Prelab 2 - HTTP and Network Performance

(100 points)

Having gotten our feet wet with the Wireshark packet sniffer in lab1, we're now ready to use Wireshark packet sniffer to investigate protocols in operation. In this lab, we'll explore several aspects of the HTTP protocol: the basic GET/response interaction, HTTP message formats, retrieving large HTML files, retrieving HTML files with embedded objects, and HTTP authentication and security. Before beginning these labs, make sure to read Section 2.2 of the text.[1]

Screenshots: For all the questions that require a screenshot, make sure that a **date timestamp** is visible next to your results (you can have a portion of another terminal open with the date command).

References: Chapter 2, Computer Networking: A Top Down Approach 1.

[24 pts] The Basic HTTP request/response interaction

Let's begin our exploration of HTTP by downloading a very simple HTML file - one that is very short, and contains no embedded objects. Do the following: ● Start up your web browser in the Mininet VM.

- Start up Wireshark. In the display-filter enter **http**, so that only captured HTTP messages will be displayed later in the packet-listing windows (We're only interested in the HTTP protocol here, and don't want to see the clutter of all captured packets). Then, start a packet capture using the interface 'eth0'. ● Clear the cache in your browser[2] and enter the following:

  http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html Your browser should display a very simple, one-line HTML file.
- Stop Wireshark packet capture.

Exercises:

Remember that the HTTP headers are extra information in the HTTP message sent by either the client or the server. You can search the vast number of HTTP headers online. By looking at the information in the HTTP GET and response message(s) in Wireshark, answer the following questions:

a. [3 pts] Is your browser running HTTP version 1.0, 1.1, or 2? What version of HTTP is the server running? How do you know?

The browser is running HTTP version 1.1 and the server is running HTTP version 1.1. In wireshark, it shows this information under the Info column.

b. [3 pts] How does your browser indicate to a server the languages it can accept? Based on what you see in Wireshark, what languages does your browser prefer?

The browser indicates to the server what languages it can accept under the "Accept-Language" HTTP request header.
Based on what I see in wireshark, the browser prefers "en-US."

c. [3 pts] In this exchange, who is the client and what is its IP address?

The client is the Mininet VM and it's IP address is 10.0.2.15

d. [3 pts] In this exchange, who is the server and what is its IP address?

The server is http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html and its IP address is 128.119.245.12

e. [3 pts] What is the status code returned from the server to your browser?

The status code is 200.

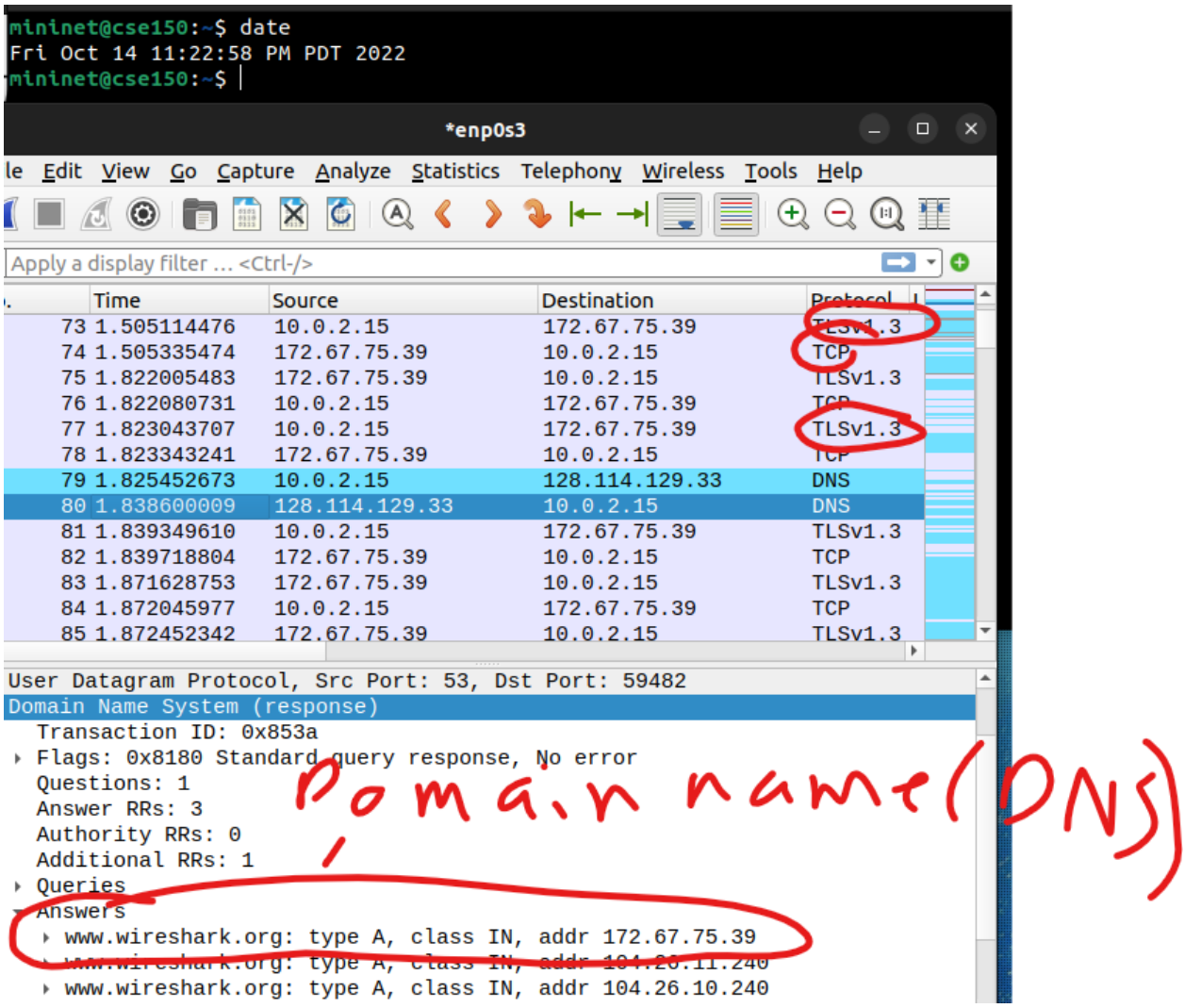
f. [3 pts] How many bytes of content are being returned to your browser?

540 bytes are being returned to the browser.

g. [6 pts] Now do a new Wireshark packet capture for "https://wireshark.org" **without http filter**- what do you observe is different compared to the previous URL? Can you read the responses from the server? (Hint: Do some investigation into the purpose of HTTPS.) Take note of the different protocols used and mention them in your observations discussion. Take a screenshot and mark it up to help explain your observations.

I am seeing mostly TCP requests and responses. TCP is used to organize data

and ensure a secure transmission between the server and the client. I am also seeing the TLS protocol, which adds a layer of security on top of the TCP protocol. TLS uses encryption to securely send private data, but it can increase latency between client and server communications. The DNS protocol is used to identify websites using readable host names instead of numerical IP addresses.



2. [40 pts] Experiment with wget

A. **[20 pts] Running the *wget* command**

In the VM terminal, run *wget* http://www2.cs.uh.edu/~gnawali/courses/cosc4377-s12/hw2/http.html to display the HTTP headers returned by the server **without downloading the actual file**. Also use Wireshark to capture the packets exchanged by wget. ***Please ignore the /favicon.ico request/response in Wireshark*.**

1. [3 pts]Take a screenshot showing your command and the output

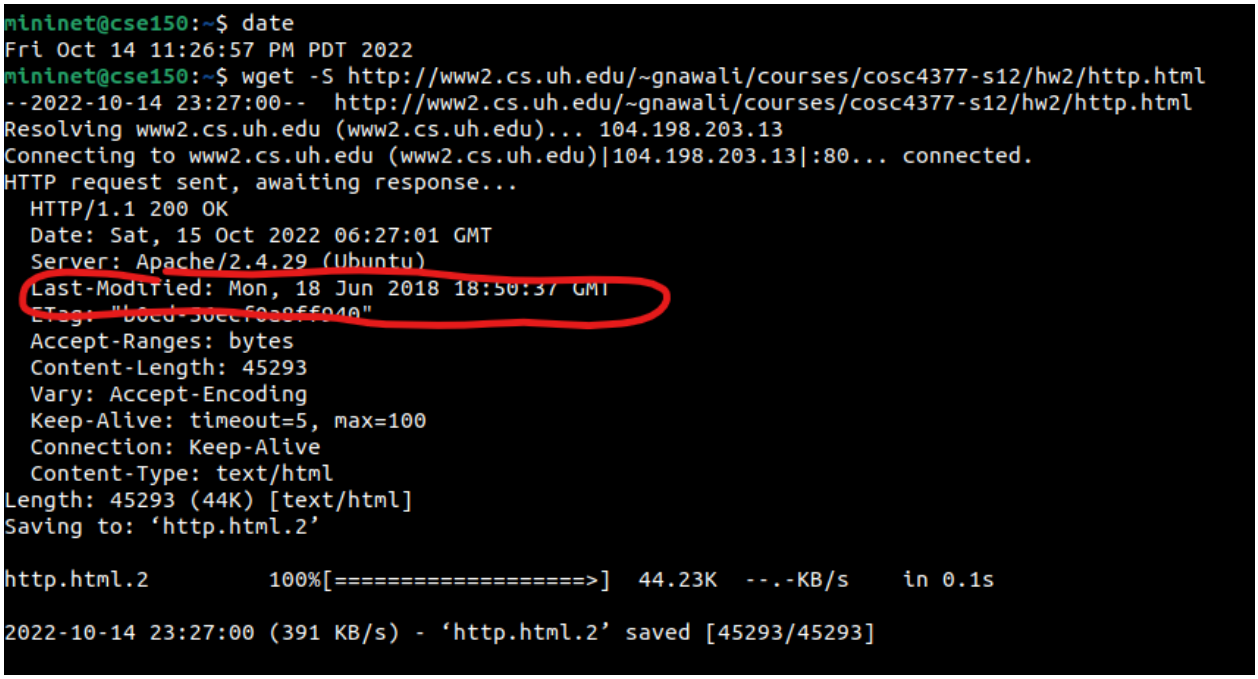
```
mininet@cse150:~$ date
Fri Oct 14 11:26:57 PM PDT 2022
mininet@cse150:~$ wget -S http://www2.cs.uh.edu/~gnawali/courses/cosc4377-s12/hw2/http.html
--2022-10-14 23:27:00--  http://www2.cs.uh.edu/~gnawali/courses/cosc4377-s12/hw2/http.html
Resolving www2.cs.uh.edu (www2.cs.uh.edu)... 104.198.203.13
Connecting to www2.cs.uh.edu (www2.cs.uh.edu)|104.198.203.13|:80... connected.
HTTP request sent, awaiting response...
  HTTP/1.1 200 OK
  Date: Sat, 15 Oct 2022 06:27:01 GMT
  Server: Apache/2.4.29 (Ubuntu)
  Last-Modified: Mon, 18 Jun 2018 18:50:37 GMT
  ETag: "b0ed-56eef0a8ff940"
  Accept-Ranges: bytes
  Content-Length: 45293
  Vary: Accept-Encoding
  Keep-Alive: timeout=5, max=100
  Connection: Keep-Alive
  Content-Type: text/html
Length: 45293 (44K) [text/html]
Saving to: ‘http.html.2’

http.html.2         100%[===================>]  44.23K  --.-KB/s    in 0.1s

2022-10-14 23:27:00 (391 KB/s) - ‘http.html.2’ saved [45293/45293]
```

Answer the following questions based on the output of wget (mark up your screenshot of the output based on questions below):

2. [2 pts] When was the web page last modified ? What does the last modified field indicate?



