Δίκτυα Υπολογιστών Ι

Θωμάς Πλιάκης
tpliakis@ece.auth.gr
AEM: 9018

February 2, 2023

```
import java.io.*;
import java.util.*;
import ithakimodem.*;

/* My virtual modem class */
public class myVirtualModem {

    int speed = 50000; // modem speed
    int timeOut = 240000; // 4 minutes per session
    private Modem modem; // my modem instance

    /* Main function where i call everything and save results */
    public static void main(String[] args) throws InterruptedException {
        myVirtualModem userApplication = new myVirtualModem();

        userApplication.echoPackage("E3181\r","/home/teras/IdeaProjects/Computer_Networks_I_Project/
        results/echo_package_time_results");
        userApplication.Image_request("M1902\r","/home/teras/IdeaProjects/Computer_Networks_I_Project/
        results/errorFreeImage.png" );
        userApplication.Image_request("G3585\r","/home/teras/IdeaProjects/Computer_Networks_I_Project/
        results/errorImage.png" );
        userApplication.arqRequest("Q6322\r",
            "R9512\r","/home/teras/IdeaProjects/Computer_Networks_I_Project/
        results/Arq.txt","/home/teras/IdeaProjects/Computer_Networks_I_Project/
        results/nackResults.txt" );
        userApplication.Image_request("P5191T=225735403737T=225740403735T=225742403733T=225735403733\r",
            "/home/teras/IdeaProjects/Computer_Networks_I_Project/
        results/gpsImage.jpeg");
        userApplication.Gps_request("P5191R=1000135\r","/home/teras/IdeaProjects/Computer_Networks_I_Project
        results/gpsPackages.txt" );

    }

    // Iniatilize my virtual modem every time i want to do something
    public void myVirtualModem() {

        int k;
        String rxmessage = ""; // buffer for writing the message from ithaki

        modem = new Modem();
        modem.setSpeed(speed);
        modem.setTimeout(timeOut);
        modem.open("ithaki");  // action to talk too ithaki
        modem.write("atd2310ithaki\r".getBytes()); // Connect local modem to remote
            ithaki modem.

        //System.out.println("end of welcome message");
        // Welcome message from ithaki when connected or error
        for (; ; ) {
            try {
```

```
            k = modem.read(); // k 0-255, blocking function
            if (k == -1) { // something went wrong
                System.out.println("error message");
                break;
            }
            //System.out.print((char) k);
            rxmessage = rxmessage + (char) k;
            if (rxmessage.indexOf("\r\n\n\n") > -1) {     // delimeter for welcome
                message
                System.out.println("end of welcome message");
                break;
            }
        } catch (Exception x) {
            break;
        }
    }
    //modem.close();  // I dont close the modem here, i do it in the end of every
        function
}

// Function of echo package
public int echoPackage(String pack_code, String path) {
    System.out.println("EchoPackage");

    // Init virtual modem
    this.myVirtualModem();
    // IO to store results
    BufferedWriter bw = null;
    File file;
    FileWriter fw;

    try {
        // connection timeout for echo package
        long connectionStart = System.currentTimeMillis(); // Connection start here to
            stop when timeout time have passed
        long connectionFinish = connectionStart + timeOut; // 240000/1000 = 240
            seconds = 4 minutes
        long packageTxTime = 0, packageRxTime = 0; // transmit and receive times for a
            package
        int numOfPackages = 0; // Number of packages received
        long avgTime = 0; // Average time for packages
        String rxmessage = ""; // Buffer for storing each package
        int k; // buffer for each read

        // File and buffer IO to store results
        file = new File(path);
        fw = new FileWriter(file);
        bw = new BufferedWriter(fw);

        // While for capturing every package
        while ((System.currentTimeMillis() < connectionFinish) && (numOfPackages <
            9000)) { // Timeout and number of packages check
            rxmessage = "";
            // Write echo package code
            modem.write(pack_code.getBytes());
            packageTxTime = System.currentTimeMillis(); // Save current time
            for (; ; ) { // Read echo package
                try {
                    k = modem.read();
                    //System.out.print((char) k);
                    \begin{verbatim} rxmessage = rxmessage + (char) k;
                    if (rxmessage.indexOf("PSTOP") > -1) { // Package delimeter
                        //System.out.println("package is here");
                        break;
                    }
```

```
                    if (k == -1) {
                        System.out.println("Maybe the packet is here\n");
                        break;
                    }
                } catch (Exception x) {
