Assignment 3
Of
Network & Distributed System Lab (CS2051)
Masters of Technology in Computer Science And Engineering

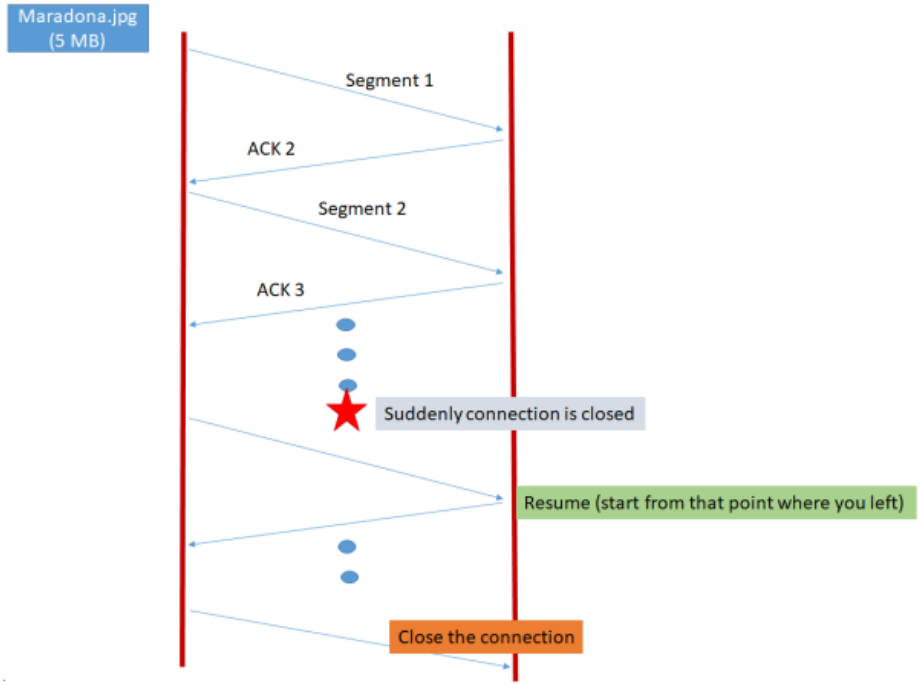
submitted to
Dr Sujoy Saha
Assistant Professor
&
Dr Suvrojit Das
Associate Professor
Dept. of CSE

National Institute of Technology, Durgapur

submitted by
Arghya Bandyopadhyay
RollNo. 20CS4103

14 June 2021

Objective: Implement naïve flow control mechanism using stop & wait protocol. Transfer files (Text, Image, Audio, Video) using TCP and UDP protocol. If during the connection suddenly connection is terminated then you have start ones again, it simply resume the process not start from being.



Write a socket program in Java for Multimodal File Transmission using TCP and UDP with **Full-Duplex Stop and Wait protocol**. The program/protocol should support the following properties/mechanism

1. The protocol will send any type of files
2. Each packet should consist of the file name, sequence number/Acknowledgement number
3. A log file should be generated with some information like,

List of uncommon files in server and client which are to be transferred, Start time, If the connection is broken then the % of the file already uploaded, How many times connections were established during the complete transmission, End time (when the file is fully transmitted), **How many packets are lost, How many time-outs are occurred, etc.**

Answer.

The following code is the implementation for Multimodal File Transmission using TCP and UDP with **Full-Duplex Stop and Wait protocol**.