The web page was last modified on Monday, June 18th 2018.
The last modified field indicates when the web-page was last updated.

3. [2 pts] What type of HTTP *connection* is used ? Describe what this connection does.

A persistent HTTP connection is being used (aka “Keep-alive” connection). This connection allows multiple objects to be sent over a single TCP connection between the client and server.

4. [2 pts] On what port number is the server listening for this request? How do you determine this?

It is using port 80. The port is located on the “connecting” line and is next to the server IP. (Screenshot below)

```
mininet@cse150:~$ date
Fri Oct 14 11:26:57 PM PDT 2022
mininet@cse150:~$ wget -S http://www2.cs.uh.edu/~gnawali/courses/cosc4377-s12/hw2/http.html
--2022-10-14 23:27:00--  http://www2.cs.uh.edu/~gnawali/courses/cosc4377-s12/hw2/http.html
Resolving www2.cs.uh.edu (www2.cs.uh.edu)... 104.198.203.13
Connecting to www2.cs.uh.edu (www2.cs.uh.edu)|104.198.203.13|:80... connected.
HTTP request sent, awaiting response...
  HTTP/1.1 200 OK
  Date: Sat, 15 Oct 2022 06:27:01 GMT
  Server: Apache/2.4.29 (Ubuntu)
  Last-Modified: Mon, 18 Jun 2018 18:50:37 GMT
  ETag: "b0ed-56eef0a8ff940"
  Accept-Ranges: bytes
  Content-Length: 45293
  Vary: Accept-Encoding
  Keep-Alive: timeout=5, max=100
  Connection: Keep-Alive
  Content-Type: text/html
Length: 45293 (44K) [text/html]
Saving to: 'http.html.2'

http.html.2         100%[===================>]  44.23K  --.-KB/s    in 0.1s

2022-10-14 23:27:00 (391 KB/s) - 'http.html.2' saved [45293/45293]
```

5. [3 pts] What is the size (in bytes) of the base HTML page?

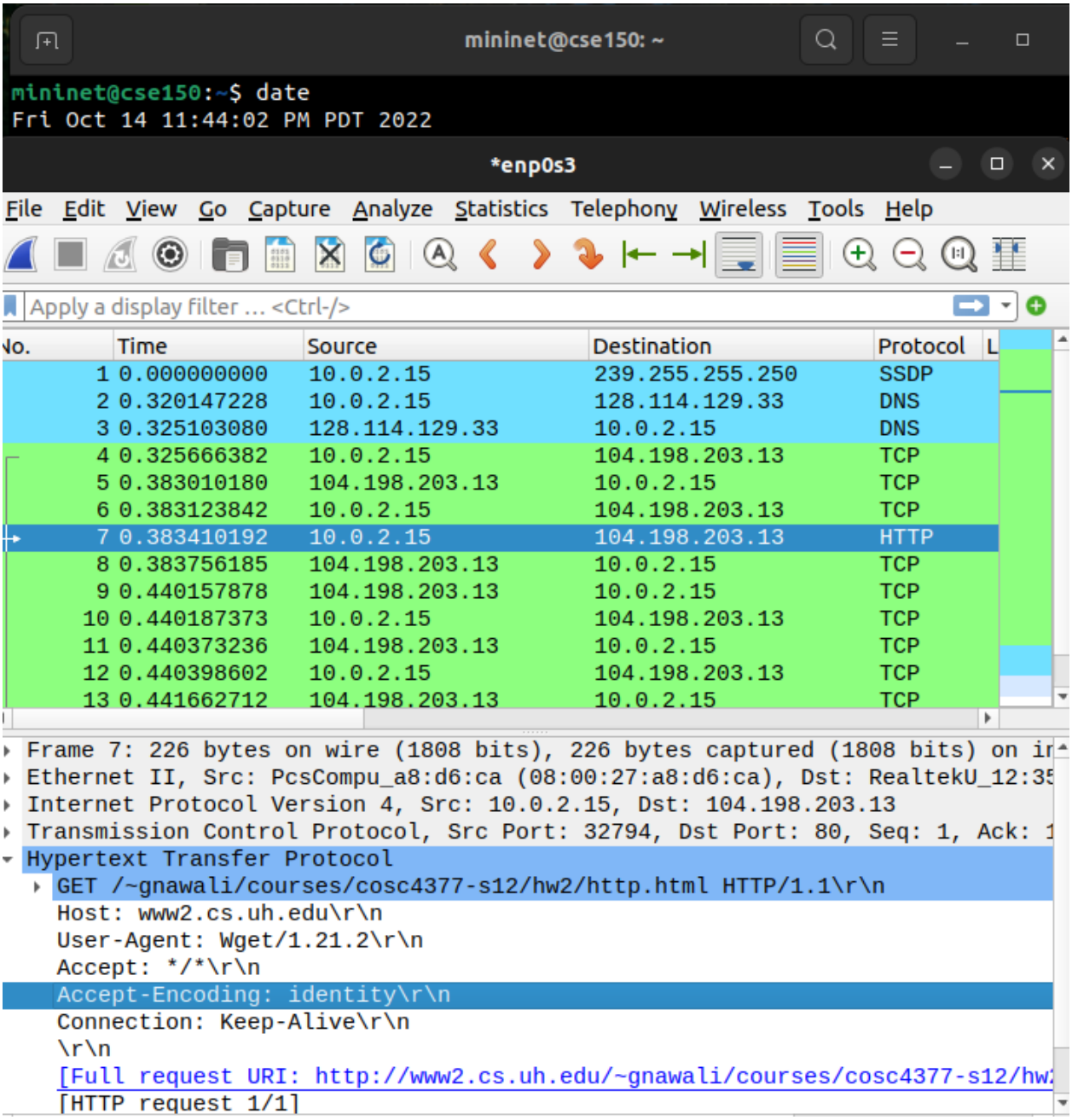
The size of the base HTML page is 45,293 bytes.


```
mininet@cse150:~$ date
Fri Oct 14 11:26:57 PM PDT 2022
mininet@cse150:~$ wget -S http://www2.cs.uh.edu/~gnawali/courses/cosc4377-s12/hw2/http.html
--2022-10-14 23:27:00--  http://www2.cs.uh.edu/~gnawali/courses/cosc4377-s12/hw2/http.html
Resolving www2.cs.uh.edu (www2.cs.uh.edu)... 104.198.203.13
Connecting to www2.cs.uh.edu (www2.cs.uh.edu)|104.198.203.13|:80... connected.
HTTP request sent, awaiting response...
  HTTP/1.1 200 OK
  Date: Sat, 15 Oct 2022 06:27:01 GMT
  Server: Apache/2.4.29 (Ubuntu)
  Last-Modified: Mon, 18 Jun 2018 18:50:37 GMT
  ETag: "b0ed-56eef0a8ff940"
  Accept-Ranges: bytes
  Content-Length: 45293
  Vary: Accept-Encoding
  Keep-Alive: timeout=5, max=100
  Connection: Keep-Alive
  Content-Type: text/html
Length: 45293 (44K) [text/html]
Saving to: 'http.html.2'

http.html.2         100%[===================>]  44.23K  --.-KB/s    in 0.1s

2022-10-14 23:27:00 (391 KB/s) - 'http.html.2' saved [45293/45293]
```

Now look at your Wireshark capture and take a screenshot:

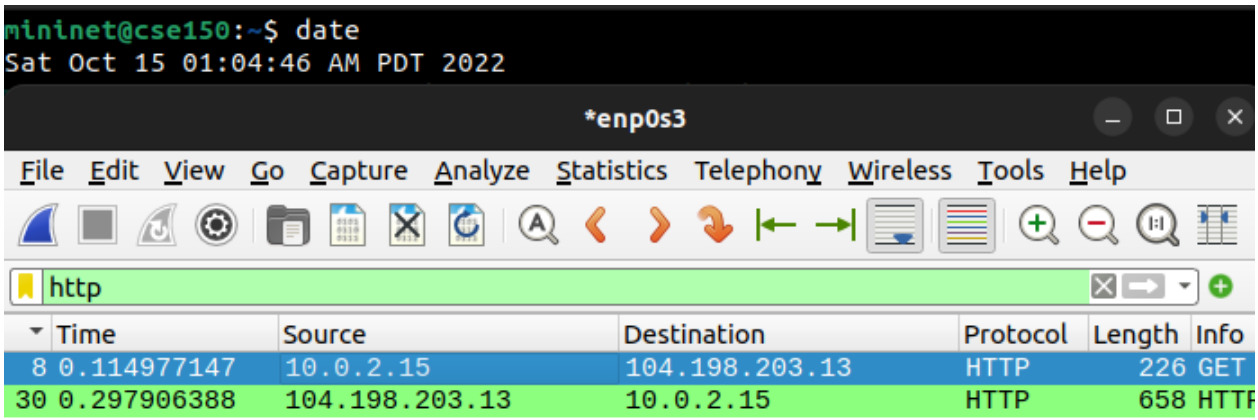


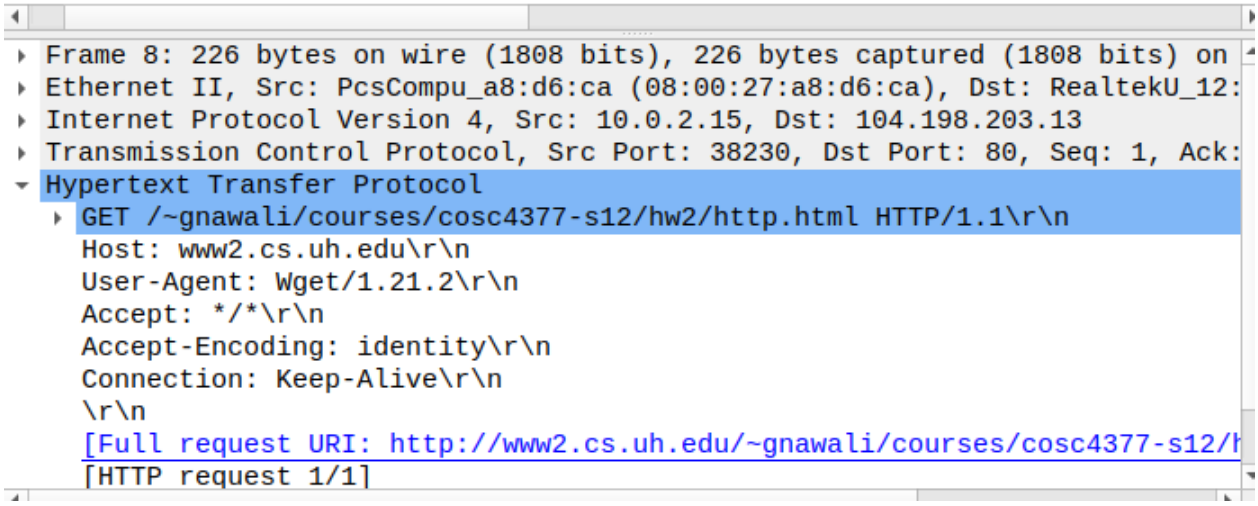
6. [2 pts] What HTTP method is used in the request message? Describe the purpose of this method.

