Theo Novak
CSE 150
6 March 2022

Discussion

The application is designed to be a simplified implementation of the *curl* command line utility, which issues a request to a given url and prints the HTML response from the webpage to the terminal on successful retrieval, or one of an assortment of error messages on unsuccessful retrieval. The file is run with either of the following formats:

(1) python tjnovakMyCurl.py [url]
(2) python tjnovakMyCurl.py [url] [hostname]

In (1), the user must provide a full url containing the server's hostname, such as *http://www.google.com*. In (2), the url argument is an IP address, in which the hostname must also be provided as a second argument for proper behavior. An example of this would be using the url: http://93.184.216.34, and the hostname: www.example.com. On successful retrieval, the program will write the HTML content received from the web server to a file, HTTPoutput.html, and append an entry with information about the connection to another file, Log.csv. On unsuccessful retrieval, the program does not write to HTTPoutput.html, but still appends an entry to Log.csv with connection information. Entries in Log.csv have the following format:

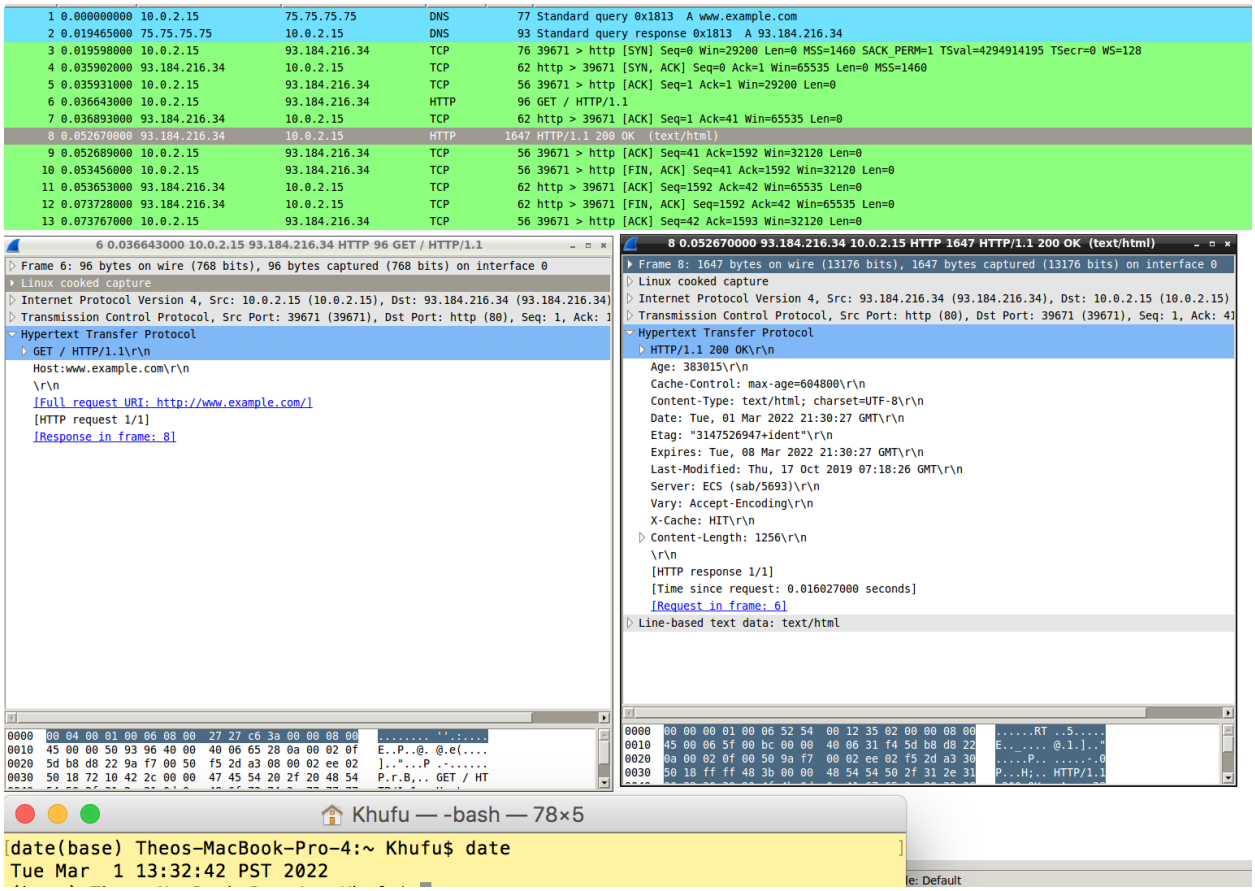
> Successful or Unsuccessful, Server Status Code, Requested URL, hostname, source IP, destination IP, source port, destination port, Server Response line (including code and phrase)

