UDP File Transfer Tool Documentation

UDP File Transfer Development by Cedric

**Contents**

[1 Introduction 3](#_Toc5218)

[1.1 Purpose 3](#_Toc5219)

[1.2 Scope 3](#_Toc5220)

[2 System Overview 3](#_Toc5221)

[3 Architecture 3](#_Toc5222)

[3.1 Frontend 3](#_Toc5223)

[3.2 Backend 4](#_Toc5224)

[3.3 Protocol 4](#_Toc5225)

[4 Installation 4](#_Toc5226)

[4.1 System Requirements 4](#_Toc5227)

[4.2 Installation Steps 4](#_Toc5228)

[5 User Manual 5](#_Toc5229)

[5.1 Getting Started 5](#_Toc5230)

[5.2 Sending a File 5](#_Toc5231)

[5.3 Receiving a File 5](#_Toc5232)

[5.4 Tips 5](#_Toc5233)

[6 Technical Specifications 6](#_Toc5234)

[6.1 Configuration Parameters 6](#_Toc5235)

[6.2 Packet Format 6](#_Toc5236)

[6.3 Error Handling 6](#_Toc5237)

[7 Troubleshooting 6](#_Toc5238)

[8 Limitations 6](#_Toc5239)

[9 Future Enhancements 7](#_Toc5240)

[10 Conclusion 7](#_Toc5241)

# Introduction

The UDP File Transfer Tool is a desktop application designed to facilitate reliable file transfers over the User Datagram Protocol (UDP). Unlike traditional TCP-based file transfer solutions, this application leverages UDP’s lightweight and fast characteristics while implementing custom reliability mechanisms to ensure robust file delivery. Built with Python and Tkinter, the tool offers a user-friendly graphical interface for sending and receiving files across networks, making it suitable for both technical and non-technical users.

## Purpose

The primary purpose of the UDP File Transfer Tool is to provide a simple, efficient, and reliable method for transferring files between two devices over a network using UDP. It addresses the inherent unreliability of UDP by incorporating packet sequencing, acknowledgments, and retry mechanisms, ensuring complete and accurate file transfers.

## Scope

Thisdocumentationcoverstheapplication’sarchitecture, installation, usermanual, technical specifications, and troubleshooting. It is intended for end-users, developers, and system administrators who wish to use, extend, or maintain the application.

# System Overview

The UDP File Transfer Tool consists of a single Python application with a Tkinter-based graphical user interface (GUI). The application operates in two modes: sender and receiver. Key components include:

* **GUI**: A Tkinter interface for file selection, IP/port configuration, and transfer status monitoring.
* **Networking Module**: A UDP socket-based communication system with custom reliability protocols.
* **Threading**: Background threads for non-blocking file transfer operations.
* **Protocol Headers**: Custom byte-string headers (META, DATA, EOF, ACK) for packet identification.

# Architecture

The application follows a client-server-like architecture, where the sender initiates the transfer, and the receiver listens for incoming packets. The architecture is divided into:

## Frontend

The frontend is built using Tkinter, providing an intuitive interface with:

* File selection for sending.
* Input fields for receiver IP and port.
* Buttons for initiating send/receive operations.
* A progress bar and status label for real-time feedback.

## Backend

The backend handles:

* **Socket Management**: Creation and management of UDP sockets for communication.
* **Packet Construction**: Formatting packets with headers (META, DATA, EOF, ACK) and sequence numbers.
* **Reliability Mechanisms**: Acknowledgments, retries, and timeouts to ensure reliable delivery.
* **File Handling**: Reading/writing file chunks and reassembling received files.

## Protocol

The application uses a custom protocol with the following packet types:

* **META**: Contains metadata (filename, filesize, total chunks).
* **DATA**: Carries file chunks with sequence numbers.
* **EOF**: Signals the end of file transfer.
* **ACK**: Acknowledges receipt of packets.

# Installation

## System Requirements

* Operating System: Windows, macOS, or Linux.
* Python: Version 3.6 or higher.
* Libraries: Tkinter(usuallyincludedwithPython), standardlibrarymodules(socket, os, struct, math, json, threading).

## Installation Steps

1. Install Python from <https://www.python.org/downloads/>.
2. Verify Tkinter is available by running python -m tkinter in a terminal.
3. Download the application source code (main\_app.py).
4. No additional dependencies are required, as the application uses Python’s standard library.
5. Run the application using python main\_app.py.

# User Manual

This section guides users on how to use the UDP File Transfer Tool for sending and receiving files.

## Getting Started

1. Launch the application by running python main\_app.py.
2. The main window displays fields for file selection, receiver IP, port, and transfer status.

## Sending a File

1. Click the **Browse** button to select a file.
2. Enter the receiver’s IP address (e.g., 192.168.1.100) in the **Receiver IP** field.
3. Specify a port number (1025–65534) in the **Port** field, or use the default (5001).
4. Click **Send File**.
5. Monitor the progress bar and status label for transfer updates.
6. A success or error message will appear upon completion.

## Receiving a File

1. Specify a port number (1025–65534) in the **Port** field, matching the sender’s port.
2. Click **Receive File**.
3. Select a folder to save the received file.
4. The application will listen for incoming packets and display progress.
5. A success or error message will appear upon completion, with the file saved to the chosen folder.

## Tips

* Ensure both sender and receiver use the same port number.
* Verify network connectivity and firewall settings to allow UDP traffic on the specified port.
* Large files may take longer to transfer due to UDP’s retransmission mechanism.

# Technical Specifications

## Configuration Parameters

* PACKET\_BUFFER\_SIZE: 1472 bytes (optimal UDP payload size).
* DATA\_CHUNK\_SIZE: 1024 bytes per data packet.
* RECEIVER\_IP: 0.0.0.0 (listens on all interfaces).
* DEFAULT\_PORT: 5001.
* SENDER\_TIMEOUT\_S: 2 seconds for ACK timeouts.
* RECEIVER\_TIMEOUT\_S: 10 seconds for packet timeouts.

## Packet Format

* **META**: META header + JSON-encoded metadata.
* **DATA**: DATA header + 4-byte sequence number + chunk data.
* **EOF**: EOF header.
* **ACK**: ACK header + 4-byte sequence number or header identifier.

## Error Handling

The application handles errors such as:

* Invalid file paths or ports.
* Network timeouts or unreachable receivers.
* Missing or corrupted packets (via retries and acknowledgments).

# Troubleshooting

* **Error: Invalid Port**: Ensure the port is between 1025 and 65534.
* **Receiver Timeout**: Check network connectivity, firewall settings, and ensure the sender is active.
* **Transfer Incomplete**: Verify the sender and receiver are using the same port and that no packets are blocked.
* **Application Not Responding**: Ensure Python and Tkinter are properly installed, and close any conflicting processes.

# Limitations

* UDP’s unreliability may lead to retransmissions, slowing transfers on unstable networks.
* No encryption or authentication, making it unsuitable for sensitive data.
* Single-file transfer per session.

# Future Enhancements

* Implement encryption for secure transfers.
* Support multiple concurrent file transfers.
* Add a command-line interface for advanced users.
* Introduce congestion control to optimize performance on congested networks.

# Conclusion

The UDP File Transfer Tool provides a reliable and user-friendly solution for file transfers over UDP. With its custom reliability mechanisms and intuitive interface, it is wellsuited for educational, experimental, or internal network use. Future enhancements will further improve its functionality and security.