**Name : Devashish Katoriya**

**Roll No. : 19CS4119**

**Ex.No: 2:** **Implementation of Full-Duplex Multimodal File Transmission using UDP protocol in JAVA.**

**DESCRIPTION:**

UDP Client and Server send a request for synchronization of files among them. Every time client communicates with the server and receives a response from it. The protocol will send any type of files. A log file will be generated with some information like, file name, progress, start time and end time.

**ALGORITHM:**

Server

1. Create a server thread and start listening on server port. Create a client thread for sending.
2. On receiving thread:
   1. Listen for new connection and when a connection arrives, start receiving it.
   2. Read the Client's message containing file information which is arriving.
   3. Get the file contents from client.
   4. Write file onto storage.
3. On sending thread:
   1. Send the second file info like length, filename on client port.
   2. Start sending file contents.
4. Close all streams.
5. Stop.

Client

1. 1. Create a receiving thread and start listening on client port. Create a sending thread for sending file to server.
2. On receiving thread:
   1. Listen for new connection and when a connection arrives, start receiving it.
   2. Read the Client's message containing file information which is arriving.
   3. Get the file contents from client.
   4. Write file onto storage.
3. On sending thread:
   1. Send the second file info like length, filename on client port.
   2. Start sending file contents.
4. Close all streams.
5. Stop.

**PROGRAM:**

**//UDPClient.java**

import **java.net.\***;

import **java.io.\***;

**public** **class** UDPClient **extends** Thread {

**private** **DatagramSocket** datagramSocket;

**int** server\_port;

**InetAddress** clientAddress = InetAddress.getLocalHost();

**String** inputFile;

**public** UDPClient(**int** port, **String** fileName, **int** serverPort) **throws** **IOException** {

        datagramSocket = new DatagramSocket(port);

        datagramSocket.setSoTimeout(9000);

        server\_port = serverPort;

        inputFile = fileName;

    }

**public** **void** run() {

        System.out.println("Client started.");

**byte** buf[] = null;

**int** byteRead;

**int** cnt = 0;

        buf = new **byte**[65000];

        try {

            sleep(3000);

*// Calculate file name and file size*

**File** f = new File(inputFile);

**long** fileSize = f.length();

**String** fileInfo = inputFile + "," + fileSize;

*// Send fileInfo*

            buf = fileInfo.getBytes();

**DatagramPacket** DpSend = new DatagramPacket(buf, buf.length, clientAddress, server\_port);

            datagramSocket.send(DpSend);

            sleep(10);

            System.out.println("File Info: " + fileInfo);

            System.out.println("File Info Sent.");

*// Open input file for reading contents*

**InputStream** inputStream = new FileInputStream(inputFile);

            System.out.println("\nSending file contents...");

            buf = new **byte**[65000];

            while ((byteRead = inputStream.read()) != -1) {

                buf[cnt % 65000] = (**byte**) byteRead;

                if ((cnt + 1) % 65000 == 0) {

*// Send 65000 bytes to server*

                    DpSend = new DatagramPacket(buf, buf.length, clientAddress, server\_port);

                    datagramSocket.send(DpSend);

                    sleep(10);

                    buf = new **byte**[65000];

                    System.out.println("Bytes Sent: " + cnt);

                }

                cnt = cnt + 1;

            }

*// Send final buffer*

            if (cnt != 0) {

                DpSend = new DatagramPacket(buf, (cnt % 65000) + 1, clientAddress, server\_port);

                datagramSocket.send(DpSend);

                buf = new **byte**[cnt + 1];

                sleep(10);

                System.out.println("Final Bytes Sent: " + cnt);

            }

            inputStream.close();

        } catch (**Exception** e) {

            e.printStackTrace();

        }

        System.out.println("\nClient done!");

    }

**public** **static** **void** main(**String**[] args) **throws** **IOException** {

**int** my\_port = 6060;

**int** server\_port = 6070;

        try {

*// Thread for sending file*

**Thread** t = new UDPClient(my\_port, "input.pdf", server\_port);

            t.start();

