Practical Work 1: TCP File Transfer

Trinh Nhat Huy - 23BI14193

November 21, 2025

1 Introduction

This report describes a simple file transfer system using TCP/IP and Java sockets. The system has one server that receives files and one client that sends files.

2 Protocol Design

The file transfer works like this. I designed a simple handshake protocol:

```
Client                          Server
  |                               |
  |------ Connect --------------->|
  |                               |
  |------ Filename -------------->|
  |------ File Size ------------->|
  |                               |
  |==== File Data ===============>|
  |                               |
  |<------- ACK ------------------|
  |                               |
  |------ Close ----------------->|
```

Figure 1: How the protocol works

The protocol sends data in this order:

1. Filename as UTF-8 string
2. File size as long (8 bytes)
3. File data in 4KB chunks
4. ACK message at the end

3 System Design

3.1 Server (Server.java)

The server does these things:

- Opens port 5000 and waits
- Accepts client connection
- Receives filename and size
- Receives file data
- Saves file as "received_filename"
- Sends ACK back to client

3.2 Client (Client.java)

The client does these things:

- Checks if file exists
- Connects to server on port 5000
- Sends filename and size
- Sends file data
- Shows progress
- Waits for ACK

4 Implementation

4.1 Server Code

Listing 1: Server.java

```java
import java.io.*;
import java.net.*;

public class Server {
    public static void main(String[] args) {
        try {
            ServerSocket serverSocket = new ServerSocket(5000);
            System.out.println("Server is waiting for client...");

            Socket socket = serverSocket.accept();
            System.out.println("Client connected!");

            DataInputStream dis = new DataInputStream(socket.
                getInputStream());

            String filename = dis.readUTF();
            System.out.println("Receiving file: " + filename);

            long filesize = dis.readLong();
            System.out.println("File size: " + filesize + " bytes");

            FileOutputStream fos = new FileOutputStream("received_" +
               filename);

            byte[] buffer = new byte[4096];
            int bytesRead;
            long totalRead = 0;

            while (totalRead < filesize) {
                bytesRead = dis.read(buffer, 0, (int)Math.min(buffer.
                   length, filesize - totalRead));
                if (bytesRead == -1) break;
                fos.write(buffer, 0, bytesRead);
                totalRead += bytesRead;
                System.out.print("Progress: " + (totalRead * 100 /
                   filesize) + "%\r");
            }

            System.out.println("\nFile received successfully!");

            DataOutputStream dos = new DataOutputStream(socket.
               getOutputStream());
            dos.writeUTF("ACK");

            fos.close();
            dis.close();
            dos.close();
            socket.close();
            serverSocket.close();

        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

4.2 Client Code

Listing 2: Client.java

```
import java.io.*;
import java.net.*;

public class Client {
    public static void main(String[] args) {
        if (args.length < 1) {
            System.out.println("Usage: java Client <filename>");
            return;
        }

        String filename = args[0];

        try {
            File file = new File(filename);
            if (!file.exists()) {
```

```java
                System.out.println("File not found: " + filename);
                return;
            }

            System.out.println("File: " + filename);
            System.out.println("Size: " + file.length() + " bytes");

            Socket socket = new Socket("localhost", 5000);
            System.out.println("Connected to server");

            DataOutputStream dos = new DataOutputStream(socket.getOutputStream());

            dos.writeUTF(file.getName());
            dos.writeLong(file.length());

            FileInputStream fis = new FileInputStream(file);
            byte[] buffer = new byte[4096];
            int bytesRead;
            long totalSent = 0;

            while ((bytesRead = fis.read(buffer)) != -1) {
                dos.write(buffer, 0, bytesRead);
                totalSent += bytesRead;
                System.out.print("Progress: " + (totalSent * 100 / file.length()) + "%\r");
            }

            System.out.println("\nFile sent successfully!");

            DataInputStream dis = new DataInputStream(socket.getInputStream());
            String ack = dis.readUTF();
            System.out.println("Server response: " + ack);

            fis.close();
            dis.close();
            dos.close();
            socket.close();

        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

5 How to Use

First compile the files:

```
javac Server.java
javac Client.java
```

Then run the server:

```
java Server
```

In another terminal, run the client:

```
java Client myfile.txt
```

6 Testing

I tested the program with different files to make sure it handles binary data correctly:

File Type	**Size**	**Result**
Text file	1 KB	OK
Image	2 MB	OK
PDF	5 MB	OK
Video	10 MB	OK

Table 1: Test results

All files transferred correctly and the md5 checksums matched.

7 Problems

Problem 1: The file transfer sometimes didn't finish.
Solution: I used a while loop to make sure all bytes are sent.

Problem 2: Large files were slow.
Solution: I used 4KB buffer size which is faster.

8 Conclusion

This project shows how to transfer files using TCP sockets in Java. The program works well and can send any type of file.

Things that could be better:

- Handle multiple clients at once
- Add file checksum to verify integrity
- Make a GUI instead of command line

I learned how TCP sockets work and how to send binary data over the network.