                    System.out.println(x);
                    return 0;
                }
            }
            packageRxTime = System.currentTimeMillis(); // Save the finish time of
                receiving a package
            numOfPackages += 1; // Count the package received
            avgTime = avgTime + (packageRxTime - packageTxTime); // Add the time of
                every package
            String time = String.valueOf(packageRxTime - packageTxTime); // save the
                time in a string

            bw.write(time); // Write the time of the package
            bw.newLine(); // add a newline
        }
        avgTime = avgTime / numOfPackages; // Calculate average time
        // Print average time, number of packages and total time
        System.out.println("avg time is " + avgTime);
        System.out.println("number of packages received " + numOfPackages);
        System.out.println("in time " + avgTime * numOfPackages);

        // Write them in the file, do not needed anynore
        //bw.newLine();
        //bw.write("avg time is " + avgTime + "\n");
        //bw.write("number of packages received " + numOfPackages + "\n");
        //bw.write("in time " + avgTime * numOfPackages + "\n");

        // Flush and close bufferwriter
        bw.flush();
        bw.close();

        // Close modem
        modem.close();
    } catch (Exception x) {
        System.out.println("\nException in echoPackage! ");
        return 0;
    }
    return 0;
}

// Image request function
public void Image_request(String pack_code, String file_path) {
    System.out.println("Image Request");
    // Init virtual modem
    this.myVirtualModem();

    try {
        // IO to store images
        File file = new File(file_path);
        OutputStream image = new FileOutputStream(file);
        // Write image package code
        modem.write(pack_code.getBytes());
        String rxmessage = ""; // buffer for writing the message from ithaki
        int k;
        boolean flag = false;

        long packageTxTime = 0, packageRxTime = 0; // transmit and receive times for a
            package
        packageTxTime = System.currentTimeMillis(); // Save current time
```

```
            for (; ; ) {
                try {
                    k = modem.read(); // k 0-255, blocking function
                    if (k == -1) break;
                    rxmessage = rxmessage + (char) k;
                    //System.out.println(k);

                    if (rxmessage.indexOf(" ") > -1) { // Image start delimeter
                        image.write(255);
                        image.write(216);
                        System.out.println("start reading of image");
                        rxmessage = "";
                        flag = true;
                    }
                    if (flag) { // Check if we started getting the image
                        image.write(k);
                    }
                    if (rxmessage.indexOf(" ") > -1) { // Image end delimeter
                        System.out.println("end of image");
                        image.write(k);
                        break;
                    }
                } catch (Exception x) {
                    break;
                }
            }
            packageRxTime = System.currentTimeMillis(); // Save time when the image is
                 received
            System.out.println("Finished receiving the image after " + (packageRxTime -
                 packageTxTime) / 1000 + "seconds!"); // Print how much time it took

            // Close bufferwriter and virtual modem
            image.close();
            modem.close();
        } catch (Exception x) {
            System.out.println("Exception in Image request");
        }
    }

    // Gps request function
    public int Gps_request(String gps_code, String path) {
        System.out.println("Gps reuest");
        // Init virtual modem
        this.myVirtualModem();
        // Write image package code
        modem.write(gps_code.getBytes());
        // IO to store gps data
        BufferedWriter bw = null;
        File file;
        FileWriter fw;
        String rxmessage = ""; // buffer for writing the message from ithaki
        int k;

        try {
            // IO init
            file = new File(path);
            fw = new FileWriter(file);
            bw = new BufferedWriter(fw);

            // Reading of gps data
            for (; ; ) {
                try {
                    k = modem.read(); // k 0-255, blocking entolh
                    if (k == -1) break;
                    //System.out.print((char) k);
```

```
                rxmessage = rxmessage + (char) k;
                if (rxmessage.indexOf("STOP ITHAKI GPS TRACKING") > -1) { // GPS
                    delimeter
                    System.out.println("\nend of gps message");
                    break;
                }
