NOVA SCHOOL OF SCIENCE & TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE

HTTP- Hyper Text Transfer Protocol (Lab-08 2023/24)

1. Introduction

In this class we will program a very simple HTTP client and server able to transfer a file using the HTTP protocol. In the last part of the class, we will see how dynamic contents is generated.

Learning Materials:

- Lecture slides available in CLIP and *Peterson & Davie Computer Networks: A Systems Approach* section 9.1.2 *https://book.systemsapproach.org/applications/traditional.html#world-wide-web-http*
- Python basic concepts can be studied in several online texts namely *Python for Everybody: Exploring Data in Python3* by Charles R. Severance et al, available (including a tar file with the code of the examples used in the book) at https://www.py4e.com/book.php
- A Python http.server tutorial at https://docs.python.org/3/library/http.server.html

2. Python built-in webserver

Python standard library comes with a built-in webserver which can be invoked for serving static files. Although not a full featured web server, it can parse simple static html files and serve them by responding with the required response codes.

How to run this http server in the terminal:

```
$python -m http.server
```

This will start a webserver waiting connections on port 8000, with the document root defaulting to the current directory, i.e., the current working directory contents will be served.

To change the default directory, use the option -d or --directory, that will specify a directory to which it should serve the files. For example, the following command uses the specific directory /tmp:

```
$python -m http.server -d /tmp/
```

By default, the server listens to port 8000. This can be changed by passing on the command line the desired port. For example, the following command will override the default value:

```
$python -m http.server 9000
```

By default, the server binds itself to all interfaces. The option -b/--bind specifies a specific address to which it should bind. Both IPv4 and IPv6 addresses are supported. For example, the following command causes the server to bind to localhost only:

```
$python -m http.server --bind 127.0.0.1
```

3. Basic HTTP client

In this assignment, you'll write a simple HTTP client that will send a HTTP request to the http.server, with the message with the following command line:

```
$python HTTPclient.py name1 name2
```

name1 – the complete URL **`http://localhost:8000/resourceName`**, corresponding to the file to be transferred.

name2 – name of the file where the client saves the body part of the server reply

Example of a simple HTTP message:

```
GET filename HTTP/1.1\r\n
host: localhost:8000\r\n
\r\n
```

4. HTTP Server

Now, instead of using the http.server, you will build your own HTTPserver. This very basic server should receive the request message from the previous developed HTTP client, build a HTTP response message containing the requested object (file) and send it to the HTTPclient.

Example:

```
HTTP/1.1 200 OK\r\n
Server: SimpleHTTP/0.6 Python/3.8.3\r\n
Date: Tue, 7 Nov 2023 10:34:27 GMT\r\n
Content-type: image/jpeg\r\n
Content-Length: 279431\r\n
Last-Modified: Tue, 1 Jan 2021 11:34:12 GMT\r\n
\r\n
… 279431 bytes of file …
```

5. Static contents

To run the http server in the terminal, write the following line in a terminal:

```
$python -m http.server -d ./www 19999
```

This will start a webserver waiting connections on port 19999, with the document root defaulting to the subdirectory *www* of the current directory. You can start the server in a folder of your choice. The chosen folder will be populated with several HTML files and a folder called *cgi-bin* (see section 6).

You can use a program editor like *notepad++*, *geany* or *vscode* to edit an HTML with the following contents that can be saved in a file *index.html*.

```
<!DOCTYPE html>
<head>
```

```
    <title> Nova School of Science & Technology </title>
</head>
<body>
    <img src="nova_4.png" alt="Logo da FCT NOVA" width="300" height="150"/>
    <h1> Welcome to the FCT NOVA site </h1>
    <section>
     <h2> What do you want to access? </h2>
     <ul>
          <li> <a href "https://www.fct.unl.pt/en/about-fct/governance/deans-
office"> Dean's office </a></li>
           <li> <a href="https://www.di.fct.unl.pt/en/study-programme-
mei"> Study Programme of MEI </a></li>
     </ul>
    </section>
</body>
</html>
```

After storing the *index.html* and *nova_4*.png files in the chosen folder, we can start the web server and verify if it is up and running by opening a tab in a browser and writing in the dialog box *http://localhost:19999/index.html*.

Please change the contents of the file *index.html* at your will.

6. Dynamic contents

In this section, you will see how to execute a program on the server that receives information from the browser and dynamically generates an HTML page that is rendered by the browser.

Let's start by creating python file *dateTime.py* with the following contents:

```
import sys
import codecs
import datetime

sys.stdout = codecs.getwriter("utf-8")(sys.stdout.detach())

print("""Content-type:text/html\n\n
<!DOCTYPE html>
<head>
    <title> Date </title>
</head>
<body>
    <h1> Date </h1>
     <h2> """)
now = datetime.datetime.now()
print(now.strftime("%Y-%m-%d %H:%M:%S"))
print("""</h2>
</body> </html>""")
```

First, you must launch the http.server with a command line option allowing the launching of scripts present in subfolder *cgi-bin* of *www*:

```
$python -m http.server -d ./www --cgi 19999
```

You can obtain a page with the current date/time by entering in the browser
http://localhost:19999/cgi-bin/dateTime.py

Study the Python code and try it. You can also change the Python code to get a page with another piece of information.

What is missing is the ability to pass parameters to the script that runs on the server. This will be discussed next week.