CPSC 471-01 Project 1

Team: Luc Dang (solo)

# Part 1 – Python Programming Web Server

**#import socket module**

**from socket import \***

**import sys # In order to terminate the program**

**serverSocket = socket(AF\_INET, SOCK\_STREAM)**

**#Prepare a sever socket**

**# TASK 1**

**#Fill in start**

**serverSocket.bind(('localhost', 45678))**

**serverSocket.listen()**

**#Fill in end**

**while True:**

**#Establish the connection**

**print('Ready to serve...')**

**# TASK 2**

**connectionSocket, addr = serverSocket.accept() #Fill in start #Fill in end**

**try:**

**# TASK 3**

**message = connectionSocket.recv(1024) #Fill in start #Fill in end**

**filename = message.split()[1]**

**f = open(filename[1:])**

**# TASK 4**

**outputdata = f.read() #Fill in start #Fill in end**

**# TASK 5**

**#Send one HTTP header line into socket**

**#Fill in start**

**header = 'HTTP/1.1 200 OK\r\n'**

**connectionSocket.send(header.encode('utf-8'))**

**#Fill in end**

**#Send the content of the requested file to the client**

**for i in range(0, len(outputdata)):**

**connectionSocket.send(outputdata[i].encode('utf-8'))**

**connectionSocket.send("\r\n".encode('utf-8'))**

**connectionSocket.close()**

**except IOError:**

**# TASK 6**

**#Send response message for file not found**

**#Fill in start**

**notFound = 'HTTP/1.1 404 Not Found\r\n'**

**connectionSocket.send(notFound.encode('utf-8'))**

**#Fill in end**

**# TASK 7**

**#Close client socket**

**#Fill in start**

**connectionSocket.close()**

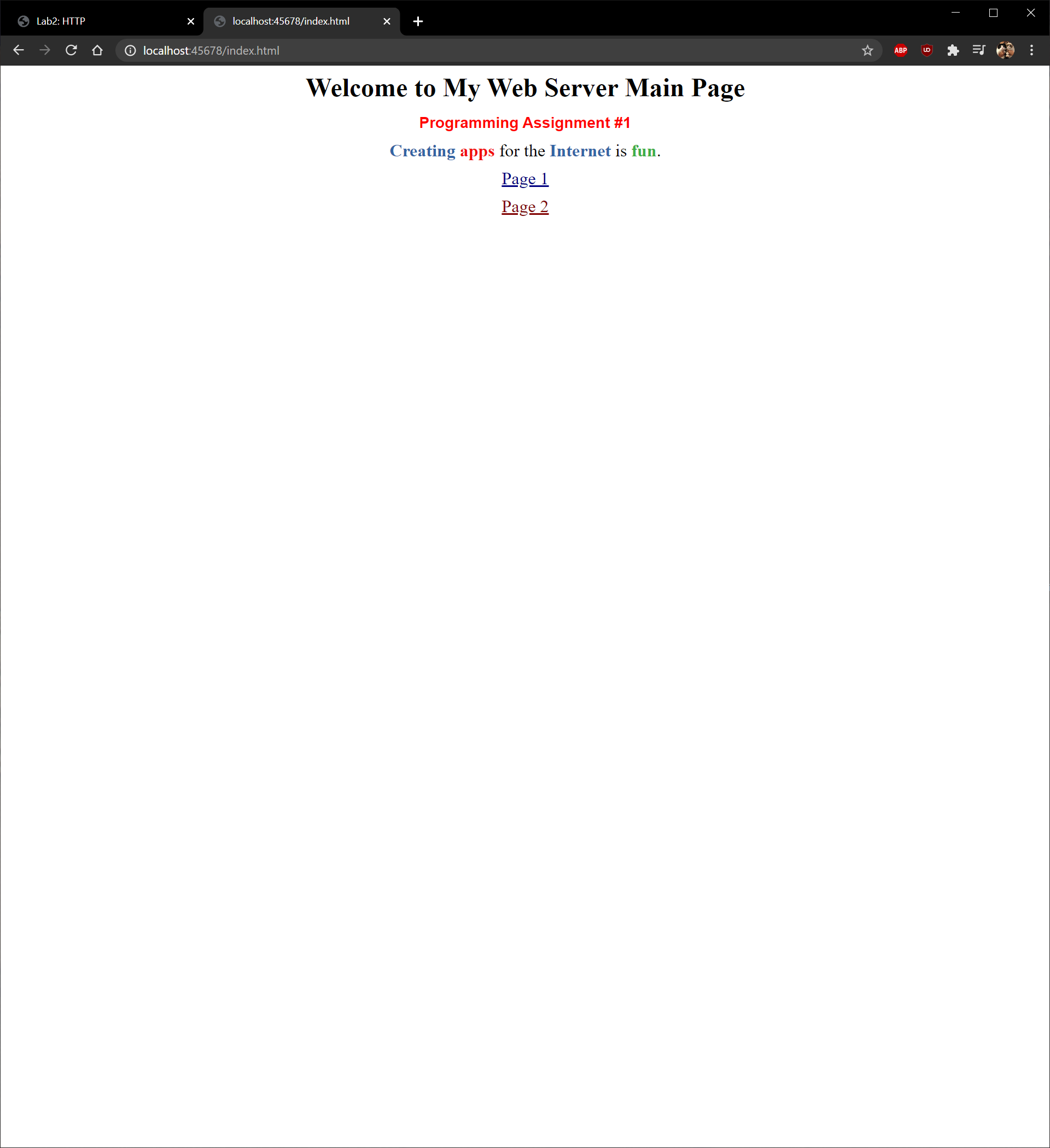
**#Fill in end**

**serverSocket.close()**

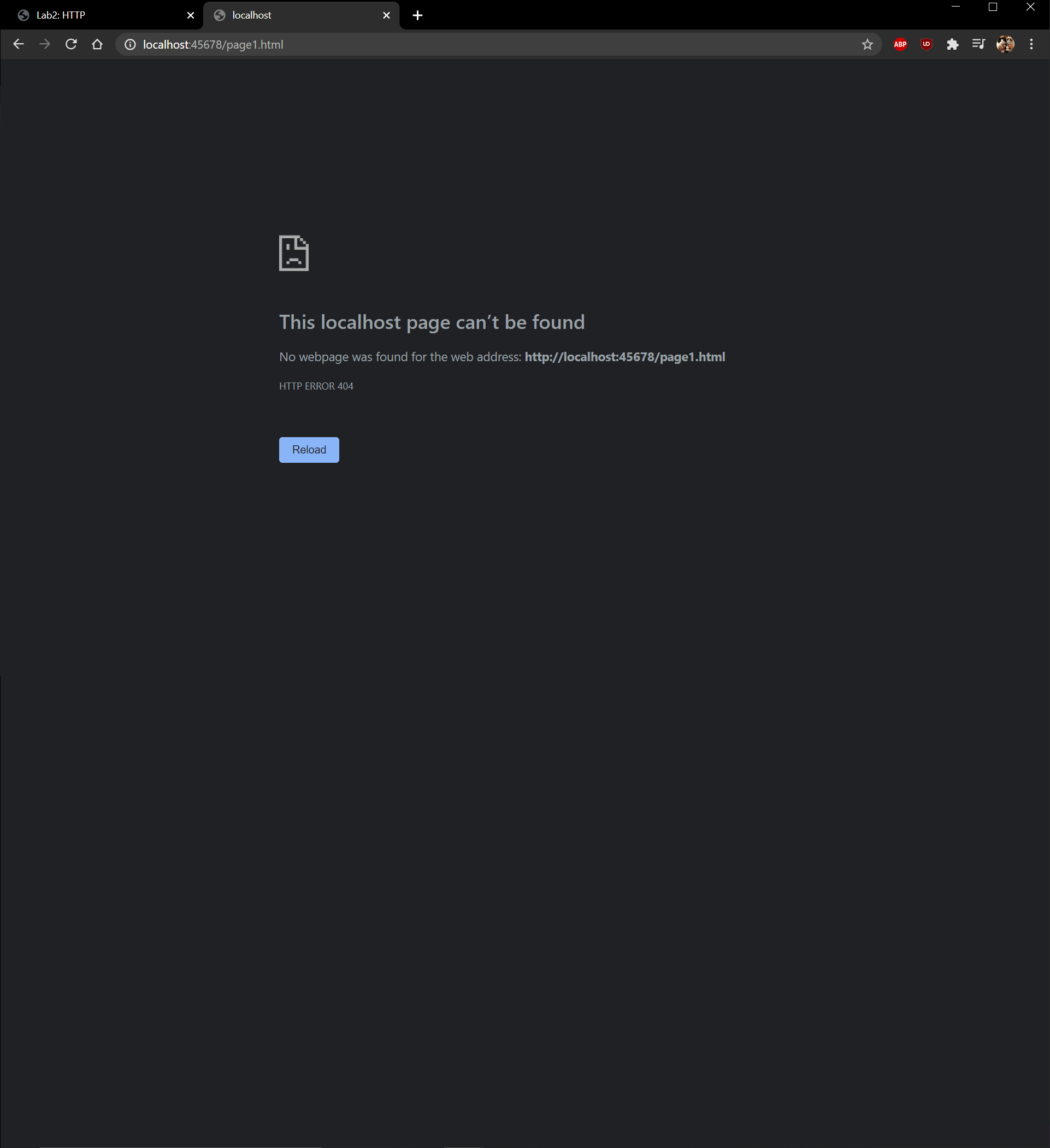
**sys.exit()#Terminate the program after sending the corresponding data**