The GET method is being used in this request message. It is used for posting/sending data to the server using a given URL.

7. [3 pts] How many HTTP request/response messages do you see in total? Take a screenshot showing the HTTP request/response messages in Wireshark.

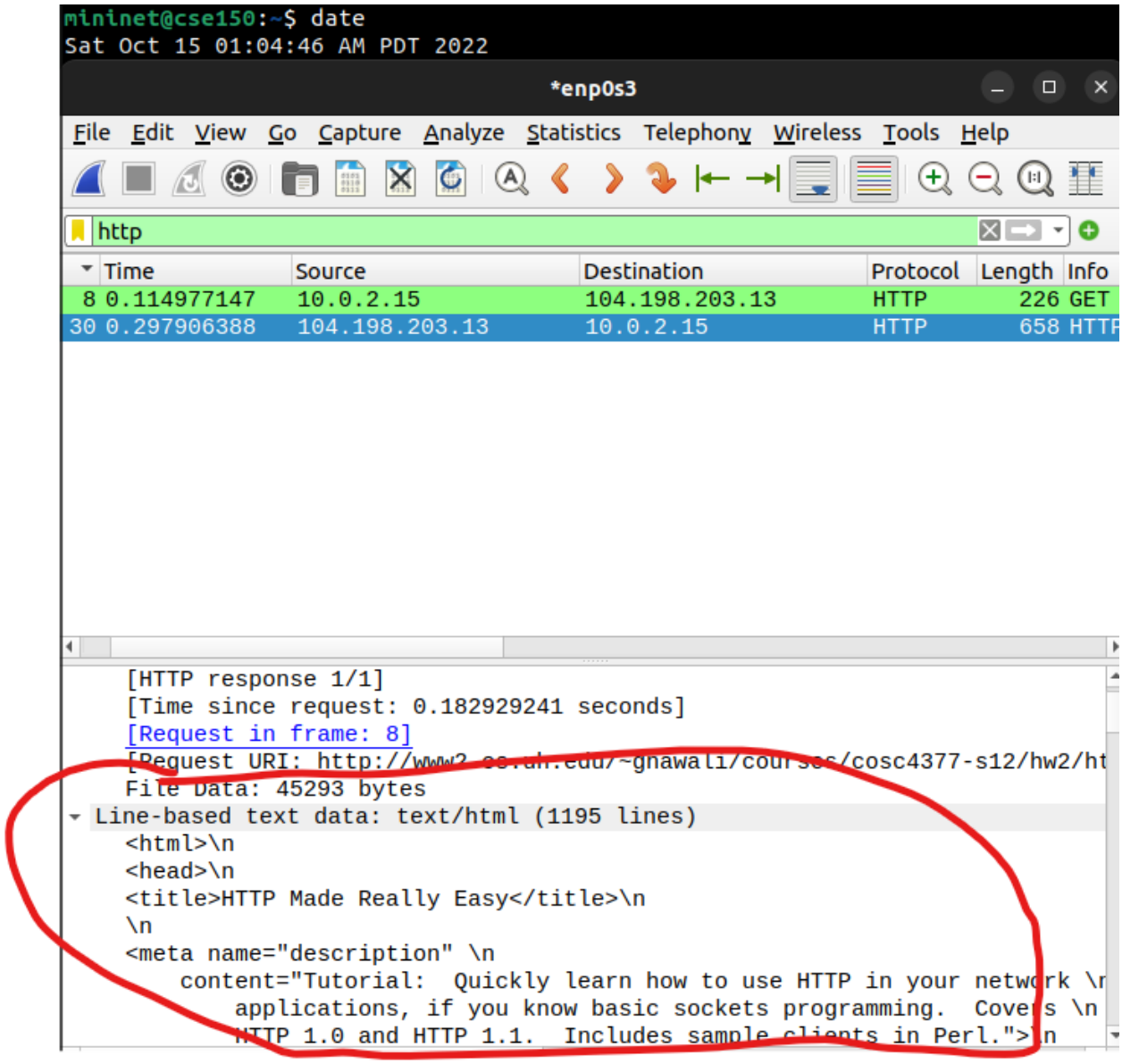
I see 2 HTTP request/response messages in Wireshark.





8. [3 pts] Is the base HTML file downloaded? How do you know based on Wireshark? Attach a screenshot showing the same.

Yes, the base HTML file is downloaded, as it shows the total number bytes of the file, and Wireshark proceeds to display the HTML code.



B. [20 pts] Embedded Objects

Let's look at what happens when your browser downloads a file with several embedded objects, i.e., a file that includes other objects that are stored on another server(s).

Do the following:

- Enter http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file4.html in your browser and capture the packets in Wireshark.
- Your browser should display a short HTML file with two images. These two images are referenced in the base HTML file. That is, the images themselves are not contained in the HTML; instead the URLs for the images are contained in the downloaded HTML file. As discussed in the textbook, your browser will have to retrieve these objects from the indicated web sites. Stop Wireshark packet capture, and enter "**http**" in the display-filter-specification window, so that only captured HTTP messages will be displayed.
- ***Please ignore the /favicon.ico request/response in the wireshark***.

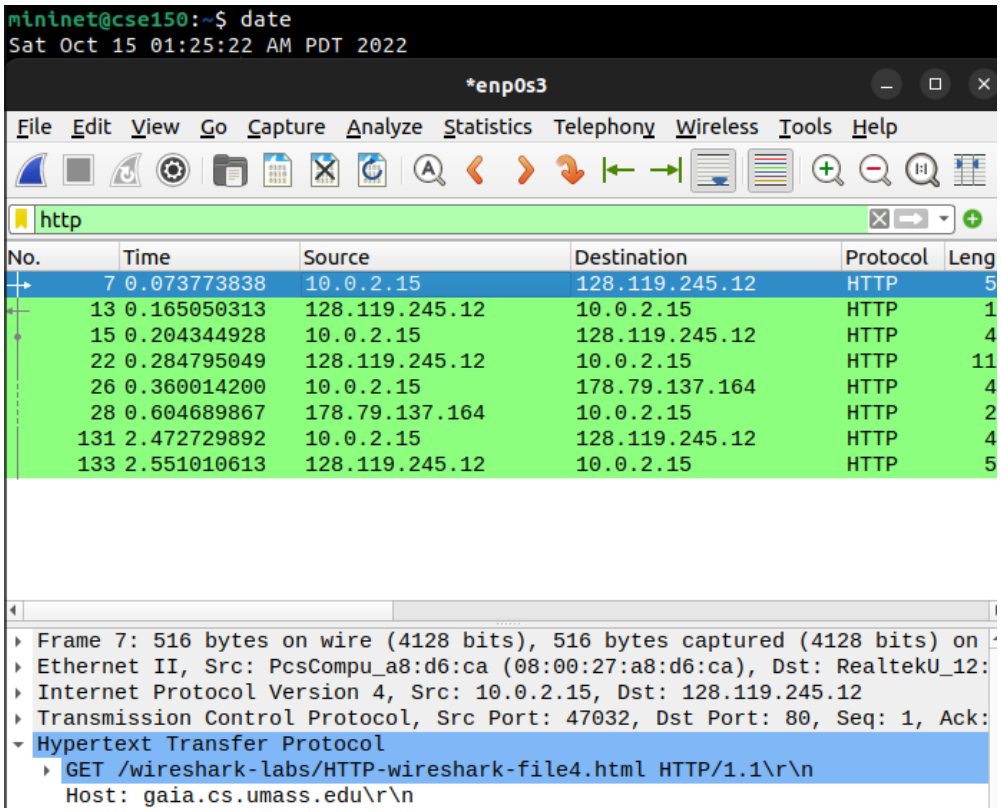
Answer the following questions:

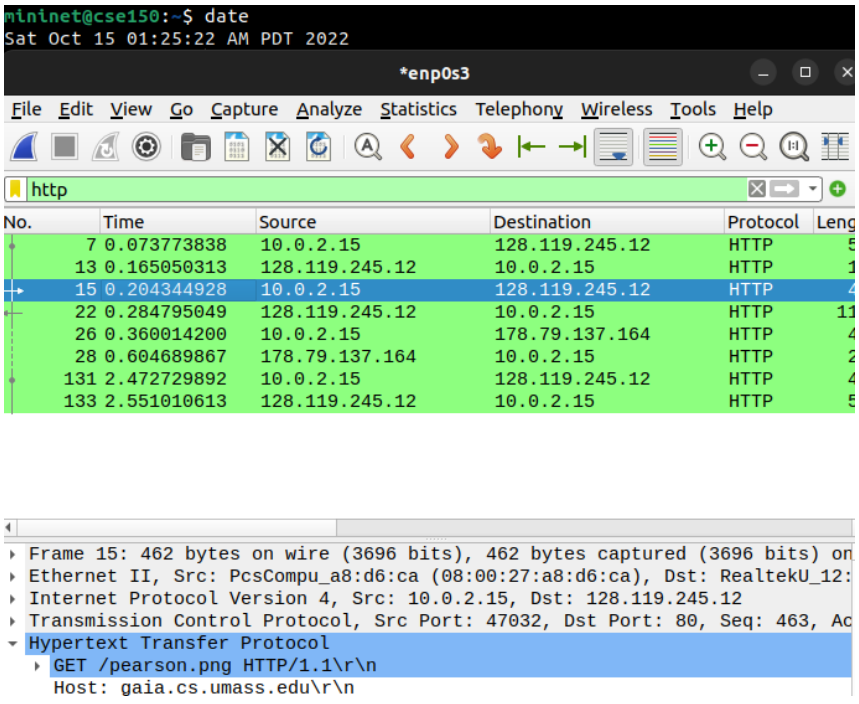
1. [3 pts] How many HTTP GET request messages did your browser send in total? To which Internet addresses were these GET requests sent? Take a screenshot showing the HTTP request/response messages in Wireshark.

