NT Assignment 4

HTTP, Start of Project

Introduction

In this assignment we are going to explore the HyperText Transfer Protocol ([HTTP](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol)[[1]](#footnote-1)).

HTTP is an Internet application protocol. It is the workhorse of the Web. HTTP enables clients to create, retrieve, update or delete web resources. It is used by browsers to fetch web pages (and associated images) from the web sites. It is also used by software programs to access remote Web Services.

HTTP is a client/server protocol. An HTTP server (e.g. Apache or IIS) is listening on a port (usually 80) for requests from clients. It will respond to HTTP requests received on this port. Servers do not take initiative. They are just there to serve. A client that wants a representation of a specific resource like a web page, or wants to get some data processed, has to connect to the right host/port combination, submit a valid HTTP request, and wait for the response from the server.

Some very useful introductory reading on HTTP:

* HTTP Made Really Easy  
  <http://www.jmarshall.com/easy/http/>
* How the web works: HTTP and CGI explained

<http://www.garshol.priv.no/download/text/http-tut.html>

Task 1: Performing HTTP requests manually (putty)

First we will try to communicate with a web server by manually submitting HTTP requests. We will use PuTTY to communicate with a webserver. To obtain PuTTY on your Ubuntu node, just enter the following command:

sudo apt-get install putty

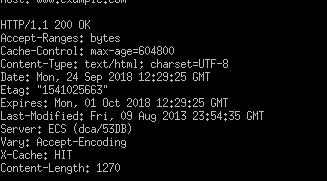
To send a HTTP Request with PuTTY you have to do the following

* Enter ‘Host Name (or IP address)’: e.g. [www.example.com](http://www.example.com)  
  (enter ‘localhost’ if your server is running on your own PC)
* Enter ‘Port’: 80 (could be different on some host)
* Select ‘Connection Type’: Raw
* Select ‘Close window on exit’: Never
* Press ‘Open’

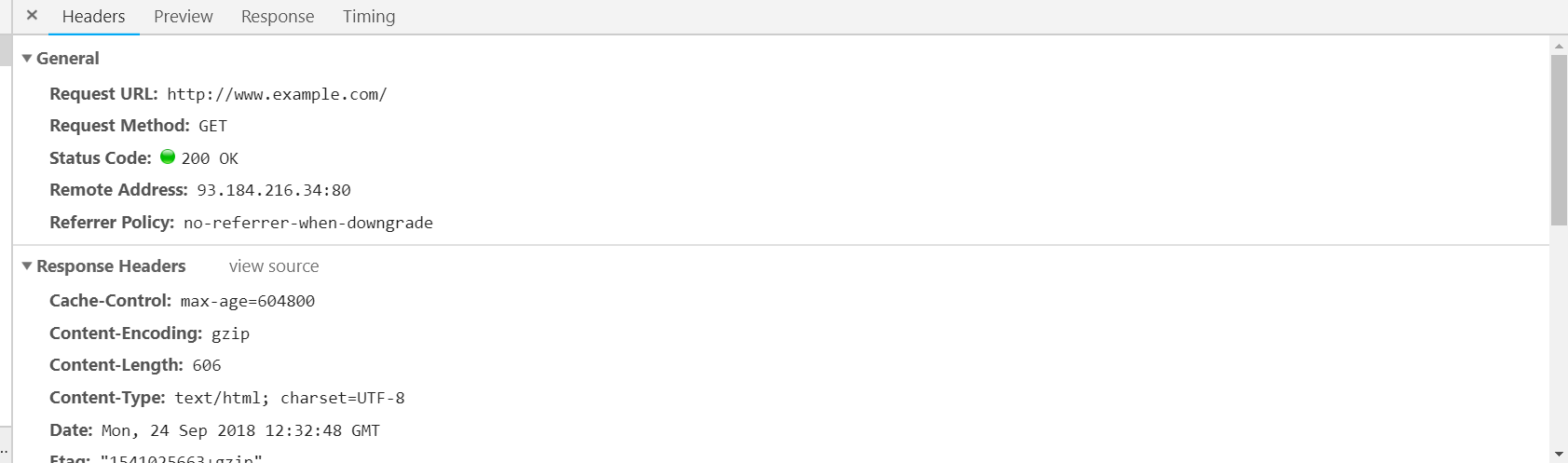
Perform the following tests and answer all questions:

1. Open your favorite browser and enter the URL [www.example.com](http://www.example.com). Look what happens.
2. Let us do what the browser does, but manually, step by step. Connect to [www.example.com](http://www.example.com) (with PuTTY). Type the following two lines in the PuttY window and after that hit the <enter> key twice:  
     
    GET /index.html HTTP/1.1  
    Host: www.example.com

What response do you get? What does it mean?



