



Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης  
Πολυτεχνική Σχολή

## Δίκτυα Υπολογιστών Ι

Θεόδωρος Κατζάλης  
ΑΕΜ:9282  
katzalis@auth.gr

April 11, 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

```
1 package src;
2
3 import ithakimodem.*;
4 import java.io.File;
5 import java.io.FileWriter;
6 import java.util.Date;
7 import java.util.Scanner;
8
9 class UserApplication {
10     public static void main(String[] args) {
11         printWelcome();
12
13         Modem modem = new Modem();
14         int speed = 80_000, timeout = 2_000;
15         setupModem(modem, speed, timeout);
16         startModem(modem, false);
17
18         // Request codes
19
20         String echoCode = "E0337";
21         String imageNoErrorCode = "M5179";
22         String imageWithErrorCode = "G7717";
23         String gpsCode = "P1848";
24         String gpsCodeComplete = gpsCode + "R=1000099";
25         String ackCode = "Q6094";
26         String nackCode = "R6506";
27
28         //String cameraSuffix = "CAM=FIX";
29         //String directionSuffix = "DIR=L";
30         //imageNoErrorCode += cameraSuffix;
31         //imageWithErrorCode += cameraSuffix;
32         String enter = "\r";
33
34         // write request to file
35         try (FileWriter requests = new FileWriter(new File("logs/requests.txt"))) {
36             requests.write("Echo: " + echoCode + "\n");
37             requests.write("Image No Error: " + imageNoErrorCode + "\n");
38             requests.write("Image Yes Error: " + imageWithErrorCode + "\n");
39             requests.write("GPS: " + gpsCode + "\n");
40             requests.write("GPS Full: " + gpsCodeComplete + "\n");
41             requests.write("ACK: " + ackCode + "\n");
42             requests.write("NACK: " + nackCode + "\n");
43             requests.write("Time start: " + new Date());
```

```

44 } catch (Exception x) {
45     System.out.println(x);
46 }
47
48 // applications
49
50 int minutes = 4;
51 final int secondsPerMinute = 60;
52 long timeInterval = minutes * secondsPerMinute;
53
54 Echo.pstopRepeat(modem, echoCode + enter, timeInterval);
55
56 Image.get(modem, imageNoErrorCode + enter);
57 Image.get(modem, imageWithErrorCode + enter);
58
59 Image.get(modem, imageNoErrorCode + "CAM=PTZ"+ enter);
60 Image.get(modem, imageWithErrorCode + "CAM=PTZ" + enter);
61
62 String maps_query = GPS.mergeDataPoints(modem, gpsCodeComplete + enter, 2);
63 System.out.println("The GPS parameter " + maps_query);
64 Image.get(modem, gpsCode + maps_query + enter);
65
66 ARQ.arqRepeat(modem, ackCode + enter, nackCode + enter, timeInterval);
67
68 modem.close();
69 }
70
71 /**
72  * Read the welcome screen when opening the virtual modem
73  *
74  * @param modem The virtual opened modem
75  * @param isPrinted Print the returned message to stdout
76  */
77 private static void startModem(Modem modem, boolean isPrinted) {
78     final int finishReadingFlag = -1;
79
80     int returnValueModem, finishCounter = 0;
81     char returnCharModem = ' ';
82     char[] finishReadingString = {'\r', '\n', '\n', '\n'};
83
84     while (true) {
85         try {
86             returnValueModem = modem.read();
87             returnCharModem = (char)returnValueModem;
88
89             if (isPrinted) {
90                 System.out.print(returnCharModem);
91                 Thread.sleep(10);
92             }
93
94             // check for breaking flag
95             // if (returnValueModem == finishReadingFlag) break;
96
97             // check for breaking sequence
98             if ((returnCharModem == finishReadingString[finishCounter])) {
99                 finishCounter++;
100                 if (finishCounter == 4)
101                     break;
102             } else
103                 finishCounter = 0;

```

```

104     } catch (Exception x) {
105         System.out.println(x);
106         break;
107     }
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