*// Thread for receiving file*

**Thread** t2 = new UDPServer(my\_port + 1);

            t2.start();

        } catch (**IOException** e) {

            e.printStackTrace();

        }

    }

}

**//UDPServer.java**

import **java.net.\***;

import **java.io.\***;

**public** **class** UDPServer **extends** Thread {

**private** **DatagramSocket** datagramSocket;

**private** **byte**[] receive = new **byte**[65000];

**private** **DatagramPacket** DpReceive = null;

**public** UDPServer(**int** port) **throws** **IOException** {

        datagramSocket = new DatagramSocket(port);

        datagramSocket.setSoTimeout(15000);

    }

**public** **void** run() {

        System.out.println("Server Listening...");

**String** outputFile;

**String** logFile;

**String**[] fileInfo;

**String** startTime, endTime;

**int** len;

        try {

*// Receive fileInfo*

            DpReceive = new DatagramPacket(receive, receive.length);

            datagramSocket.receive(DpReceive);

            fileInfo = data(receive).toString().split(",");

            receive = new **byte**[65000];

            sleep(2);

            System.out.println("File Info Recv.");

            System.out.println("File Info: " + fileInfo[0] + "," + fileInfo[1]);

            startTime = java.time.LocalDateTime.now().toString();

*// Create log file stream*

            logFile = "log\_" + fileInfo[0] + ".txt";

**OutputStream** logStream = new FileOutputStream(logFile);

*// Create output file stream*

            outputFile = "output\_files/" + fileInfo[0];

**OutputStream** outputStream = new FileOutputStream(outputFile);

**int** cnt = 0;

            len = Integer.parseInt(fileInfo[1]);

            logStream.write("\n------------".getBytes());

            logStream.write(("\n" + outputFile).getBytes());

            logStream.write(("\nStart Time: " + startTime).getBytes());

            logStream.write(("\n" + fileInfo[0]).getBytes());

            System.out.println("\nReceiving file contents...");

**double** perc = 0.0;

            perc = (**double**) (cnt / len) \* 100.0;

            System.out.println("Progress: " + perc);

            while (cnt <= len) {

*// Receive 65000 bytes from client*

                DpReceive = new DatagramPacket(receive, receive.length);

                datagramSocket.receive(DpReceive);

*// Write to output file*

                outputStream.write(receive);

                receive = new **byte**[65000];

                sleep(2);

                cnt = cnt + 65000;

                perc = (**double**) ((cnt \* 100) / len);

                if (perc > 100)

                    perc = 100;

                System.out.println("Progress: " + perc);

*// Write to log file*

                logStream.write(("\nProgress: " + perc).getBytes());

            }

            endTime = java.time.LocalDateTime.now().toString();

            logStream.write(("\nEnd Time: " + endTime).getBytes());

            logStream.write("\n------------".getBytes());

            logStream.close();

            outputStream.close();

        } catch (**Exception** e) {

            e.printStackTrace();

        }

        System.out.print("\nServer done.\n");

    }

**public** **static** **void** main(**String**[] args) {

**int** my\_port = 6070;

**int** client\_port = 6060;

        try {

*// Thread for receiving file*

**Thread** t = new UDPServer(my\_port);

            t.start();

*// Thread for sending file*

**Thread** t2 = new UDPClient(my\_port + 1, "input2.pdf", client\_port + 1);

            t2.start();

        } catch (**IOException** e) {

            e.printStackTrace();

        }

    }

**public** **static** **StringBuilder** data(**byte**[] a) {

        if (a == null)

            return null;

**StringBuilder** ret = new StringBuilder();

**int** i = 0;

        while (a[i] != 0) {

            ret.append((**char**) a[i]);

            i++;

        }

        return ret;