## Screen Captures

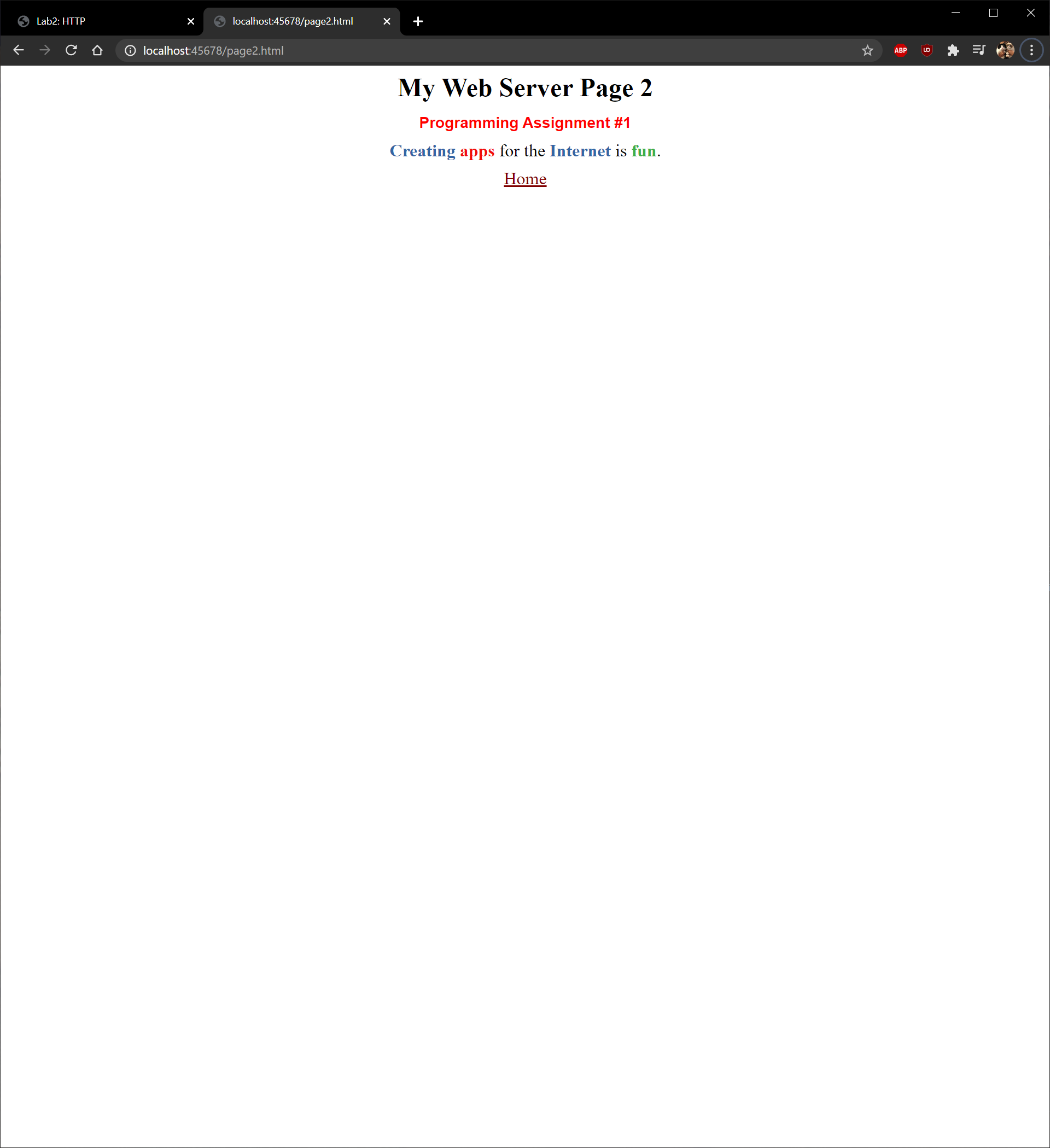
### The Main Page



### The 404 Not Found



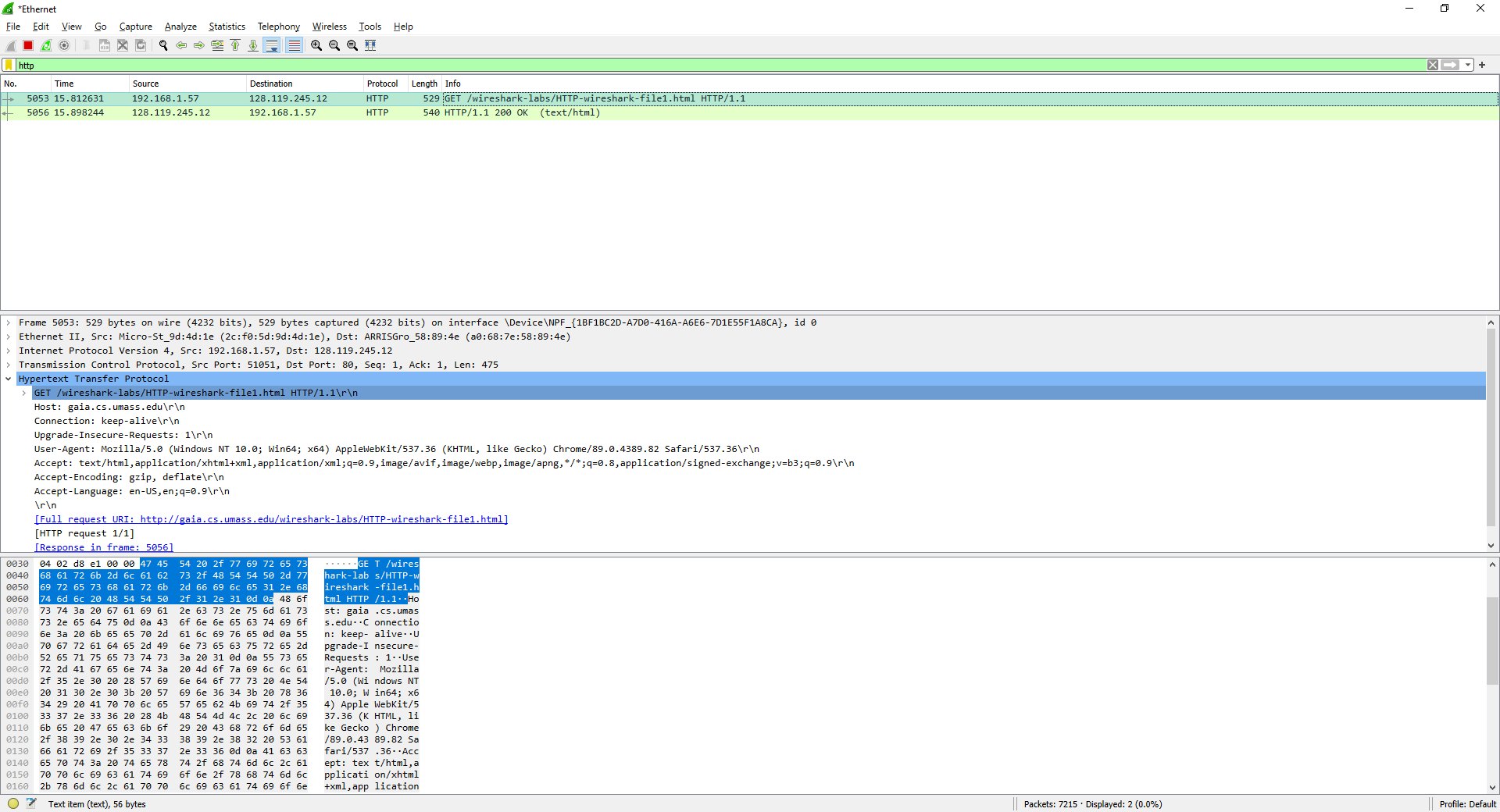