Java Code:

```java
//Receiver Class Code
import java.net.*;
import java.io.*;
import java.util.*;
public class ReceiverClass extends Thread {
    private DatagramSocket datagramSocket;
    private byte[] receive = new byte[65000];
    private DatagramPacket DpReceive = null;
    private String myDir;
    private int client_port;

    public ReceiverClass(int port, String my_dir, int c_port) throws IOException {
        datagramSocket = new DatagramSocket(port);
        datagramSocket.setSoTimeout(20000);
        myDir = my_dir;
        client_port = c_port;
    }

    public void run() {
        System.out.println("\nReceiver thread listening...");

        String outputFile;
        String logFile;
        String[] fileInfo;
        String startTime, endTime;
        int len;

        try {
            while (true) {
                int packets_recv = 0;
                int ack_sent = 0;

                // Receive fileInfo
                DpReceive = new DatagramPacket(receive, receive.length);
                datagramSocket.receive(DpReceive);
                fileInfo = data(receive).toString().split(",");
                receive = new byte[65000];
                packets_recv = packets_recv + 1;
                sleep(10);
                System.out.println("File Info Recv.");
                System.out.println("File Info: " + fileInfo[0] + "," + fileInfo[1]);

                // Send ACK to client
                InetAddress clientAddress = InetAddress.getLocalHost();
                byte[] buf = ("--ACK--").getBytes();
                DatagramPacket DpSend = new DatagramPacket(buf, buf.length, clientAddress, client_port);
                datagramSocket.send(DpSend);
```

```
                ack_sent = ack_sent + 1;
                sleep(2);

                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 = myDir + 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(("\nPriority: " + FileComparator.getPriority(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);
                    packets_recv = packets_recv + 1;

                    // Write to output file
                    outputStream.write(receive);
                    receive = new byte[65000];
                    sleep(2);

                    cnt = cnt + 65000;

                    // Send ACK to client
                    buf = ("ACK#" + (int) (cnt / 65000)).getBytes();
                    DpSend = new DatagramPacket(buf, buf.length, clientAddress, client_port);
                    datagramSocket.send(DpSend);
                    ack_sent = ack_sent + 1;

                    // Display progress
                    perc = (double) ((cnt * 100) / len);
                    if (perc > 100)
                        perc = 100;
                    System.out.println("Progress: " + perc + " Packet #" + (int) (cnt / 65000));

                    // Write to log file
                    logStream.write(("\nProgress: " + perc + " Packet #" + (int) (cnt / 65000)).getBytes());
```

```
                }

                endTime = java.time.LocalDateTime.now().toString();
                logStream.write(("\nEnd Time: " + endTime).getBytes());
                logStream.write(("\nNo. of packets recv: " + packets_recv).getBytes());
                logStream.write(("\nNo. of ACK sent: " + ack_sent).getBytes());
                logStream.write(("\n*Note: 1 packet is for file info.").getBytes());
                logStream.write("\n------------".getBytes());

                logStream.close();
                outputStream.close();

                System.out.println("\nFile: " + fileInfo[0] + " received.");
                System.out.println("\nFor this file:");
                System.out.println("No. of packets recv: " + packets_recv);
                System.out.println("No. of ACK sent: " + ack_sent);
            }

        } catch (Exception e) {
            // e.printStackTrace();
            System.out.println("\nClosing receiver due to in-activity.");
        }
        System.out.println("\nReceiver done.\n");
    }

    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;
    }
}
```

```java
//Sender class code
import java.net.*;
import java.io.*;

public class SenderClass extends Thread {

    private DatagramSocket datagramSocket;

    int server_port;
    InetAddress clientAddress = InetAddress.getLocalHost();

    String inputFile;

    public SenderClass(int port, String fileName, int serverPort) throws IOException {
        datagramSocket = new DatagramSocket(port);
        datagramSocket.setSoTimeout(8000);
        server_port = serverPort;

        inputFile = fileName;
    }

    public void run() {
        System.out.println("\nSender thread started.");

        byte buf[] = null;

        int byteRead;
        int cnt = 0;
        buf = new byte[65000];

        try {
            sleep(3000);
            int packets_sent = 0;
            int ack_recv = 0;
            int packets_req = 0;
            int failed_tries = 0;

            // Calculate file name and file size
            File f = new File(inputFile);
            long fileSize = f.length();
            String fileInfo = f.getName() + "," + fileSize;

            DatagramPacket DpSend;
            System.out.println("File Info: " + fileInfo);
            packets_req = packets_req + 1;
            while (true) {
                // Send fileInfo
                System.out.println("Sending info...");
                buf = fileInfo.getBytes();
                DpSend = new DatagramPacket(buf, buf.length, clientAddress, server_port);
                datagramSocket.send(DpSend);
                packets_sent = packets_sent + 1;

