Practical Work 1: TCP File Transfer

Le Duy Anh

26/11/2024

1 Introduction

This project uses Python sockets to implement a 1-to-1 file transfer system over TCP/IP. The system allows a client to send files to a server while ensuring integrity via checksum validation. Enhancements include progress tracking and error handling, making it robust for large files.

2 Protocol Design

The file transfer protocol includes the following steps:

1. The client connects to the server.
2. The client sends file metadata (name and size).
3. The server acknowledges and prepares to receive data.
4. The client sends the file in chunks.
5. The server verifies file integrity using a checksum.

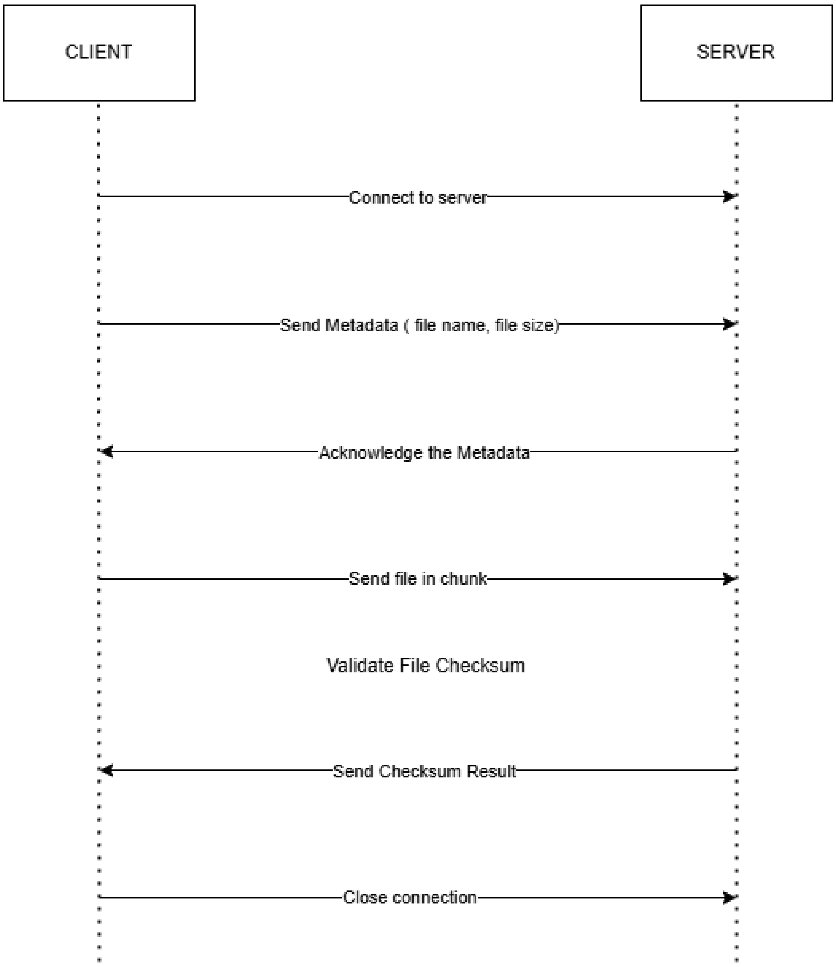

Figure 1: Protocol Flow for File Transfer

3 System Organization

The system uses a client-server model, where the server listens for incoming connections, and the client initiates the file transfer.

4 Implementation

The project uses Python's socket library for communication. Key features include:

- File transfer in chunks to support large files.
- MD5 checksum for file integrity verification.

- Progress tracking using the `tqdm` library.
- Error handling for network and file issues.