### The Page 2



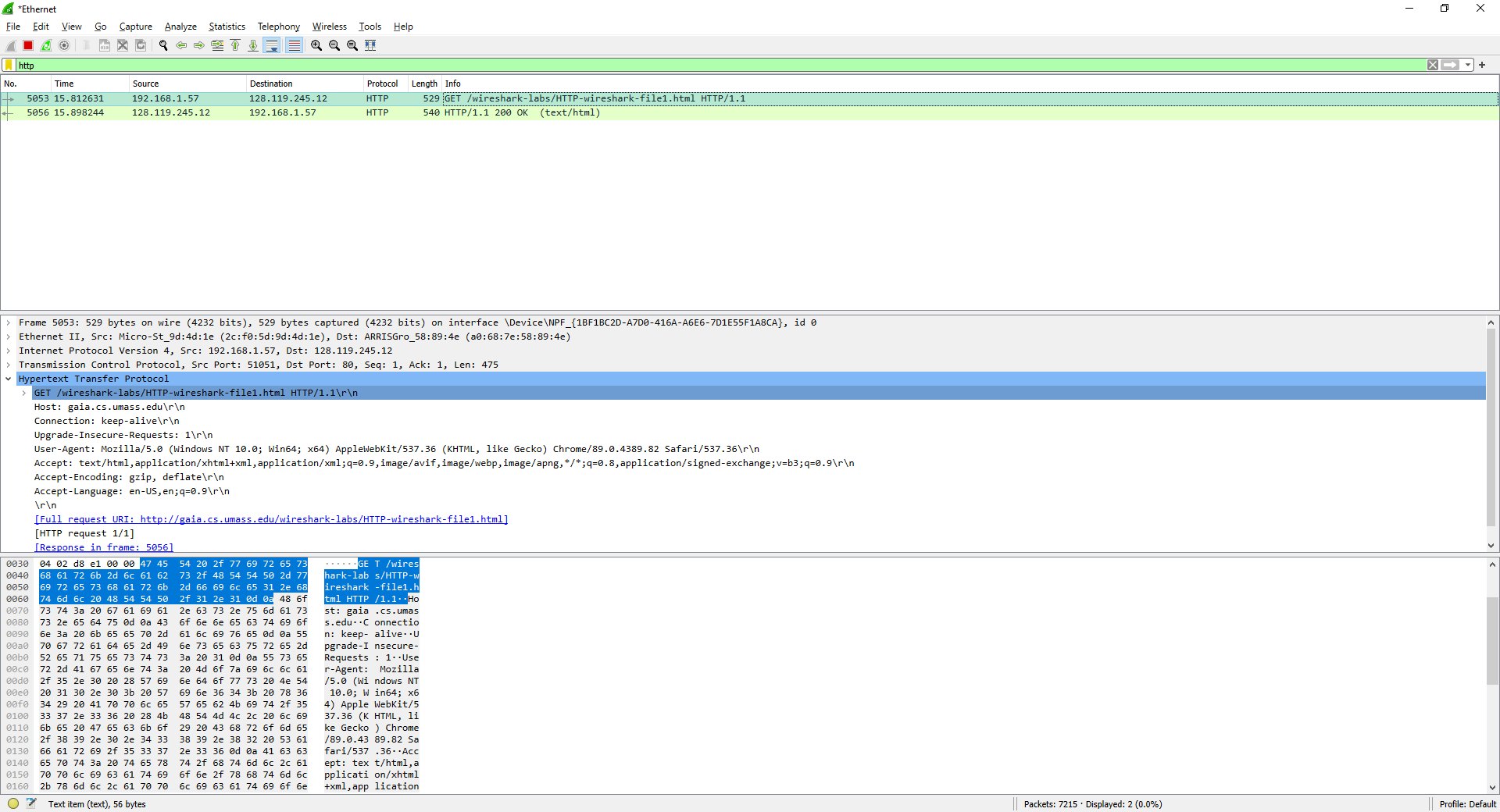
# Part 2 – Analyzing HTTP messages using Wireshark

