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:

<u>Server</u>

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

<u>Client</u>

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

PROGRAM:

//UDPClient.java

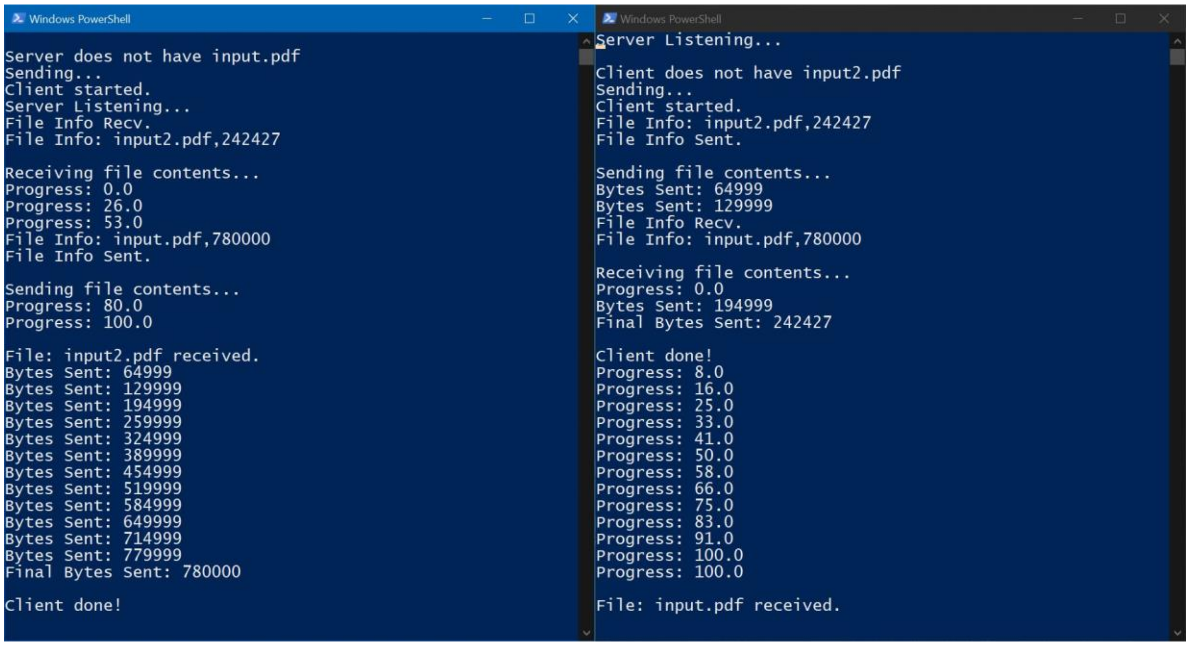
```
import java.net.*;
import java.io.*;

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 IOExce
ption {
        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, clien
tAddress, 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, clientAddres
s, 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, clientAdd
ress, 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

```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;
```

```java
        while (a[i] != 0) {
            ret.append((char) a[i]);
            i++;
        }
        return ret;
    }