```

```
                // sleep(1);

                try {
                    // Receive ACK from Server
                    byte[] receive = new byte[65000];
                    DatagramPacket DpReceive = new DatagramPacket(receive, receive.length);
                    datagramSocket.receive(DpReceive);
                    System.out.println("Got " + data(receive));

                    ack_recv = ack_recv + 1;
                    sleep(10);
                    break;
                } catch (Exception e) {
                    System.out.println("ACK not received. Re-sending...");
                    if (failed_tries == 10) {
                        System.out.println("\nTimeout occurred after 10 reties. Aborting.");
                        break;
                    }
                    failed_tries = failed_tries + 1;
                }

            }
            if (failed_tries == 10)
                return;
            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 packet until ACK is received or failed tries == 10
                    packets_req = packets_req + 1;
                    failed_tries = 0;
                    while (true) {
                        // Send 65000 bytes to server
                        DpSend = new DatagramPacket(buf, buf.length, clientAddress, server_port);
                        datagramSocket.send(DpSend);
                        packets_sent = packets_sent + 1;

                        // sleep(1);

                        try {
                            // Receive ACK from Server
                            byte[] receive = new byte[65000];
                            DatagramPacket DpReceive = new DatagramPacket(receive, receive.length);
                            datagramSocket.receive(DpReceive);
                            System.out.println("Got " + data(receive));
                            ack_recv = ack_recv + 1;
```

```
                    sleep(10);
                    break;
                } catch (Exception e) {
                    System.out.println("ACK not received. Re-sending...");
                    if (failed_tries == 10) {
                        System.out.println("\nTimeout occurred after 10 reties. Aborting.");
                        break;
                    }
                    failed_tries = failed_tries + 1;
                }

            }
            if (failed_tries == 10) {
                // System.out.println("\nTimeout occurred after 10 reties. Aborting.");
                break;
            }
            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();

    System.out.println("\nFor this file:");
    System.out.println("No. of packets req to send: " + packets_req);
    System.out.println("No. of packets sent: " + packets_sent);
    System.out.println("No. of ACK received: " + ack_recv);
    System.out.println("No. of packets lost: " + (packets_sent - packets_req));

    // Append packets info to log file
    String logFile = "log_" + f.getName() + ".txt";
    FileWriter f2 = new FileWriter(logFile, true);
    BufferedWriter b2 = new BufferedWriter(f2);
    PrintWriter p2 = new PrintWriter(b2);
    p2.println("\nNo. of packets req to send: " + packets_req);
    p2.println("No. of packets sent: " + packets_sent);
    p2.println("No. of ACK received: " + ack_recv);
    p2.println("No. of packets lost: " + (packets_sent - packets_req));

    p2.close();
    b2.close();
    f2.close();

    datagramSocket.close();

```

```
160         } catch (Exception e) {
161             e.printStackTrace();
162             System.out.println("\nTimeout occurred after many reties. Aborting.");
163         }
164         System.out.println("\nSender thread done!");
165     }
166 
167     public static StringBuilder data(byte[] a) {
168         if (a == null)
169             return null;
170         StringBuilder ret = new StringBuilder();
171         int i = 0;
172         while (a[i] != 0) {
173             ret.append((char) a[i]);
174             i++;
175         }
176         return ret;
177     }
178 
179 }
```

```java
//FileComparator Class Code
import java.util.*;

public class FileComparator implements Comparator<String> {

    // Overriding compare method
    public int compare(String s1, String s2) {
        if (getPriority(s1) > getPriority(s2))
            return 1;
        else if (getPriority(s1) < getPriority(s2))
            return -1;
        return 0;
    }

    // Function to get priority of a file
    public static int getPriority(String fName) {
        // Document files
        if (fName.contains(".txt"))
            return 10;
        else if (fName.contains(".doc"))
            return 11;
        else if (fName.contains(".docx"))
            return 12;
        else if (fName.contains(".pdf"))
            return 13;
        else if (fName.contains(".ppt"))
            return 14;
        else if (fName.contains(".pptx"))
            return 15;
        // Audio files
        else if (fName.contains(".mp3"))
            return 30;
        else if (fName.contains(".aac"))
            return 31;
        else if (fName.contains(".flac"))
            return 32;
        else if (fName.contains(".wav"))
            return 33;
        // Video files
        else if (fName.contains(".mp4"))
            return 40;
        else if (fName.contains(".mkv"))
            return 41;
        else if (fName.contains(".avi"))
            return 42;
        // Other files
        return 99;
    }
}
```

```java
//Server class code
import java.io.*;
import java.util.*;

public class Server {

    public static void main(String[] args) {
        int my_port = 56070;
        int client_port = 56060;
        try {
            System.out.println("This is Server.java");

            // Thread for receiving file
            Thread t = new ReceiverClass(my_port, "server_files/", client_port);
            t.start();

            // List of my files
            File curDir = new File("./server_files");
            String fileList = getAllFiles(curDir);
            String files[] = fileList.split(",");

            File curDir2 = new File("./client_files");
            String fileList2 = getAllFiles(curDir2);

            PriorityQueue<String> pq = new PriorityQueue<String>(50, new FileComparator());

            // Prioritizing files
            System.out.println("\nPrioritizing files...\n");
            for (String f : files) {
                if (!fileList2.contains(f)) {
                    System.out.println("Client does not have " + f);
                    pq.add(f);
                } else {
                    if (!f.equalsIgnoreCase(""))
                        System.out.println("Client already has " + f);
                }
            }

            // Sending files
            System.out.println("\nNow begin sending...");
            while (!pq.isEmpty()) {
                String f = pq.poll();
                System.out.println("Sending: " + f);

                Thread t2 = new SenderClass(my_port + 1, "./server_files/" + f, client_port + 1);
                t2.start();
                t2.join();
                t2.interrupt();
            }
            // System.out.println("\nSending files from server done.");

        } catch (Exception e) {
            e.printStackTrace();
```