## THE BASIC HTTP GET/RESPONSE INTERACTION

1.

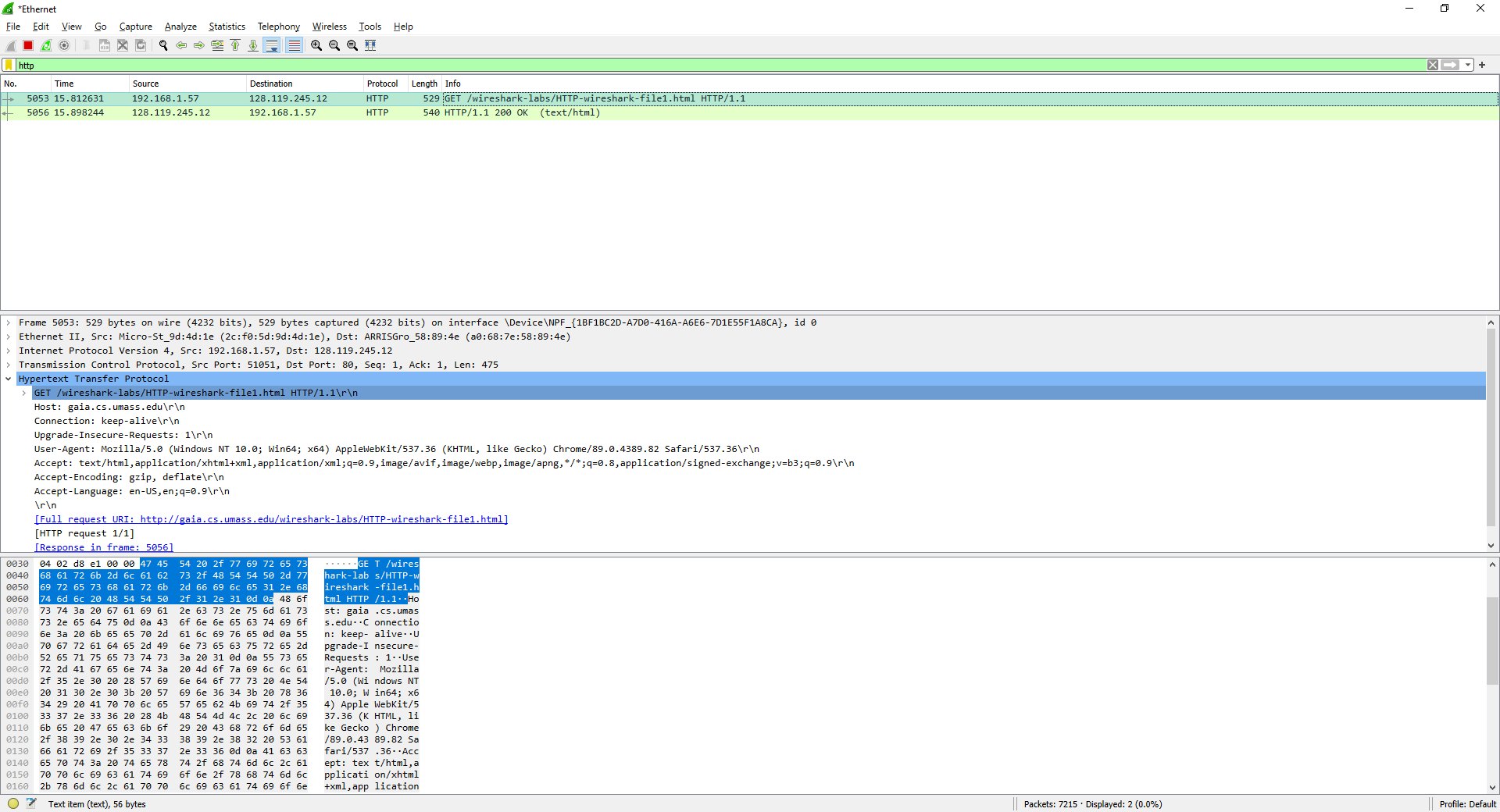


My browser is running HTTP version 1.1 as seen by the GET request. The Server is running version 1.1 as well as seen with “HTTP/1.1 200 OK”.