```
1 package src;
2 import ithakimodem.*;
3 import java.io.File;
4 import java.io.FileWriter;
5 import java.util.LinkedList;
6 import java.util.Queue;
7
8 public class Echo {
9
10     /**
11      * Echo packet contain info dependent of the request code.
12      *
13      * Stop communication when detect "PSTOP"
14      *
15      * @param modem The virtual opened modem
16      * @param code Echo request code
17      */
18     public static String pstop(Modem modem, String code) {
19         // System.out.println("Echo application");
20
21         char returnModem = ' ';
22         String message = "";
23         Queue<Character> breakingChars = new LinkedList<>();
24
25         modem.write(code.getBytes());
26
27         while (true) {
28             try {
29
30                 // all the echo packets ends to "PSTOP"
31                 if (breakingChars.size() != 5) {
32                     returnModem = (char)modem.read();
33                     breakingChars.add(returnModem);
34                 } else {
35                     String breakingString = queueChar2String(breakingChars);
36                     if (breakingString.equals("PSTOP"))
37                         break;
38                     else {
39                         breakingChars.remove();
40                         returnModem = (char)modem.read();
41                         breakingChars.add(returnModem);
42                     }
43                 }
44             } catch (Exception x) {
45                 System.out.println(x);
46             }
47
48             message += returnModem;
49         }
50
51         return message;
52     }
53
54     /**
55      * Stop communication when -1 returns
56      *
57      * @param modem
58      * @param code
```

```

59  */
60  public static void generic(Modem modem, String code) {
61      final int finishReadingFlag = -1;
62
63      int returnValueModem = 0;
64      char returnCharModem = ' ';
65
66      modem.write(code.getBytes());
67      while (true) {
68          try {
69              returnValueModem = modem.read();
70              returnCharModem = (char)returnValueModem;
71
72              System.out.print(returnCharModem);
73
74              // check for breaking flag
75              if (returnValueModem == finishReadingFlag)
76                  break;
77          } catch (Exception x) {
78              System.out.println(x);
79          }
80      }
81      System.out.println();
82  }
83
84  public static void pstopRepeat(Modem modem, String code, long timeInterval) {
85      float start = System.currentTimeMillis() / 1000f;
86      int counter = 0;
87
88      try (FileWriter echo = new FileWriter(new File("logs/echo.txt"))) {
89          while ((System.currentTimeMillis() / 1000f - start) < timeInterval) {
90              System.out.print("Packet No" + counter + ": ");
91
92              long tic = System.currentTimeMillis();
93              String message = Echo.pstop(modem, code);
94              System.out.println(message);
95              long toc = System.currentTimeMillis();
96
97              System.out.println("Total time: " + (toc - tic) + " (ms)\n");
98              counter += 1;
99
100             echo.write((toc - tic) + "\n");
101         }
102     } catch (Exception x) {
103         System.out.println(x);
104     }
105 }
106
107 private static String queueChar2String(Queue<Character> queue) {
108     String message = "";
109     for (Character character : queue) {
110         message += character;
111     }
112     return message;
113 }
114 }

```

## 4 Image.java

```

1 package src;

```

```

2 import ithakimodem.*;
3 import java.io.ByteArrayOutputStream;
4 import java.io.File;
5 import java.io.FileOutputStream;
6
7 public class Image {
8     /**
9      * Create image file. The breaking flag to stop reading is the delimiter
10     * "0xFFD9"
11     *
12     * @param modem The virtual opened modem
13     * @param code Image request code
14     */
15     public static void get(Modem modem, String code) {
16         System.out.println("\nImage application...");
17
18         ByteArrayOutputStream buffer = new ByteArrayOutputStream();
19         int returnValueModem = 0;
20         byte first, second;
21
22         long tic = System.currentTimeMillis();
23
24         modem.write(code.getBytes());
25         returnValueModem = modem.read();
26         first = int0(returnValueModem);
27         buffer.write(first);
28
29         while (true) {
30             try {
31                 returnValueModem = modem.read();
32                 second = int0(returnValueModem);
33                 buffer.write(second);
34
35                 // System.out.print(String.format("%02X", first));
36                 // System.out.print(String.format("%02X", second));
37
38                 if ((String.format("%02X", first).equals("FF")) &&
39                     (String.format("%02X", second).equals("D9")))
40                     break;
41
42                 first = second;
43
44             } catch (Exception x) {
45                 System.out.println(x);
46             }
47         }
48         byte[] dataImage = buffer.toByteArray();
49
50         // write image file
51
52         String path = "";
53         if (code.substring(0, 1).equals("P")) {
54             path = "media/gps.jpg";
55         } else if (code.substring(0, 1).equals("M")) {
56             path = code.contains("PTZ") ? "media/image_error_free_ptz.jpg"
57                 : "media/image_error_free_fix.jpg";
58         } else if (code.substring(0, 1).equals("G")) {
59             path = code.contains("PTZ") ? "media/image_with_errors_ptz.jpg"
60                 : "media/image_with_errors_fix.jpg";
61         } else {

```



