Design introduction: the customized reliable communication protocol over UDP,including message format, ACK scheme, and transaction sequence, and communication performance.

**Introduction**

**Background and Motivation**

User Datagram Protocol (UDP) is a connectionless transport layer protocol that provides minimal services, making it ideal for low-latency communication. However, UDP lacks reliability mechanisms such as error correction, acknowledgments (ACKs), and retransmissions, which are present in Transmission Control Protocol (TCP). In applications where low latency is essential but reliable data delivery is still required, a customized reliable communication protocol over UDP becomes necessary.

**Objective**

The goal of this project is to design a reliable communication protocol over UDP by incorporating acknowledgment schemes, message sequencing, error handling, and retransmission mechanisms. This protocol ensures data integrity and delivery while maintaining the lightweight and low-latency advantages of UDP.

**Key Features**

- **Message Format**: Structuring data packets with headers containing sequence numbers, checksums, and acknowledgment indicators.
- **ACK Scheme**: Implementing selective or cumulative acknowledgments to ensure reliable data delivery.
- **Transaction Sequence**: Defining a communication sequence for message transmission, acknowledgment, and retransmission upon packet loss.
- **Communication Performance**: Evaluating throughput, latency, and packet loss recovery efficiency compared to TCP.

**Structure of the Document**

This document details the design of the protocol, including:

- **Section 2: Message Format** – Explains the structure of data packets, including headers and payloads.

| Command | Hex Value | Description |
|---|---|---|
| **UNKNOWN** | 0x00 | Unrecognized command |
| **REQ_QUIT** | 0x01 | Client requests to disconnect |
| **REQ_DOWNLOAD** | 0x02 | Client requests a file download |
| **RSP_DOWNLOAD** | 0x03 | Server response with file details |
| **REQ_LIST_FILES** | 0x04 | Client requests available files |
| **RSP_LIST_FILES** | 0x05 | Server sends file list |
| **DOWNLOAD_ERROR** | 0x30 | Indicates file request error |

- **Section 3: ACK Scheme** – Describes acknowledgment mechanisms to ensure reliability.

File Download Process
1. Client initiates a REQ_DOWNLOAD request, specifying the file name.
2. Server responds with RSP_DOWNLOAD, including file details and a session ID.
3. The server transmits data packets, each with a unique sequence number.
4. The client acknowledges received packets via ack_packet responses.
5. If a packet is lost, the server retransmits after a timeout period.
6. The process continues until all packets are received and acknowledged.

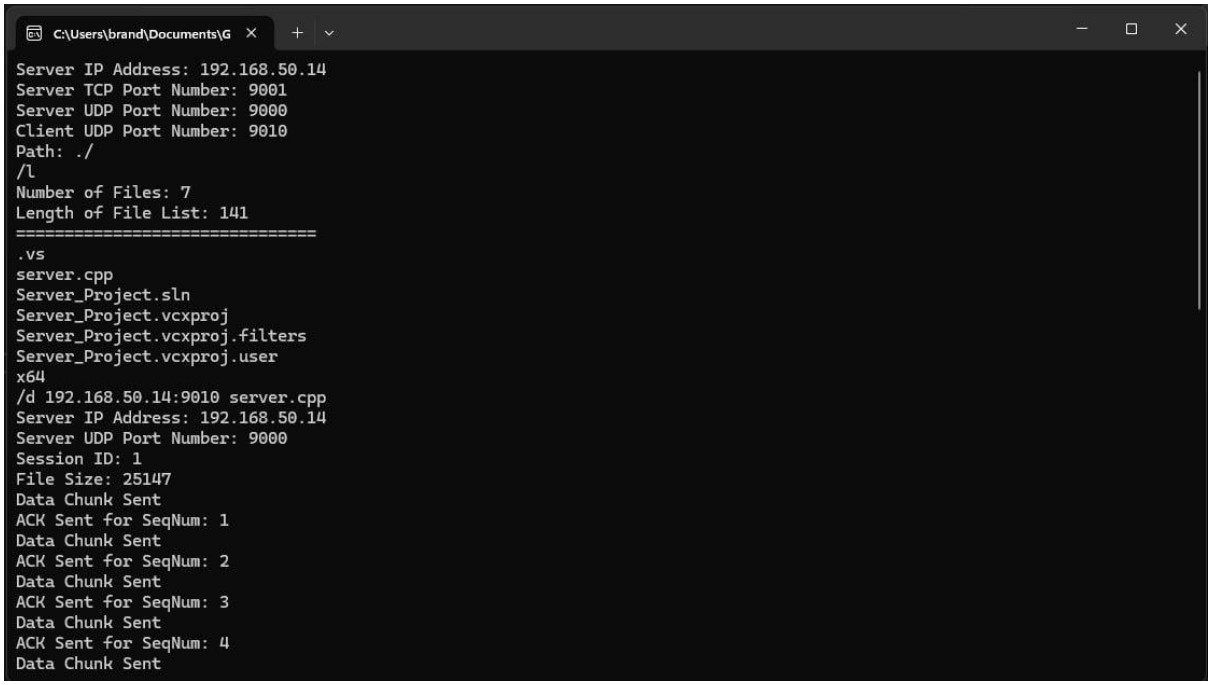
Error Handling
• If the requested file is unavailable, the server responds with DOWNLOAD_ERROR.
• If an ACK is not received within timeout_ms = 350ms, the packet is retransmitted.
• The client ensures correct packet ordering before writing to disk.

