INF142

Obligatorisk oppgave 1

2018

Levert av:

Andrey Belinskiy, zur008

Oppgave 1 - description of communication process between client “K”, proxy server “WPS” and remote host “RHOST”.

0. **Starting point**

   There are three entities at the starting point: client “K”, proxy “WPS” and remote host “RHOST”.
   “K” is waiting for input from the user. “WPS” waits for the request from the “K”. “RHOST” waits for any remote request.

1. “K” receives a URL from user input containing ether just domain name (f.ex. www.uib.no or uib.no) or full address to a page or file.

2. “K” converts this data to bytes and sends it to “WPS” through DatagramSocket using DatagramPacket. The socket closes if server is not responding after 120 seconds.

3. “WPS” receives the data from “K” through DatagramSocket. After that “WPS” performs rough validation of the received information. If said information is valid “WPS” separates the host name and webpage/file address into two separate variables for further usage. If received information is not valid “WPS” sends an error message to the “K” through DatagramSocket.

4. “WPS” creates a TCP socket and tries to connect to the specified remote address. If it can’t connect it sends an error message to the “K”.

5. After establishing TCP connection “WPS” sends the “HEAD” (with obligatory for HTTP/1.1 “Host: “) request to the “RHOST” via TCP socket. “WPS” automatically assigns the contents of the variables for this request. If user provided just the domain address, the “HEAD” request will use single “/”. If user provided more specific address with a path to a file or web page, this information will be used instead of a single “/”. Contents of the variable for “Host: ” are also assigned automatically based

on the URL from the user. If the remote host does not provide a response after 120 seconds, then the connection times out and closes. At the same time “K” prints out the same error message about timed out connection.

6. “RHOST” receives the request from “WPS”, processes it and sends the response to the “WPS” via TCP socket.

7. “WPS” receives the response form the “RHOST”, converts it to byte data and sends this data through DatagramSocket to “K”. After that “WPS” closes the sockets and shuts down.

8. “K” receives the data from “WPS” via DatagramSocket and prints it to the terminal. After that “K” closes the sockets and shuts down.

In the process of working “WPS” and “K” print out the status messages to the terminal. The user will be able to make several requests to the proxy server and receive multiple responses if we make small adjustments to the code (we can add simple loops to listen for the user input and shut down the client and server after specific key phrase).