}
```

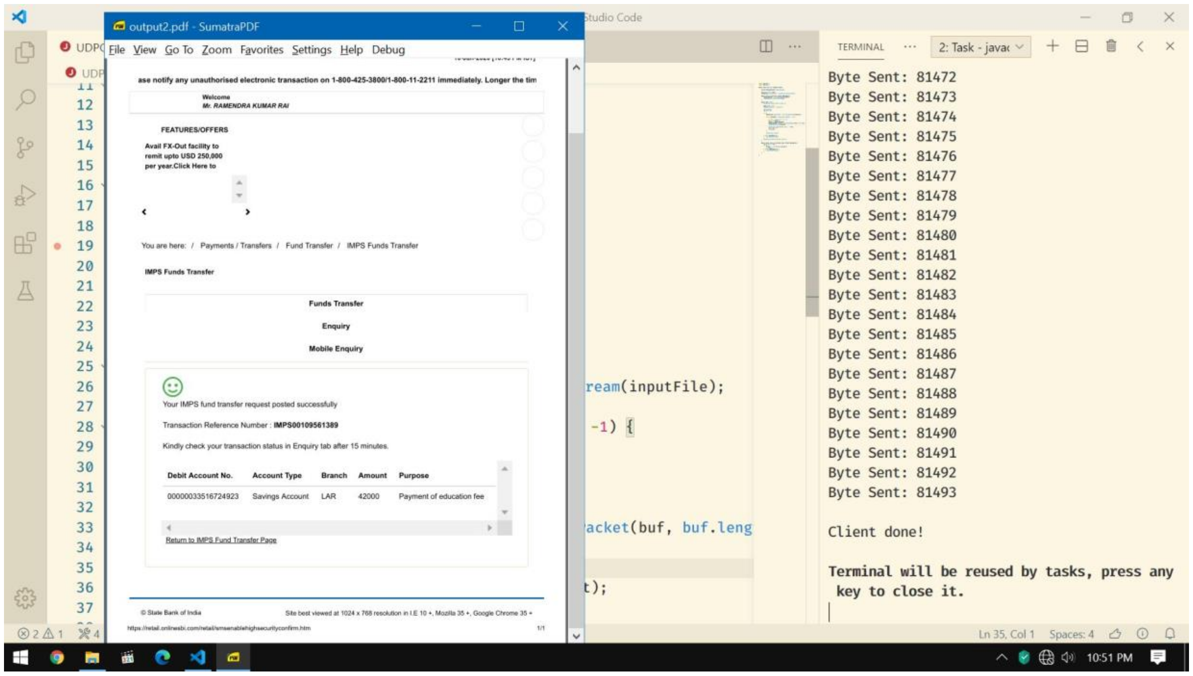
OUTPUT



```
Server does not have input.pdf
Sending...
Client started.
Server Listening...
File Info Recv.
File Info: input2.pdf,242427

Receiving file contents...
Progress: 0.0
Progress: 26.0
Progress: 53.0
File Info: input.pdf,780000
File Info Sent.

Sending file contents...
Progress: 80.0
Progress: 100.0

File: input2.pdf received.
Bytes Sent: 64999
Bytes Sent: 129999
Bytes Sent: 194999
Bytes Sent: 259999
Bytes Sent: 324999
Bytes Sent: 389999
Bytes Sent: 454999
Bytes Sent: 519999
Bytes Sent: 584999
Bytes Sent: 649999
Bytes Sent: 714999
Bytes Sent: 779999
Final Bytes Sent: 780000

Client done!
```

```
Server Listening...

Client does not have input2.pdf
Sending...
Client started.
File Info: input2.pdf,242427
File Info Sent.

Sending file contents...
Bytes Sent: 64999
Bytes Sent: 129999
File Info Recv.
File Info: input.pdf,780000

Receiving file contents...
Progress: 0.0
Bytes Sent: 194999
Final Bytes Sent: 242427

Client done!
Progress: 8.0
Progress: 16.0
Progress: 25.0
Progress: 33.0
Progress: 41.0
Progress: 50.0
Progress: 58.0
Progress: 66.0
Progress: 75.0
Progress: 83.0
Progress: 91.0
Progress: 100.0
Progress: 100.0