SCREENSHOTS BELOW

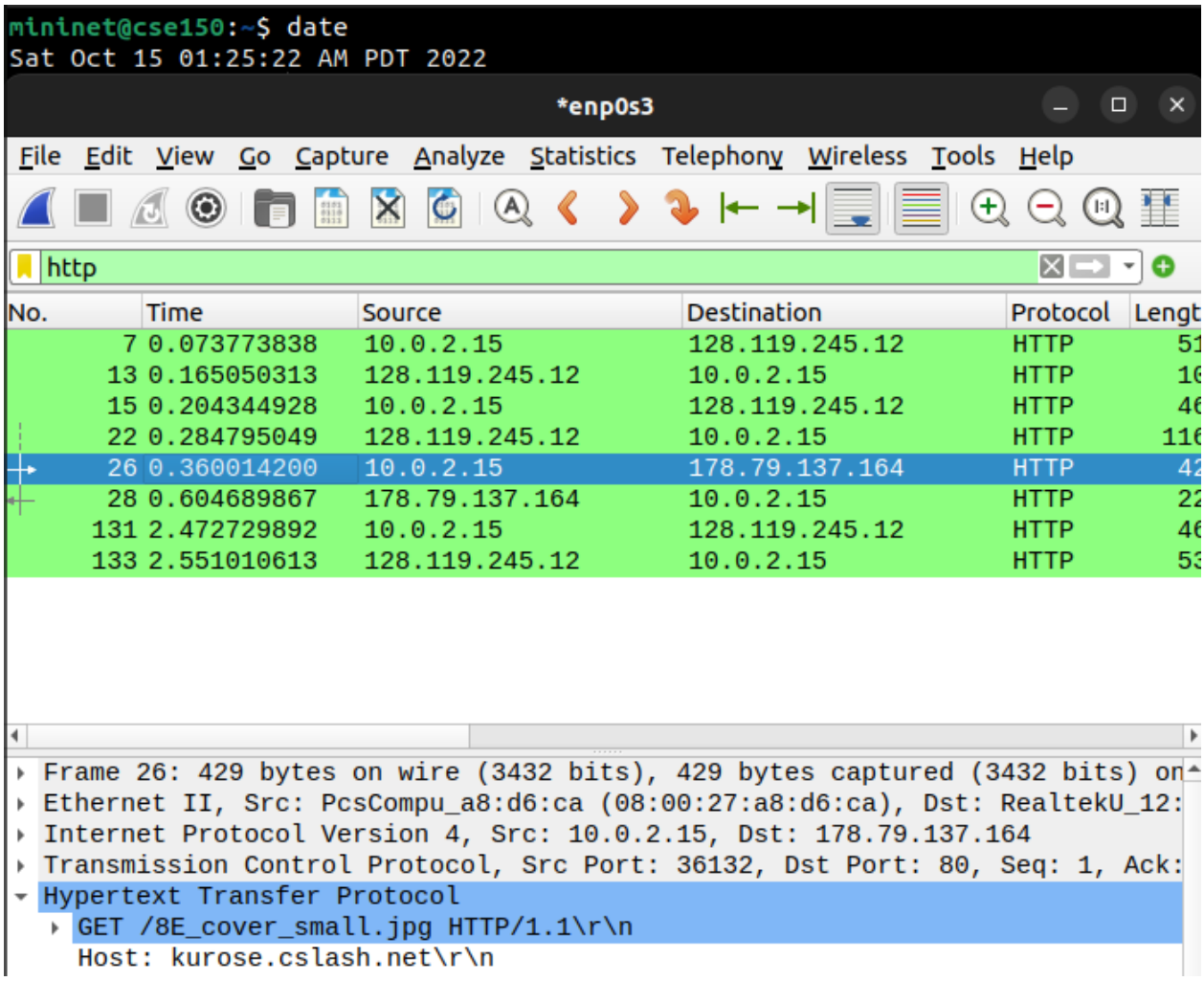
There are a total of 3 HTTP GET request messages

The first screenshot shows the GET address is sent to 128.119.245.12





The second screenshot shows the GET request is sent to 128.119.245.12



The third screenshot shows the GET request being sent to 178.79.137.164

2. [3 pts] Describe the purpose of each of the Request/Response messages

The purpose of the 1st request/response message is to download the content of the base HTML file.

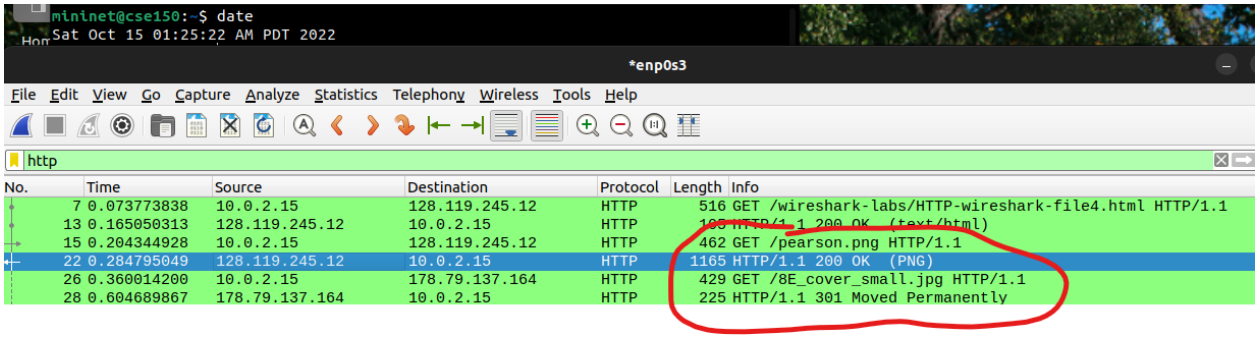
The purpose of the 2nd request/response message is to download the content for the Pearson logo image.

The purpose of the 3rd request/response message is to download the content for the textbook cover image.

3.[5 pts] Can you tell whether your browser downloaded the embedded images serially, or whether they were downloaded from the two websites in parallel? Explain and support with screenshots from your Wireshark trace.

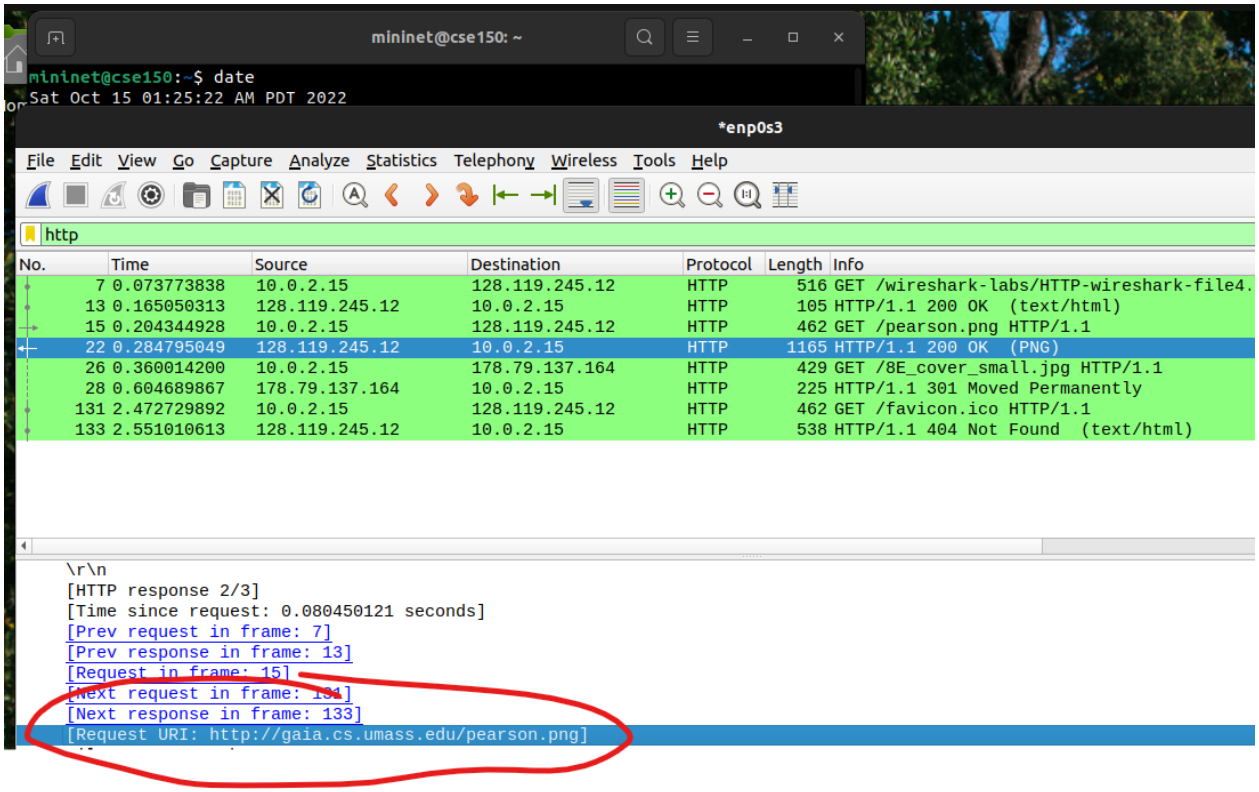
The browser downloaded the embedded images serially as the first image

is downloaded after the second GET request, and the second image is downloaded after the 3rd GET request.

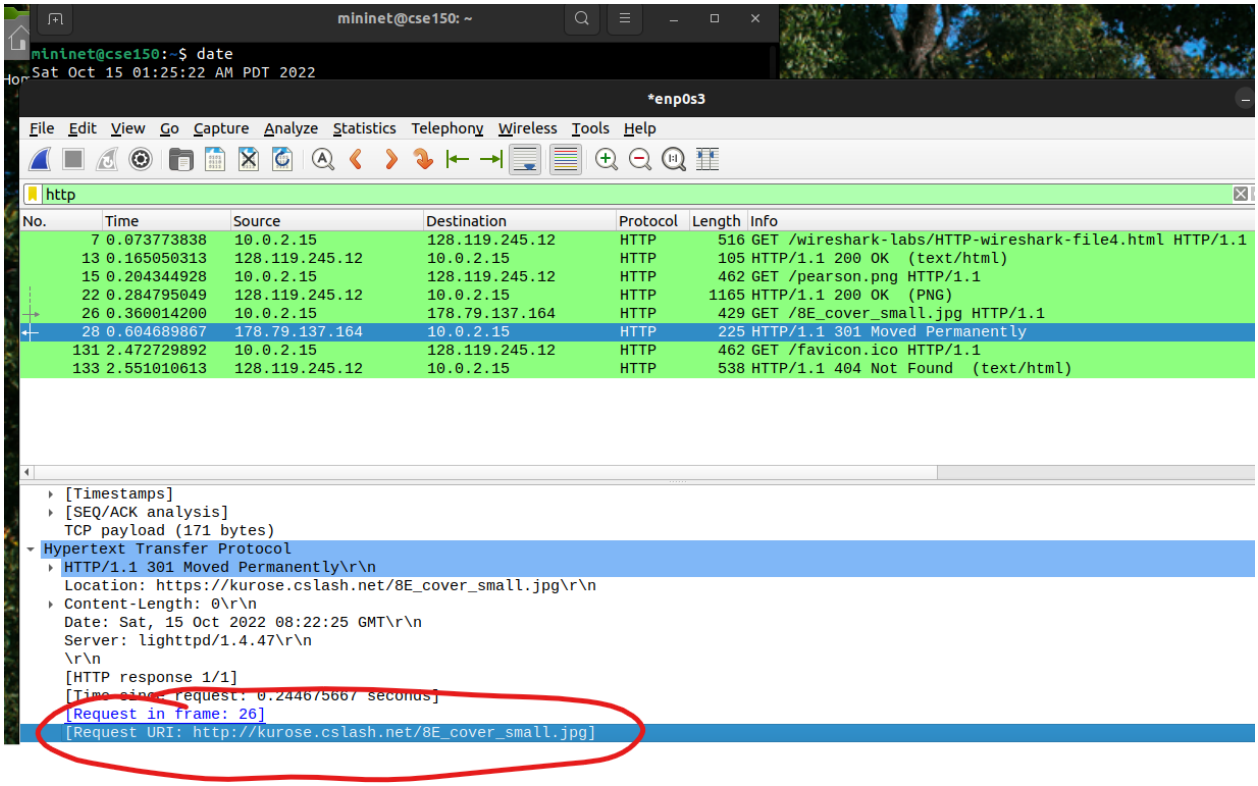


4. [3 pts] What are the URLs for the images? Support with screenshots from your Wireshark trace.

