

CSc 361: Computer Communications and Networks (Spring 2022)

Programming Assignment 1: Smart Web Client

Spec Out: Jan 12, 2022
Final Due: 23:55 pm, Jan 28, 2022

1 Goal

The project is to build a tool at web client to collect information regarding a web server. The purpose of this project is twofold:

- to provide students with hands-on experience with socket programming **in Python**,
- to help students understand the application-layer protocols HTTP/HTTPS. Note that HTTPS is not a standalone protocol, but instead, it is HTTP over Transport Layer Security (TLS). In this assignment, your main focus is HTTP, not TLS.

2 Background

2.1 HTTP

HTTP stands for Hyper Text Transfer Protocol and is used for communication for web services.

The web client initiates a conversation by opening a connection to a web server. Once a connection is set up, the client sends an HTTP request to the web server. The server sends an HTTP response back to the client. An HTTP request consists of two parts: a header and a body. Whether a body follows a header or not is specified in the header.

Using *single-line header of HTTP request* as an example, the first line of any request header should be:

- the method field: The method field can take on several different values, including GET, POST, HEAD, and so on.
- the URL field: It is the field to identify a network resource, e.g., “http://www.csc.uvic.ca/index.html”.
- the HTTP version field

The response from a server also has two parts: a header and a body. The first line of a header should be:

- the HTTP version field,
- the status code field,

- the phrase field.

Two main status codes include 200 and 404. The status code 200 means that the request succeeded and the information is returned in the response. The status code 404 means that the requested document does not exist on this server. Two example response messages are: “*HTTP/1.0 404 Not Found\r\n\r\n*” and “*HTTP/1.0 200 OK\r\n\r\n data data data ...*” Another two status codes 505: “HTTP Version Not Supported”, and 302: “302 found” for URL redirection are also useful for this assignment.

2.2 URI

URI stands for Uniform Resource Identifier and is also known as the combination of Uniform Resource Locators (URL) and Uniform Resource Names (URN). It is a formatted string that identifies a network resource. It generally has the format: *protocol://host[:port]/filepath*. When a port is not specified, the default HTTP port number is 80, and the default HTTPS port number is 443.

2.3 Cookies

An HTTP cookie is a small piece of data that a server sends to the user’s web browser. The browser may store it and send it back with the next request to the same server. Typically, it’s used to tell if two requests came from the same browser — keeping a user logged-in, for example. It remembers stateful information for the stateless HTTP protocol. Cookies have many applications in web, such as tracking, authentication, and web analytics. Due to this reason, cookies also cause many concerns about security and privacy breach.

The textbook includes simple introduction on cookies. More detailed information could be found at: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies>. Python includes dedicated modules to handle Cookies: <https://docs.python.org/3/library/http.cookies.html>. Nevertheless, you are **no allowed** to use this package because it defeats the purpose of this assignment: understanding the nuts and bolts of HTTP.

3 Project Description

You are required to build a smart web client tool, called *SmartClient*, in Python. **Note that for consistency, program in other languages will not be accepted!**

Given the URL of a web page, your *SmartClient* needs to find out the following information regarding the web server:

- 1. whether or not the web server supports http2,
- 2. the cookie name, the expire time (if any), and the domain name (in any) of cookies that the web server will use,
- 3. whether or not the requested web page is password-protected.

Your program first accepts URI from stdin and parses it. Then it connects to a server, sends an HTTP request, and receives an HTTP response. You should also implement a routine that prints out the response from the server, marking the header and the body. When you finish the client,

66 you can try to connect to any HTTP server. For instance, type “www.uvic.ca” as the input to the
67 client program and see what response you get.

68 As an *example* output, after you run your code with

```
69 % python SmartClient.py www.uvic.ca
```

70 Your *SmartClient* may output the received response from the server (**optional**), e.g.,

```
71 ---Request begin---
```

```
72 GET http://www.uvic.ca/index.html HTTP/1.1
```

```
73 Host: www.uvic.ca
```

```
74 Connection: Keep-Alive
```

```
75
```

```
76 ---Request end---
```

```
77 HTTP request sent, awaiting response...
```

```
78
```

```
79 ---Response header ---
```

```
80 HTTP/1.1 200 OK
```

```
81 Date: Tue, 03 Jan 2017 22:42:27 GMT
```

```
82 Expires: Thu, 23 Nov 2017 08:52:00 GMT
```

```
83 Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
```

```
84 Pragma: no-cache
```

```
85 Set-Cookie: SESSID_UV_128004=VD3v0JhqL3YUbmazSTJre1; path=/; domain=www.uvic.ca
```

```
86 Set-Cookie: uvic_bar=deleted; expires=Thu, 04-Jan-2018 00:00:01 GMT; Max-Age=0; path=/; dom
```

```
87 Keep-Alive: timeout=5, max=100
```

```
88 Connection: close
```

```
89 Content-Type: text/html; charset=UTF-8
```

```
90 Set-Cookie: www_def=2548525198.20480.0000; path=/
```

```
91 Set-Cookie: TS01a564a5=0183e07534a2511a2dcd274bee873845d67a2c07b7074587c948f80a42c427b1f7ea
```

```
92 Set-Cookie: TS01c8da3c=0183e075346a73ab4544c7b9ba9d7fa022c07af441fc6214c4960d6a9d0db2896; p
```

```
93 Set-Cookie: TS014bf86f=0183e075347c174a4754aeb42d669781e0fafb1f43d3eb2783b1354159a9ad8d81f7
```

```
94
```

```
95 --- Response body ---
```

```
96 Body Body .... (the actual content)
```

```
97
```

