CSE 156/L Programming Lab 4 - Reliable File Transfer Application Using UDP with Packet Loss Implementation and File Redundancy

Author: Anurag Kache
Student ID: 1801243

Files:

---1. src/
Contains source files for the client and server.
- myserver.cpp
- myclient.cpp

2. bin/
Executable generated by the Makefile.
- myserver
- myclient
- myserver.o
- myclient.o

3. doc/
Documentation of your application in plain text or PDF format, containing information on the internal design, usage instructions, and limitations. It should also list the files in the submission with a brief description.
- README.pdf

4. Makefile
Used to build the client program from the required source files.

---Documentation
---

1) Overview

This project aims to extend reliable file transfer capabilities using UDP with redundancy. The application replicates a local file to multiple servers, ensuring correct and reliable transmission even in the face of packet loss or reordering during transit.

2) Project Structure

The project consists of two main components: myclient and myserver, which represent the client and server applications, respectively. The client sends the file to the server over UDP, and the server reassembles and saves the file locally.

3) Building the Project

Clone the repository to your local machine.
Navigate to the project directory.
Run make command to build the executables.
Executables will be generated in the bin/ directory.

4) Running the Applications

Server (myserver)
./myserver <port_number> <droppc> <root_folder_path>
<port number>: Port number for receiving packets.
<droppc>: Percentage value (0-100) to drop packets.
<root_folder_path>: Root directory path for saving files.

Client (myclient)
./myclient <servn> <servaddr.conf> <mtu> <winsz> <in_file_path> <out_file_path>
<servn>: Number of servers to replicate to.
<servaddr.conf>: Server configuration file path.
<mtu>: Maximum UDP payload size.
<winsz>: Window size for packet transmission.
<in_file_path>: Path of the input file to transfer.
<out_file_path>: Path for saving the output file on servers.

5) Test Cases (Tested All Five Test Cases below with .pdf, .txt, and .jpeg files)

Small File Transfer without Loss:

Description: Transfer a small file (e.g., 1KB) from the client to multiple servers without any simulated packet loss.
Expected Outcome: All servers should receive the file successfully without any errors. The client should exit with a success status.
Server Command: ./myserver 9090 0 serv_folder
Client Command: ./myclient 3 serv.conf 512 10 input/small_file.txt output/small_file.txt

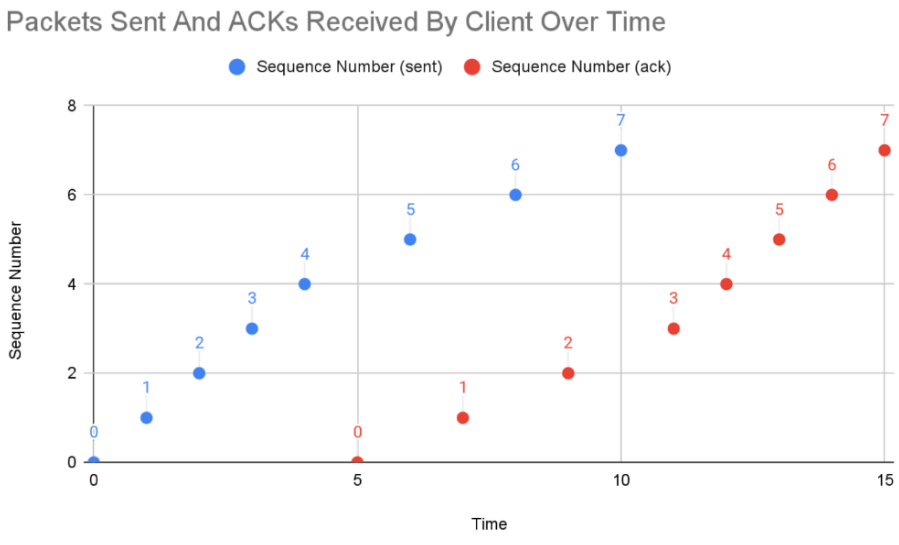
Moderate-sized File Transfer with Packet Loss:
Description: Transfer a moderately-sized file (e.g., 10MB) from the client to servers with a simulated packet loss rate of 10%.
Expected Outcome: Despite the packet loss, all servers should eventually receive the complete file. The client should handle retransmissions properly and exit with a success status.

Server Command: ./myserver 9090 10 serv_folder
Client Command: ./myclient 3 serv.conf 512 10 input/medium_file.txt output/medium_file.txt

Large Binary File Transfer with High Packet Loss:
Description: Transfer a large binary file (e.g., 100MB) from the client to servers with a simulated packet loss rate of 50%.
Expected Outcome: The transfer may take longer due to the high packet loss rate, but all servers should eventually receive the file intact. The client should handle retransmissions and exit with a success status.
Server Command: ./myserver 9090 50 serv_folder
Client Command: ./myclient 3 serv.conf 512 10 input/large_file.bin output/large_file.bin

Client Behavior on Server Unavailability:
Description: Start the servers but keep one server unreachable. Attempt to transfer a file from the client to all servers.
Expected Outcome: The client should detect the unreachable server and print an error message indicating the inability to detect the server. It should continue transferring the file to the available servers and exit with a success status.
Server Commands:
./myserver 9090 0 serv0_folder
./myserver 9091 0 serv1_folder
Client Command: ./myclient 3 serv.conf 512 10 input/small_file.txt output/small_file.txt

Handling Excessive Packet Loss:
Description: Transfer a file from the client to servers with a high simulated packet loss rate (e.g., 90%).
Expected Outcome: The client should handle excessive packet loss by retransmitting lost packets up to the specified limit (10 times). If all servers become unreachable due to packet loss, the client should print an error message indicating the maximum re-transmission limit reached and exit with a failure status.
Server Command: ./myserver 9090 90 serv_folder
Client Command: ./myclient 3 serv.conf 512 10 input/small_file.txt output/small_file.txt



6) References

Class Textbook: Unix Network Programming, Volume 1: The Sockets Networking API, 3/EW. Richard Stevens Bill Fenner M. Rudoff
https://www.w3schools.com/cybersecurity/cybersecurity_network_transport.php
https://www.geeksforgeeks.org/udp-server-client-implementation-c/
https://stackoverflow.com/questions/35568996/socket-programming-udp-client-server-in-c
ChatGPT to help with overall pseudocode/brainstorming ideas to contruct my code for this project