These are the headers. These headers can also be found when you press F12 under network> headers as shown below:



Task 2: Using cURL to do HTTP

In this assignment you are going to use cURL to talk to an HTTP server.

To obtain cURL on your Ubuntu node, just enter the following command:

sudo apt-get install curl

Get acquainted with cURL first. Here are some useful links to get you going:

* <http://curl.haxx.se/docs/httpscripting.html>
* <http://www.codediesel.com/tools/6-essential-curl-commands/>
* <http://www.thegeekstuff.com/2012/04/curl-examples/>

Note: cURL built-in help can be activated with the command ‘curl --help’

Now do the following tasks (and test if your answers work!):

* Define a cURL command to do everything (except nice display) that the browser did in task 1.

Write down the used curl command.

We used curl [www.example.com](http://www.example.com) -v and this gave the following result:



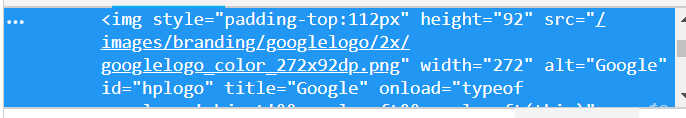
* Define a cURL command to download and save the Google logo from the main page of the website [www.google.com](http://www.google.com).

What extension (based on image type) should the downloaded file have? How can you find that in the HTTP response header?

Write down the used curl command.

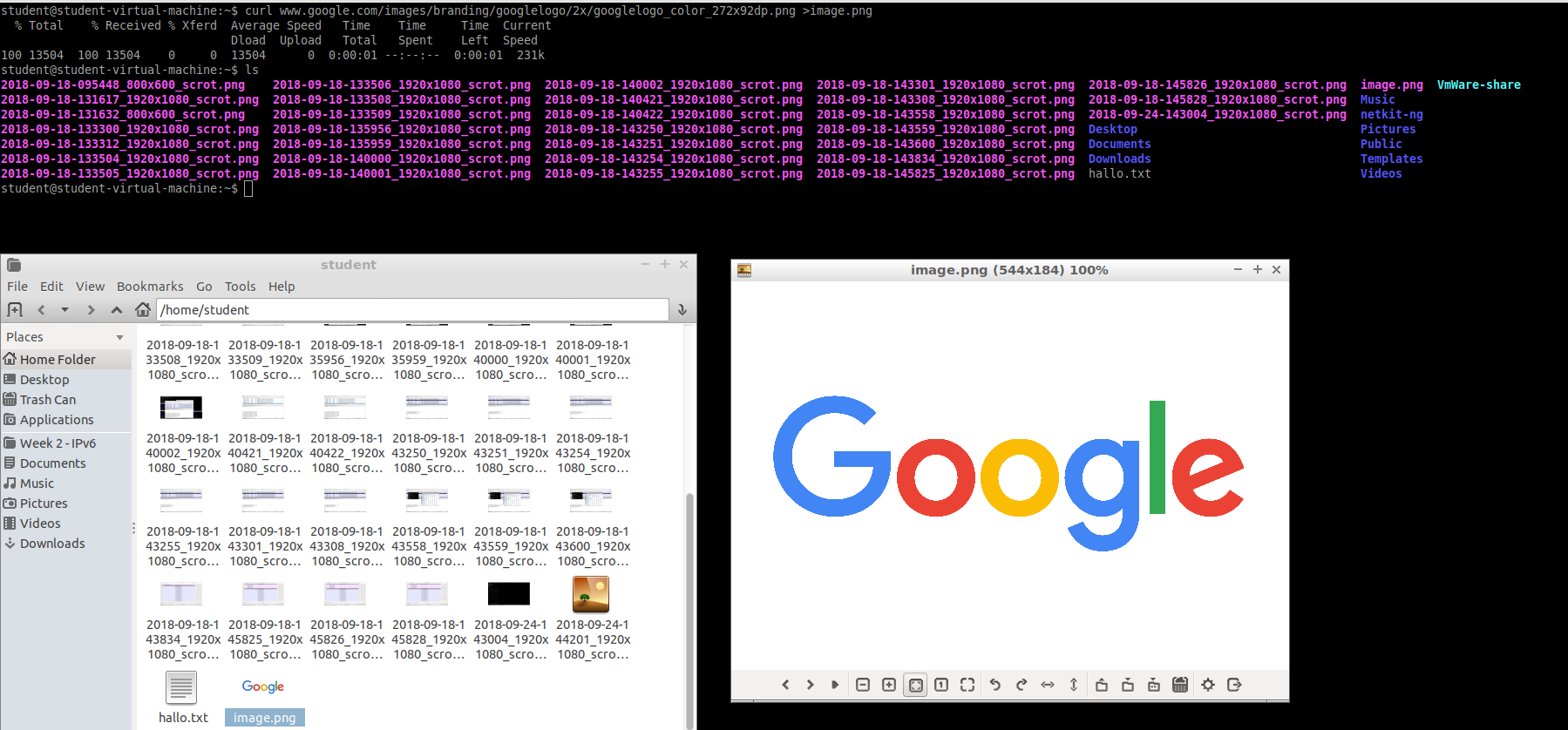
The image should be a .png, because the image on google was already a png.