The URL for the first image is http://gaia.cs.umass.edu/pearson.png

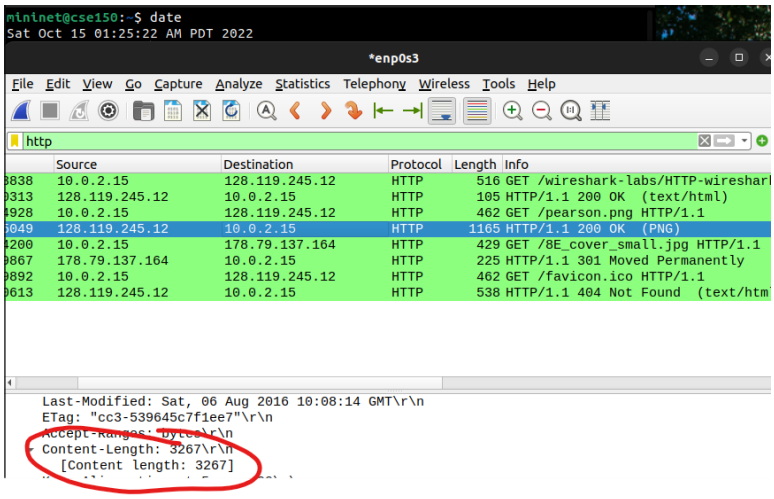


The URL for the second image is http://kurose.cslash.net/8E_cover_small.jpg

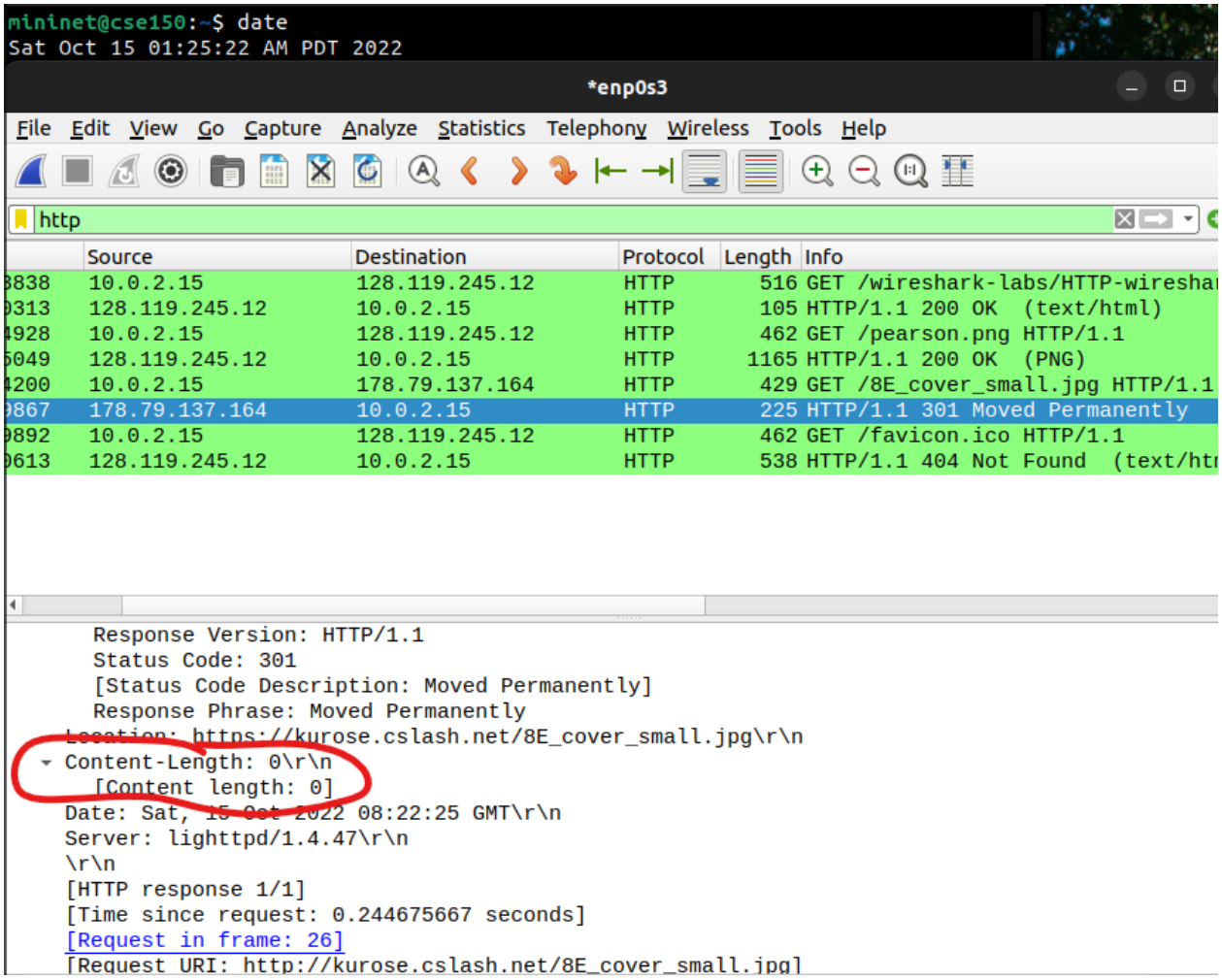


5. [3 pts] Draw a picture with a client and server, and use arrows to indicate the Requests and Responses to fully load the web page.
6. [3 pts] What are the sizes of the embedded images? Attach the screenshot with markup on the size of the image.

First image size is 3,267 bytes



Wireshark says the second image size is 0 bytes, but that is because the image is at a different location.



3. [16 pts] HTTP Conditional GET

Recall from Section 2.2.5 of the text, that most web browsers perform object caching and thus often perform a conditional GET when retrieving an HTTP object. Before performing the steps below, make sure your browser's cache is empty[2]. Now do the following:

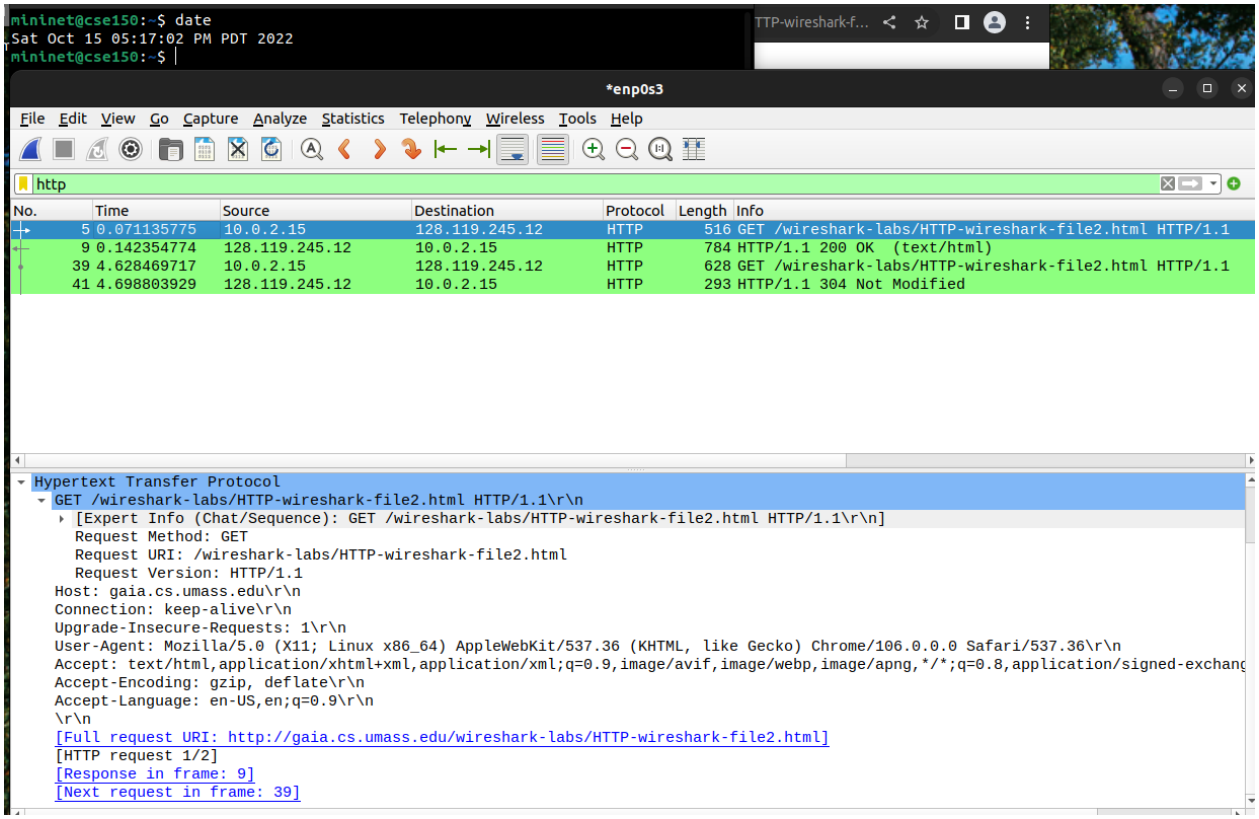
- Start up your web browser, and make sure your browser's cache is cleared.
- Start up the Wireshark packet sniffer
- Enter the following URL into your browser
  http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file2.htm
  l Your browser should display a very simple five-line HTML file.
- Quickly enter the same URL into your browser again (or simply select the refresh button on your browser)
- Stop Wireshark packet capture, and enter "**http**" (again, in lower case without the quotation marks) in the display-filter-specification window, so that only captured HTTP messages will be displayed later in the packet-listing window.

- ***Please ignore the /favicon.ico request/response in Wireshark***.