98 Note that some lines in the above output were truncated.

99 Your code may need to send **multiple requests** in order to find out the required information.

100 **In particular, if you get an HTTP response with code 302 or 301, you need to send**
101 **further HTTP requests to the new URI provided by the Location header.**

102 Your code should output the final results (**mandatory**), for example:

```
103 website: www.uvic.ca
```

```
104 1. Supports http2: no
```

```
105 2. List of Cookies:
```

```
106 cookie name: SESSID_UV_128004, domain name: www.uvic.ca
```

```
107 cookie name: uvic_bar, expires time: Thu, 04-Jan-2018 00:00:01 GMT; domain name: .uvic.ca
```

```
108 cookie name: www_def,
```

```
109 cookie name: TS01a564a5
110 cookie name: TS01c8da3c, domain name: www.uvic.ca
111 cookie name: TS014bf86f, domain name: .uvic.ca
112 3. Password-protected: no
```

113 **Note that the above output may be outdated and does not necessarily reflect the**
114 **ground truth of the current configuration of www.uvic.ca.**

115 3.1 Other Notes

- 116 1. Regarding other printouts: Anything not specified in Assignment 1 is optional. For example,
117 you can decide whether or not to print out the IP address, port number, and so on. When
118 TAs test your code, if your code works fine without any problem, you are fine even if you
119 do not print out anything not required in Assignment 1. Nevertheless, if your code does not
120 work, TAs will not spend time to figure out what is wrong and you get a zero mark on the
121 required function (Refer to the table in Section 5 of Assignment 1). In this case, if your code
122 includes some printout to show intermediate results, TAs will have an idea of how far you
123 have achieved and give you some partial mark based on their judgement.
- 124 2. Regarding the Python network/http packages: those that you are allowed to use include:
125 socket, ssl, re, time, os, sys, select, threading, etc; those that you are **not** allowed to use
126 include: httplib, TCPClient, hyper, requests, and other similar third-party packages, since
127 these packages define classes that already implement the client side of the HTTP and HTTPS
128 protocols. By calling the functions in these packages, you essentially ask Python to do all the
129 jobs for you. This defeats the purpose of a network course.
- 130 3. Regarding the readme file: Readme file is important. Without it TAs will not know how to
131 compile your code and how to run your code. It would waste our time to deal with your
132 complaint if TAs cannot run your code and give you a zero.
- 133 4. For more information on HTTP, HTML, URI, etc., please refer to <http://www.w3.org>. It is
134 the home page of W3 Consortium, and you will find many useful links to subjects related to
135 the World Wide Web.

136 4 Schedule

137 In order to help you finish this programming assignment successfully, the schedule of this assignment
138 has been synchronized with both the lectures and the tutorials/ labs. Before the final deadline, there
139 are three tutorial sessions to help you finish the assignment. A schedule is listed as follows:

Session	Tutorial	Milestones
Tut 1	P1 spec go-through, design hints, python	design and code skeleton
Tut 2	socket programming and testing	alpha code done
Tut 3	socket programming and last-minute help	beta code done

140

5 Deliveries and Marking Scheme

141

For your final submission of each assignment you are required to submit your source code to brightSpace in a single zip file (double-check your zip file to make sure all required files have been included before submission!). You should include a readme file to tell TA how to compile and run your code.

145

Note: For consistency and ease of test, you should test/run your code on the server *linux.csc.uvic.ca* by running python3 and the python packages supported by the server *linux.csc.uvic.ca*. In other words, TAs will test your code on *linux.csc.uvic.ca* and give marks based on the test results over *linux.csc.uvic.ca* rather than the results from your local computer.

149

The marking scheme is as follows:

150

Components	Weight
Error handling	10
Correct output for “support of http2 ”	20
handling http redirect 302/301	20
List of Cookies	30
Correct output for “password-protected”	15
Readme.txt	5
Total Weight	100

151

Important Note: listing cookies is a very tricky business, and it is possible that you will not get a unique, static answer due to the dynamic changes in some cookies created dynamically based on users’ interactive input. Some online tool, such as <http://www.cookie-checker.com/>, can find cookies that are triggered by javascript or php code. Nevertheless, finding those cookies is optional for this Assignment.

155

156

6 Plagiarism

157

This assignment is to be done individually. You are encouraged to discuss the design of your solution with your classmates, but each person must implement their own assignment.

158

159

The End
