Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης
Πολυτεχνική Σχολή

Δίκτυα Υπολογιστών Ι

Θεόδωρος Κατζάλης
ΑΕΜ:9282
katzalis@auth.gr

April 18, 2021

Περιεχόμενα

1 Δομή του προγράμματος 2

2 UserApplication.java 2

3 Echo.java 5

4 Image.java 6

5 GPS.java 8

6 ARQ.java 10

7 Plots 13

1 Δομή του προγράμματος

```
src
   ARQ.java
   Echo.java
   GPS.java
   Image.java
   plot.ipynb
   UserApplication.java
```

2 UserApplication.java

```java
package src;

import ithakimodem.*;
import java.io.File;
import java.io.FileWriter;
import java.util.Date;
import java.util.Scanner;

class UserApplication {
  public static void main(String[] args) {
    printWelcome();

    Modem modem = new Modem();
    int speed = 80_000, timeout = 2_000;
    setupModem(modem, speed, timeout);
    startModem(modem, false);

    // Request codes

    String echoCode = "E0838";
    String imageNoErrorCode = "M8687";
    String imageWithErrorCode = "G1618";
    String gpsCode = "P3510";
    String gpsCodeComplete = gpsCode + "R=1000099";
    String ackCode = "Q1589";
    String nackCode = "R9523";

    //String cameraSuffix = "CAM=FIX";
    //String directionSuffix = "DIR=L";
    //imageNoErrorCode += cameraSuffix;
    //imageWithErrorCode += cameraSuffix;
    String enter = "\r";

    // write request to file
    try (FileWriter requests = new FileWriter(new File("logs/requests.txt"))) {
      requests.write("Echo: " + echoCode + "\n");
      requests.write("Image No Error: " + imageNoErrorCode + "\n");
      requests.write("Image Yes Error: " + imageWithErrorCode + "\n");
      requests.write("GPS: " + gpsCode + "\n");
      requests.write("GPS Full: " + gpsCodeComplete + "\n");
      requests.write("ACK: " + ackCode + "\n");
      requests.write("NACK: " + nackCode + "\n");
      requests.write("Time start: " + new Date());
```

```
    } catch (Exception x) {
      System.out.println(x);
    }

    // applications

    int minutes = 4;
    final int secondsPerMinute = 60;
    long timeInterval = minutes * secondsPerMinute;

    Echo.pstopRepeat(modem, echoCode + enter, timeInterval);

    Image.get(modem, imageNoErrorCode + enter);
    Image.get(modem, imageWithErrorCode + enter);

    Image.get(modem, imageNoErrorCode + "CAM=PTZ"+ enter);
    Image.get(modem, imageWithErrorCode + "CAM=PTZ" + enter);

    String maps_query = GPS.mergeDataPoints(modem, gpsCodeComplete + enter, 2);
    System.out.println("The GPS parameter " + maps_query);
    Image.get(modem, gpsCode + maps_query + enter);

    ARQ.arqRepeat(modem, ackCode + enter, nackCode + enter, timeInterval);

    modem.close();
  }

  /**
   * Read the welcome screen when opening the virtual modem
   *
   * @param modem The virtual opened modem
   * @param isPrinted Print the returned message to stdout
   */
  private static void startModem(Modem modem, boolean isPrinted) {
    final int finishReadingFlag = -1;

    int returnValueModem, finishCounter = 0;
    char returnCharModem = ' ';
    char[] finishReadingString = {'\r', '\n', '\n', '\n'};

    while (true) {
      try {
        returnValueModem = modem.read();
        returnCharModem = (char)returnValueModem;

        if (isPrinted) {
          System.out.print(returnCharModem);
          Thread.sleep(10);
        }

        // check for breaking flag
        // if (returnValueModem == finishReadingFlag) break;

        // check for breaking sequence
        if ((returnCharModem == finishReadingString[finishCounter])) {
          finishCounter++;
          if (finishCounter == 4)
            break;
        } else
          finishCounter = 0;
```

```
104
105       } catch (Exception x) {
106         System.out.println(x);
107         break;
108       }
109     }
110   }
111
112   /**
113    * Configure modem parameters
114    *
115    * @param modem The virtual opened modem
116    * @param speed The data speed of the communication
117    * @param timeout The time interval waiting for message
118    */
119   private static void setupModem(Modem modem, int speed, int timeout) {
120     modem.setSpeed(speed);
121     modem.setTimeout(timeout);
122     modem.open("ithaki");
123   }
124
125   /**
126    * Send "TEST" code and print output
127    *
128    * @param modem The virtual opened modem
129    */
130   private static void testModem(Modem modem) {
131     modem.write("TEST\r".getBytes());
132     while (true) {
133       int returnMessage = modem.read();
134       System.out.print((char)returnMessage);
135       if (returnMessage == -1)
136         break;
137     }
138   }
139
140   /**
141    * Print welcome ASCII text
142    */
143   private static void printWelcome() {
144     try {
145       Scanner welcome = new Scanner(new File("welcome.txt"));
146       while (welcome.hasNextLine())
147         System.out.println(welcome.nextLine());
148     } catch (Exception x) {
149       System.out.println(x + "\nWelcome text failed to open.");
150     }
151   }
152 }
```