```

62     path = "image.jpg";
63 }
64
65 File image = new File(path);
66 try (FileOutputStream fos = new FileOutputStream(image)) {
67     fos.write(dataImage);
68     System.out.println("File " + path + " has been created successfully");
69 } catch (Exception x) {
70     System.out.println(x);
71 }
72
73 long toc = System.currentTimeMillis();
74 System.out.print("Total time creating image: " + (toc - tic) / 1000.0 +
75                 " (s)\n");
76 }
77
78 /**
79  * source:
80  * https://stackoverflow.com/questions/1936857/convert-integer-into-byte-array-java
81  */
82 private static byte int3(int x) { return (byte)(x >> 24); }
83 private static byte int2(int x) { return (byte)(x >> 16); }
84 private static byte int1(int x) { return (byte)(x >> 8); }
85 private static byte int0(int x) { return (byte)(x >> 0); }
86 }

```

## 5 GPS.java

```

1 package src;
2
3 import ithakimodem.*;
4 import java.io.File;
5 import java.io.FileWriter;
6 import java.util.ArrayList;
7
8 public class GPS {
9     private static ArrayList<String> parser(Modem modem, String code) {
10         ArrayList<String> coordinates = new ArrayList<String>();
11
12         modem.write(code.getBytes());
13         while (true) {
14             try {
15                 String line = returnLine(modem);
16                 // System.out.println(line);
17
18                 // check for breaking flag
19                 if (line.equals("STOP ITHAKI GPS TRACKING\r"))
20                     break;
21
22                 // skip first message and track longitude and latitude
23                 if (!line.equals("START ITHAKI GPS TRACKING\r")) {
24                     String[] nmea_split = line.split(",");
25                     coordinates.add(nmea_split[1]); // time
26                     coordinates.add(nmea_split[2]); // latitude
27                     coordinates.add(nmea_split[4]); // longitude
28                 }
29             } catch (Exception x) {
30                 System.out.println(x);
31             }
32         }
33     }
34 }

```

```

33     }
34
35     return coordinates;
36 }
37
38 /**
39  * Create a formatted string merging gps data points
40  */
41 public static String mergeDataPoints(Modem modem, String code,
42                                     int numPoints) {
43     System.out.println("Creating GPS parameter special format...");
44     String finalPoints = "";
45     String[] adjust_coords = {"", ""};
46
47     // coordinates: time, latitude, longitude
48     ArrayList<String> coordinates = new ArrayList<String>();
49     // filtered: merge latitude and longitude
50     ArrayList<String> filteredCoords = new ArrayList<String>();
51
52     coordinates = parser(modem, code);
53
54     // adjust to degrees, minutes, seconds
55     for (int i = 0; i < coordinates.size(); i++) {
56         if (!(i % 3 == 0)) {
57             String lat_long = coordinates.get(i);
58
59             String hour =
60                 i % 3 == 1 ? lat_long.substring(0, 2) : lat_long.substring(1, 3);
61
62             String minutes = i % 3 == 1 ? lat_long.substring(2, lat_long.length())
63                                     : lat_long.substring(3, lat_long.length());
64             minutes = String.valueOf(Float.parseFloat(minutes));
65             int intPart = Integer.parseInt(minutes.split("\\.")[0]);
66             float decimalPart = Float.parseFloat("0." + minutes.split("\\.")[1]);
67             minutes = String.valueOf(intPart);
68
69             String seconds = String.valueOf(decimalPart * 60f);
70             seconds = seconds.substring(0, 2);
71
72             adjust_coords[i % 3 - 1] = hour + minutes + seconds;
73         }
74         if (i % 3 == 2) {
75             // first store the longitude and after the latitude
76             filteredCoords.add("T=" + adjust_coords[1] + adjust_coords[0]);
77         }
78     }
79
80     // keep track of the indices to find the timestamps of the unique data
81     // points
82     ArrayList<Integer> indices = new ArrayList<Integer>();
83     finalPoints = findUniqueDataPoints(filteredCoords, indices);
84
85     ArrayList<String> timestamps = new ArrayList<String>();
86     for (Integer i : indices) {
87         timestamps.add(coordinates.get(i * 3));
88     }
89
90     try (FileWriter time = new FileWriter(new File("logs/gps_timestamps"))) {
91         for (String string : timestamps) {
92             time.write(string + "\n");

```