            } catch (Exception x) {
                break;
            }
        }
        // Write and flush GPS data
        bw.write(rxmessage);
        bw.flush();
        bw.close();
        // Close modem
        modem.close();
    }catch (Exception e){
        System.out.println("\nException in echoPackage! ");
        return 0;
    }
    return 0;
}

// ARQ request function
public void arqRequest(String ackCode, String nackCode, String pathArq, String
    pathResults) {
    System.out.println("arq Request");
    // Init virtual modem
    this.myVirtualModem();

    // IO to store gps data
    BufferedWriter bwArq = null, bwResults = null;
    File fileArq, fileResults;
    FileWriter fwArq, fwResults;

    String rxmessage = ""; // buffer for writing the message from ithaki
    long connectionStart = System.currentTimeMillis();
    long connectionEnd = connectionStart + timeOut; // 240000/1000 = 240 seconds = 4
        minutes
    long packageTxTime = 0, packageRxTime = 0, packageTime; // transmit and receive
        times for a package
    int numOfAttemps = 0;
    boolean correctTransmit = true; // If the FCS == XOR result

    int k, fcs;
    String[] parseRxmessage; // Save parsed package to check if it is correct
    char[] xxx16;  // 16 byte message form package
    byte x; // to do the XOR check
    int numOfPackages = 0; // NUmber of packages

    try {
        // File and buffer io
        fileArq = new File(pathArq);
        fileResults = new File(pathResults);
        fwArq = new FileWriter(fileArq);
        bwArq = new BufferedWriter(fwArq);
        fwResults = new FileWriter(fileResults);
        bwResults = new BufferedWriter(fwResults);

        while (System.currentTimeMillis() < connectionEnd) {

            // If the previous package is correct then we ask for a new, else we write
                nack code to retransmit previous package
            if(correctTransmit){
                numOfAttemps = 1; // Save how many attempts we did for one package
```

```
                    packageTxTime = System.currentTimeMillis();
                    modem.write(ackCode.getBytes());
                }else {
                    numOfAttemps++; // increase if we have a wrong transmit
                    modem.write(nackCode.getBytes());
                }
                rxmessage = ""; // buffer for writing the message from ithaki
                for (;;) {
                    try {
                        k = modem.read(); // k 0-255, blocking function
                        //System.out.print((char) k);
                        rxmessage = rxmessage + (char) k;

                        if (rxmessage.indexOf("PSTOP") > -1 || k == -1) // Package delimeter
                            break;

                    } catch (Exception e) {
                        System.out.println("Catched exception e");
                        break;
                    }
                }
                //System.out.println(rxmessage);
                packageRxTime = System.currentTimeMillis(); // Save time of package arrival
                packageTime = packageRxTime - packageTxTime; // Calculate time
                parseRxmessage = rxmessage.split(" "); // Parse package based on spaces

                fcs = Integer.parseInt(parseRxmessage[5]); // FCS code is the 6th element
                //System.out.println(fcs);
                xxx16 = parseRxmessage[4].toCharArray(); // 16byte message is the 5th
                    element
                //System.out.println(xxx16);
                x = (byte) xxx16[1]; // But the message is in <16xxx> so we take the 2 to
                    17 element
                for(int i=2; i<xxx16.length-1; i++)
                    x = (byte) (x^xxx16[i]); // XOR bytes of message

                //System.out.println(x);
                if((int)x == fcs){ // Check if FCS == XOR in decimal
                    // If correct increase packages, save true transmit and save package
                    numOfPackages++;
                    correctTransmit = true;
                    bwResults.write((packageTime + " " + numOfAttemps + "\n"));
                    bwResults.flush();
                    bwArq.write(rxmessage + "\n");
                    bwArq.flush();
                    //arrayOfAttempts[numOfAttemps]++;
                }else{
                    // Else false transmit, save fals transmit
                    correctTransmit = false;
                    bwArq.write( rxmessage + " wrong package: " + parseRxmessage[3] + ") "
                        + "\n");
                    bwArq.flush();
                }
            }
            // Close bufferwritters
            bwArq.close();
            bwResults.close();

            // Close virtual modem
            modem.close();
        } catch (Exception e) {
            System.out.println("\nException in ArqRequest! ");
        }
    }
}
```