4.1 Server Code Snippet

4.1.1 Receiving File Metadata

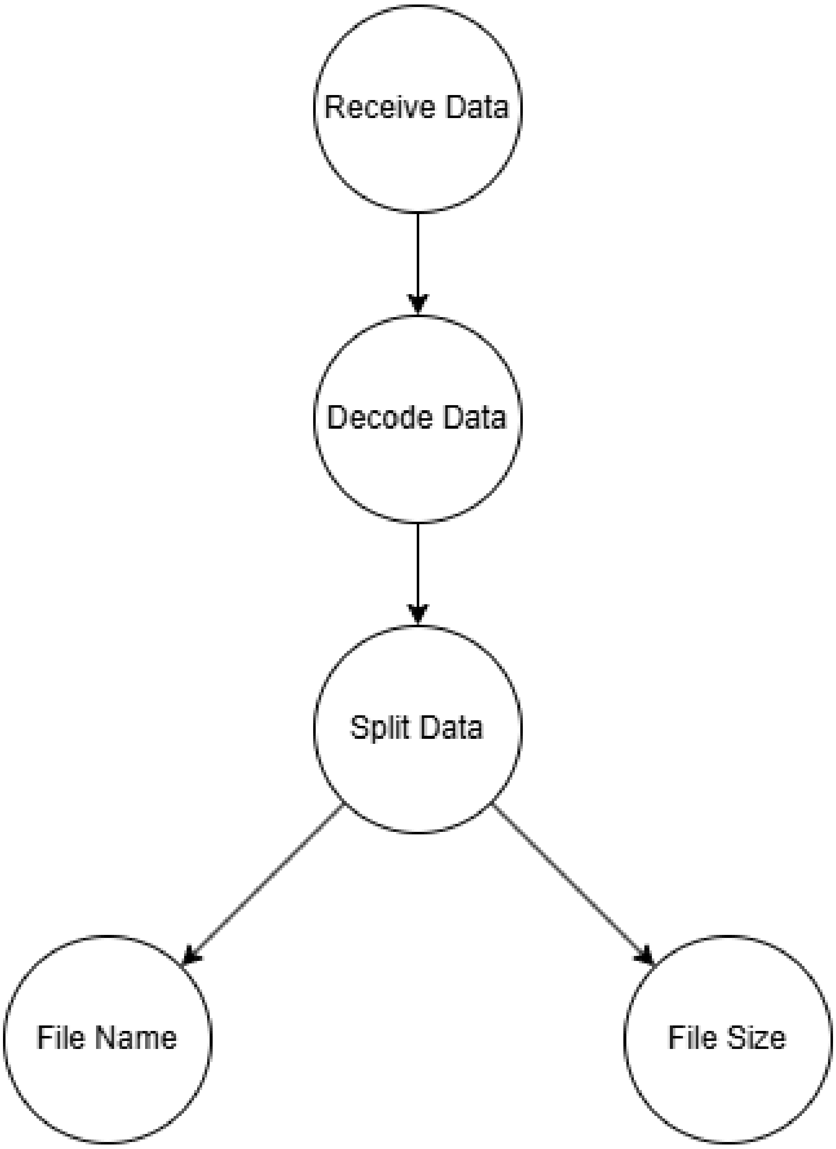

Figure 2: Receiving File Metadata Diagram

```
# Receiving file metadata
file_metadata = conn.recv(1024).decode()
filename, filesize = file_metadata.split(",")
filesize = int(filesize)
```

4.1.2 Receiving File in Chunks

Figure 3: Receiving File in Chunks Diagram

```
# Receiving file in chunks
received_bytes = 0
with open(f"received_{filename}", "wb") as f:
    progress = tqdm(total=filesize, unit="B", unit_scale=True, desc="R
    while received_bytes < filesize:
        chunk = conn.recv(1024)
        if not chunk:
            break
        f.write(chunk)
        received_bytes += len(chunk)
        progress.update(len(chunk))
    progress.close()
```