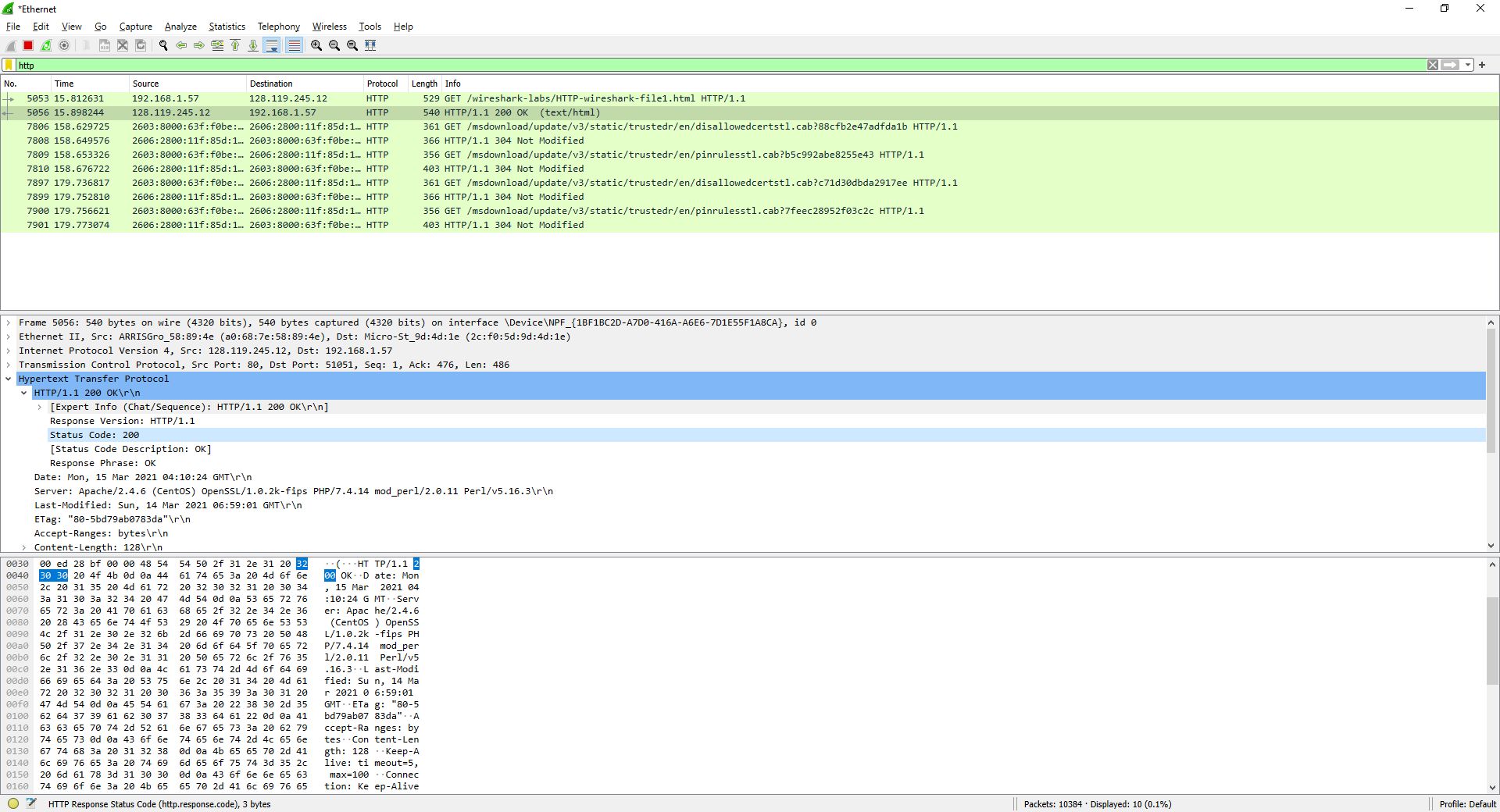
2. 

My browser indicates that it accepts the English language as seen at the bottom of the screenshot with “en-US,en;”

3.

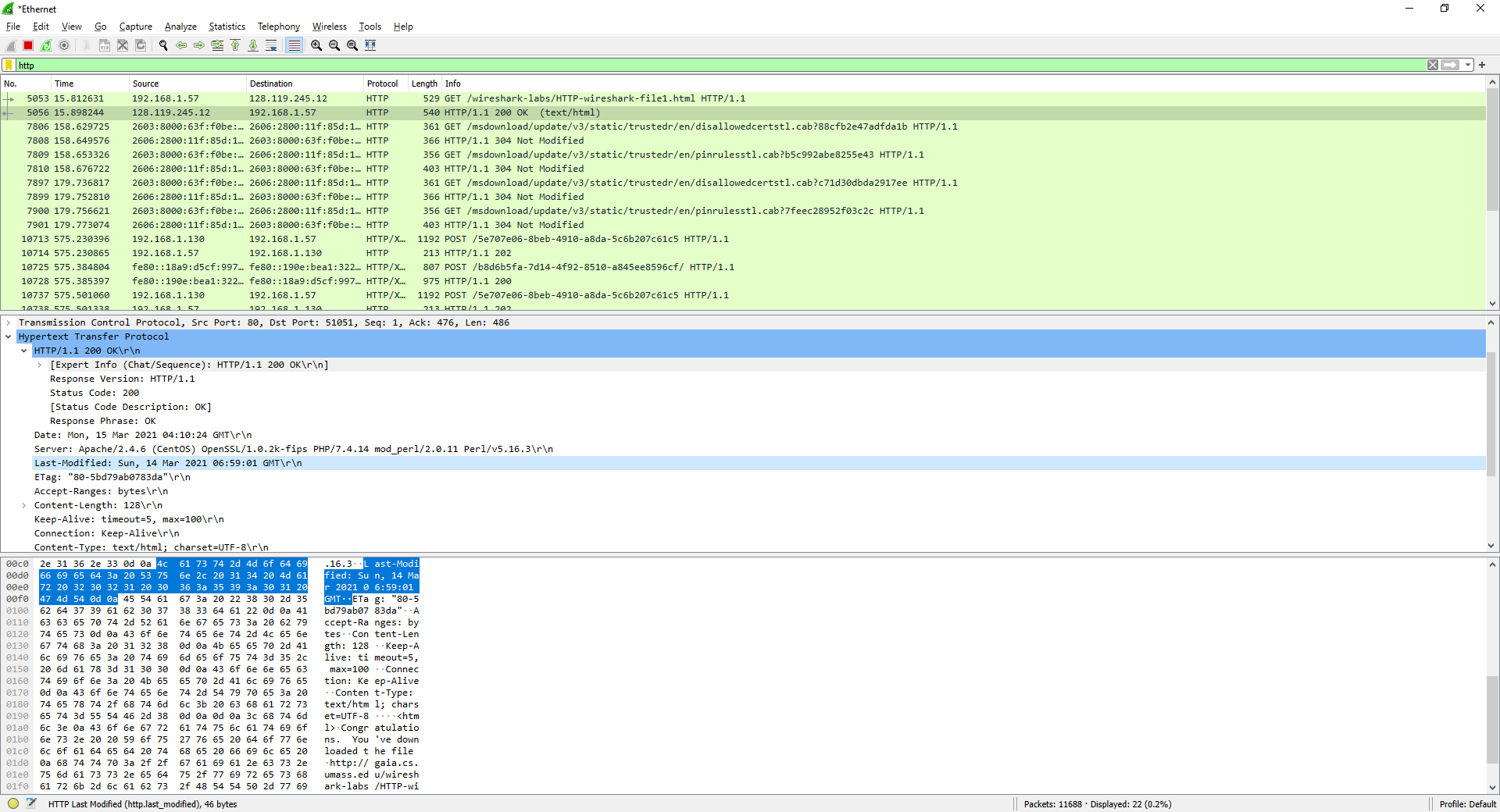


From the GET request we can from the source that my IP address is “192.168.1.57” and the address of the server is the destination at “128.119.245.12”

4. 