```

93     }
94 } catch (Exception x) {
95     System.out.println(x);
96 }
97
98     return finalPoints;
99 }
100
101 /**
102  * Find different GPS points based on latitude and longitude
103  *
104  * @param coordinates Format T=AABBCCDDEEFF longitude and latitude in hourse,
105  *     minutes, seconds
106  * @return
107  */
108 private static String findUniqueDataPoints(ArrayList<String> coordinates,
109                                           ArrayList<Integer> indices) {
110     ArrayList<String> filtered = new ArrayList<String>();
111     int lengthSamples = Math.min(4, coordinates.size());
112     String parsedString = "";
113
114     // exclude "T=" and find unique data points
115     for (int i = 0; i < coordinates.size(); i++) {
116         if (!filtered.contains(coordinates.get(i))) {
117             filtered.add(coordinates.get(i));
118             indices.add(i);
119         }
120     }
121
122     for (int i = 0; i < filtered.size(); i++) {
123         parsedString += filtered.get(i);
124     }
125
126     return parsedString;
127 }
128
129 private static String returnLine(Modem modem) {
130     String line = "";
131     char returnCharModem = ' ';
132
133     while (true) {
134         try {
135             returnCharModem = (char)modem.read();
136             if (returnCharModem != '\n')
137                 line += returnCharModem;
138             else
139                 break;
140         } catch (Exception x) {
141             System.out.println(x);
142         }
143     }
144
145     return line;
146 }
147
148 }

```

## 6 ARQ.java

```

1 package src;

```

```

2
3 import ithakimodem.*;
4 import java.io.File;
5 import java.io.FileWriter;
6
7 public class ARQ {
8     /**
9      * Automatic repeat request
10     *
11     * @param modem
12     * @param ackCode
13     * @param nackCode
14     * @return The number of nack packets
15     */
16     private static Integer run(Modem modem, String ackCode, String nackCode) {
17         System.out.println("\nARQ application...");
18         Integer xorResult = 0, fcs = 0;
19
20         String message = Echo.pstop(modem, ackCode);
21         System.out.println("ACK: " + message);
22         Integer counter = 0;
23
24         while (true) {
25             // parse message to find the encoding sequence
26             String[] parsedAck = message.split("\\s+");
27             String encodedSequence =
28                 parsedAck[4].substring(1, parsedAck[4].length() - 1);
29             char[] encodedChar = encodedSequence.toCharArray();
30
31             xorResult = xorCharArray(encodedChar);
32             fcs = Integer.parseInt(parsedAck[5]);
33             System.out.println("Xor: " + xorResult + " FCS: " + fcs);
34
35             if (xorResult != fcs) {
36                 message = Echo.pstop(modem, nackCode);
37                 System.out.println("NACK: " + message);
38                 counter += 1;
39             } else {
40                 break;
41             }
42         }
43         System.out.println("Number of nack " + counter);
44         return counter;
45     }
46
47     private static Integer xorCharArray(char[] array) {
48         int xorResult = (int)array[0];
49         for (int i = 1; i < array.length; i++) {
50             xorResult = xorResult ^ array[i];
51         }
52
53         return xorResult;
54     }
55
56     public static void arqRepeat(Modem modem, String ackCode, String nackCode,
57                                 long timeInterval) {
58         float start = System.currentTimeMillis() / 1000f;
59         int ackCounter = 1, nackCounter = 0;
60         try (FileWriter arq = new FileWriter(new File("logs/arq.txt"))) {
61             while ((System.currentTimeMillis() / 1000f - start) < timeInterval) {

```

