CSCD 330 - Computer Networks

Lab 4, HTTP and Sockets

Overview

Write a TCP client similar to `curl` and `wget`. Your client must be able to retrieve a complete webpage via HTTP, whether that page is the root or **another path**. Handling HTTPS is not expected.

Instructions

Complete the above program following the steps in the next section. Call this file lab4.py and include your name at the top of the file in a comment – #author: YOUR NAME. Your output should match the provided example output. You must also create a README explaining how to run your program and answering the questions below. Furthermore, you must create a test script in Bash. Call this file test.sh.

You are expected to read section 2.7 before beginning this lab.

You may only use the following imports:

```
from sys import argv
from socket import *
from urllib.parse import urlparse
```

Steps:

0. Setup

Your client must take 3 command line arguments as follows:

1. A flag indicating whether to print to `STDOUT` or to save the output to a file called `output.txt`. The flag should be `-p` for printing and `-f` for outputting to said file.
2. The target port.
3. The URL to retrieve. Note that `urlparse()` expects a properly formatted URL such as `http://google.com/` and not `google.com`.

1. Create a TCP connection

Use sockets to setup a client **TCP** connection.

2. Create and send an HTTP GET request

Create the string necessary to communicate with a web server. You'll only need to worry about unencrypted connections (i.e., port 80 HTTP traffic).

Once you have crafted the correct string you must send it over the previously created socket connection (Step 1 above).

3. Receive the full webpage

You will likely have to call `recv()` multiple times. If your program waits until timeout, you will be deducted points. Ideally the server will initiate closing the TCP connection after sending the full page.

4. Output to either STDOUT or `output.txt`

Omit the header i.e., your output should look like the example output.

Questions:

1. Why did you have to `encode()` your request and `decode()` the response(s)? What do these functions do?
2. What changes would you have to make to create a UDP socket?
3. If you wanted to create a TCP server, what would you have to change?
4. Can your TCP client create or process HTTPS traffic? What happens if you send a request to port 443?

Turn in:

Submit a tarball with the following:

- Your source code (in Python) called `lab4.py`
- Your test script (in Bash) called test.sh – test at least 3 inputs.
- Your README answering the above questions and explaining how to run your program.

In case you have forgotten:

```
tar -czvf lab4_YOURNAME.tar.gz *.py *.sh README
```

Example output:

python3 lab4.py -p 80 http://httpforever.com/

```
<!DOCTYPE HTML>
<html>
    <head>
        <title>HTTP Forever</title>
        # Removed for brevity.
    </body>
</html>
```

python3 lab4.py -f 80 http://httpforever.com/

```
# Create a file called ‘output.txt‘.
# The file contents are expected to be indentical to the "-p" output
above.
```

python3 lab4.py -p 80 http://httpforever.com/login

```
<html>
<head><title>404 Not Found</title></head>
<body>
<center><h1>404 Not Found</h1></center>
<hr><center>nginx/1.18.0 (Ubuntu)</center>
</body>
</html>
```