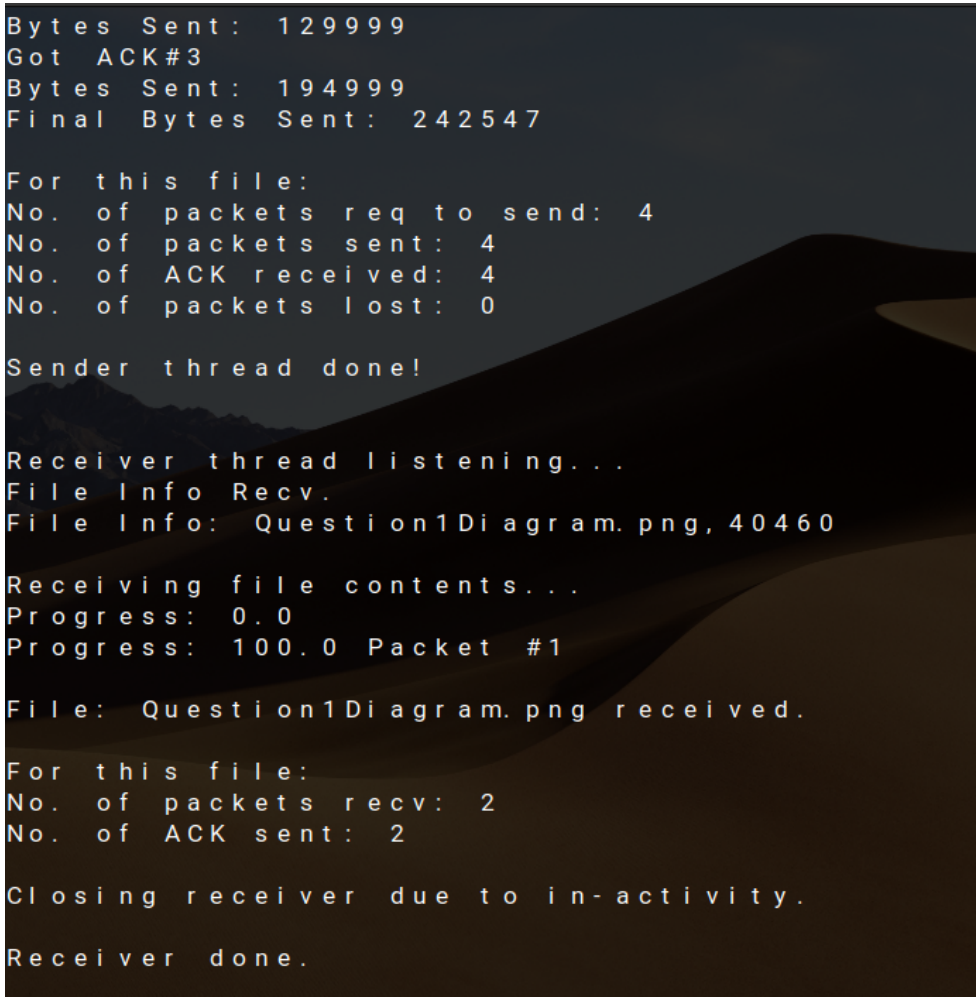
```
54          }
55      }
56
57      private static String getAllFiles(File curDir) {
58
59          String files = "";
60
61          File[] filesList = curDir.listFiles();
62          for (File f : filesList) {
63              if (f.isDirectory())
64                  getAllFiles(f);
65              if (f.isFile()) {
66                  files = files + "," + f.getName();
67                  // System.out.println(f.getName());
68              }
69          }
70          return files;
71      }
72
73  }
```