Answer the following questions:

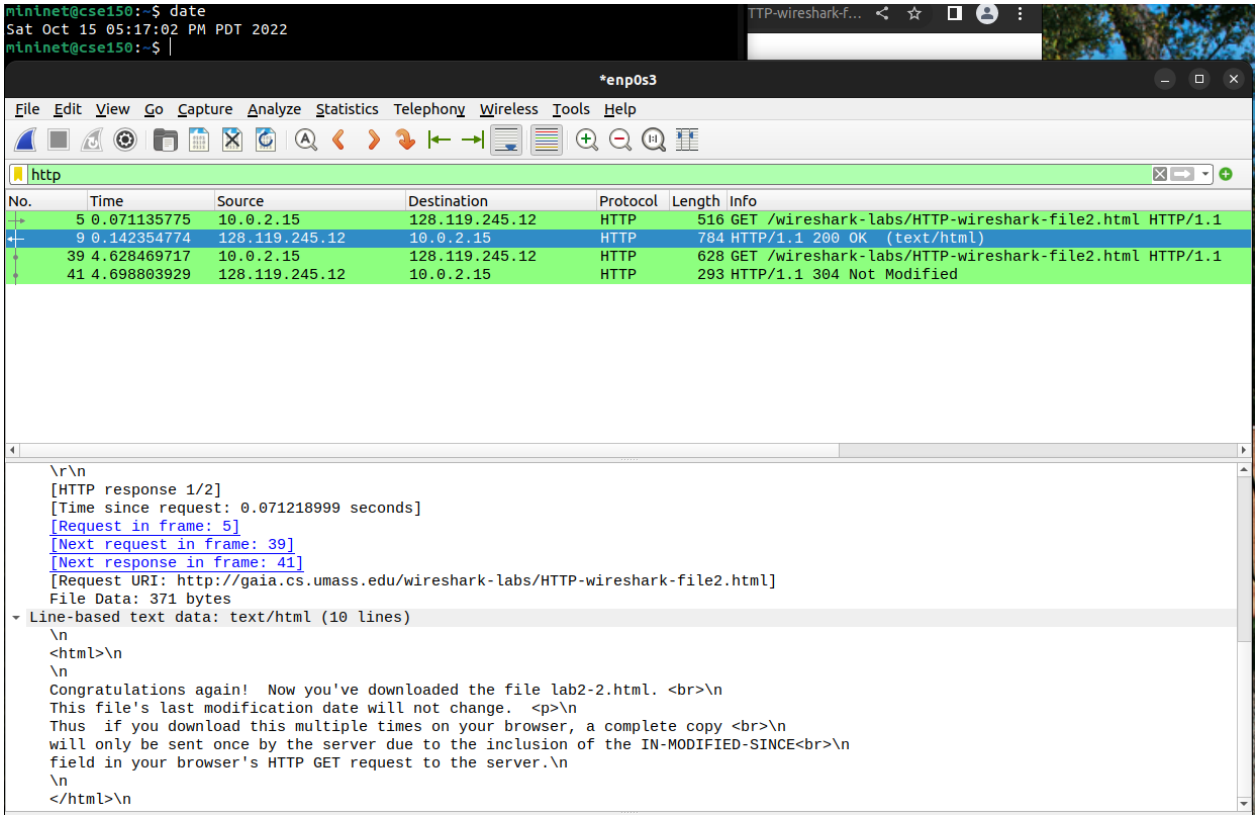
a. [4 pts] Inspect the contents of the first HTTP GET[3] request from your browser to the server. Do you see an “IF-MODIFIED-SINCE” line in the HTTP GET? Show a screenshot. What is the purpose of this header?

There is no “IF-MODIFIED-SINCE” line in the HTTP GET request. The purpose of the header is to show the last time the browser received the requested data from the server.



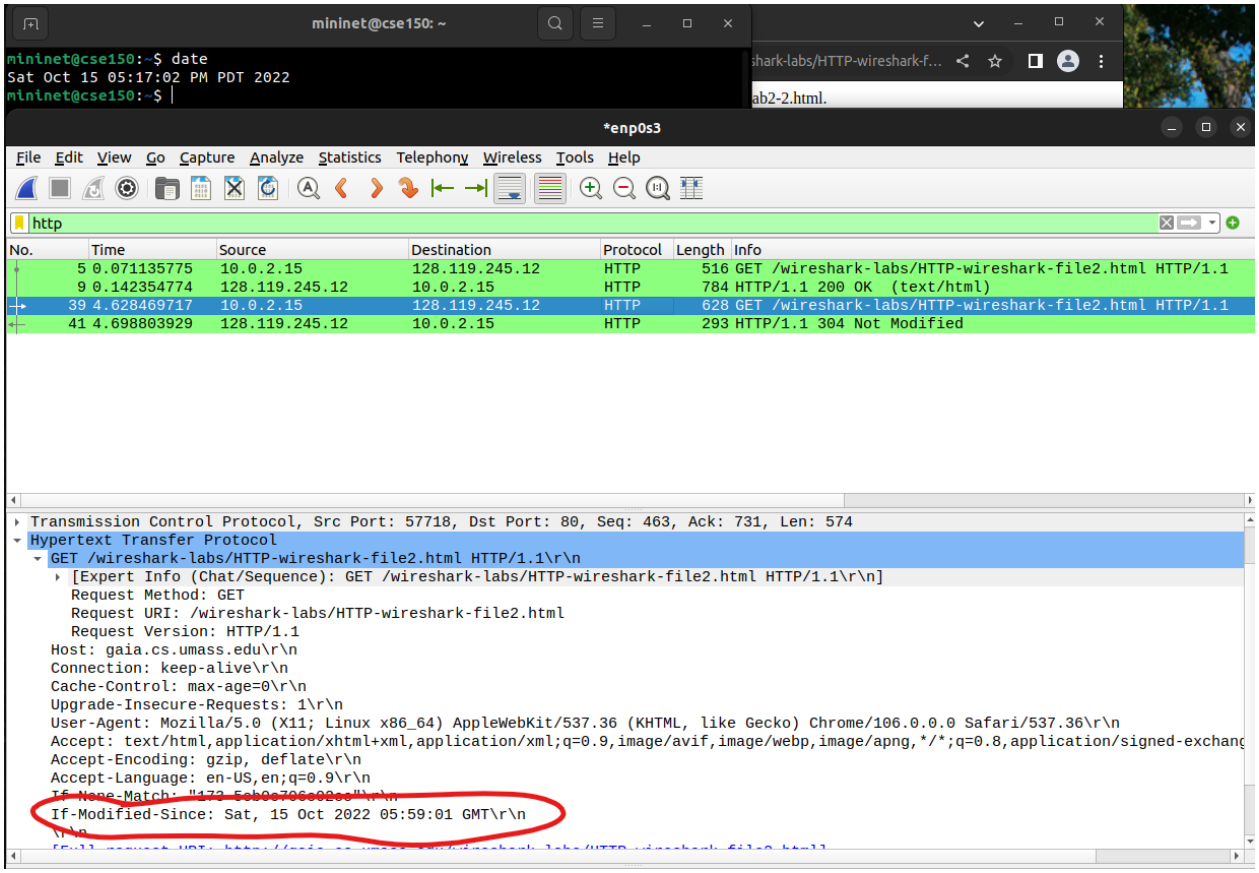
b. [4 pts] Inspect the contents of the server response. Did the server explicitly return the contents of the file? How can you tell? Support with screenshots from your Wireshark trace.

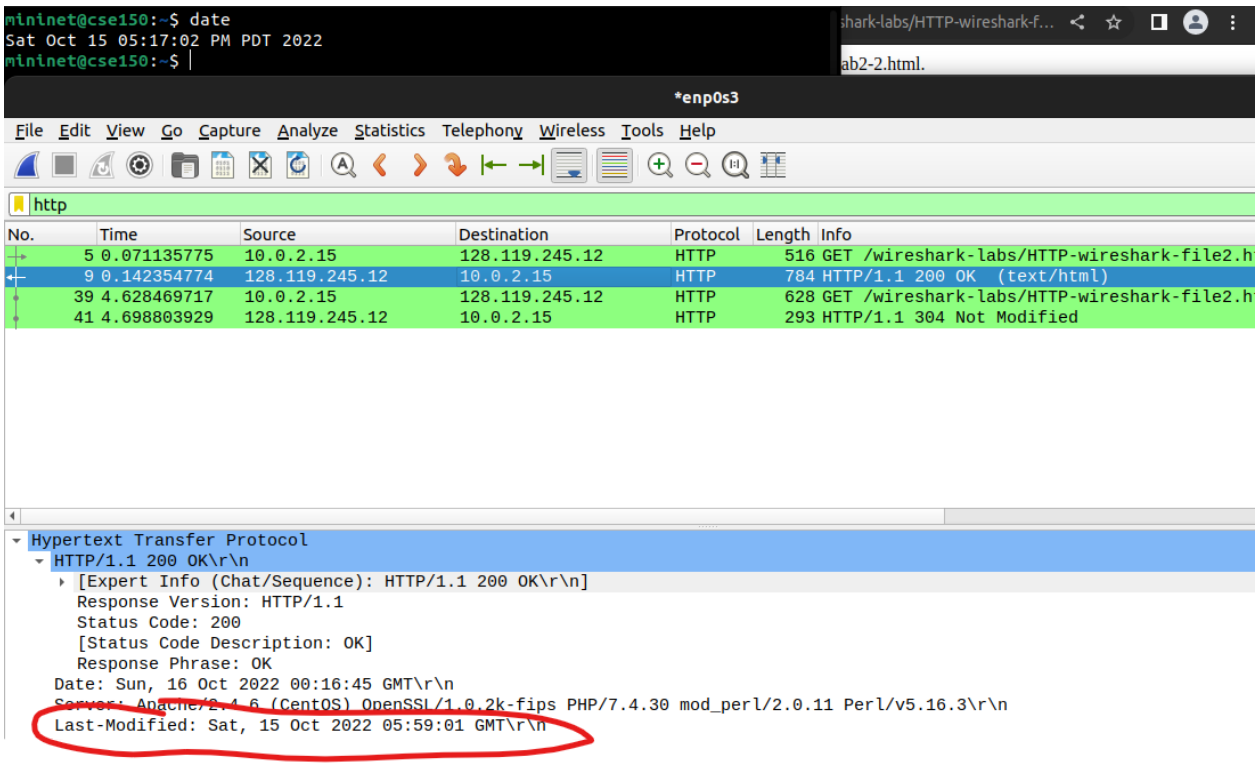
Yes the server did explicitly return the contents of the file as it shows the content of the message in the response. (SCREENSHOT BELOW)

