Lab 12 Unreliable UDP Communication & Basic Reliability Patterns

Assessments	@October 27, 2025 → October 28, 2025
Dates	@October 28, 2025 → October 28, 2025
Topics	**Unreliable UDP Communication & Basic Reliability Patterns**
Instructor	Kanthanet (Nam)

UDP Client-Server

Objective: Implement a simple UDP client-server application in Go. First, build the basic unreliable communication. Then, add a simple retry and acknowledgment mechanism to make it reliable.

Lab Part 1: Unreliable UDP (30 mins)

1. **Step 1: Write the UDP Server.**
   - Create a Go file named `server.go`.
   - Import the `net` and `fmt` packages.
   - Use `net.ListenPacket("udp", ":8080")` to create a UDP listener.
   - Use a `for` loop to continuously read incoming datagrams with `ReadFromUDP`.
   - Print the message received and the address of the sender.
   - Don't forget to close the listener with a `defer` statement.
2. **Step 2: Write the UDP Client.**
   - Create a Go file named `client.go`.
   - Use `net.Dial` to establish a "connection" to the server's address (`localhost:8080`).
   - In a `for` loop, use `conn.Write` to send a message to the server.
   - You can use `fmt.Println` to print what you're sending.
   - Don't forget to close the connection with a `defer` statement.
3. **Step 3: Test and Observe Unreliability.**
   - Run the server first: `go run server.go`.
   - In another terminal, run the client: `go run client.go`.
   - Verify that messages are being received.
   - **Demonstrate Unreliability:** Stop the server (`Ctrl+C`). Continue sending messages from the client. The client will not receive an error, and the messages will be lost.

Lab Part 2: Implementing Basic Reliability (30 mins)

1. **Step 1: Add a Sequence Number to the Client.**
   - Modify `client.go` to add a simple sequence number to each message. For example, send a message like `1:Hello`, `2:Hello`, etc.
2. **Step 2: Add an Acknowledgment to the Server.**
   - Modify `server.go`. After receiving a message, parse the sequence number from it.
   - Send an acknowledgment (ACK) back to the sender using `WriteToUDP`. The ACK message should contain the sequence number you received, e.g., `ACK:1`.
3. **Step 3: Implement Retries and Timeout in the Client.**
   - Modify `client.go`. After sending a message, use `conn.SetReadDeadline` to set a timeout on the read operation.
   - Use a nested `for` loop to retry sending the message if the read operation times out.
   - When an ACK is received, verify that the sequence number matches the sent message before breaking out of the retry loop.
4. **Step 4: Final Test.**
   - Run the server and client again.
   - Now, briefly stop the server while the client is running. The client should fail on the first attempt, timeout, and then successfully resend the message after you restart the server.