WRITEUP for lab 4

Dongjing Wang

02/24/2024

File List

- **README.md:** A brief explanation of the project, including how to compile and run the program, as well as the directory structure.
- **src/:**
  - **myclient.cc:** A UDP-based client that can send data packets to multiple servers concurrently. It involves breaking the file into packets, sending them to the server, and implementing a sliding window based on the ACK packets sent by the server. Each server corresponds to one thread. Each thread can complete this task independently.
  - **myserver.cc:** Source code for UDP-based server. It listens for incoming packets from clients and simulates packet loss based on random numbers. At the same time, an ACK packet will be returned for the received data packet. This file is also restored based on the received packets and stored in the output file provided by the client. Because it is based on UDP, the server can naturally process data packets from multiple clients and assemble the data packets in the corresponding file according to the `out_file_path` field in each packet.
- **bin/:**
  - Directory for storing executable files generated during the build process. As per project guidelines, this directory is to be kept empty in the repository.
- **doc/:**
  - **WRITEUP.tex:** The LaTeX source file for detailed project documentation. It outlines the project's design, implementation details, instructions for use, test cases, and any known limitations.
  - **WRITEUP.pdf:** Compiled from WRITEUP.tex, provides a readable format of the project documentation for easy sharing and review.

Test Cases Overview

This section outlines the test cases conducted to validate the functionality and robustness of the UDP-based file echo service. Each test case aims to verify a specific aspect of the client-server interaction under different scenarios. The corresponding shell script 'testscript.sh' facilitates the automation of these tests (in the src/ folder. If you want, please follow the tutorial in README.md to run it).

1. **Simple Test:** Tests basic file transmission and echo functionality with two files, ensuring data integrity by comparing the echoed file with the original.
2. **Multiple Server Configurations Test:** Verifies the client's ability to handle multiple server configurations, testing with a PNG file to ensure binary data is correctly echoed back.
3. **Server Unavailability Test:** Examines the client's error handling when the server is unreachable, expecting the client to exit with an error.
4. **Multiple Clients Test:** Assesses server capability to manage concurrent client connections without data loss or corruption.
5. **Large Binary File Test (Optional):** *Not explicitly detailed in the provided script but can be inferred as a potential test case.* Tests the system's performance with a large binary file (e.g., 100MB.bin), focusing on the application's scalability and reliability.

Design Overview

This section delves into the architecture and operational dynamics of our UDP-based file echo system, which integrates both client and server components. The design is tailored for efficient file transmission over UDP, with a focus on packetization, loss management, and integrity preservation through advanced protocols and acknowledgment processes.

Client Design:

- **Sliding Window Protocol:** Employs a sliding window mechanism, enhancing packet flow control in alignment with server acknowledgments, thereby facilitating effective data transmission and congestion management.
- **Timeout and Retransmission Mechanisms:** Incorporates a strategic timeout feature. If acknowledgments from the server are not received within a 30-second window, it triggers the client to abort transmission, signaling an error, and to reinitiate packet transmission, ensuring reliability and efficiency.

- **Acknowledgment Handling:** Features a dedicated thread for acknowledgment reception, fostering a seamless communication channel with the main thread, thereby enhancing the retransmission strategy based on timeout without direct implementation.
- **Implementation Insights:** The approach initially considered non-blocking I/O but pivoted to a multi-threading strategy due to challenges encountered, effectively embracing a simplified version of the Go-Back-N protocol that meets the core project requirements.

Server Design:

- **Packet Management:** Focuses on data packet reception, orderly assembly, and acknowledgment issuance. Adhering to the Go-Back-N protocol logic is vital for accurate sequence number acknowledgment and client window adjustment.
- **Loss Simulation:** Integrates packet loss simulation capabilities to test system resilience and reliability under varied network scenarios.
- **Operational Logging:** Implements comprehensive logging for packet statuses (e.g., DROP, DATA) to aid in debugging and evaluating system performance.
- **Multi-client Processing:** Equipped to handle concurrent client requests efficiently by maintaining distinct client states, ensuring that client-specific variables like sequence numbers are managed accurately without interference from parallel processes.

Limitations: Although I'm sure my lab 4 is much better than lab 3, I feel like the transfer speed is still slower. Especially when the MTU and window limits are relatively small.