3 Echo.java

```java
package src;
import ithakimodem.*;
import java.io.File;
import java.io.FileWriter;
import java.util.LinkedList;
import java.util.Queue;

public class Echo {

  /**
   * Echo packet contain info dependent of the request code.
   *
   * Stop communication when detect "PSTOP"
   *
   * @param modem The virtual opened modem
   * @param code Echo request code
   */
  public static String pstop(Modem modem, String code) {
    // System.out.println("Echo application");

    char returnModem = ' ';
    String message = "";
    Queue<Character> breakingChars = new LinkedList<>();

    modem.write(code.getBytes());

    while (true) {
      try {

        // all the echo packets ends to "PSTOP"
        if (breakingChars.size() != 5) {
          returnModem = (char)modem.read();
          breakingChars.add(returnModem);
        } else {
          String breakingString = queueChar2String(breakingChars);
          if (breakingString.equals("PSTOP"))
            break;
          else {
            breakingChars.remove();
            returnModem = (char)modem.read();
            breakingChars.add(returnModem);
          }
        }
      } catch (Exception x) {
        System.out.println(x);
      }

      message += returnModem;
    }

    return message;
  }

  /**
   * Stop communication when -1 returns
   *
   * @param modem
   * @param code
```

```
   */
  public static void generic(Modem modem, String code) {
    final int finishReadingFlag = -1;

    int returnValueModem = 0;
    char returnCharModem = ' ';

    modem.write(code.getBytes());
    while (true) {
      try {
        returnValueModem = modem.read();
        returnCharModem = (char)returnValueModem;

        System.out.print(returnCharModem);

        // check for breaking flag
        if (returnValueModem == finishReadingFlag)
          break;
      } catch (Exception x) {
        System.out.println(x);
      }
    }
    System.out.println();
  }

  public static void pstopRepeat(Modem modem, String code, long timeInterval) {
    float start = System.currentTimeMillis() / 1000f;
    int counter = 0;

    try (FileWriter echo = new FileWriter(new File("logs/echo.txt"))) {
      while ((System.currentTimeMillis() / 1000f - start) < timeInterval) {
        System.out.print("Packet No" + counter + ": ");

        long tic = System.currentTimeMillis();
        String message = Echo.pstop(modem, code);
        System.out.println(message);
        long toc = System.currentTimeMillis();

        System.out.println("Total time: " + (toc - tic) + " (ms)\n");
        counter += 1;

        echo.write((toc - tic) + "\n");
      }
    } catch (Exception x) {
      System.out.println(x);
    }
  }

  private static String queueChar2String(Queue<Character> queue) {
    String message = "";
    for (Character character : queue) {
      message += character;
    }
    return message;
  }
}
```