- **Section 4: Transaction Sequence** – Outlines the communication flow, including message exchange and error handling.
- **Section 5: Communication Performance** – Analyzes the efficiency and reliability of the proposed protocol through simulations or empirical results.

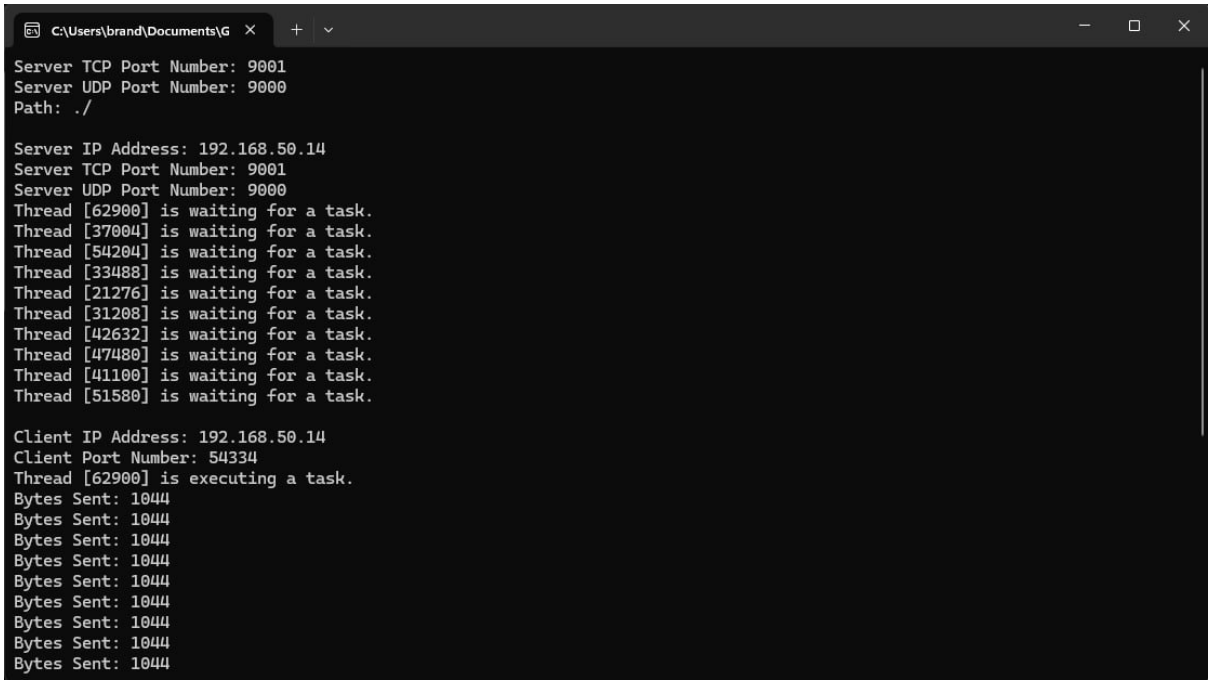
The proposed protocol aims to provide an effective solution for applications requiring reliable and efficient data transmission over UDP, such as real-time communications, gaming, and lightweight data transfer services.

Verification: how did your group verify the design? Screenshots/photos, log files can be provided.

CLIENT



SERVER




Individual contribution: including all members' full names, SIT IDs, and DigiPen IDs.

/******************************START HEADER*******************************/

/* \file README

\ author Zulfami, b.zulfamiashrafi 2301298

\ author Brandon Poon, b.poon 2301224

\ author Gabriel Peh, peh.j 2301454

\par email: b.zulfamiashrafi@digipen.edu

\par email: b.poon@digipen.edu

\par email: peh.j@digipen.edu

\date 19 March 2025



/******************************END HEADER*******************************/

Zulfami:

- Set up and configured the TCP connection between the server and client.

- Managed ACK (Acknowledgment) handling to ensure reliable data transfer.

- Developed and optimized connection initialization and teardown processes.

- Conducted testing, debugging, and performance analysis to improve network stability.

Brandon:

- Developed server-side multi-threading to handle concurrent client requests effectively.

- Implemented packet queuing mechanisms to manage incoming data efficiently.

- Conducted stress testing, debugging, and performance optimization for the UDP transmission process.

- Enhanced error detection and recovery to minimize packet loss and improve reliability.

Gabriel:

- Developed the client-side message processing and logging system.

- Implemented message validation and error handling to ensure accurate communication.

- Conducted extensive testing and debugging to verify client-server interactions.