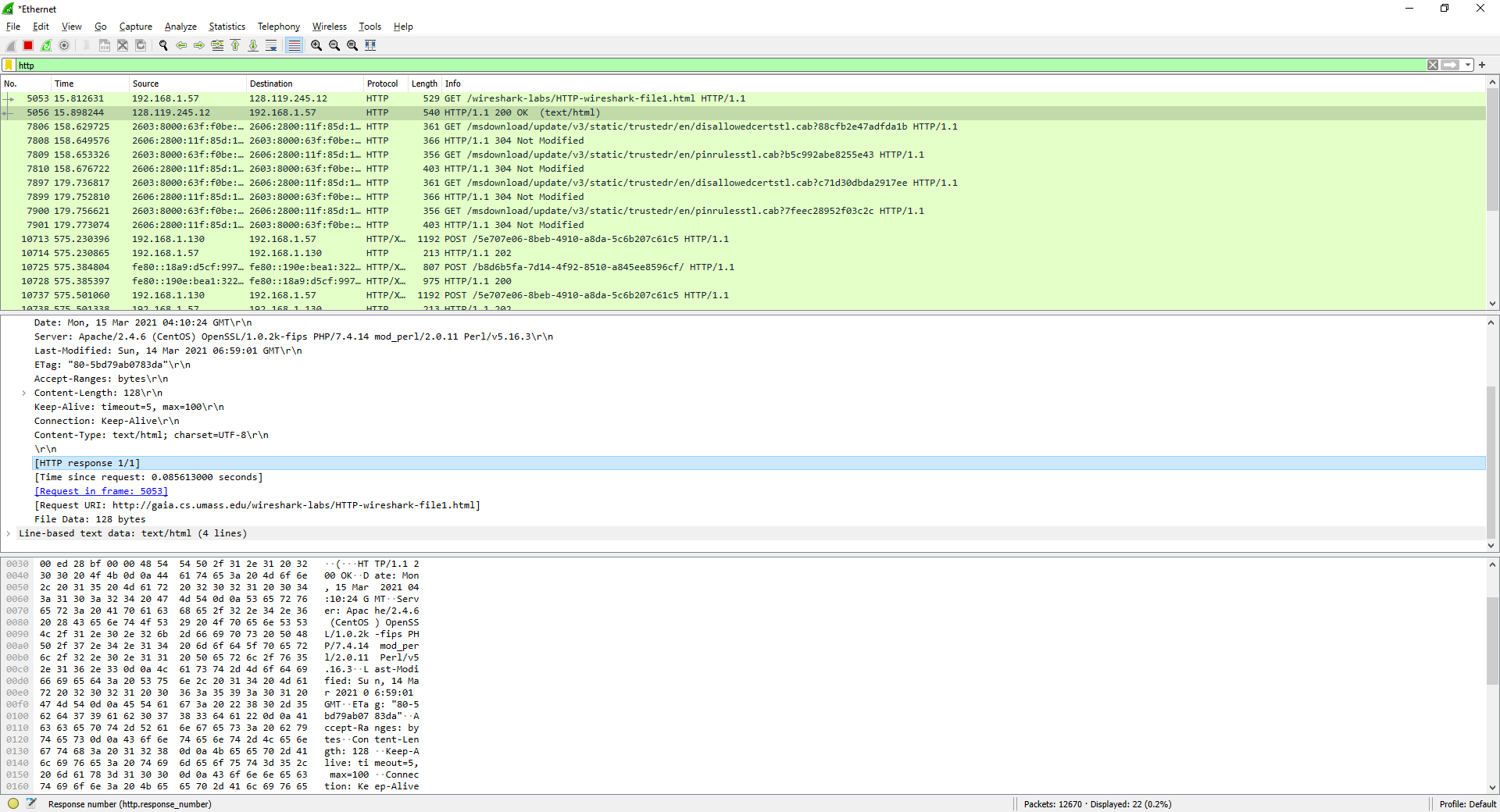
The status code is 200 as seen in the highlighted section of the screenshot.

5.