4 Image.java

```
package src;
```

```
import ithakimodem.*;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileOutputStream;

public class Image {
  /**
   * Create image file. The breaking flag to stop reading is the delimiter
   * "0xFFD9"
   *
   * @param modem The virtual opened modem
   * @param code Image request code
   */
  public static void get(Modem modem, String code) {
    System.out.println("\nImage application...");

    ByteArrayOutputStream buffer = new ByteArrayOutputStream();
    int returnValueModem = 0;
    byte first, second;

    long tic = System.currentTimeMillis();

    modem.write(code.getBytes());
    returnValueModem = modem.read();
    first = int0(returnValueModem);
    buffer.write(first);

    while (true) {
      try {
        returnValueModem = modem.read();
        second = int0(returnValueModem);
        buffer.write(second);

        //System.out.print(String.format("%02X", first));
        //System.out.print(String.format("%02X", second));

        if ((String.format("%02X", first).equals("FF")) &&
            (String.format("%02X", second)).equals("D9"))
          break;

        first = second;

      } catch (Exception x) {
        System.out.println(x);
      }
    }
    byte[] dataImage = buffer.toByteArray();

    // write image file

    String path = "";
    if (code.substring(0, 1).equals("P")) {
      path = "media/gps.jpg";
    } else if (code.substring(0, 1).equals("M")) {
      path = code.contains("PTZ") ? "media/image_error_free_ptz.jpg"
                                  : "media/image_error_free_fix.jpg";
    } else if (code.substring(0, 1).equals("G")) {
      path = code.contains("PTZ") ? "media/image_with_errors_ptz.jpg"
                                  : "media/image_with_errors_fix.jpg";
    } else {
```

```
      path = "image.jpg";
    }

    File image = new File(path);
    try (FileOutputStream fos = new FileOutputStream(image)) {
      fos.write(dataImage);
      System.out.println("File " + path + " has been created successfully");
    } catch (Exception x) {
      System.out.println(x);
    }

    long toc = System.currentTimeMillis();
    System.out.print("Total time creating image: " + (toc - tic) / 1000.0 +
                     " (s)\n");
  }

  /**
   * source:
   * https://stackoverflow.com/questions/1936857/convert-integer-into-byte-array-java
   */
  private static byte int3(int x) { return (byte)(x >> 24); }
  private static byte int2(int x) { return (byte)(x >> 16); }
  private static byte int1(int x) { return (byte)(x >> 8); }
  private static byte int0(int x) { return (byte)(x >> 0); }
}
```