```

62     System.out.print("Packet No" + ackCounter + ": ");
63
64     long tic = System.currentTimeMillis();
65     nackCounter += ARQ.run(modem, ackCode, nackCode);
66     long toc = System.currentTimeMillis();
67
68     System.out.println("Total time: " + (toc - tic) + " (ms)\n");
69     ackCounter += 1;
70
71     arq.write((toc - tic) + "\n");
72 }
73 } catch (Exception x) {
74     System.out.println(x);
75 }
76
77 System.out.println("BER:" + berCalculation(ackCounter, nackCounter));
78 }
79
80 private static double berCalculation(int ackCounter, int nackCounter) {
81     float successProbability = ackCounter / (float)(nackCounter + ackCounter);
82     System.out.println("Success prob: " + successProbability * 100 + "%");
83
84     final int numberOfEncodedChars = 16, bitsPerByte = 8;
85     int bitsSequence = numberOfEncodedChars * bitsPerByte;
86
87     double ber = 1 - Math.pow(successProbability, 1 / (float)bitsSequence);
88
89     return ber;
90 }
91 }

```

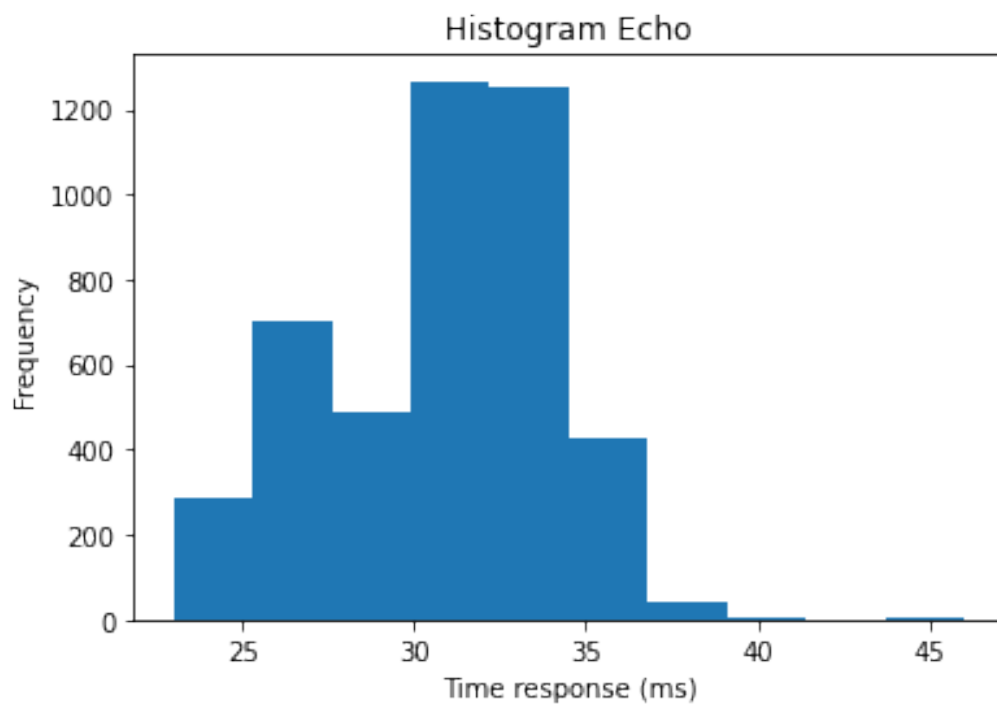
## 7 Plots

plot

April 11, 2021

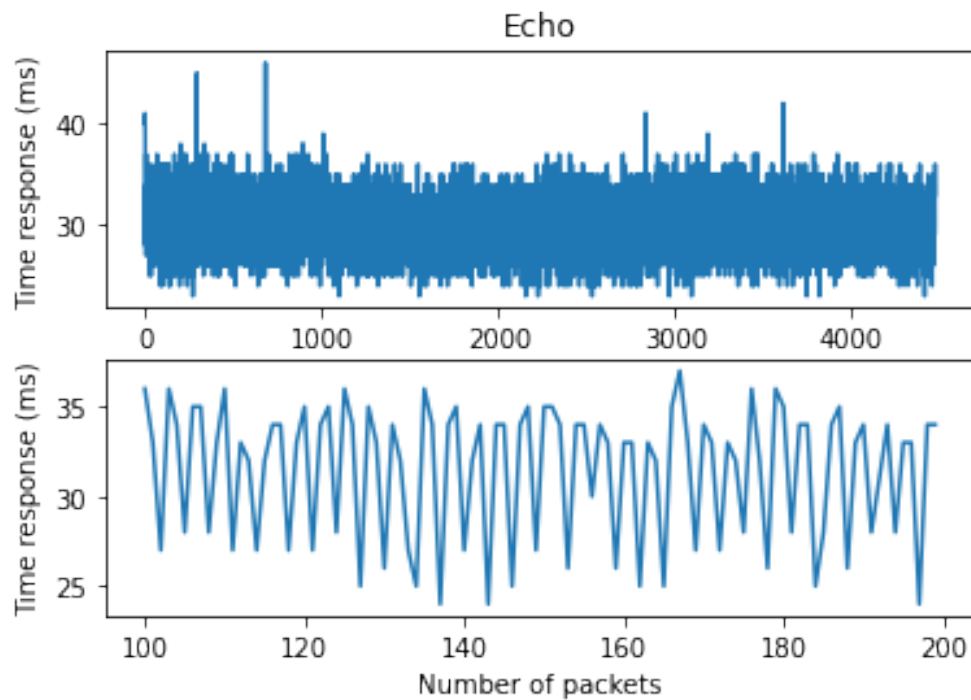
```
[4]: import seaborn as sns
import matplotlib.pyplot as plt
import scipy
import numpy as np
import csv
import pandas as pd
```

