UDP File Transfer with RDT 1.0 - Design Document

**UDP File Transfer with RDT 1.0**

Author: Ayush Pandey Date: 02/05/2025

# Introduction

This document describes the design and implementation of a file transfer system using UDP sockets and the RDT 1.0 protocol. The system consists of a client and a server that communicate over a network to transfer a BMP image file.

# Project Structure

The project is divided into two main phases:

 **Phase 1(a):** Basic UDP Communication

A diagram of a communication system

Description automatically generated

A screenshot of a computer

Description automatically generated

## 3. Phase 1(a): Basic UDP Communication

This phase implements basic UDP communication between client and server with message echo functionality.

### Client Implementation

A screenshot of a computer program

Description automatically generated

**Explanation:**

* Imports the socket module for network communication.
* Defines the server's IP address and port number.
* Creates a UDP socket using the socket.socket() function.
* Enters a loop to continuously prompt the user for messages until the user types 'quit'.
* Converts the message to binary format.
* Sends the binary message to the server using the client\_socket.sendto() function.
* Receives the binary response from the server using the client\_socket.recvfrom() function.
* Converts the binary response back to string format.
* Prints the server's response.
* Closes the socket using the client\_socket.close() function

### Server Implementation

A screenshot of a computer program

Description automatically generated

**Explanation:**

* Imports the socket module for network communication.
* Defines the server's port number.
* Creates a UDP socket using the socket.socket() function.
* Binds the socket to the specified port using the server\_socket.bind() function.
* Prints a message indicating that the server is listening on the specified port.
* Enters a loop to continuously listen for incoming messages.
* Receives the binary message from the client using the server\_socket.recvfrom() function.
* Converts the binary message back to string format.
* Prints the received message and the client's address.
* Converts the processed message to binary format.
* Sends the binary processed message back to the client using the server\_socket.sendto() function.

## 4. Phase 1(b): File Transfer with RDT 1.0

A screenshot of a computer program

Description automatically generated

**Explanation:**

* The process starts by reading the image file.
* If successful, the file data is divided into packets.
* The client sends a packet and waits for an ACK from the server.
* This process repeats until all packets are sent.
* The server reassembles the received packets.
* If successful, the server saves the reassembled file.
* The process ends.

A diagram of a process

Description automatically generated

This phase implements the RDT 1.0 protocol for reliable BMP file transfer.

### Client Implementation

A screenshot of a computer program

Description automatically generated

**Explanation:**

* Imports the socket module for network communication.
* Defines the server's IP address and port number.
* Creates a UDP socket using the socket.socket() function.
* Specifies the filename of the image to be transferred.
* Sets the packet size for data transmission.
* Reads the image file data in binary mode.
* Calculates the number of packets required to transmit the entire file.
* Converts the number of packets to binary format.
* Sends the binary number of packets to the server.
* Iterates through the packets, sending each one to the server.
* Closes the socket.

### Server Implementation

A screenshot of a computer program

Description automatically generated

**Explanation:**

* Imports the socket module for network communication.
* Defines the server's port number.
* Creates a UDP socket using the socket.socket() function.
* Binds the socket to the specified port.
* Prints a message indicating that the server is listening.
* Receives the binary number of packets from the client.
* Converts the binary number of packets to an integer.
* Initializes an empty byte string to store the received file data.
* Iterates through the expected number of packets, receiving each one from the client and appending it to the file\_data.
* Opens a new file named "received\_image.bmp" in binary write mode and writes the received file data to it.
* Closes the socket.

