| Course | COMP 7005 |
|--------|-----------|
| Program | Bachelor of Science in Applied Computer Science |
| Term | April 2025 |

- This is a **pair or individual (your choice)** [programming](programming) assignment.

# Objective

- This assignment introduces students to implementing reliable communication over UDP by simulating network unreliability and building a simple protocol using retransmissions and acknowledgments.
- You will write three programs: a client, a server, and a proxy server that introduces packet loss and delay.
- The goal is understanding how to make communication reliable over an unreliable transport layer.

# Learning Outcomes

- Understand the limitations of UDP and the need for reliability mechanisms.
- Design a message-based protocol with identifiers and acknowledgments.
- Implement and evaluate retry logic, timeout handling, and error cases.
- Simulate packet loss and delay using a configurable proxy server.
- Measure and describe how the system performs under degraded conditions.

# Assignment Details

- The programs should also work without the proxy (client and server communicating directly).

## Client

- The client reads and sends messages from standard input to the UDP server. It implements a reliability mechanism as follows:
  - Assigns a sequence number to each message.
  - Sends the message to the server and waits for an acknowledgment.
  - The client retransmits the message if no acknowledgment is received within the timeout period.

- ○ After a maximum number of retries (e.g., 5), the client gives up on that message and prints an error.

- ● The client supports the following command-line arguments:
  - ○ --target-ip: IP address of the server
  - ○ --target-port: Port number of the server
  - ○ --timeout: Timeout (in seconds) for waiting for acknowledgments
  - ○ --max-retries: Maximum number of retries per message (optional, default 5)
- ● The client does not attempt to communicate with more than one server and does not implement any connection or handshake logic.

## Server

- ● The server listens on a UDP socket and receives messages from a client. For each valid message:
  - ○ It prints the message to standard output.
  - ○ It returns an acknowledgment (including the original sequence number) to the client.
- ● The server does not respond to duplicate messages or out-of-order delivery; it is to acknowledge and display what it receives. It supports the following arguments:
  - ○ --listen-ip: IP address to bind to
  - ○ --listen-port: UDP port to listen on
- ● The server only handles one client at a time and is not required to support concurrent connections.

## Proxy Server

- ● The proxy server sits between the client and the server. It forwards UDP packets in both directions while simulating unreliable network conditions.
- ● It is responsible for:
  - ○ Listening for packets from the client on a specified IP and port.
  - ○ Forwarding those packets to the actual server address.
  - ○ Listening for packets from the server and forwarding them back to the client.
  - ○ Randomly dropping packets based on configured drop probabilities.
  - ○ Randomly delaying packets based on configured delay probabilities and delay ranges.
- ● The proxy must support independent configuration for each direction (client-to-server and server-to-client). Delay times must be specified as a millisecond range using min and max values.
- ● The proxy supports the following arguments:
  - ○ --listen-ip          IP address to bind for client packets
  - ○ --listen-port          Port to listen on for client packets
  - ○ --target-ip          Server IP address to forward packets to
  - ○ --target-port          Server port number

- ○ --client-drop          Drop chance (%) for packets from client
- ○ --server-drop          Drop chance (%) for packets from server
- ○ --client-delay          Delay chance (%) for packets from client
- ○ --server-delay          Delay chance (%) for packets from server
- ○ --client-delay-time-min    Minimum delay time (ms) for client packets
- ○ --client-delay-time-max    Maximum delay time (ms) for client packets
- ○ --server-delay-time-min    Minimum delay time (ms) for server packets
- ○ --server-delay-time-max    Maximum delay time (ms) for server packets
- Example:

```
proxy --listen-ip 0.0.0.0 \
      --listen-port 4000 \
      --target-ip 192.168.1.10 \
      --target-port 5000 \
      --client-drop 10 \
      --server-drop 5 \
      --client-delay 20 \
      --server-delay 15 \
      --client-delay-time-min 100 \
      --client-delay-time-max 200 \
      --server-delay-time-min 150 \
      --server-delay-time-max 300
```

# Message Format

- Your protocol must define a message format that at least includes:
    - A sequence number so acknowledgments can be matched to original messages.
    - The message payload (text entered by the user).
- The acknowledgment must include the matching sequence number.
- Design your message format to be simple, clearly structured, and robust to loss or duplication.

# Constraints

- All communication must use UDP.
- Do not implement any connection setup or teardown. The system is message-oriented.
- Only one client is supported at a time. No concurrency is needed.
- Do not use TCP or any external reliability libraries.

# Required Testing

- You must test your protocol with these combinations:

| Drop Chance (%) | Delay Chance (%) | Delay Time (ms range) |
| --- | --- | --- |
| 0 | 0 | N/A |
| 50 | 0 | N/A |
| 100 | 0 | N/A |
| 0 | 50 | 100–500 |
| 0 | 100 | ≥ client timeout |
| 50 | 50 | ≥ client timeout |

- Run a test without the proxy server.
- Run additional tests as proof that your programs work.

# Report

- Your report must clearly describe:
  - For each configuration:
    - How many messages were attempted
    - How many were acknowledged
    - How many were retransmitted
    - How many were lost
  - How the client behaved when delays exceeded the timeout
  - Any anomalies or edge cases discovered during testing
  - Any changes you made to improve robustness

# Submission

- Ensure your submission meets all the guidelines, including formatting, file type, and submission.
- Follow the AI usage guidelines.
- Be aware of the late submission policy to avoid losing marks.
- ***Note: Please strictly adhere to the submission requirements to ensure you don't lose any marks.***

# Evaluation

| Topic | Value |
| --- | --- |
| Design | 20% |

| Testing | 20% |
|---|---|
| Client Implementation | 15% |
| Server Implementation | 15% |
| Proxy Implementation | 20% |
| Analysis | 10% |
| Total | 100% |

# Hints

- Build and test the client and server before adding the proxy.
- Use select() or poll() for timeouts.
- Simulate high drop or delay percentages to test retransmissions.
- Use sudo if needed to bind to ports under 1024.
- Keep your protocol simple - just enough structure to ensure reliability.