As noted at the beginning of the document, the program is not a perfect replica of *curl*'s capabilities. Our program only operates with the HTTP protocol, and cannot handle HTTPS connections to web servers. It also requires that urls begin with the *http://* prefix, designating that HTTP connection should be used, and aborts execution prematurely if this prefix is not present. The program also does not work with chunk-encoded webpages, such as *http://www.google.com*, and will print an error message to the terminal specifying that chunk-encoding is not supported. The *curl* command also does not abide by our restrictions on command line argument input, and can work with a single IP address argument without supplying a hostname. This is because *curl* can issue a DNS query to find the hostname if omitted, which our program does not support.

As part of the project deliverables, we were tasked with testing two successful downloads, two unsuccessful downloads, one redirect, and a request to port 443. These tests are provided in the submitted Log.csv file with the appropriate connection data.

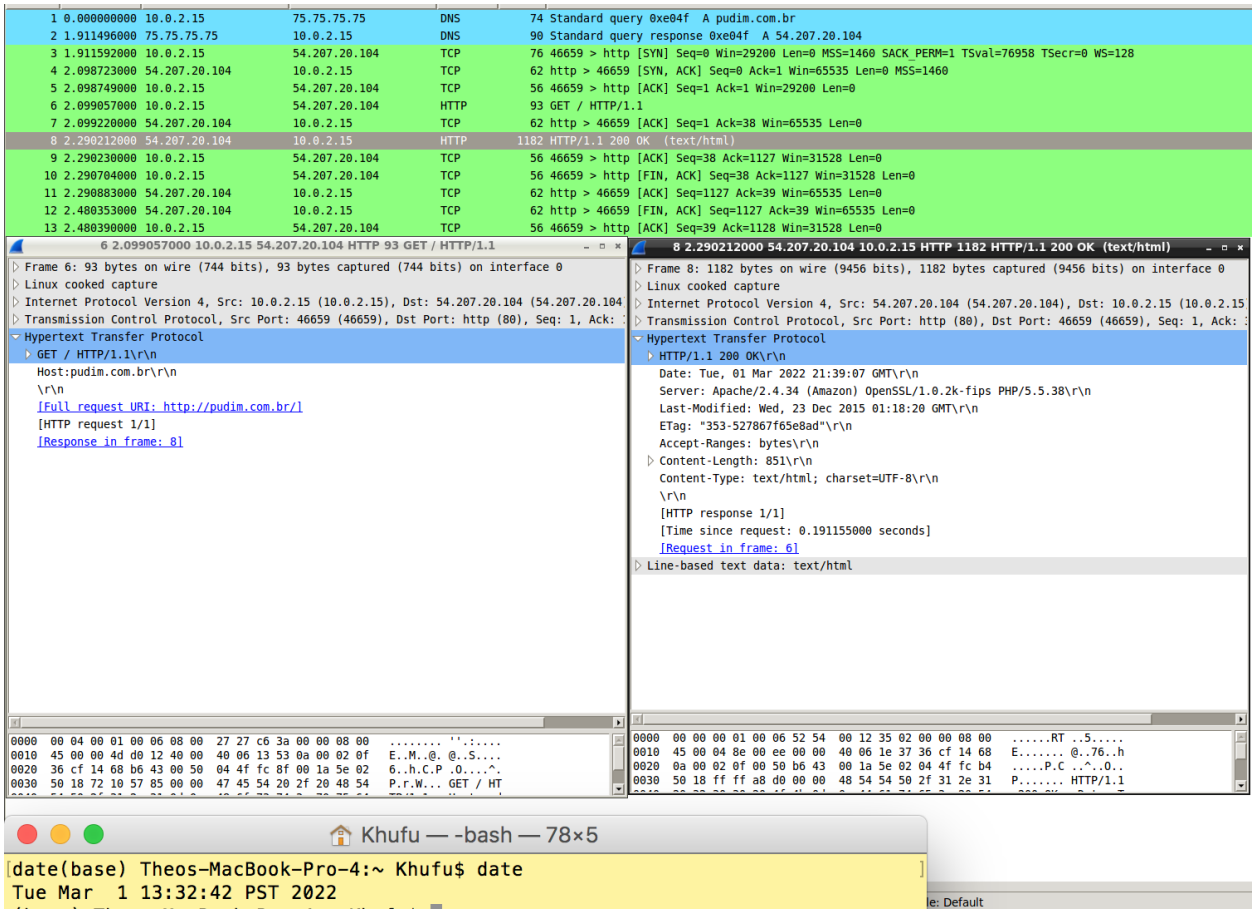
Successful Downloads:

(1) http://www.example.com



This is the first of our successful downloads, whose output matches that of *curl*.