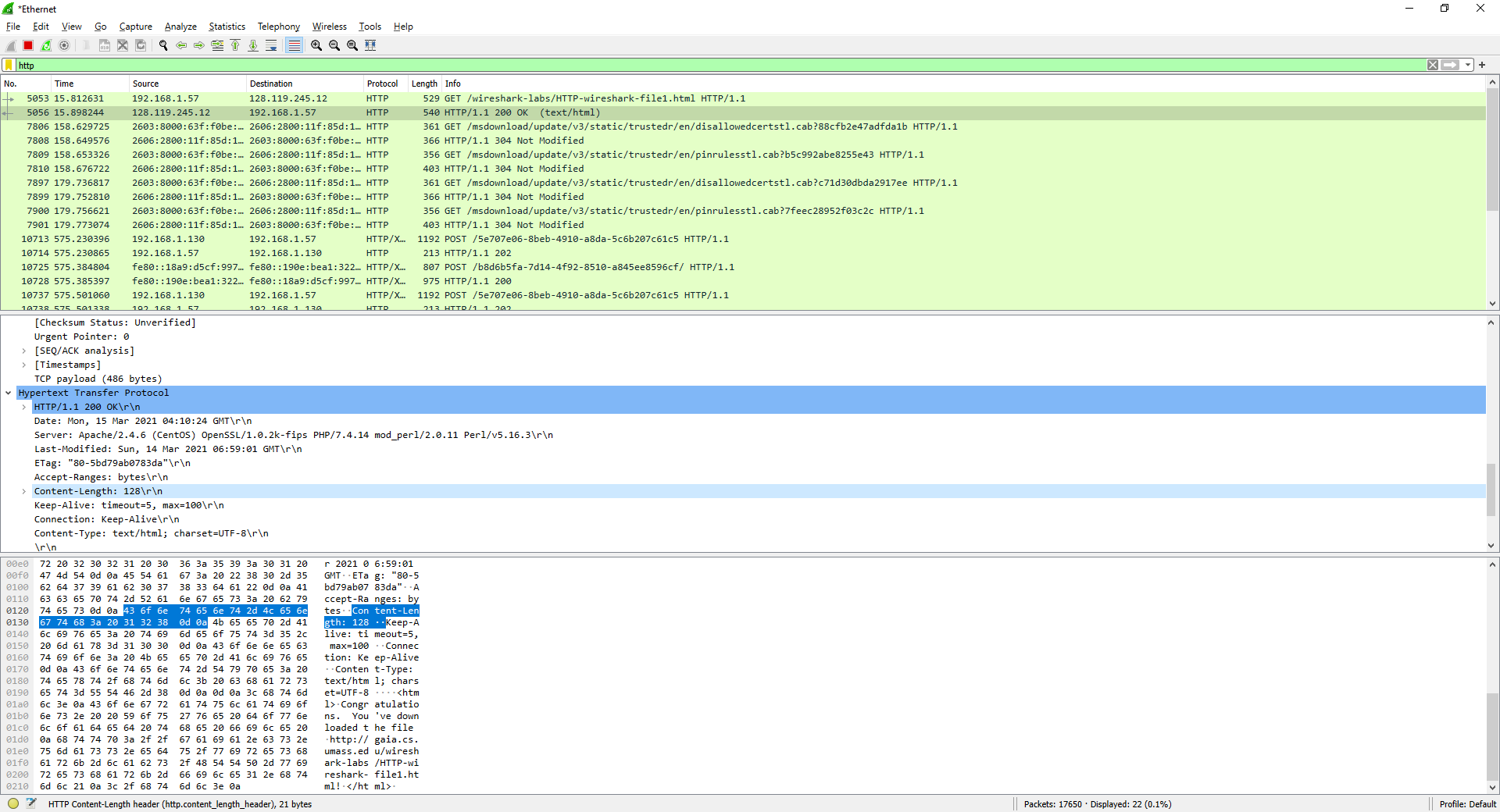


The HTML file was last modified “Sun, 14 Mar 2021 06:59:01 GMT” as seen in the highlighted section of the screenshot

6.

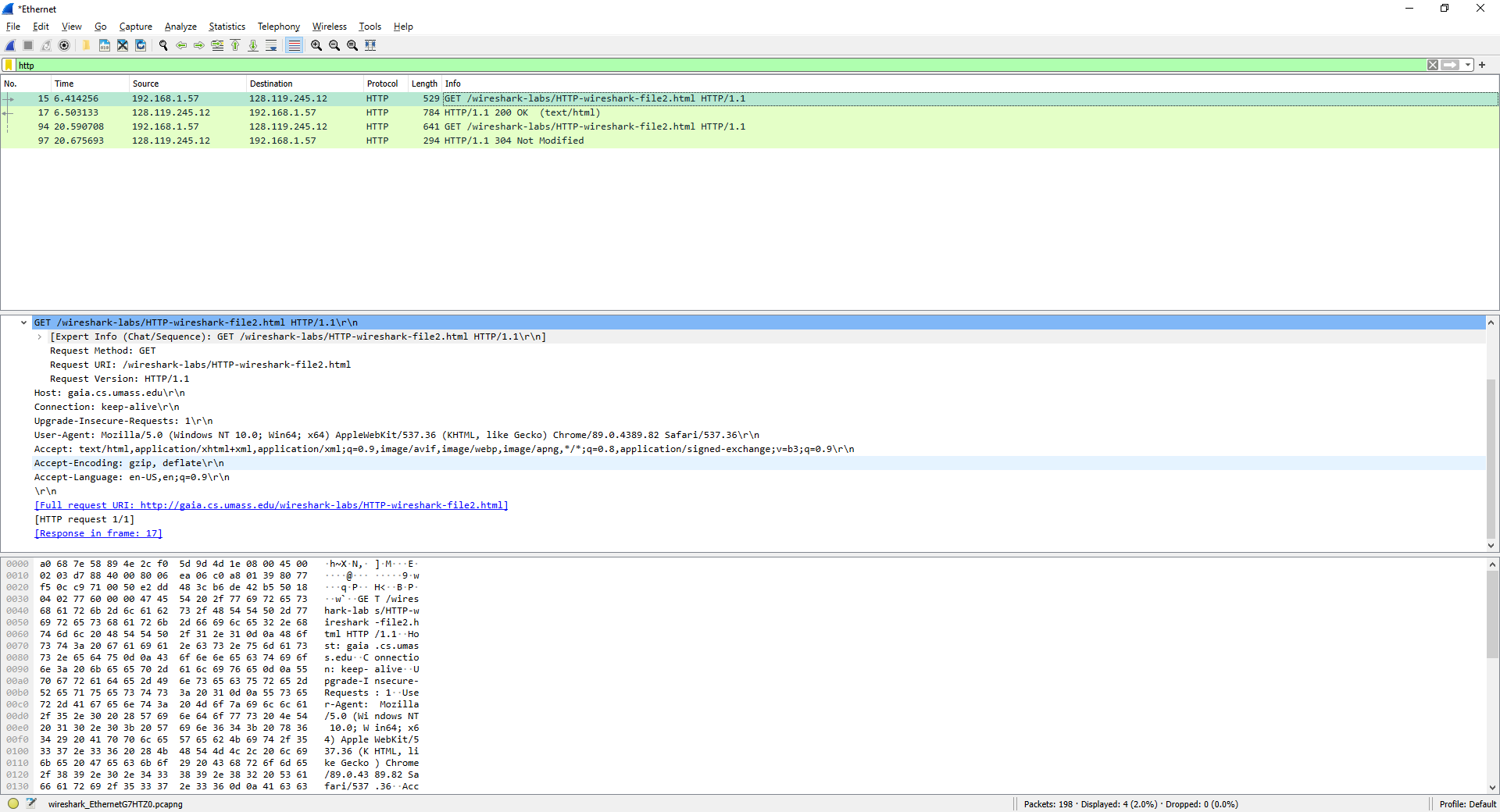


From the screen shot we see that “Content-Length: 128” and “File Data: 128 bytes” so we know 128 bytes are being returned to the browser.