    }

}

**OUTPUT**

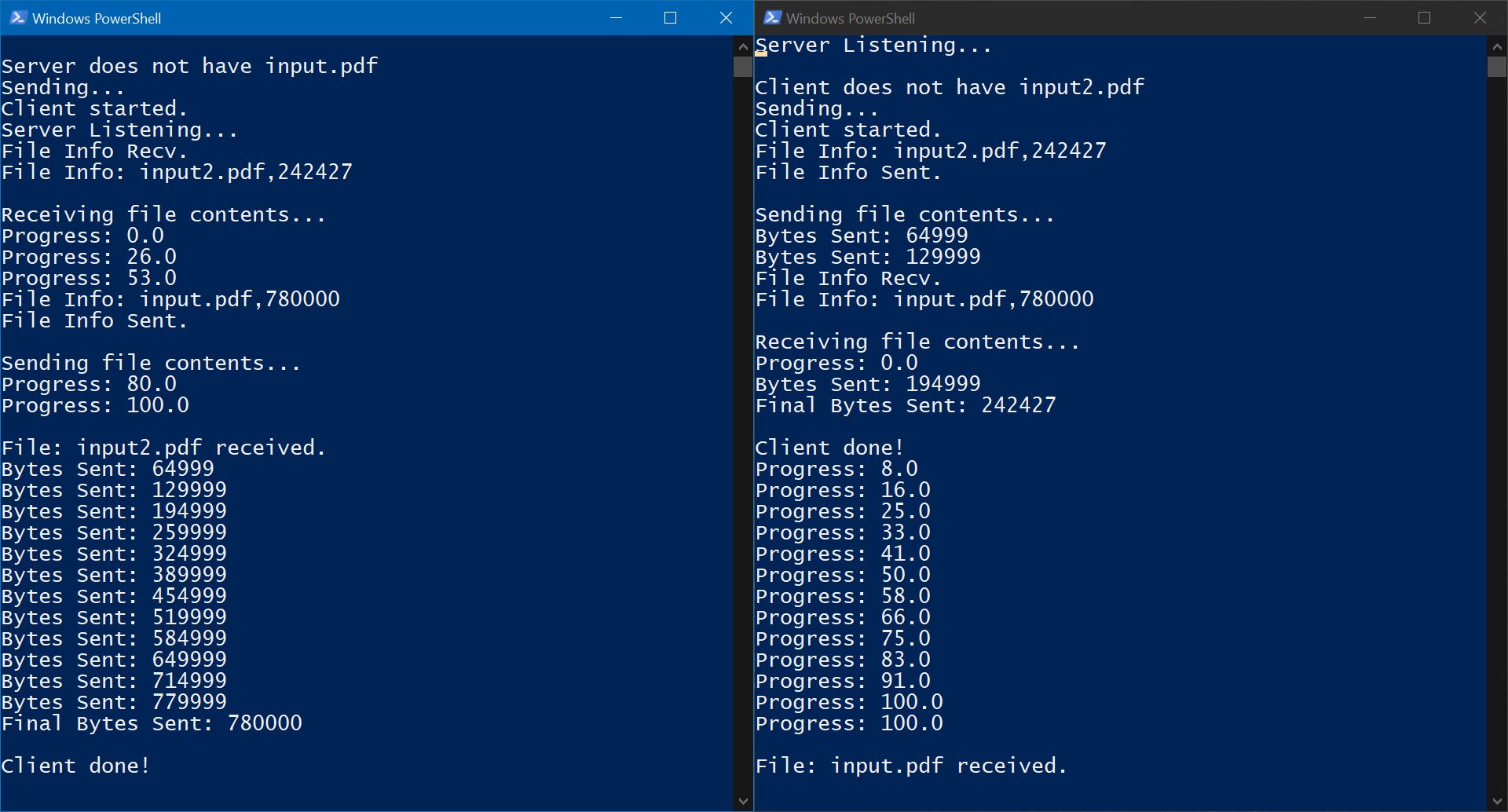


Fig: UDP Client-Server checking & transferring file in Full Duplex mode.

