CS 421- Computer Networks

Programming Assignment 1 Report

31.03.2023

Instructor: Ezhan Karaşan

Section: 01

Zeynep Selcen Öztunç - 21902941

Part 1

In part 1, I have used the Netcat working tool to observe what message the Firefox is sending. I have used the -l option to specify the port number, which was 12345 in my case. I tried this with 3 HTTP websites.(The “nc -l” gave an error in my case so I have used “nc -L -p” which does the same thing). Fields such as Host, User-Agent, Accept-Language etc. of the HTTP request message can be observed from the Netcat results.

```
C:\Users\admin\Desktop\network>nc -L -p 12345
GET http://go.com/ HTTP/1.1
Host: go.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/111.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1

GET http://wa.gov/ HTTP/1.1
Host: wa.gov
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/111.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1

GET http://ssense.com/ HTTP/1.1
Host: ssense.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/111.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

Part 2

In the second part of this assignment, I have implemented a proxy server using Java. In order to do so, in my main method, after doing the necessary checks, I created a client socket and server socket objects and got the input stream from the client socket. After doing that, I print out the HTTP request message on the console. I do this on a while loop so that the program continues to execute as long as the user requests websites. I also parsed the request message so that I would be able to obtain a url and a filename like “bilkent.txt”.

```
109
110         //create server socket
111         serverSocket=new ServerSocket(port);
112
113         while(true){
114             clientSocket=serverSocket.accept();//a socket object representing client's connection
115
116             BufferedReader clientReq= new BufferedReader(new InputStreamReader
117             (clientSocket.getInputStream()));
118             String line;//lines of the HTTP request message
119
120             StringBuilder msgBuilder = new StringBuilder();
121             String clientRequest="";
122             int i = 0;
123             //build the http get message
124             while((line=clientReq.readLine()) != null && !line.isEmpty()){
125                 if(i == 0){
126                     clientRequest= line;//first line is the request from the client
127                 }
128                 msgBuilder.append(line).append(str:"\n");
129                 i++;
130             }
131
```

```
131
132         //divide the request into parts
133         String requestParts[]=clientRequest.split(regex:" ");
134         String url= requestParts[1];
135
136         //only print the bilkent server requests/responses
137         if(!url.contains(s:"www.cs.bilkent.edu.tr")){
138             System.out.println();
139             continue;
140         }
141
142         //print the get request
143         String requestMessage = msgBuilder.toString();
144         System.out.println(x:"Retrieved request from Firefox :\n");
145         System.out.println(requestMessage);
146
```

Then, I created an URL object using that filename and opened a connection on that object. I set the request method of the connection to “GET”.

```
163
164     //create an URL object to open a connection
165     URL server= new URL(url);
166     HttpURLConnection connection= (HttpURLConnection)server.openConnection();//connection object
167     connection.setRequestMethod(method:"GET");
168
169     //get the HTTP status code
170     int status= connection.getResponseCode();
171
```

If the response message from the server is “OK”, the sendResponse() method is called.

```
//if the status code is ok
if(status == HttpURLConnection.HTTP_OK){

    //if the url is valid, send response to the client socket
    if(url != null){
        sendResponse(clientSocket, connection, url);
    }
```

The sendResponse method sends an HTTP response message to the client socket by using the output stream. The response message includes the status code and message, content type and content length, which are obtained by using the getContentLength() and getContentType() methods of the URLConnection. After sending the HTTP response message, to display the contents on the browser, the read method reads the contents of the input stream and the write method writes the contents to the output stream.

```java
public static void sendResponse(Socket clientSocket, URLConnection connection,
String url) throws IOException {

    //init the content length and type
    int cLength = connection.getContentLength();
    String cType = connection.getContentType();

    //init the input and output streams
    InputStream inputStream = connection.getInputStream();
    OutputStream outputStream = clientSocket.getOutputStream();
    PrintWriter out = new PrintWriter(outputStream, autoFlush:true);

    //the http response message
    out.println(x:"HTTP/1.1 200 OK");
    out.println("Content-Type: " + cType);//this line is necessary
    out.println("Content-Length: " + cLength);
    out.println();

    byte[] buffer = new byte[1024];
    int numOfBytes;

    //display the output in the browser
    while ((numOfBytes = inputStream.read(buffer)) != -1) {
        outputStream.write(buffer, off:0, numOfBytes);
    }
    outputStream.close();
}
```

After this method returns, I print a message indicating I have retrieved the content and then I invoke the downloadFile() method with the url of the website as an argument.

```java
//if the status code is ok
if(status == HttpURLConnection.HTTP_OK){

    //if the url is valid, send response to the client socket
    if(url != null){
        sendResponse(clientSocket, connection, url);
    }

    //print the HTTP status code and message
    System.out.println("Retrieved "+ status +" " +connection.getResponseMessage());

    //download the files
    downloadFile(url);
    System.out.println();
    System.out.println();

}
```

The downloadFile() method downloads the files in the url which was given in function arguments. It first creates a connection using the url in the argument, then if the response code is “OK”, it downloads the files into the project directory, otherwise it gives an error message and terminates. When the response message is “OK”, it downloads the files by first creating an input stream from the connection’s getInputStream() method, then it creates an output stream and a byte array which will store the link content. By using the read() method for the input stream and the write() method for the output stream, the content is written to the

output stream and therefore it is downloaded.

```java
public static void downloadFile(String url) throws IOException {

    // create a new connection with the url in the method argument
    URL targetUrl = new URL(url);
    HttpURLConnection connection = (HttpURLConnection) targetUrl.openConnection();
    connection.setRequestMethod(method:"GET");//with get request method

    //get the response code from connection
    int response = connection.getResponseCode();

    // if the response message is OK
    if (response == HttpURLConnection.HTTP_OK) {

        // get filename
        String filename = url.substring(url.lastIndexOf(str:"/") + 1);

        //array to store the read data
        byte[] buffer = new byte[4096];
        int numOfBytes = -1;

        // create an output stream with the filename
        FileOutputStream oStream = new FileOutputStream(filename);

        //read the input stream
        InputStream iStream = connection.getInputStream();// get the input stream
        while ((numOfBytes = iStream.read(buffer)) != -1) {
            oStream.write(buffer, off:0, numOfBytes);//write it to the output stream
        }
        oStream.close();
```

Then using Pattern and Matcher objects, all of the hyperlinks in the response message are found. A new URL object is created with the name of each new hyperlink and the method is called recursively.

```java

        //finds all the hyperlinks in the response
        String responseMessage = new String(buffer, StandardCharsets.UTF_8);
        Pattern hyperLink = Pattern.compile(regex:"<a\\s+[^>]*href=\"([^\"]*)\"[^>]*>");
        Matcher match = hyperLink.matcher(responseMessage);

        //while there are still hyperlinks
        while (match.find()) {
            String hLink = match.group(group:1);

            // create an url object
            URL absoluteUrl = new URL(url);
            URL linkUrl = new URL(absoluteUrl, hLink);//new url object with absolute url as base
            String downloadUrl = linkUrl.toString();

            downloadFile(downloadUrl);//recursively call the method
        }
        System.out.println("File downloaded successfully: " + url);
    }
    else {
        // print an error message if the response code is not 200
        System.out.println("Error, could not download the files!: " + response
        + " " + connection.getResponseMessage());
    }
}
```