5 GPS.java

```
package src;

import ithakimodem.*;
import java.io.File;
import java.io.FileWriter;
import java.util.ArrayList;

public class GPS {
  private static ArrayList<String> parser(Modem modem, String code) {
    ArrayList<String> coordinates = new ArrayList<String>();

    modem.write(code.getBytes());
    while (true) {
      try {
        String line = returnLine(modem);
        // System.out.println(line);

        // check for breaking flag
        if (line.equals("STOP ITHAKI GPS TRACKING\r"))
          break;

        // skip first message and track longitude and latitude
        if (!line.equals("START ITHAKI GPS TRACKING\r")) {
          String[] nmea_split = line.split(",");
          coordinates.add(nmea_split[1]); // time
          coordinates.add(nmea_split[2]); // latitude
          coordinates.add(nmea_split[4]); // longitude
        }

      } catch (Exception x) {
        System.out.println(x);
      }
```

```
    }

    return coordinates;
  }

  /**
   * Create a formated string merging gps data points
   */
  public static String mergeDataPoints(Modem modem, String code,
                                       int numPoints) {
    System.out.println("Creating GPS parameter special format...");
    String finalPoints = "";
    String[] adjust_coords = {"", ""};

    // coordinates: time, latitude, longitude
    ArrayList<String> coordinates = new ArrayList<String>();
    // filtered: merge latitude and longitude
    ArrayList<String> filteredCoords = new ArrayList<String>();

    coordinates = parser(modem, code);

    // adjust to degrees, minutes, seconds
    for (int i = 0; i < coordinates.size(); i++) {
      if (!(i % 3 == 0)) {
        String lat_long = coordinates.get(i);

        String hour =
            i % 3 == 1 ? lat_long.substring(0, 2) : lat_long.substring(1, 3);

        String minutes = i % 3 == 1 ? lat_long.substring(2, lat_long.length())
                                    : lat_long.substring(3, lat_long.length());
        minutes = String.valueOf(Float.parseFloat(minutes));
        int intPart = Integer.parseInt(minutes.split("\\.")[0]);
        float decimalPart = Float.parseFloat("0." + minutes.split("\\.")[1]);
        minutes = String.valueOf(intPart);

        String seconds = String.valueOf(decimalPart * 60f);
        seconds = seconds.substring(0, 2);

        adjust_coords[i % 3 - 1] = hour + minutes + seconds;
      }
      if (i % 3 == 2) {
        // first store the longitude and after the latitude
        filteredCoords.add("T=" + adjust_coords[1] + adjust_coords[0]);
      }
    }

    // keep track of the indices to find the timestamps of the unique data
    // points
    ArrayList<Integer> indices = new ArrayList<Integer>();
    finalPoints = findUniqueDataPoints(filteredCoords, indices);

    ArrayList<String> timestamps = new ArrayList<String>();
    for (Integer i : indices) {
      timestamps.add(coordinates.get(i * 3));
    }

    try (FileWriter time = new FileWriter(new File("logs/gps_timstamps"))) {
      for (String string : timestamps) {
        time.write(string + "\n");
```

```
      }
    } catch (Exception x) {
      System.out.println(x);
    }

    return finalPoints;
  }

  /**
   * Find different GPS points based on latitude and longitude
   *
   * @param coordinates Format T=AABBCCDDEEFF longitude and latitude in hourse,
   *     minutes, seconds
   * @return
   */
  private static String findUniqueDataPoints(ArrayList<String> coordinates,
                                             ArrayList<Integer> indices) {
    ArrayList<String> filtered = new ArrayList<String>();
    int lengthSamples = Math.min(4, coordinates.size());
    String parsedString = "";

    // exclude "T=" and find unique data points
