CNT 4713 – Project 1

Overview:

This project develops a web status monitor (simplified version of uptimerobot.com) to practice web programming and understand the web related protocols: HTTP and TLS/SSL. You must implement your own HTTP client socket to interact with the web server, and cannot use any existing HTTP client library. You may use an existing SSL library to help implement the HTTPS client to earn extra credits. You may use Python3, Java, C++, or C as the programming language.

Background:

The Hyper Text Transfer Protocol (HTTP) is a TCP based application layer protocol for the World Wide Web. The HTTP RFC [1] is the official document that defines the protocol. HTTP is explained in detail in Chapter 2.4 of the textbook.

The Transport Layer Security (TLS) (replacing the Secure Socket Layer (SSL) protocol) is a security protocol that provides authentication and encryption to protect TCP traffic. The TLS RFC [2] is the official document that defines the protocol. TLS/SSL is explained in detail in Chapter 8.6 of the textbook.

Instructions:

The web monitor should start by typing:

`monitor urls-file` or
`java monitor urls-file` or
`python monitor.py urls-file`

where `urls-file` is the name of a text file that has the list of URLs to be monitored. Your program should then fetch each URL, analyze the response from the web server, and print the status in the terminal. **Please make sure that your program name is exactly "monitor". Submissions will be graded by an automated script. Unrecognized program names will result in a zero.**

A production web monitor will repeatedly fetch the provided URLs for continuous status monitoring. For easy grading, the web monitor developed in this project should exit after fetching all the URLs once.

A program skeleton is provided in Canvas for you to get started. Detailed steps to implement the web monitor are explained below.

1. Parse URL

The first step is to parse the URLs provided in `urls-file`. As explained in Chapter 2.2.1 of the textbook, a URL specifies the protocol, host, and path that will be needed to fetch the URL. For the following example URL
`http://www.someSchool.edu/someDepartment/picture.gif`

- `http://` Indicates that the protocol is HTTP, and hence the web server is listening at the port 80. If the URL starts with `https://` instead, the protocol will be HTTPS, and the web server will be listening at the port 443;
- `www.someSchool.edu` is the host name, which will be translated to an IP address where the web monitor should connect to;
- `/someDepartment/picture.gif` is path of the requested object on the server, which must be presented in the HTTP request message sent to the server.

2. Establish TCP connection

Since HTTP is a TCP based application layer protocol, a TCP connection must be established before HTTP messages can be exchanged. As explained above, the host name and port number can be obtained from the URL. Note that the default server port number is 80 for HTTP and 443 for HTTPS.

3. Network error

If the TCP connection cannot be established because of a network error, such as a dead server, network outage, or timeout, the web monitor should report such an error. A sample output for the network error is as follows.

```
URL: http://fiu.gov
Status: Network Error
```

If the TCP connection is established, but messages cannot be successfully sent or received. The web monitor should report the network error in a similar way.

4. Construct HTTP request message

Once the TCP connection is established, the web monitor is ready to send the first HTTP request message. The request message must follow the specified format for the server to understand. The HTTP request format is explained in detail in Chapter 2.2.3 of the textbook.

Again, if the HTTP request message cannot be successfully sent, a network error should be reported.

5. Analyze HTTP response message

If the request message is correctly constructed, the web monitor will receive a HTTP response message from the server, and then should report the URL status. The HTTP response format is explained in detail in Chapter 2.2.3 of the textbook.

The HTTP response message contains a status code for the requested URL. In this project, we will focus only on the following status codes:

- 2XX successful responses
- 3XX redirection messages
- 4XX client error messages

The web monitor should parse the status returned for each URL and print it. Sample outputs of for the 2XX, 3XX, and 4XX statuses are as follows.

```
URL: http://inet.cs.fiu.edu/
Status: 200 OK

URL: http://google.com/
Status: 301 Moved Permanently

URL: http://google.com/404
Status: 404 Not Found
```

If the HTTP response message cannot be successfully received, a network error should be reported.

5. Follow URL redirection

If the status code in the HTTP response message is 301 or 302, it means that the requested object has been redirected permanently or temporarily to a new URL, which is also presented in the HTTP response message. The web monitor should continue fetching the redirected URL and report its status.

