.
 - **In-Order vs. Out-of-Order:**
 - If a received packet's sequence number matches expectedSeq, it is accepted, and expectedSeq is incremented.
 - If a packet arrives out of order, it is discarded. The GBN receiver does not buffer out-of-order packets.
 - **Cumulative ACKs:** After every packet arrival (whether accepted or discarded), the receiver sends an ACK for the last correctly received, in-order packet (expectedSeq - 1). This efficiently informs the sender of its progress.
-

Test Cases

Test Case 1: Perfect channel (no loss, no corruption)

- Expected: Packets 0 → 9 delivered in order, ACKs 0 → 9 sent.
- Not fully seen in this run (because I had losses + corruption).

Test Case 2: Packet loss

- Condition: Random packet loss simulated ($\text{rand}()\%10 < 2$).
- Expected: Missing packet triggers timeout → retransmit full window.
- Matches expectation: packet loss triggered timeouts → sender resent entire window.

```
[Sender: Slide Window] base moved to 6  
[Sender: Ready] Making packet 9  
[Sender: Lost] Simulated loss for Packet 9  
[Sender: ACK received] ackNo=6  
[Sender: Slide Window] base moved to 7  
[Sender: Timeout] Resending window...  
[Sender: Resent] Packet 7  
[Sender: Resent] Packet 8  
[Sender: Resent] Packet 9
```

Test Case 3: Packet corruption

- Condition: Random corruption simulated ($\text{rand}()\%10 < 2$).
- Expected: Corrupted packet discarded, receiver keeps ACKing last in-order. Sender retransmits after timeout.
- Matches expectation: corrupted packets discarded, receiver keeps ACKing last good packet (ackNo=6) until the missing ones were correctly received.

```
[Receiver: Corrupted] Packet 7 discarded  
[Receiver: Corrupted] Packet 8 discarded  
[Receiver: Corrupted] Packet 7 discarded
```

Test Case 4: Out-of-order arrival

- Condition: Receiver discards out-of-order packets.
- Expected: ACK stays at last correct packet. Sender retransmits missing one.
- Matches expectation: out-of-order packets discarded, receiver kept ACKing last in-order packet.

```
[Receiver: Discard] Out-of-order packet 3 (expected 2) discarded  
[Receiver: Sent ACK] ackNo=1  
[Receiver: Discard] Out-of-order packet 4 (expected 2) discarded
```

Test Case 5: Complete transmission

- Expected: All packets 0 → 9 delivered eventually, final ACK = 9.
- Seen at the end of both outputs:
- Matches expectation: all packets 0 → 9 eventually delivered, final ACK = 9.

```
[Receiver: Sent ACK] ackNo=9  
[Receiver: Done] All packets received.  
ayushi@Ayushi:~/cn_lab/6day$  
[Sender: Slide Window] base moved to 10  
[Sender: Done] All packets acknowledged.  
ayushi@Ayushi:~/cn_lab/6day$
```

SAMPLE OUTPUT:

```
ayushi@Ayushi:~/cn_lab/6day$ ./a.out  
[Receiver: Ready] Waiting for packets...  
[Receiver: Ready] In-order packet 0 received → Delivering message  
[Receiver: Sent ACK] ackNo=0  
[Receiver: Ready] In-order packet 1 received → Delivering message  
[Receiver: Sent ACK] ackNo=1  
[Receiver: Discard] Out-of-order packet 3 (expected 2) discarded  
[Receiver: Sent ACK] ackNo=1  
[Receiver: Discard] Out-of-order packet 4 (expected 2) discarded  
[Receiver: Ready] In-order packet 2 received → Delivering message  
[Receiver: Sent ACK] ackNo=2  
[Receiver: Ready] In-order packet 3 received → Delivering message  
[Receiver: Sent ACK] ackNo=3  
[Receiver: Ready] In-order packet 4 received → Delivering message  
[Receiver: Sent ACK] ackNo=4  
[Receiver: Ready] In-order packet 5 received → Delivering message  
[Receiver: Sent ACK] ackNo=5  
[Receiver: Ready] In-order packet 6 received → Delivering message  
[Receiver: Sent ACK] ackNo=6  
[Receiver: Corrupted] Packet 7 discarded  
[Receiver: Corrupted] Packet 8 discarded  
[Receiver: Corrupted] Packet 7 discarded  
[Receiver: Discard] Out-of-order packet 8 (expected 7) discarded  
[Receiver: Sent ACK] ackNo=6  
[Receiver: Corrupted] Packet 9 discarded  
[Receiver: Discard] Out-of-order packet 8 (expected 7) discarded  
[Receiver: Sent ACK] ackNo=6  
[Receiver: Discard] Out-of-order packet 9 (expected 7) discarded  
[Receiver: Ready] In-order packet 7 received → Delivering message  
[Receiver: Sent ACK] ackNo=7  
[Receiver: Ready] In-order packet 8 received → Delivering message  
[Receiver: Sent ACK] ackNo=8  
[Receiver: Corrupted] Packet 9 discarded  
[Receiver: Corrupted] Packet 9 discarded  
[Receiver: Ready] In-order packet 9 received → Delivering message  
[Receiver: Sent ACK] ackNo=9  
[Receiver: Done] All packets received.  
ayushi@Ayushi:~/cn_lab/6day$  
ayushi@Ayushi:~/cn_lab/6day$ ./a.out  
[Sender: Ready] Starting Go-Back-N ARQ...  
[Sender: Ready] Making packet 0  
[Sender: Sent] Packet 0 (seqNo=0)  
[Sender: Ready] Making packet 1  
[Sender: Sent] Packet 1 (seqNo=1)  
[Sender: Ready] Making packet 2  
[Sender: Lost] Simulated loss for Packet 2  
[Sender: Ready] Making packet 3  
[Sender: Sent] Packet 3 (seqNo=3)  
[Sender: ACK received] ackNo=0  
[Sender: Slide Window] base moved to 1  
[Sender: Ready] Making packet 4  
[Sender: Sent] Packet 4 (seqNo=4)  
[Sender: ACK received] ackNo=1  
[Sender: Slide Window] base moved to 2  
[Sender: Ready] Making packet 5  
[Sender: Lost] Simulated loss for Packet 5  
[Sender: ACK received] ackNo=1  
[Sender: ACK received] ackNo=1  
[Sender: Timeout] Resending window...  
[Sender: Resent] Packet 2  
[Sender: Resent] Packet 3  
[Sender: Resent] Packet 4  
[Sender: Resent] Packet 5  
[Sender: ACK received] ackNo=2  
[Sender: Slide Window] base moved to 3  
[Sender: Ready] Making packet 6  
[Sender: Sent] Packet 6 (seqNo=6)  
[Sender: ACK received] ackNo=3  
[Sender: Slide Window] base moved to 4  
[Sender: Ready] Making packet 7  
[Sender: Sent] Packet 7 (seqNo=7)  
[Sender: ACK received] ackNo=4  
[Sender: Slide Window] base moved to 5  
[Sender: Ready] Making packet 8  
[Sender: Sent] Packet 8 (seqNo=8)  
[Sender: ACK received] ackNo=5  
[Sender: Slide Window] base moved to 6  
[Sender: Ready] Making packet 9
```