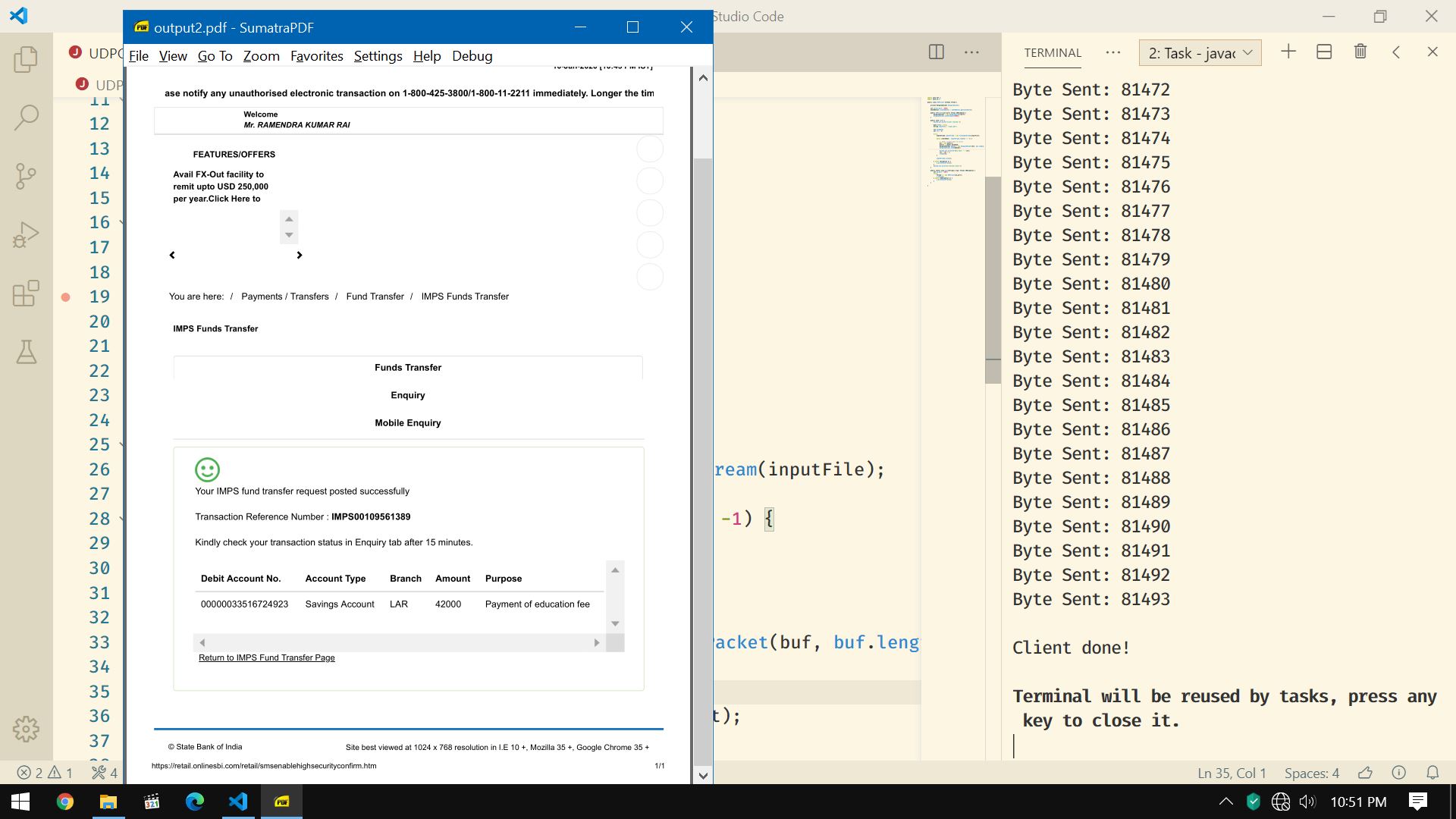


Fig: UDP Client-Server file contents after transfer.