## 5. Sample Scenarios

### Phase 1(a): Basic UDP Echo

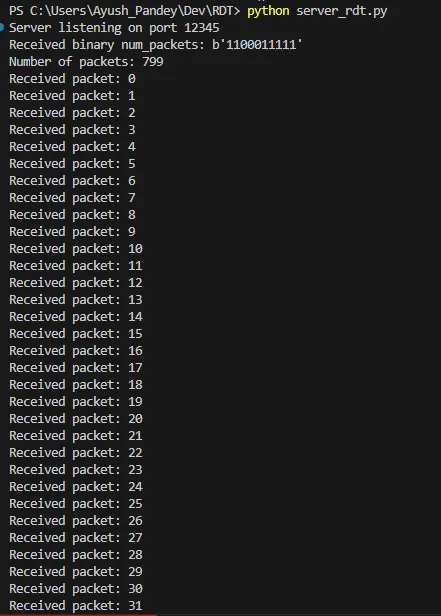
Terminal 1 (Server):

A screenshot of a phone

Description automatically generated

### Phase 1(b): BMP File Transfer

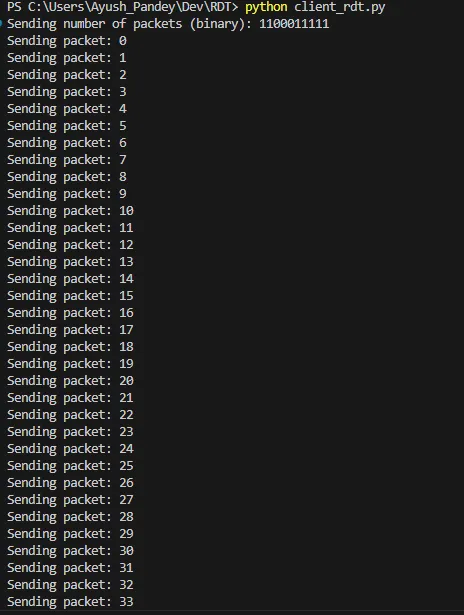
Terminal 1 (Server):



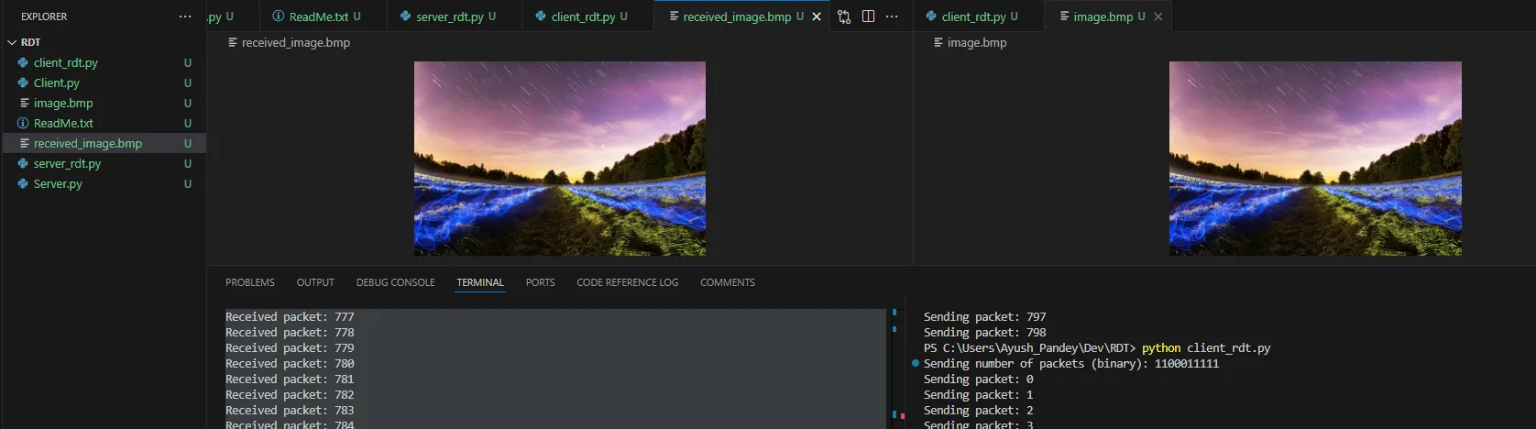
A screenshot of a computer program

Description automatically generated

Terminal 1 (Client):



The client will send the `image.bmp` file to the server, and the server will save the received file as `received\_image.bmp` in the same directory as shown below:



## 6. Future Improvements

* Implement RDT 2.0/3.0 for error handling
* Add checksum verification
* Implement flow control
* Add progress monitoring and GUI
* Support for different file types