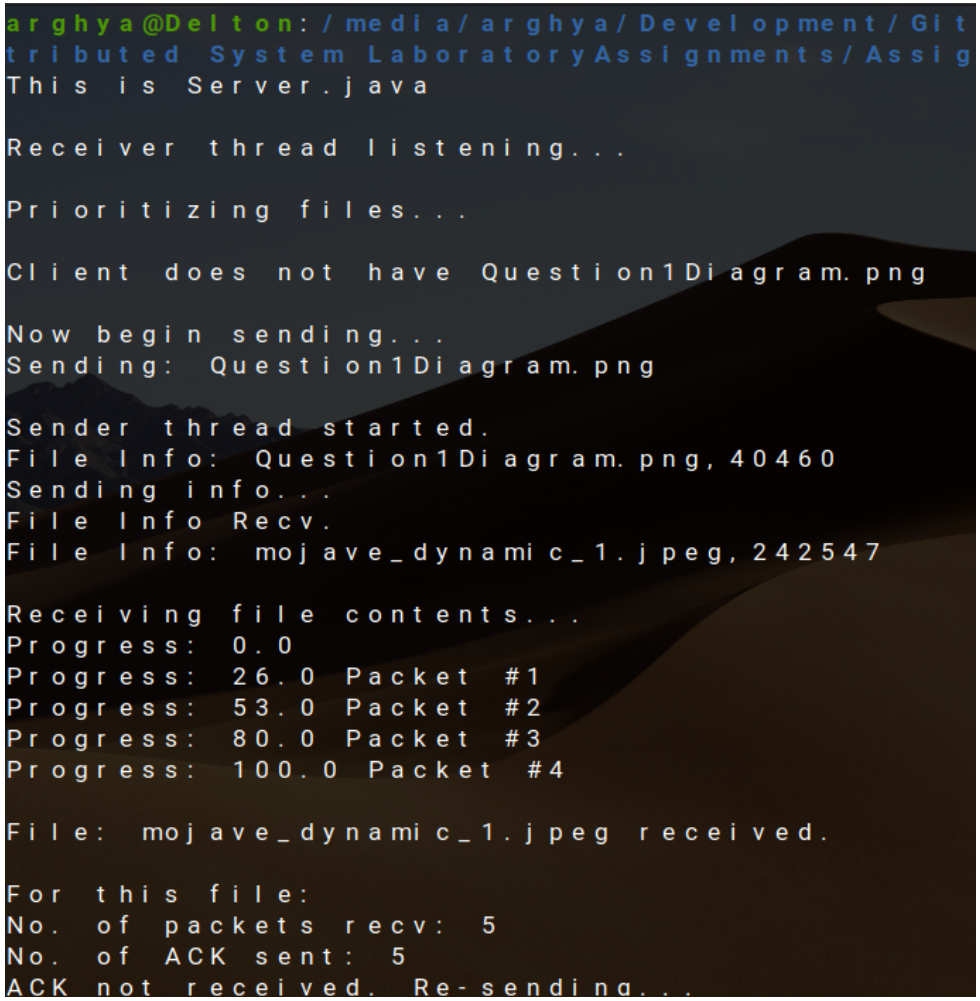
```java
//Client class code
import java.io.*;
import java.util.*;

public class Client {

    public static void main(String[] args) throws Exception {
        int my_port = 56060;
        int server_port = 56070;
        try {
            System.out.println("This is Client.java");

            // List of my files
            File curDir = new File("./client_files");
            String fileList = getAllFiles(curDir);
            String files[] = fileList.split(",");

            File curDir2 = new File("./server_files");
            String fileList2 = getAllFiles(curDir2);

            PriorityQueue<String> pq = new PriorityQueue<String>(50, new FileComparator());

            // Prioritizing files
            System.out.println("\nPrioritizing files...\n");
            for (String f : files) {
                if (!fileList2.contains(f)) {
                    System.out.println("Server does not have " + f);
                    pq.add(f);
                } else {
                    if (!f.equalsIgnoreCase(""))
                        System.out.println("Server already has " + f);
                }
            }

            // Sending files
            System.out.println("\nNow begin sending...");
            while (!pq.isEmpty()) {
                String f = pq.poll();
                System.out.println("Sending: " + f);

                Thread t = new SenderClass(my_port, "./client_files/" + f, server_port);
                t.start();
                t.join();
                t.interrupt();
            }
            System.out.println("");
            // System.out.println("\nSending files from client done.");

            // Thread for receiving file
            Thread t2 = new ReceiverClass(my_port + 1, "client_files/", server_port + 1);
            t2.start();
        } catch (IOException e) {
            e.printStackTrace();
```

```
54          }
55      }
56
57      private static String getAllFiles(File curDir) {
58
59          String files = "";
60
61          File[] filesList = curDir.listFiles();
62          for (File f : filesList) {
63              if (f.isDirectory())
64                  getAllFiles(f);
65              if (f.isFile()) {
66                  files = files + "," + f.getName();
67                  // System.out.println(f.getName());
68              }
69          }
70          return files;
71      }
72  }
```