    for (int i = 0; i < coordinates.size(); i++) {
      if (!filtered.contains(coordinates.get(i))) {
        filtered.add(coordinates.get(i));
        indices.add(i);
      }
    }

    for (int i = 0; i < filtered.size(); i++) {
      parsedString += filtered.get(i);
    }

    return parsedString;
  }

  private static String returnLine(Modem modem) {
    String line = "";
    char returnCharModem = ' ';

    while (true) {
      try {
        returnCharModem = (char)modem.read();
        if (returnCharModem != '\n')
          line += returnCharModem;
        else
          break;
      } catch (Exception x) {
        System.out.println(x);
      }
    }

    return line;
  }

}
```

6 ARQ.java

```
package src;
```

```java

import ithakimodem.*;
import java.io.File;
import java.io.FileWriter;

public class ARQ {
  /**
   * Automatic repeat request
   *
   * @param modem
   * @param ackCode
   * @param nackCode
   * @return The number of nack packets
   */
  private static Integer run(Modem modem, String ackCode, String nackCode) {
    System.out.println("\nARQ application...");
    Integer xorResult = 0, fcs = 0;

    String message = Echo.pstop(modem, ackCode);
    System.out.println("ACK:  " + message);
    Integer counter = 0;

    while (true) {
      // parse message to find the encoding sequence
      String[] parsedAck = message.split("\\s+");
      String encodedSequence =
          parsedAck[4].substring(1, parsedAck[4].length() - 1);
      char[] encodedChar = encodedSequence.toCharArray();

      xorResult = xorCharArray(encodedChar);
      fcs = Integer.parseInt(parsedAck[5]);
      System.out.println("Xor: " + xorResult + " FCS: " + fcs);

      if (xorResult != fcs) {
        message = Echo.pstop(modem, nackCode);
        System.out.println("NACK: " + message);
        counter += 1;
      } else {
        break;
      }
    }
    System.out.println("Number of nack " + counter);
    return counter;
  }

  private static Integer xorCharArray(char[] array) {
    int xorResult = (int)array[0];
    for (int i = 1; i < array.length; i++) {
      xorResult = xorResult ^ array[i];
    }

    return xorResult;
  }

  public static void arqRepeat(Modem modem, String ackCode, String nackCode,
                               long timeInterval) {
    float start = System.currentTimeMillis() / 1000f;
    int ackCounter = 1, nackCounter = 0;
    try (FileWriter arq = new FileWriter(new File("logs/arq.txt"));
         FileWriter arqNack = new FileWriter(new File("logs/arqNack.txt"))) {
```

```
      while ((System.currentTimeMillis() / 1000f - start) < timeInterval) {
        System.out.print("Packet No" + ackCounter + ": ");
        int nackCounterPerPacket = 0;

        long tic = System.currentTimeMillis();
        nackCounterPerPacket = ARQ.run(modem, ackCode, nackCode);
        nackCounter += nackCounterPerPacket;
        long toc = System.currentTimeMillis();

        System.out.println("Total time: " + (toc - tic) + " (ms)\n");
        ackCounter += 1;

        arq.write((toc - tic) + "\n");
        arqNack.write(nackCounterPerPacket + "\n");
      }
    } catch (Exception x) {
      System.out.println(x);
    }

    System.out.println("BER:" + berCalculation(ackCounter, nackCounter));
  }

  private static double berCalculation(int ackCounter, int nackCounter) {
    float successProbability = ackCounter / (float)(nackCounter + ackCounter);
    System.out.println("Success prob: " + successProbability * 100 + "%");

    final int numberOfEncodedChars = 16, bitsPerByte = 8;
    int bitsSequence = numberOfEncodedChars * bitsPerByte;

    double ber = 1 - Math.pow(successProbability, 1 / (float)bitsSequence);

    return ber;
  }
}
```