```
[5]: echo = np.genfromtxt("../logs/session1/echo.txt")
plt.hist(echo)
plt.title("Histogram Echo")
plt.xlabel("Time response (ms)")
plt.ylabel("Frequency")
plt.savefig("../logs/session1/hist_echo.png")
```

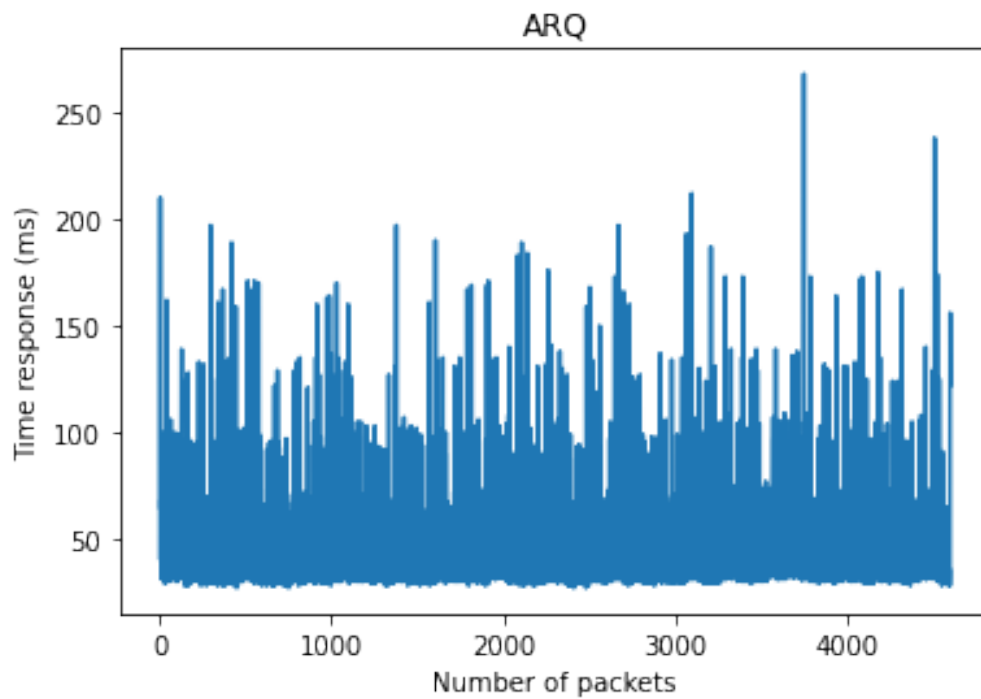


```
[6]: echo = np.genfromtxt("../logs/session1/echo.txt")
echoRange = range(0, len(echo))
plt.subplot(211)
plt.plot(echoRange, echo)
plt.title("Echo")
plt.xlabel("Number of packets")
plt.ylabel("Time response (ms)")

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")
```



```
[7]: 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")
```



```
[8]: arq = np.genfromtxt("../logs/session1/arq.txt")
plt.hist(arq)
plt.title("Histogram ARQ")
plt.xlabel("Time response (ms)")
plt.ylabel("Frequency")
plt.savefig("../logs/session1/hist_arq.png")
```