The following sample output shows that the original URL returns a 301 status, and the redirected URL returns a 200 status.

```
URL: http://google.com/
Status: 301 Moved Permanently
Redirected URL: http://wwww.google.com/
Status: 200 OK
```

The following sample output shows that the original URL returns a 302 status, and the redirected URL returns a 403 status.

```
URL: http://inet.cs.fiu.edu/temp/deny.html
Status: 302 Found
```

```
Redirected URL: http://inet.cs.fiu.edu/temp/denied.html
Status: 403 Forbidden
```

6. Fetch referenced object

The HTTP response in most cases is a HTML page, which may have referenced objects, such as images or videos. The web monitor should also fetch the referenced objects and report their statuses.

In this project, we will focus only on images that are referenced using the HTML `img` tag as follows:
`<img src=image_url alt=alternative_text>`
in which `src=image_url` gives the URL of the referenced image. `image_url` may be a complete URL like `http://abc.com/images/pic.jpg`, or only the path like `/images/pic.jpg` if the referenced image is located on the same server as the original HTML file.

The following sample output shows that the original URL has a referenced URL, and the referenced URL returns a 200 status.

```
URL: http://inet.cs.fiu.edu/page.html
Status: 200 OK
Referenced URL: http://inet.cs.fiu.edu/fiu.jpg
Status: 200 OK
```

The following sample output shows that the original URL has a referenced URL, and the referenced URL returns a 404 status.

```
URL: http://inet.cs.fiu.edu/temp/page.html
Status: 200 OK
Referenced URL: http://inet.cs.fiu.edu/temp/fiu.jpg
Status: 404 Not Found
```

7. Monitor HTTPS URLs (extra credits)

If the URL starts with `https://`, it means that the HTTP session is protected by the TLS/SSL protocol, and the requests and responses are encrypted. In this case, the web monitor needs to encrypt requests and decrypt responses, which can be achieved with an existing TLS/SSL library, such as the Python ssl library [3] or Java javax.net.ssl package [4].

Test:

A sample URLs text file is provided in Canvas for testing. The expected output is as follows.

```
URL: http://fiu.gov
```

```
Status: Network Error

URL: http://inet.cs.fiu.edu/
Status: 200 OK

URL: http://google.com/404
Status: 404 Not Found

URL: http://google.com/
Status: 301 Moved Permanently
Redirected URL: http://wwww.google.com/
Status: 200 OK

URL: http://inet.cs.fiu.edu/temp/deny.html
Status: 302 Found
Redirected URL: http://inet.cs.fiu.edu/temp/denied.html
Status: 403 Forbidden

URL: http://inet.cs.fiu.edu/page.html
Status: 200 OK
Referenced URL: http://inet.cs.fiu.edu/fiu.jpg
Status: 200 OK

URL: http://inet.cs.fiu.edu/temp/page.html
Status: 200 OK
Referenced URL: http://inet.cs.fiu.edu/temp/fiu.jpg
Status: 404 Not Found

URL: https://wwww.fiu.edu
Status: 200 OK
```

References:

- [1] Hypertext Transfer Protocol -- HTTP/1.1, https://www.ietf.org/rfc/rfc2616.txt.
- [2] The Transport Layer Security (TLS) Protocol Version 1.3, https://www.rfc-editor.org/rfc/rfc8446.
- [3] Python ssl library, https://docs.python.org/3/library/ssl.html.
- [4] Java javax.net.ssl package, https://docs.oracle.com/javase/8/docs/api/javax/net/ssl/package-summary.html.

Group:

You may work in a group of up to three. Only one submission is necessary for a group.

Submission:

Submit a copy of your final code to Canvas. (Source code only, no NetBeans/Eclipse… project files)

Submit a readme.txt to Canvas with the following information:

1. Member names and IDs,
2. Language used: Python3/Java/C++/C,
3. SSL library used if any,
4. Compiling instructions (not necessary for Python): command-line compiling instructions only, no NetBeans/Eclipse… compiling.

Grading:

Item	**Percentage**
Network error	10%
2XX status	15%
3XX status, URL redirection success	15%
3XX status, URL redirection failure	15%
4XX status	15%
Referenced object, success	15%
Referenced object, failure	15%
HTTPS	20% (extra credit)

Code plagiarism will be reported for academic dishonesty.