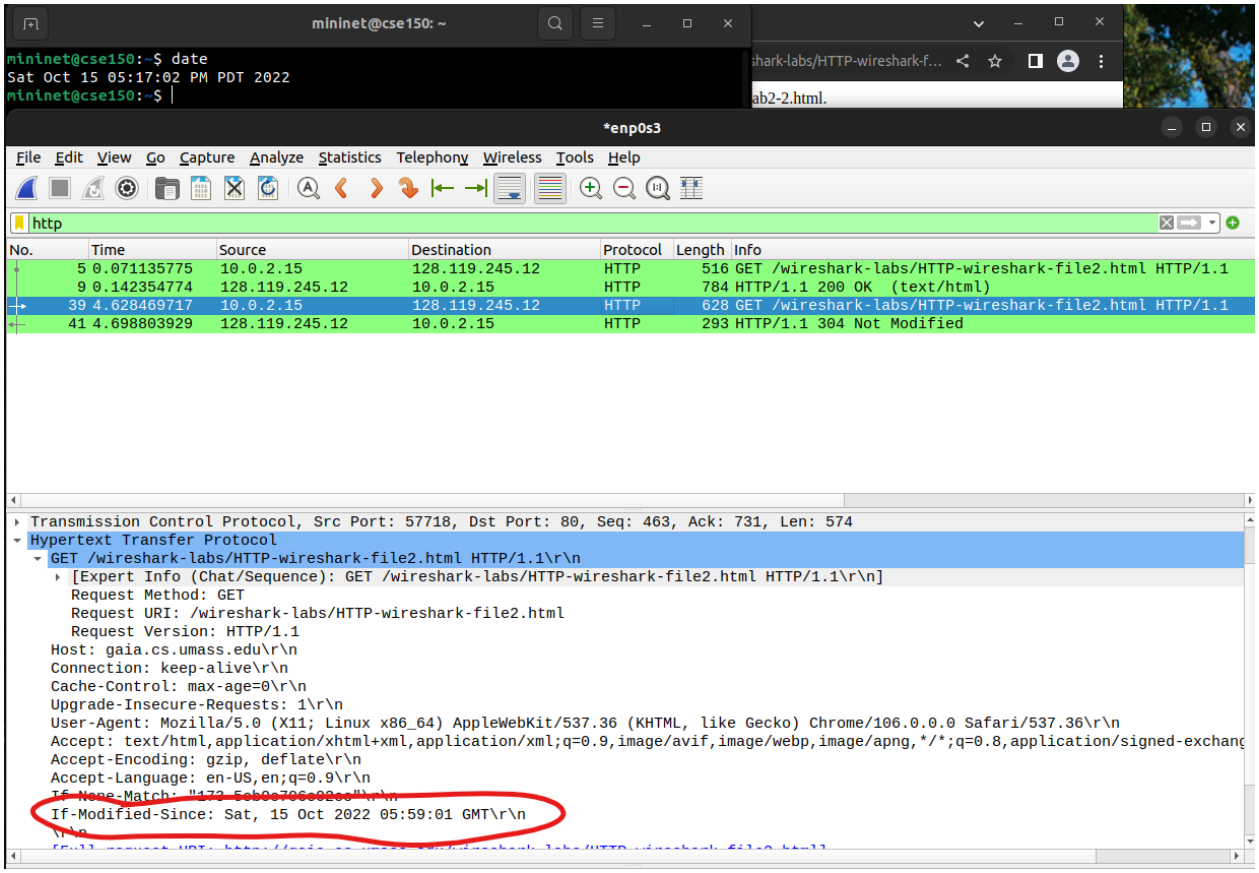


c. [4 pts] Now inspect the contents of the second HTTP GET request from your browser to the server. Do you see an “IF-MODIFIED-SINCE:” line in the HTTP GET? If so, what information follows the “IF-MODIFIED-SINCE:” header? Support with screenshots from your Wireshark trace.

There is an “IF-MODIFIED-SINCE” line in the second HTTP GET request. It shows the date: Saturday, October 15, 2022, which is the date in the first server response. (SCREENSHOTS BELOW)


mininet@cse150: ~
mininet@cse150:~$ date
Sat Oct 15 05:17:02 PM PDT 2022
mininet@cse150:~$
ab2-2.html.
*enp0s3
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http
No. Time Source Destination Protocol Length Info
5 0.071135775 10.0.2.15 128.119.245.12 HTTP 516 GET /wireshark-labs/HTTP-wireshark-file2.html HTTP/1.1
9 0.142354774 128.119.245.12 10.0.2.15 HTTP 784 HTTP/1.1 200 OK (text/html)
39 4.628469717 10.0.2.15 128.119.245.12 HTTP 628 GET /wireshark-labs/HTTP-wireshark-file2.html HTTP/1.1
41 4.698803929 128.119.245.12 10.0.2.15 HTTP 293 HTTP/1.1 304 Not Modified
Transmission Control Protocol, Src Port: 57718, Dst Port: 80, Seq: 463, Ack: 731, Len: 574
Hypertext Transfer Protocol
GET /wireshark-labs/HTTP-wireshark-file2.html HTTP/1.1\r\n
[Expert Info (Chat/Sequence): GET /wireshark-labs/HTTP-wireshark-file2.html HTTP/1.1\r\n]
Request Method: GET
Request URI: /wireshark-labs/HTTP-wireshark-file2.html
Request Version: HTTP/1.1
Host: gaia.cs.umass.edu\r\n
Connection: keep-alive\r\n
Cache-Control: max-age=0\r\n
Upgrade-Insecure-Requests: 1\r\n
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.0.0 Safari/537.36\r\n
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchang
Accept-Encoding: gzip, deflate\r\n
Accept-Language: en-US,en;q=0.9\r\n
If-None-Match: "173-5eb0e706c02ee"\r\n
If-Modified-Since: Sat, 15 Oct 2022 05:59:01 GMT\r\n
\r\n

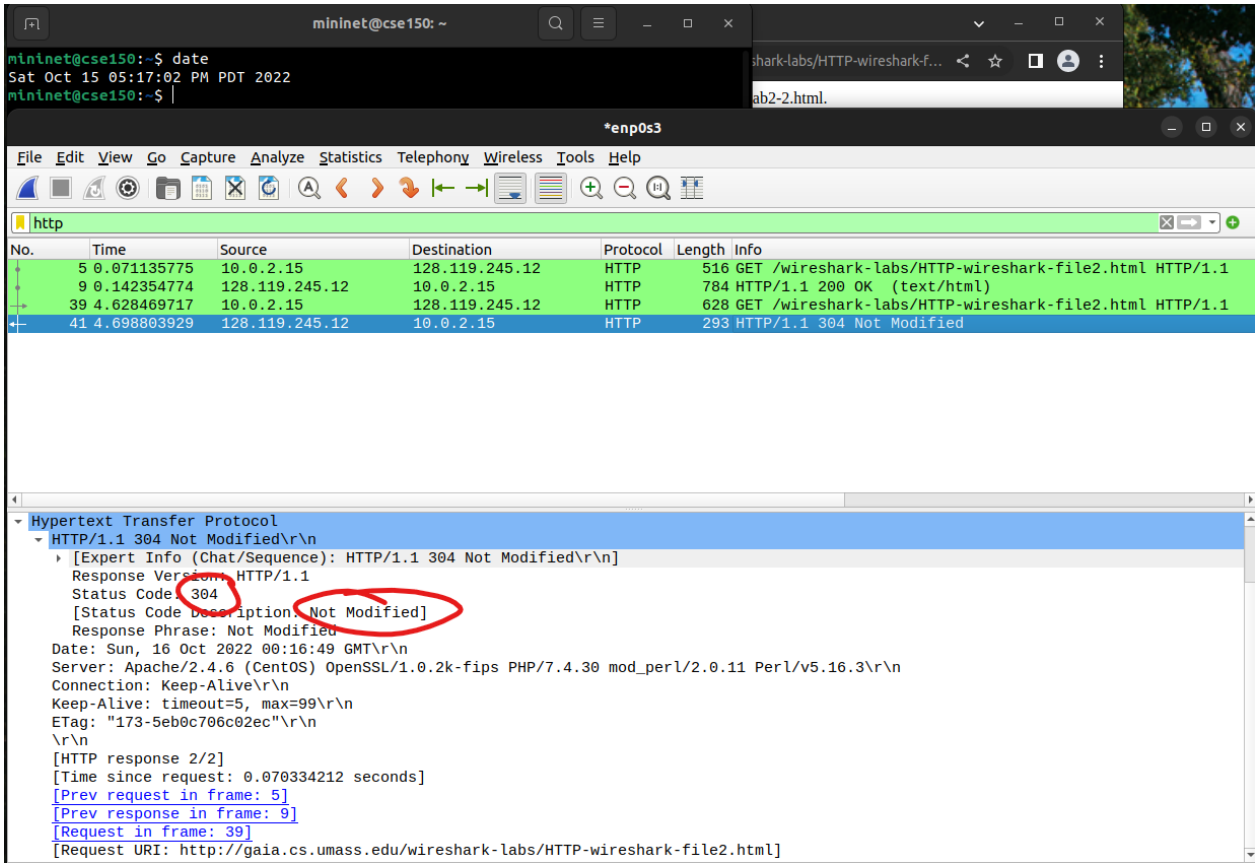

mininet@cse150:~$ date
Sat Oct 15 05:17:02 PM PDT 2022
mininet@cse150:~$
ab2-2.html.
*enp0s3
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http
No. Time Source Destination Protocol Length Info
5 0.071135775 10.0.2.15 128.119.245.12 HTTP 516 GET /wireshark-labs/HTTP-wireshark-file2.h
9 0.142354774 128.119.245.12 10.0.2.15 HTTP 784 HTTP/1.1 200 OK (text/html)
39 4.628469717 10.0.2.15 128.119.245.12 HTTP 628 GET /wireshark-labs/HTTP-wireshark-file2.h
41 4.698803929 128.119.245.12 10.0.2.15 HTTP 293 HTTP/1.1 304 Not Modified
Hypertext Transfer Protocol
HTTP/1.1 200 OK\r\n
[Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
Response Version: HTTP/1.1
Status Code: 200
[Status Code Description: OK]
Response Phrase: OK
Date: Sun, 16 Oct 2022 00:16:45 GMT\r\n
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/7.4.30 mod_perl/2.0.11 Perl/v5.16.3\r\n
Last-Modified: Sat, 15 Oct 2022 05:59:01 GMT\r\n



d. [4 pts] What is the HTTP status code and phrase returned from the server in response to this second HTTP GET? Did the server explicitly return the contents of the file? Explain and support with screenshots from your Wireshark trace.

The status code and phrase of the server response to the second HTTP GET is 304: Not Modified. THe server did not explicitly return the contents of the file because nothing was modified since the first HTTP GET response.

(SCREENSHOT BELOW)



4. [20 pts] Measuring Performance in Mininet

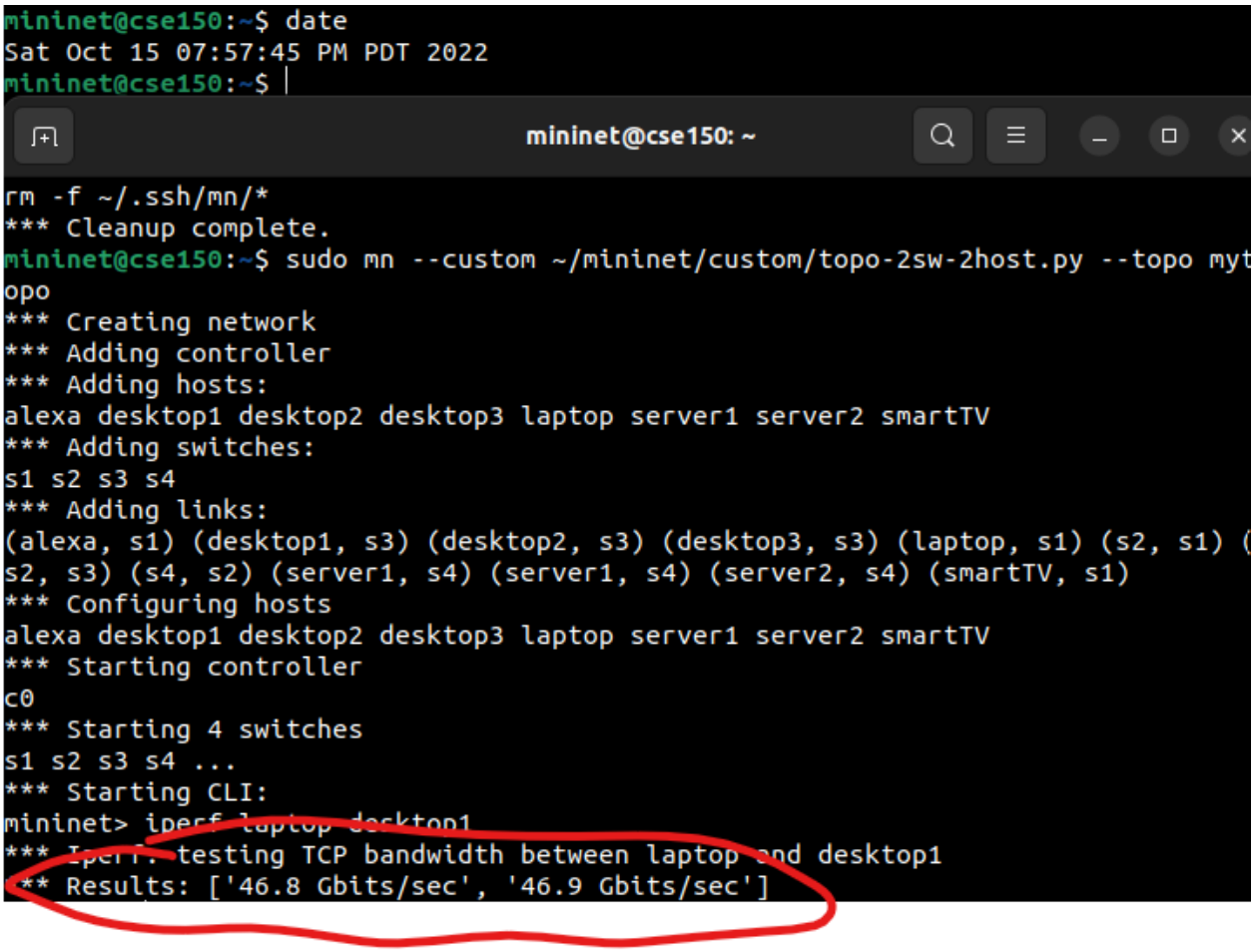
Using the iperf tool **on your topology from lab1**:

a. [2 pts] Research the *iperf* tool and describe it in your own words.