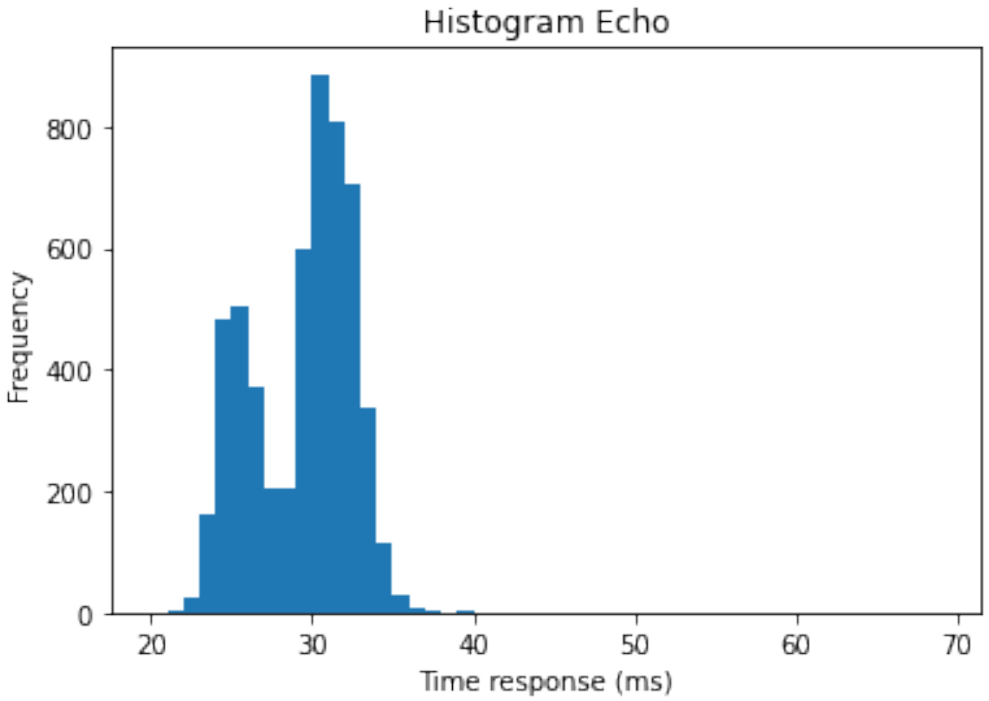
7 Plots

plot

April 18, 2021

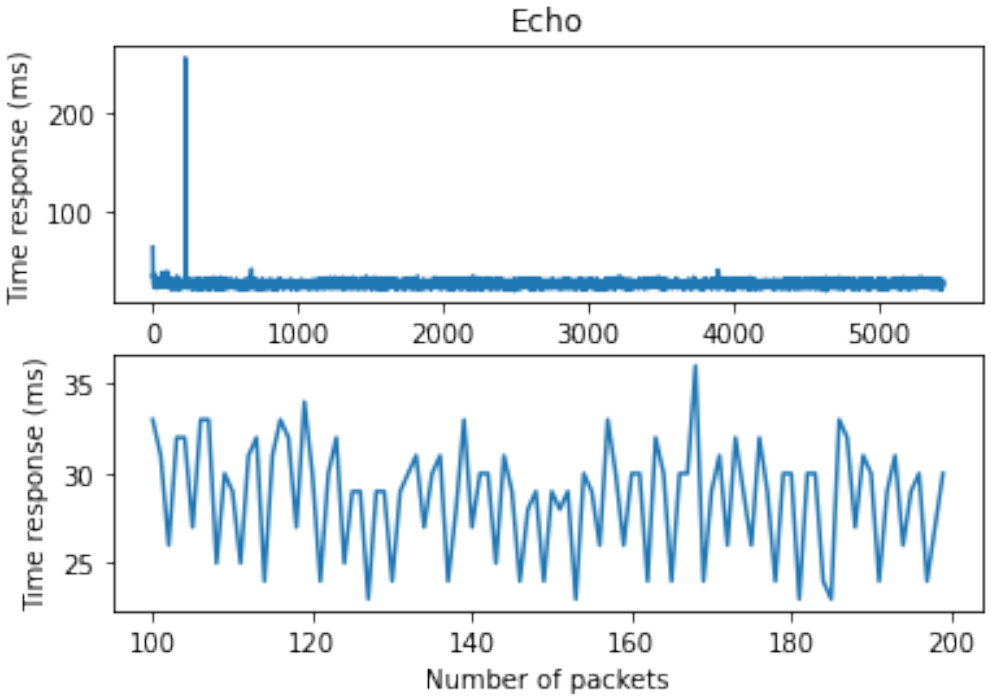
[11]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import scipy
import numpy as np
import csv
import pandas as pd
```

[12]:
```
echo = np.genfromtxt("../logs/session1/echo.txt")
bins = range(20,70,1)
plt.hist(echo, bins)
plt.title("Histogram Echo")
plt.xlabel("Time response (ms)")
plt.ylabel("Frequency")
plt.savefig("../logs/session1/hist_echo.png")
```

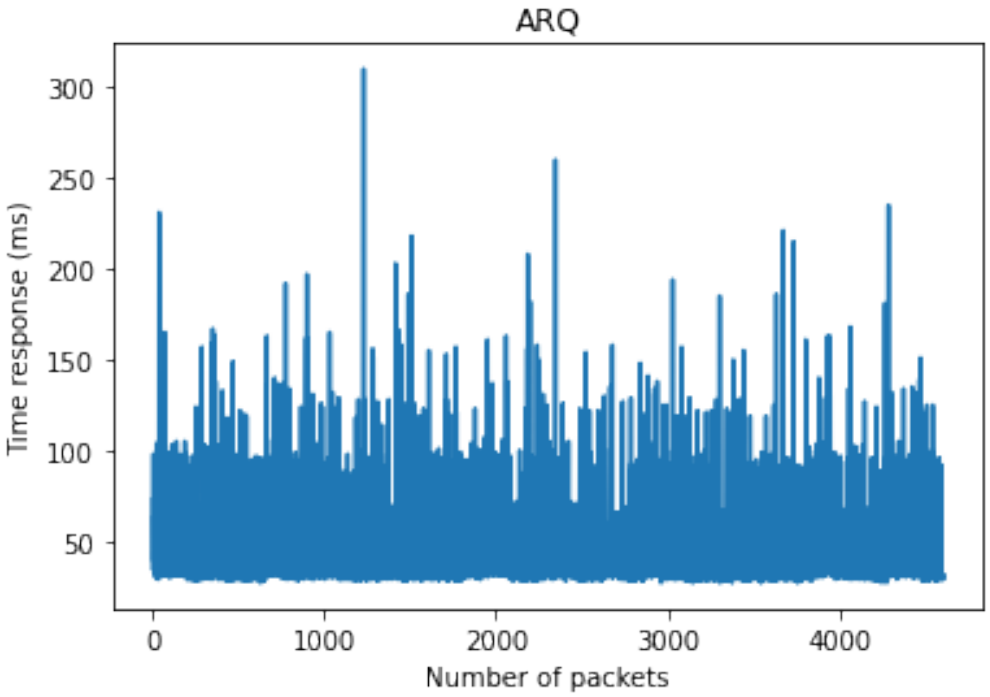


1

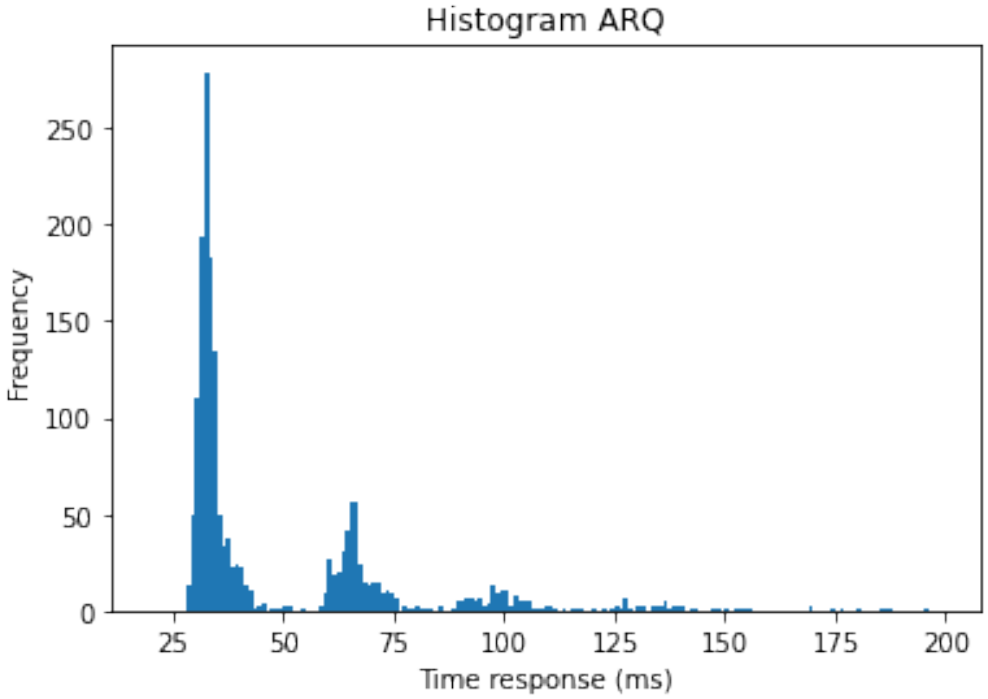
[13]:
```
echo = np.genfromtxt("../logs/session1/echo.txt")
echoRange = range(0, len(echo))
plt.subplot(211)
plt.plot(echoRange, echo)
plt.title("Echo")
plt.xlabel("Number of packets")
plt.ylabel("Time response (ms)")

plt.subplot(212)
shortRange = range(100, 200)
plt.plot(shortRange, echo[shortRange])
plt.xlabel("Number of packets")
plt.ylabel("Time response (ms)")
plt.savefig("../logs/session1/echo.png")
```



[14]:
```
arq = np.genfromtxt("../logs/session1/arq.txt")
arqRange = range(0, len(arq))
plt.plot(arqRange, arq)
plt.title("ARQ")
plt.xlabel("Number of packets")
plt.ylabel("Time response (ms)")
plt.savefig("../logs/session1/arq.png")
```



[17]:
```
arq = np.genfromtxt("../logs/session1/arq.txt")
bins = range(20, 200, 1)
plt.hist(arq, bins)
plt.title("Histogram ARQ")
plt.xlabel("Time response (ms)")
plt.ylabel("Frequency")
plt.savefig("../logs/session1/hist_arq.png")
```


Histogram ARQ
Frequency
250
200
150
100
50
0
25
50
75
100
125
150
175
200
Time response (ms)