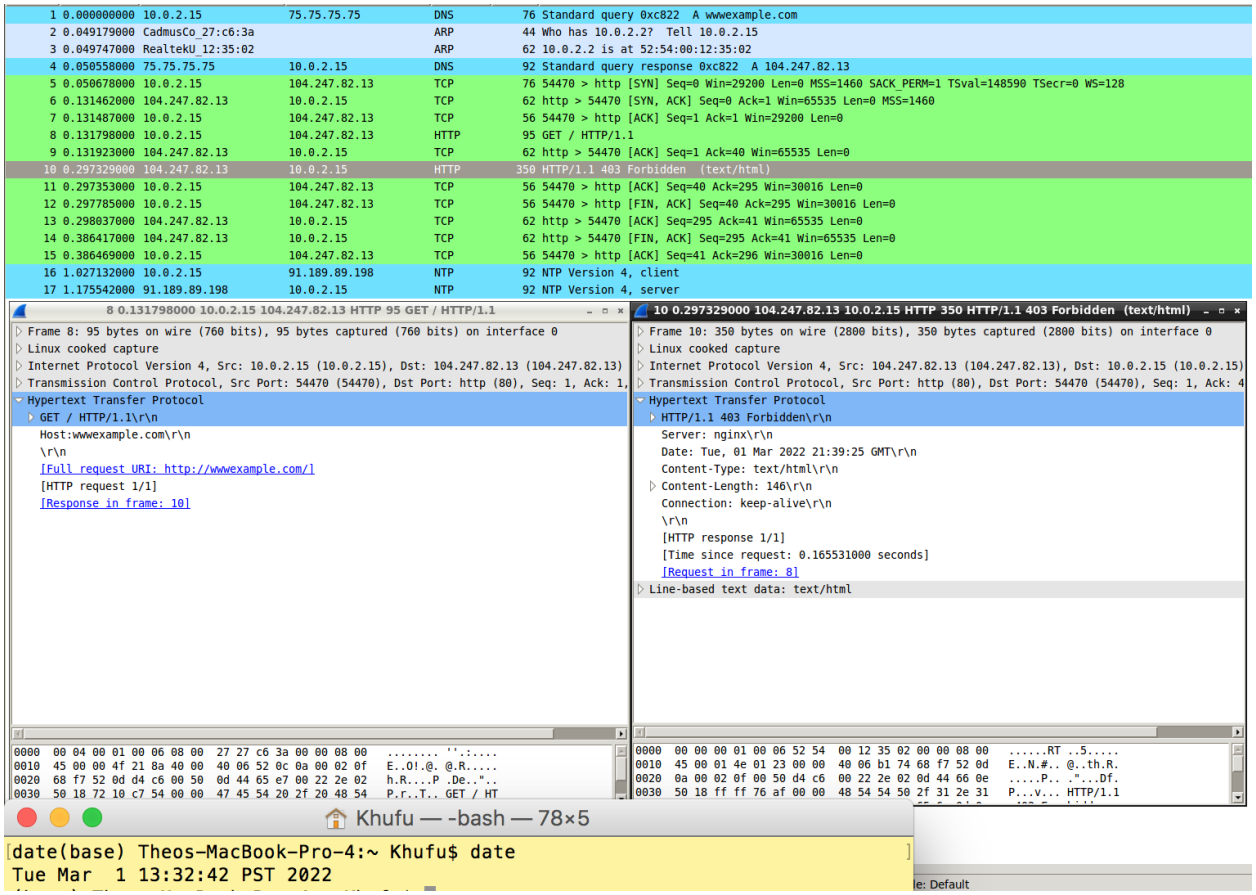
(2) http://www.pudim.com.br.com



This is the second of our successful downloads, whose output matches that of *curl*.