We found out that the image was a png by pressing f12, then clicking on the google image, and we found the following:



Here you can see that the image is a .png

After this we used the following command as shown below, using the filepath of the image that we found in the f12 page. We saved the image as image.png, and it was copied to the home directory.



* Define a cURL command that does the same POST as the HTML-form in the following w3schools.com example:  
  <http://www.w3schools.com/tags/tryit.asp?filename=tryhtml_form_method_post>

Write down the used curl command.

First we had to find out what the url was for the result after putting in the names:

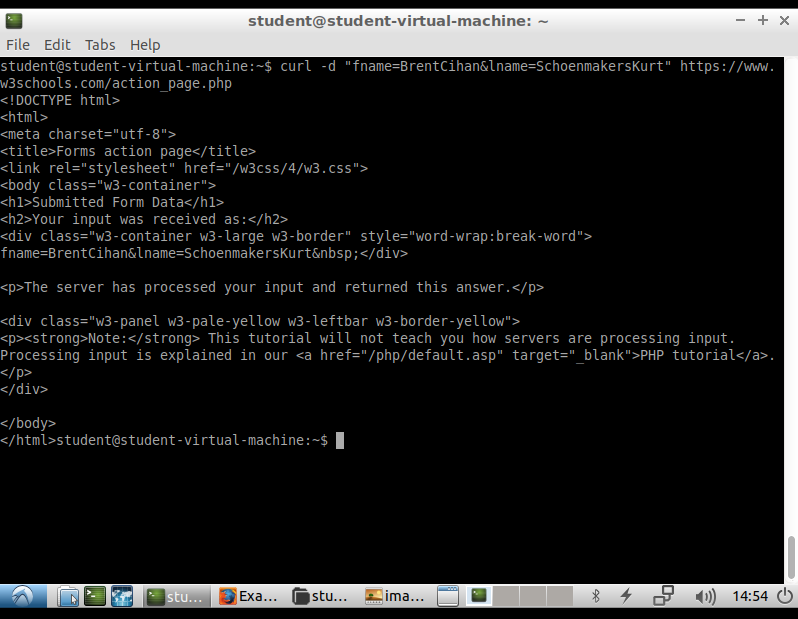
<https://www.w3schools.com/action_page.php>

in the example code, you could see that the firstname textbox was named fname, and the lastname textbox was named lname.

After knowing this, you could do the following code:

curl -d “fname=BrentCihan&lname=SchoenmakersKurt” <https://www.w3schools.com/action_page.php>

this command gave the following result:



In this screenshot, it is seen that the fname was BrentCihan and the lname was SchoenmakersKurt, just as we gave as an input in the command:



Task 3: Apache Web Server in the Netkit Lab

For this assignment we’re going to reuse Netkit lab from the previous assignment. We’re going to use *WebServer* node to set up simple Apache Web Server. Our client will be the *Laptop* node.

Find out a start up command for the Apache server and add it to WebServer.startup.

**We put the following line in our webserver.startup:**

**etc/init.d/apache2 start**

You can check your correct configuration by entering the following command on *WebServer*:

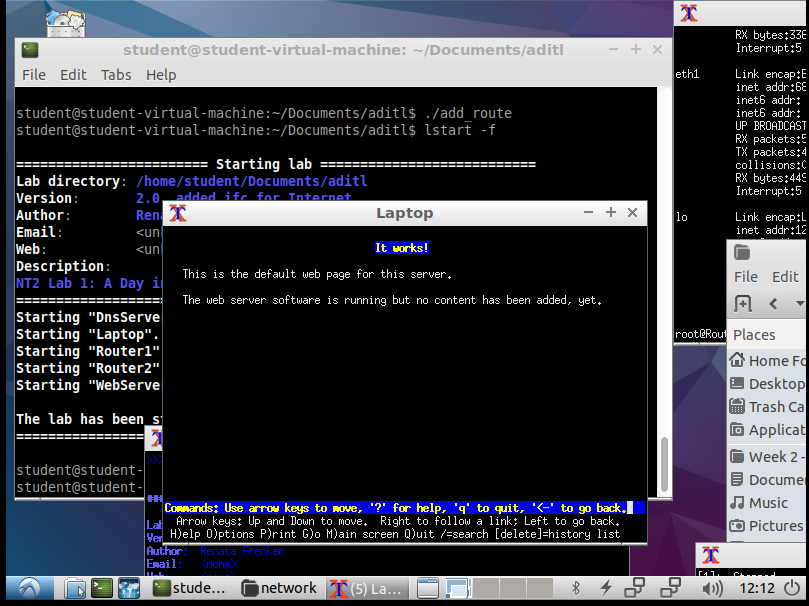
/etc/init.d/apache2 status

The default apache2 setup offers a test html page located in /var/www/index.html.