The *iperf* tool is used to measure throughput of a network and tune it. It can be used to test TCP and UDP.

b. [4 pts] Then run the *iperf* command between Laptop and Desktop1 in the Mininet command line and screenshot the output. Describe the results, referencing your screenshot, and determine how fast the connection is.

Laptop is streaming data to desktop1 at a speed of 46.8 Gigabits/sec and vice-versa is 46.9 Gigabits/sec.



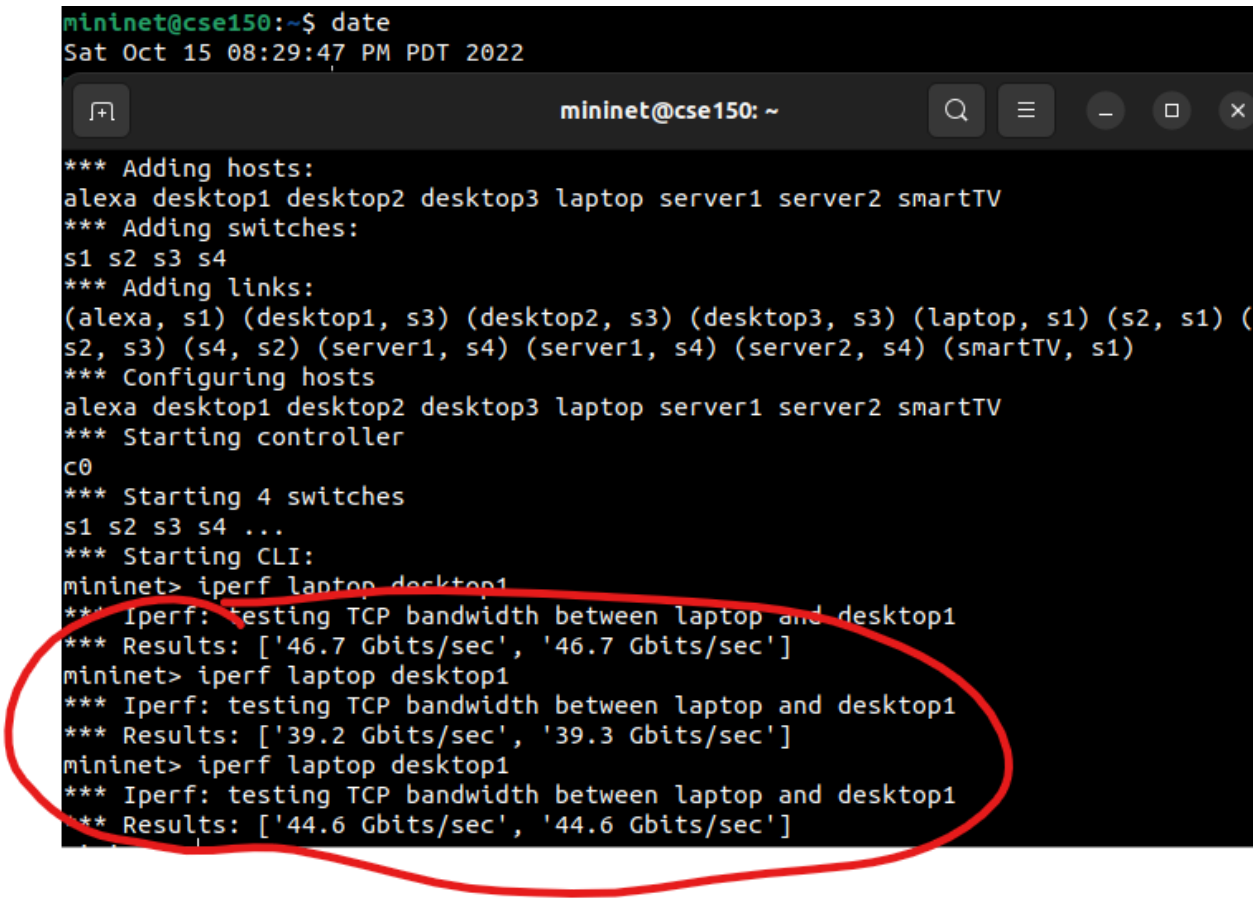
c. [3 pts] Given an example of an application or test case that would benefit from running the iperf tool.

An application that would benefit from running the iperf tool is testing the network speed between a central company server and the employee desktop stations of the company, to make sure the connection is healthy for data uploads and downloads.

Set the bandwidth between all the links between Laptop and Desktop1 (4 links total) to 50 Mbps by adding a `bw=` parameter to `addLink`, which takes its argument as a number representing the bandwidth in Mbps.

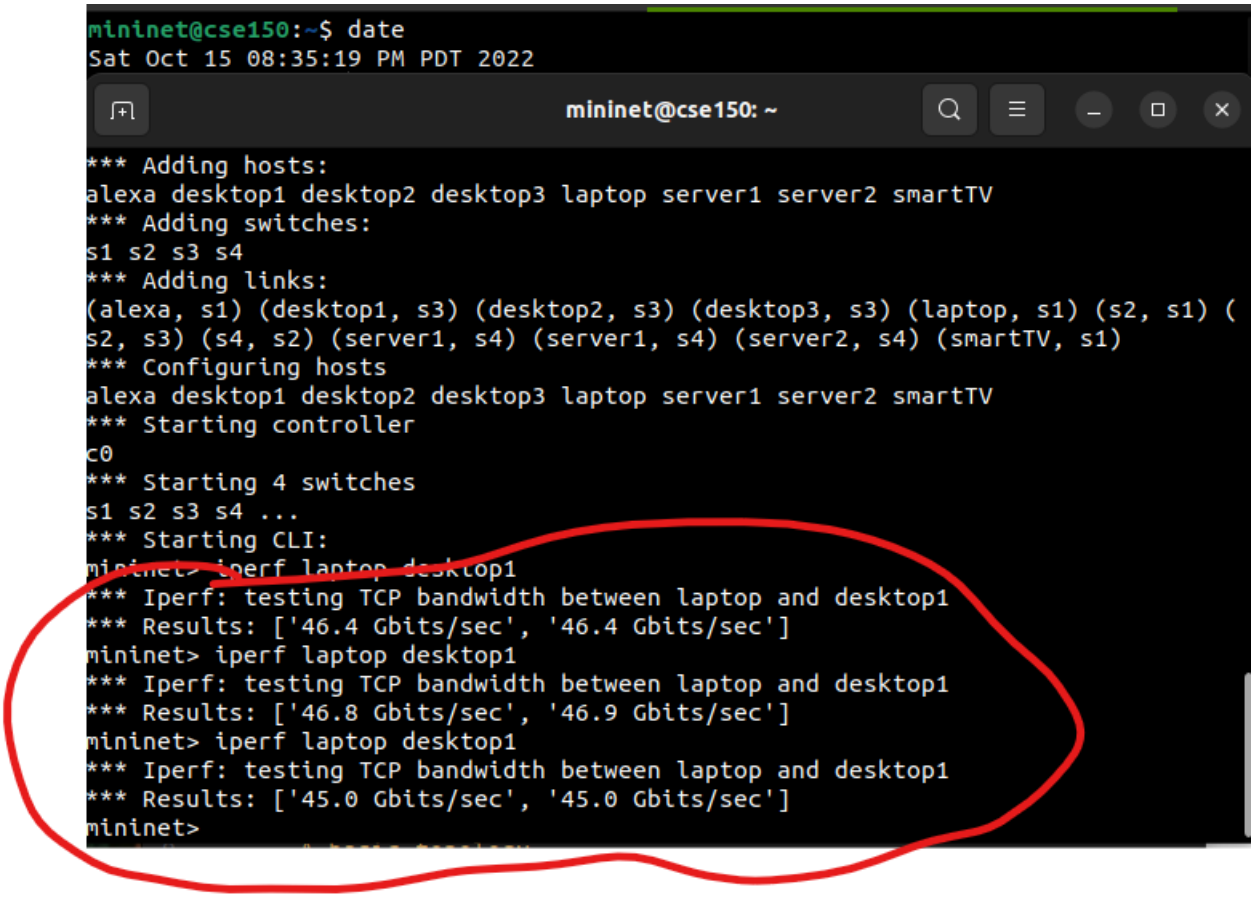
d. [4 pts] Measure the throughput between Laptop and Desktop1 with *iperf* three times and take the average of all the Mbits/sec figures printed. What is your result? Include a screenshot of the results.

43.51 Gbits/sec is the average number of all the results



e. [4 pts] Now remove all the `bw=` parameters you added except for the one between switch1 and switch2. Run iperf between the Laptop and Desktop1 another three times and take the average of all the Mbits/sec figures printed. How many Mbits/sec do you get now? Include a screenshot of the results.

46.09 Gbits/sec is my new average



f. [3 pts] Do your answers for d. and e. vary significantly? Discuss why or why not?

The answers do not vary significantly because the hosts are all based on the same machine.

Submission:

You will submit 1 file for this assignment directly on Canvas: a PDF with all of your solutions to the questions.
Naming convention: name the PDF file as [YourCruzID].pdf.

[1] References to figures and sections are for the 8th edition of our text, *Computer Networks, A Top-down Approach, 8h ed., J.F. Kurose and K.W. Ross, Addison-Wesley/Pearson, 2020.*

[2] See https://www.howtogeek.com/304218/how-to-clear-your-history-in-any-browser/ for instructions on clearing your browser cache.

[3] *Hint:* ideally, you should see an If-Modified-Since header since you've just downloaded this page a few seconds ago. However, depending on the browser you're using, and the format of the server's earlier response to your initial GET, your browser may not include an If-Modified-Since even if the document has been downloaded and cached. The Chrome browser is pretty good at regularly using If-Modified-Since. But Safari and Firefox are much more finicky about when to use If-Modified-Since. Life isn't always as easy in practice as it is in theory!