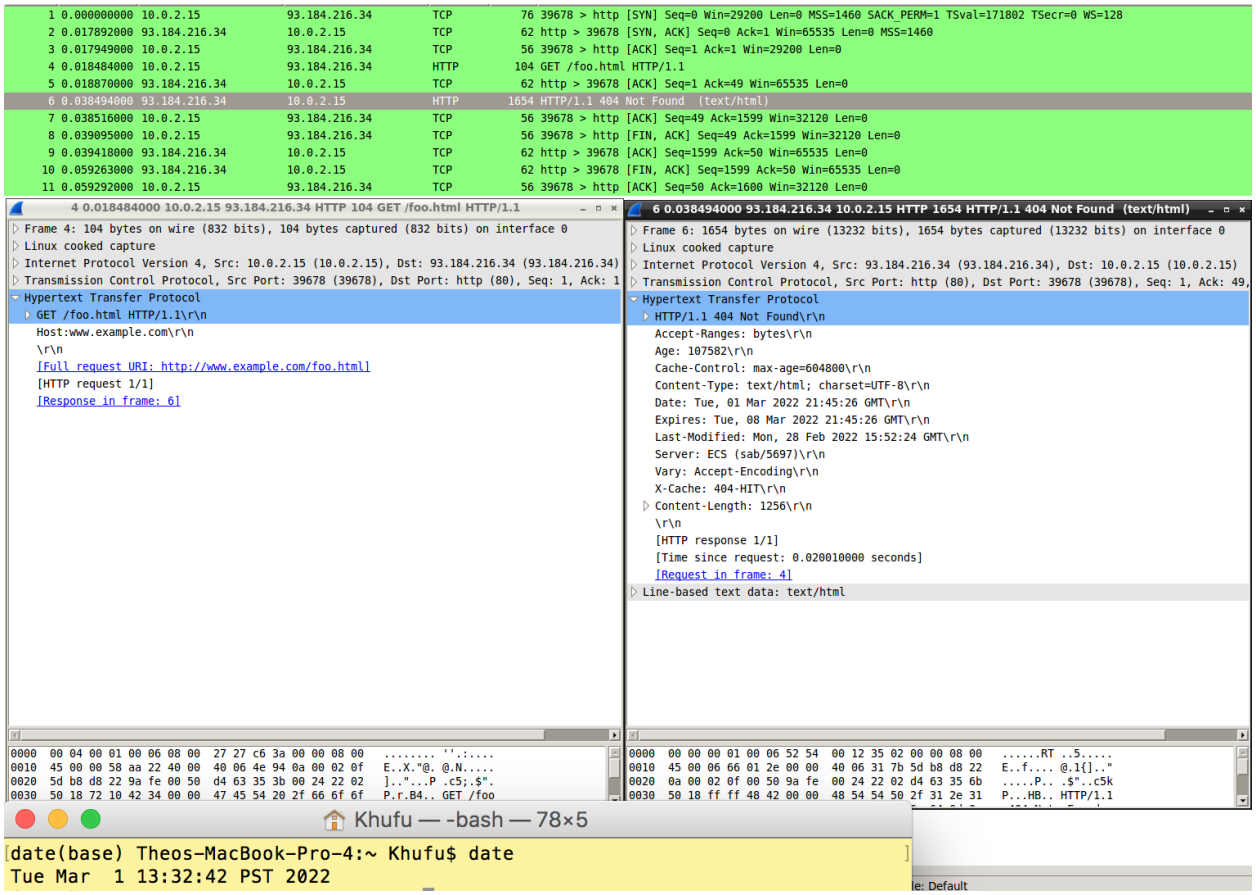
Unsuccessful Downloads:

(1) http://wwwexample.com



This is the first of our unsuccessful downloads, whose output matches that of *curl*. Both receive a 403 Forbidden status code, meaning that both programs do not have access to the given website.