Start the links[[2]](#footnote-2) browser from your *Laptop* node and connect to your *WebServer* node (“g” for entering URL). You should now be able to see the default page from the *WebServer*.

Provide a screenshot of what you see in links after successful connection to the *WebServer*.

We put in the following command in laptop: lynx 64.233.64.1 (IP of webserver).

This gave us the following screen in the laptop host: 

Task 4: Start up of the embedded project

In this task you’ll start to work on your ongoing project for the next assignments.

In the next assignments you’re going to work on an embedded project of your choice.

The requirements:

1. Your project has to have at least one embedded networking system (e.g. Arduino with Ethernet shield, ESP8266, Raspberry Pi, …) and one PC application that are going to communicate
2. Embedded systems has to have some sensors/actuators to manage
3. PC is going to run a dashboard through which you can manage your hardware and also get some status notifications from it.
4. Your project has to be implementd by using HTTP protocol (or a Web Services protocol) and at least one more IoT protocol of your choice (further details on this in the next assignment)

This project is going to be used to exercise several networking protocols for the management of an embedded system and to measure and compare their performance.

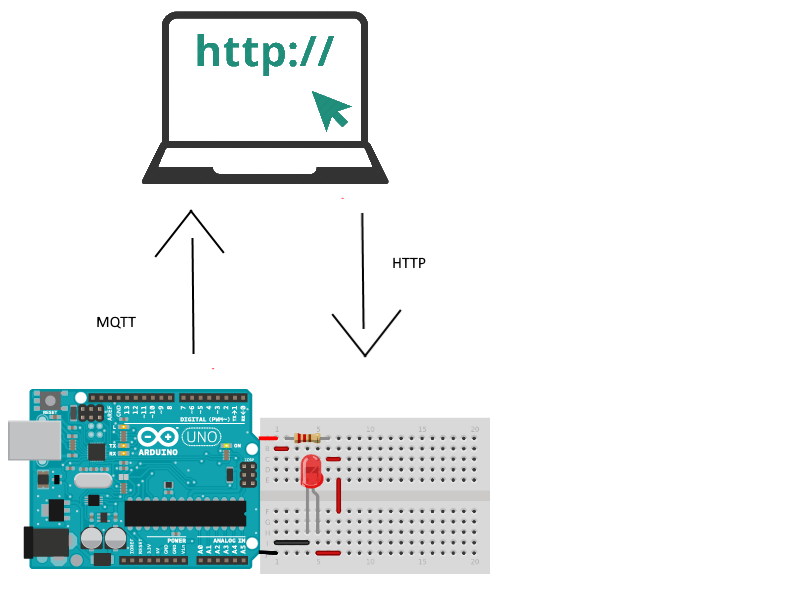
Of course, any own valuable extensions and creativity will be taken into account during assessment of this project.

For this assignment you have to deliver:

1. **Short description of what you intend to do**

We are going to host a webpage on our laptop. The laptop will also be a MQTT Message Broker.  
The Arduino will have a LED. The Arduino will be subscribed to the broker’s topic.   
  
Our Arduino will have multiple buttons as well. When one of the buttons is pressed, the Arduino will send a message with MQTT to the laptop > webpage.  
The webpage > laptop will answer with a message over HTTP.   
After receiving the response, the Arduino will change the LED.

1. **Architecture of your system (components and connections to be used)**



1. Requirements of your project

- An Arduino  
- A LED  
- A Button  
- A Laptop

Evaluation Table for Assignment 4

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Insufficient (O)** | **Sufficient (V)** | **Good (G)** |
| Task 1 : manual HTTP requests | No proof of successful completion | Task execution successful but no satisfactory explanation | Task executed correctly and completely |
| Task 2 :  CURL | No proof of successful completion | N.A. | Task executed correctly and completely |
| Apache Web Server | No proof of successful completion | N.A. | Task executed correctly and completely |
| Project Description/  Requirements/  Architecture | Not provided | Project Documentation is provided | Project is professionally described |
| PO | The assignment is not documented. | The assignment is documented. | The assignment documentation is neat and professionally done. |

1. HTTP: <http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol> [↑](#footnote-ref-1)
2. You have to install the “links” browser on your *Laptop* node by issuing this command:

   apt-get install links. But before you do this you should run “add\_route” script provided in the main directory of the Netkit lab. This should take care of being able to route externally. [↑](#footnote-ref-2)