File: input.pdf received.
```

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



ase notify any unauthorised electronic transaction on 1-800-425-3800/1-800-11-2211 immediately. Longer the tim

Welcome
Mr. RAMENDRA KUMAR RAI

FEATURES/OFFERS

Avail FX-Out facility to remit upto USD 250,000 per year.Click Here to

You are here: / Payments / Transfers / Fund Transfer / IMPS Funds Transfer

IMPS Funds Transfer

Funds Transfer

Enquiry

Mobile Enquiry

Your IMPS fund transfer request posted successfully

Transaction Reference Number : **IMPS00109561389**

Kindly check your transaction status in Enquiry tab after 15 minutes.

Debit Account No.	Account Type	Branch	Amount	Purpose
00000033516724923	Savings Account	LAR	42000	Payment of education fee

Return to IMPS Fund Transfer Page

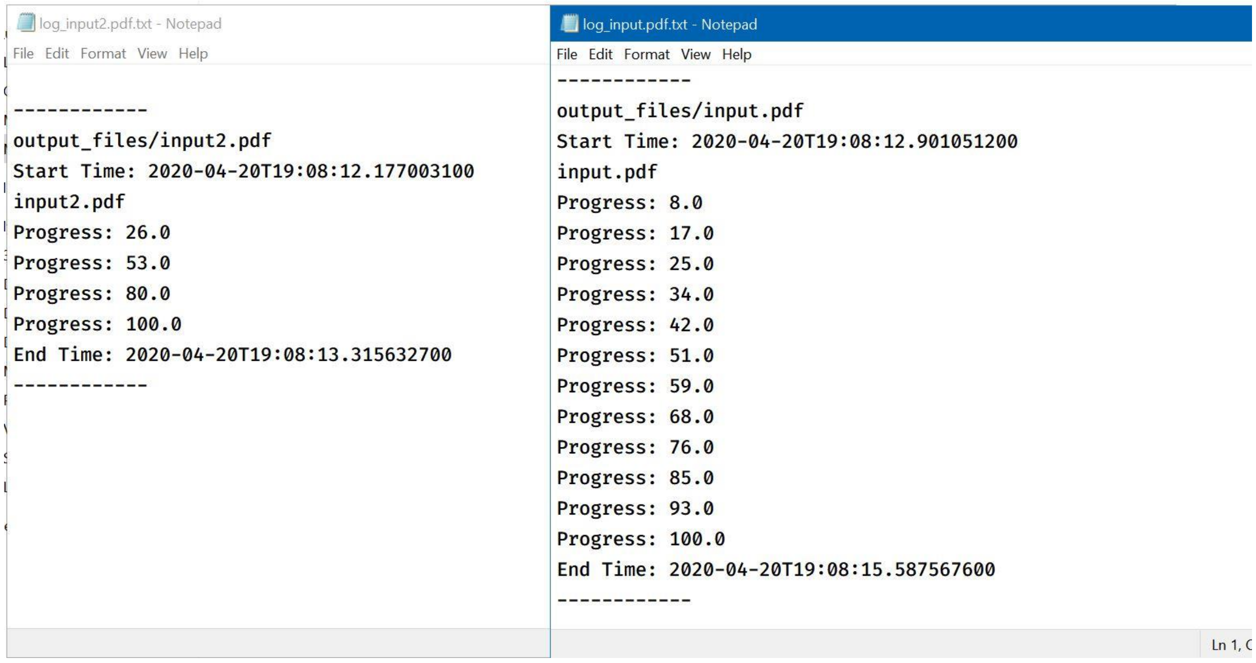
© State Bank of India   Site best viewed at 1024 x 768 resolution in I.E 10 +, Mozilla 35 +, Google Chrome 35 +

https://retail.onlinesbi.com/retail/smsenablehighsecurityconfirm.htm   1/1

```
ream(inputFile);
-1) {
acket(buf, buf.leng
t);
```

```
Byte Sent: 81472
Byte Sent: 81473
Byte Sent: 81474
Byte Sent: 81475
Byte Sent: 81476
Byte Sent: 81477
Byte Sent: 81478
Byte Sent: 81479
Byte Sent: 81480
Byte Sent: 81481
Byte Sent: 81482
Byte Sent: 81483
Byte Sent: 81484
Byte Sent: 81485
Byte Sent: 81486
Byte Sent: 81487
Byte Sent: 81488
Byte Sent: 81489
Byte Sent: 81490
Byte Sent: 81491
Byte Sent: 81492
Byte Sent: 81493

Client done!

Terminal will be reused by tasks, press any
 key to close it.
```

Fig: UDP Client-Server file contents after transfer.


log_input2.pdf.txt - Notepad
File Edit Format View Help

```
------------
output_files/input2.pdf
Start Time: 2020-04-20T19:08:12.177003100
input2.pdf
Progress: 26.0
Progress: 53.0
Progress: 80.0
Progress: 100.0
End Time: 2020-04-20T19:08:13.315632700
------------
```

log_input.pdf.txt - Notepad
File Edit Format View Help

```
------------
output_files/input.pdf
Start Time: 2020-04-20T19:08:12.901051200
input.pdf
Progress: 8.0
Progress: 17.0
Progress: 25.0
Progress: 34.0
Progress: 42.0
Progress: 51.0
Progress: 59.0
Progress: 68.0
Progress: 76.0
Progress: 85.0
Progress: 93.0
Progress: 100.0
End Time: 2020-04-20T19:08:15.587567600
------------
```

Fig: Contents of Log files.

RESULT:

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