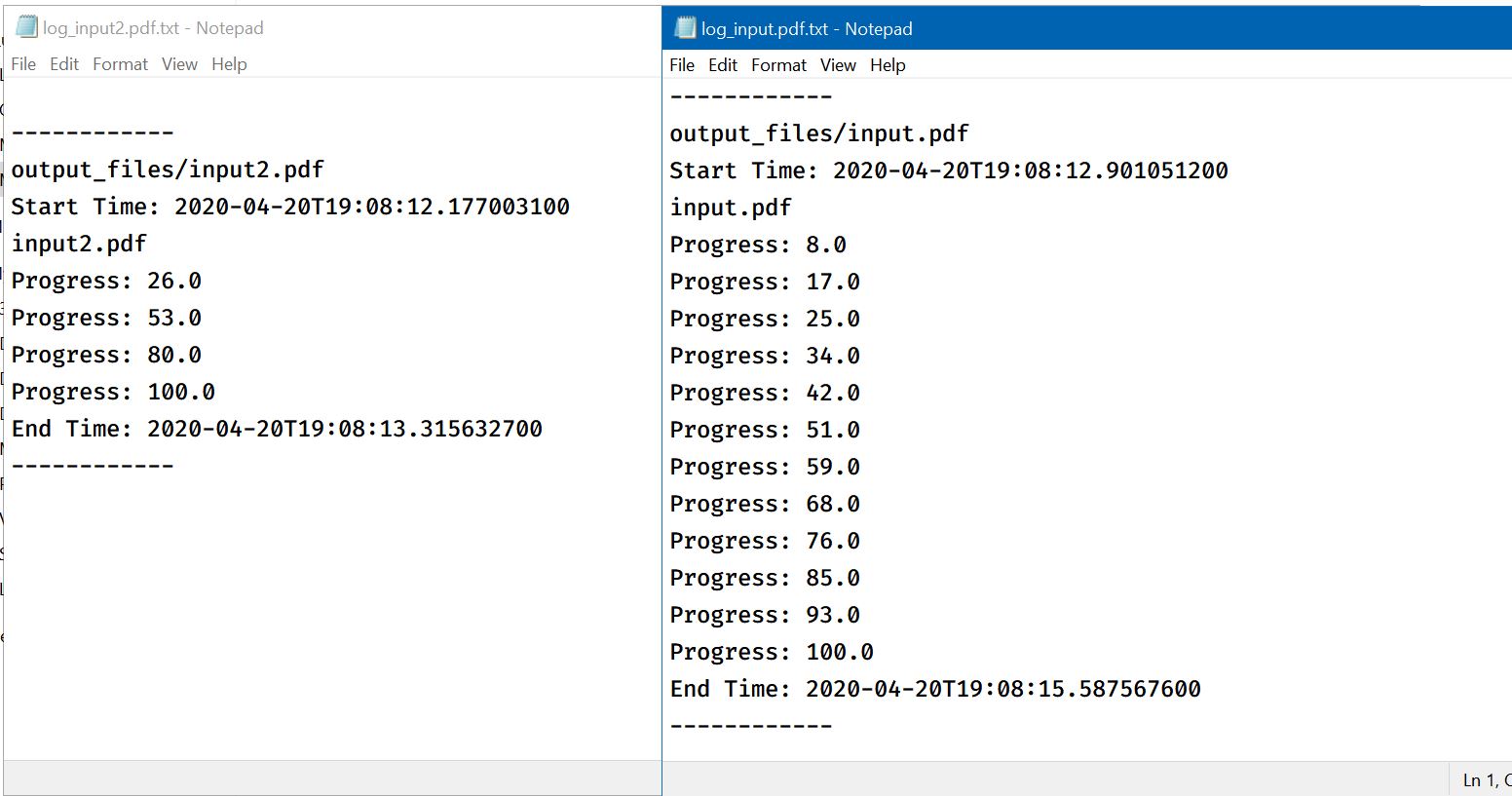


Fig: Contents of Log files.

**RESULT:**

Thus both the client and server exchange files using UDP along with generation of log files.