4.2 Client Code Snippet

4.2.1 Sending File Metadata

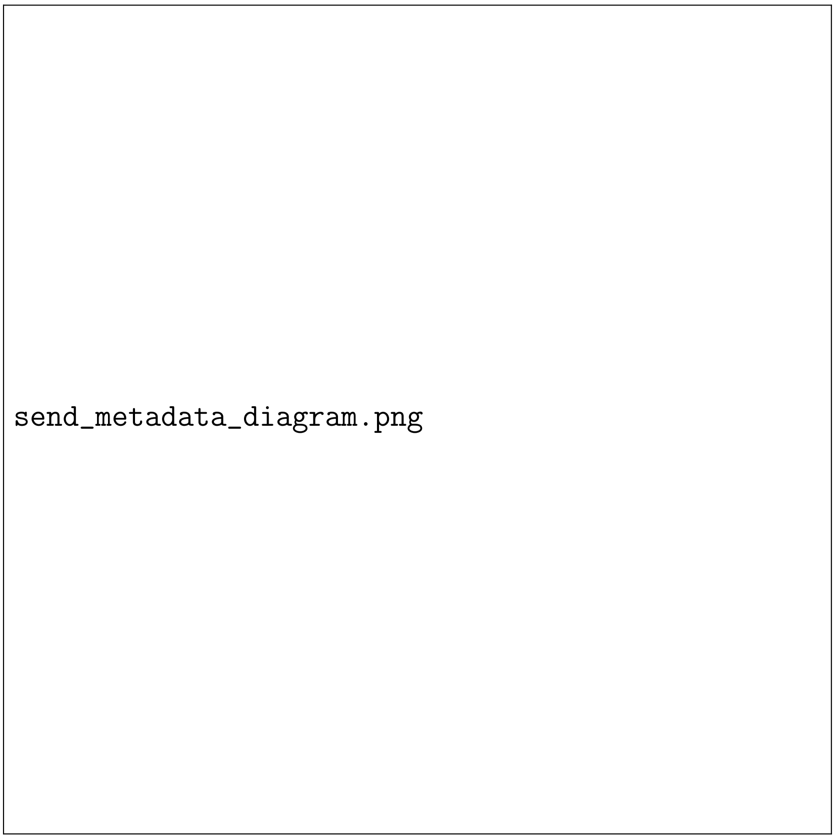

Figure 4: Sending File Metadata Diagram

```
# Sending file metadata
file_metadata = f"{filename},{filesize}"
client_socket.send(file_metadata.encode())
```

4.2.2 Sending File in Chunks

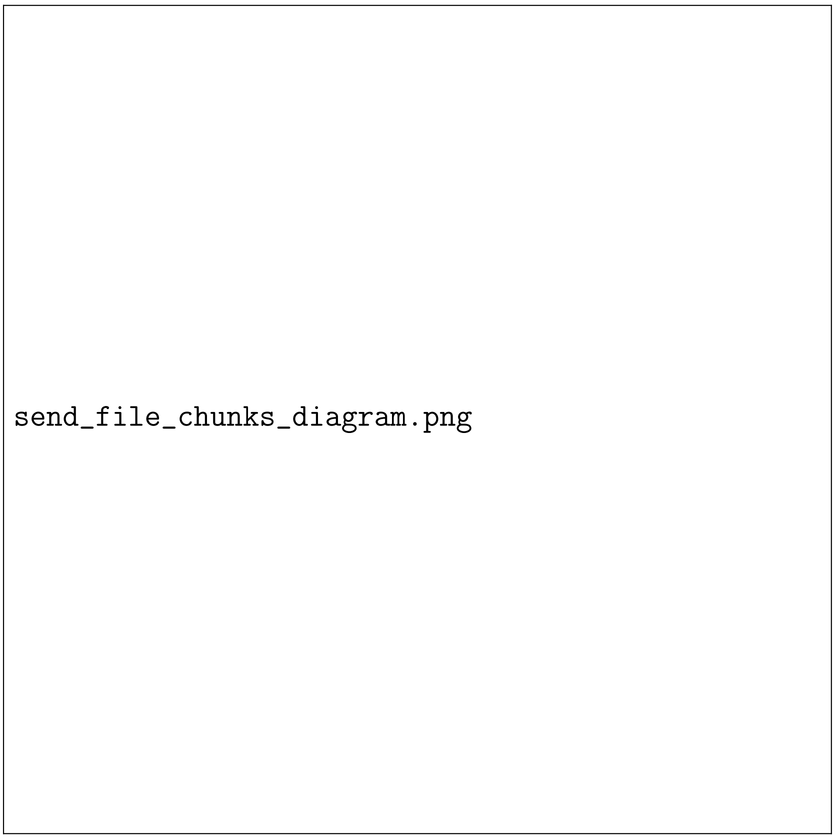

Figure 5: Sending File in Chunks Diagram

```
# Sending file in chunks
with open(filename, "rb") as f:
    for chunk in tqdm(iter(lambda: f.read(1024), b""), desc="Sending")
        client_socket.send(chunk)
```

5 Testing

The system was tested using:

- Small text files (e.g., `test.txt`).
- Large binary files (e.g., images up to 100MB).
- Verification tools for checksum (e.g., `md5sum`).

The table below summarizes the results:

File Type	**Size**	**Result**
Text File	1KB	Successful
Image File	10MB	Successful
Large File	100MB	Successful

Table 1: Testing Results

6 Challenges and Solutions

Challenges included:

- Handling large files efficiently.
- Ensuring data integrity during transfer.

Solutions involved using chunked transfers and MD5 checksums.

7 Conclusion

This project demonstrates the implementation of a robust TCP file transfer system. It highlights the importance of protocol design and error handling in distributed systems.