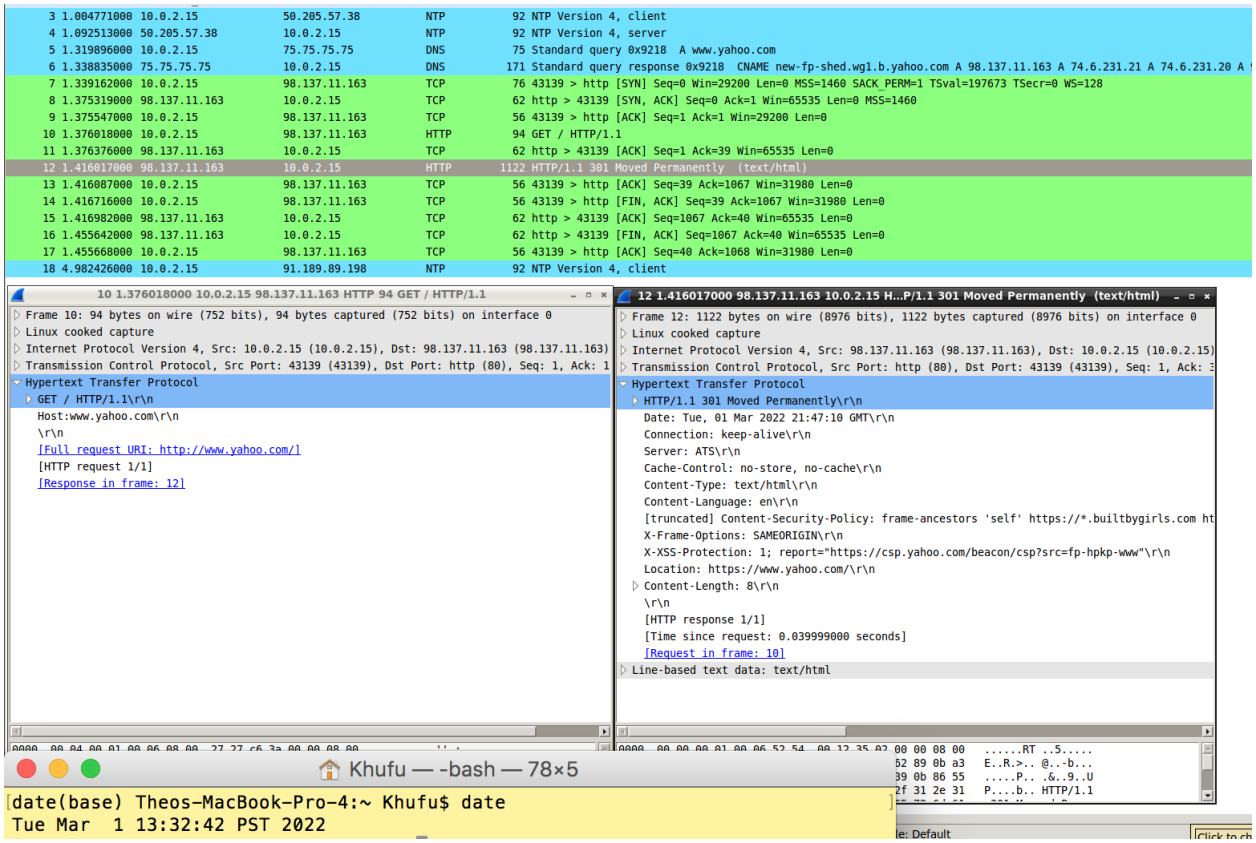
(2) http://93.184.216.34/foo.html www.example.com



This is the second of our unsuccessful downloads, whose output does not match that of *curl*. The retrieval is unsuccessful because the page *foo.html* does not exist on that site, which responds with 404 Not Found, but also replies with the base HTML page www.example.com. The difference in outputs between our program and *curl* occurs because we do not write to HTTPoutput.html if the status code is not 200, indicating a successful download, while *curl* will instead return the base HTML page of www.example.com despite also receiving a 404 Not Found status line.

Redirect:

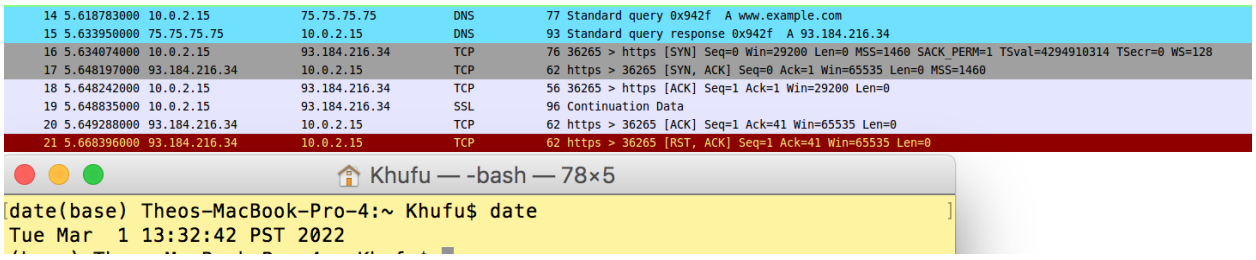
(1) http://www.yahoo.com



This is our redirect url test, whose output does not match that of *curl*. Our program will reply to the terminal with: “Unsuccessful: www.yahoo.com, 301”, as specified in the project document. *Curl* instead responds with “redirect%”. The essence of both of the responses is the same, however, as status code 301 indicates that the site has been moved permanently and is used for permanent redirecting.

Request to port 443:

(1) http://www.example.com:443



This is our test using port 443 in a url, whose output closely resembles that of *curl*. Our program responds to the exception by printing: “Recv failure: connection reset by peer”, while *curl* responds with: “curl: (56) Recv failure: connection reset by peer”. This is because port 443 is explicitly for HTTPS connections, and our HTTP connection is quickly rebuffed as it is incompatible.