Output:


arghya@Delton:/media/arghya/Development/GitHub/Distributed System LaboratoryAssignments/Assig
This is Client.java
Prioritizing files...
Server does not have mojave_dynamic_1.jpeg
Now begin sending...
Sending: mojave_dynamic_1.jpeg
Sender thread started.
File Info: mojave_dynamic_1.jpeg,242547
Sending info...
Got --ACK--
File Info Sent.
Sending file contents...
Got ACK#1
Bytes Sent: 64999
Got ACK#2
Bytes Sent: 129999
Got ACK#3
Bytes Sent: 194999
Final Bytes Sent: 242547
For this file:
No. of packets req to send: 4
No. of packets sent: 4
No. of ACK received: 4
No. of packets lost: 0


Bytes Sent: 129999
Got ACK#3
Bytes Sent: 194999
Final Bytes Sent: 242547
For this file:
No. of packets req to send: 4
No. of packets sent: 4
No. of ACK received: 4
No. of packets lost: 0
Sender thread done!
Receiver thread listening...
File Info Recv.
File Info: Question1Diagram.png,40460
Receiving file contents...
Progress: 0.0
Progress: 100.0 Packet #1
File: Question1Diagram.png received.
For this file:
No. of packets recv: 2
No. of ACK sent: 2
Closing receiver due to in-activity.
Receiver done.


arghya@Delton:/media/arghya/Development/GitHub/Distributed System LaboratoryAssignments/Assig
This is Server.java
Receiver thread listening...
Prioritizing files...
Client does not have Question1Diagram.png
Now begin sending...
Sending: Question1Diagram.png
Sender thread started.
File Info: Question1Diagram.png,40460
Sending info...
File Info Recv.
File Info: mojave_dynamic_1.jpeg,242547
Receiving file contents...
Progress: 0.0
Progress: 26.0 Packet #1
Progress: 53.0 Packet #2
Progress: 80.0 Packet #3
Progress: 100.0 Packet #4
File: mojave_dynamic_1.jpeg received.
For this file:
No. of packets recv: 5
No. of ACK sent: 5
ACK not received. Re-sending...


Receiving file contents...
Progress: 0.0
Progress: 26.0 Packet #1
Progress: 53.0 Packet #2
Progress: 80.0 Packet #3
Progress: 100.0 Packet #4
File: mojave_dynamic_1.jpeg received.
For this file:
No. of packets recv: 5
No. of ACK sent: 5
ACK not received. Re-sending...
Sending info...
Got --ACK--
File Info Sent.
Sending file contents...
Final Bytes Sent: 40460
For this file:
No. of packets req to send: 1
No. of packets sent: 2
No. of ACK received: 1
No. of packets lost: 1
Sender thread done!
Closing receiver due to in-activity.
Receiver done.

Log File

1. Log File for the files in client files.

```
1
2 ------------
3 server_files/mojave_dynamic_1.jpeg
4 Priority: 99
5 Start Time: 2021-06-14T12:11:45.593714
6 mojave_dynamic_1.jpeg
7 Progress: 26.0 Packet #1
8 Progress: 53.0 Packet #2
9 Progress: 80.0 Packet #3
10 Progress: 100.0 Packet #4
11 End Time: 2021-06-14T12:11:46.033982
12 No. of packets recv: 5
13 No. of ACK sent: 5
14 *Note: 1 packet is for file info.
15 ------------
16 No. of packets req to send: 4
17 No. of packets sent: 4
18 No. of ACK received: 4
19 No. of packets lost: 0
```

2. Log File for the files in server files.

```
1
2 ------------
3 client_files/Question1Diagram.png
4 Priority: 99
5 Start Time: 2021-06-14T12:11:50.982433
6 Question1Diagram.png
7 Progress: 100.0 Packet #1
8 End Time: 2021-06-14T12:11:51.088751
9 No. of packets recv: 2
10 No. of ACK sent: 2
11 *Note: 1 packet is for file info.
12 ------------
```