COMP 7005
Project 1
Report

Brandon Rada
A01345707
Dec 2nd, 2025

Purpose

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.
Implement reliable communication over UDP by simulating network unreliability and developing a simple protocol that utilizes retransmissions and acknowledgments.
Three programs: a client, a server, and a proxy server that introduces packet loss and delay.

Requirements

Task	Status
Client	
The client reads and sends messages from standard input to the UDP server.	Fully implemented
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.	Fully implemented
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	Fully implemented
The client does not attempt to communicate with more than one server and does not implement any connection or handshake logic.	Fully implemented
Server	
The server listens on a UDP socket and receives messages from a client.	Fully implemented

For each valid message: ○ It prints the message to standard output. ○ It returns an acknowledgment (including the original sequence number) to the client.	Fully implemented
The server does not respond to duplicate messages or out-of-order delivery; it simply acknowledges and displays what it receives.	Fully implemented
It supports the following arguments: ○ --listen-ip IP address to bind to ○ --listen-port UDP port to listen on	Fully implemented
The server only handles one client at a time and is not required to support concurrent connections.	Fully implemented
Proxy	
The proxy server sits between the client and the server. It forwards UDP packets in both directions while simulating unreliable network conditions.	Fully implemented
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.	Fully implemented
The proxy must support independent configuration for each direction (client-to-server and server-to-client).	Fully implemented
Delay times must be specified as a millisecond range, using minimum and maximum values.	Fully implemented
The proxy supports the following arguments: ○ --listen-ip ○ --listen-port IP address to bind for client packets Port to listen on for client packets	Fully implemented

○ --target-ip ○ --target-port ○ --client-drop ○ --server-drop ○ --client-delay ○ --server-delay Server IP address to forward packets to Server port number Drop chance (%) for packets from client Drop chance (%) for packets from server Delay chance (%) for packets from client 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	🟩

Platforms

dc_shell has been tested on:

- Manjaro
- Ubuntu
- Fedora

Language

- Python3

Documents

- Design
- Testing
- User Guide

EXTRA:
In the same folder the 3 programs are run, there is an extra program called clear_logs.py.
Run it with python3 clear_logs.py to clear all 3 log files of their contents.
I made this as a tool to help quickly move onto the next test without manually clearing the contents of each file.