Graphic of Client Log:

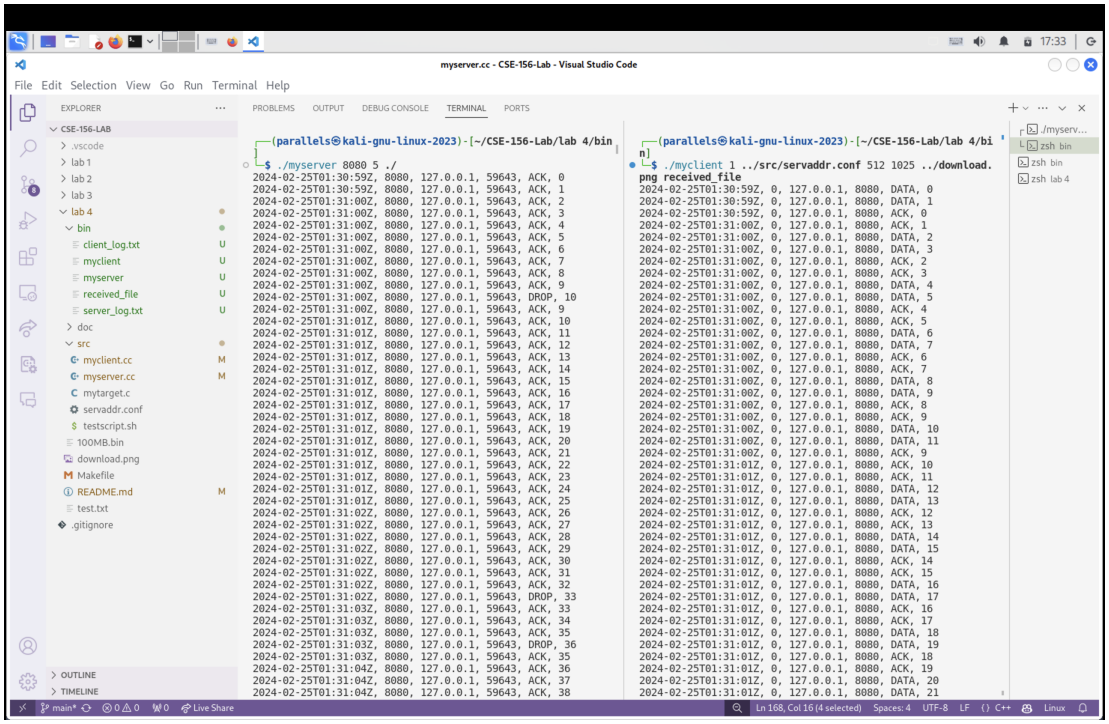

Figure 1: The graph of both server log and client log

Visualization of ACK and DATA:

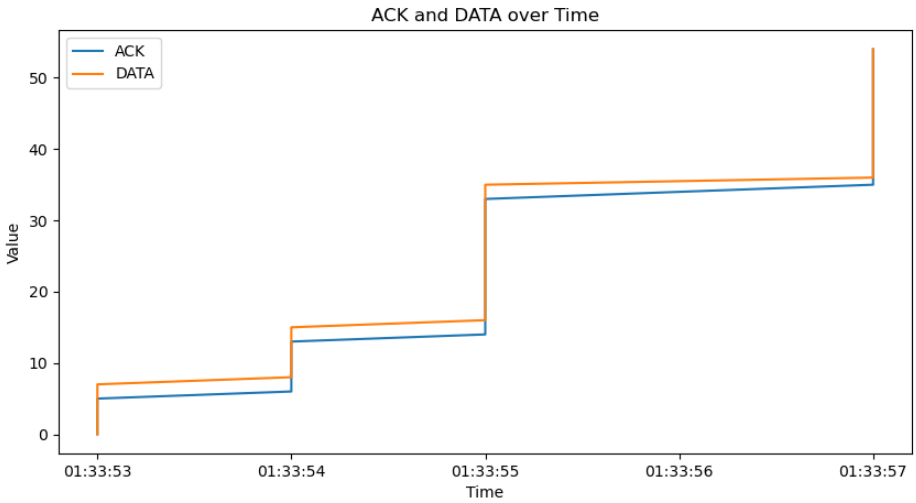

Figure 2: The graph for visualizing ACK and DATA