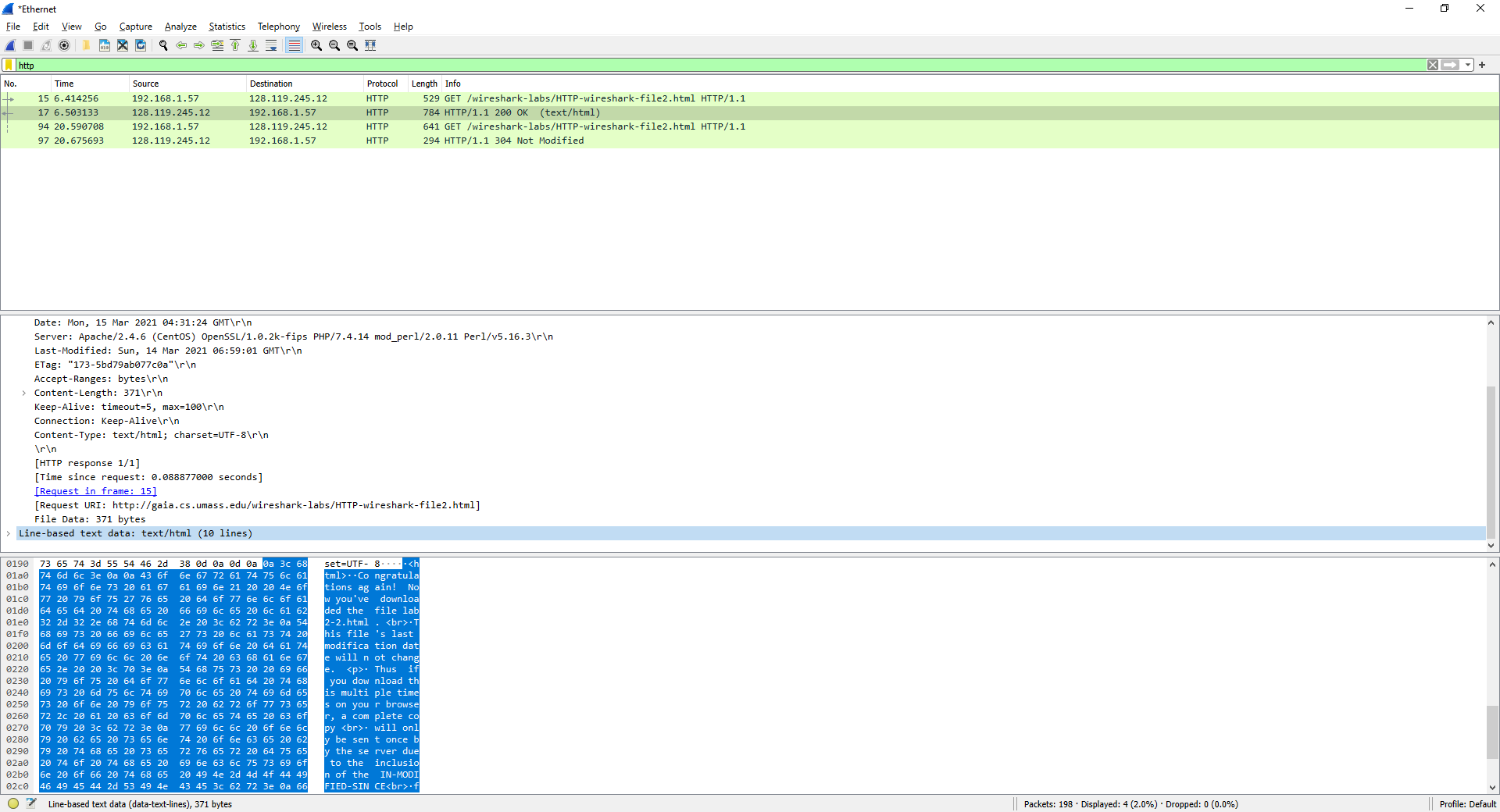
7. 

In the packet listing window, only the status code is seen, so headers like “Content-Length” and “Keep-Alive” are only displayed in the packet content window.

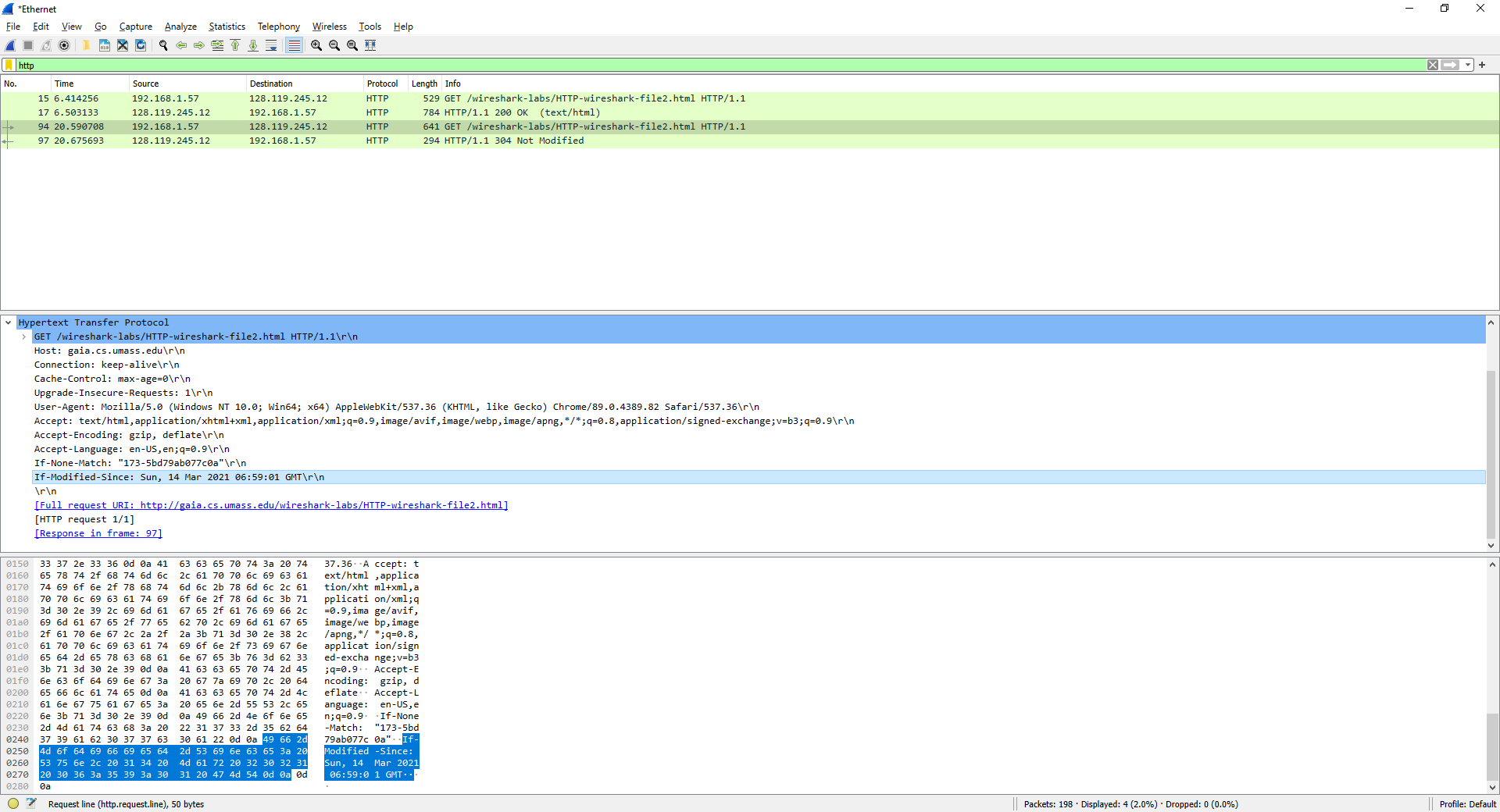
## The HTTP CONDITIONAL GET/RESPONSE INTERACTION

8.

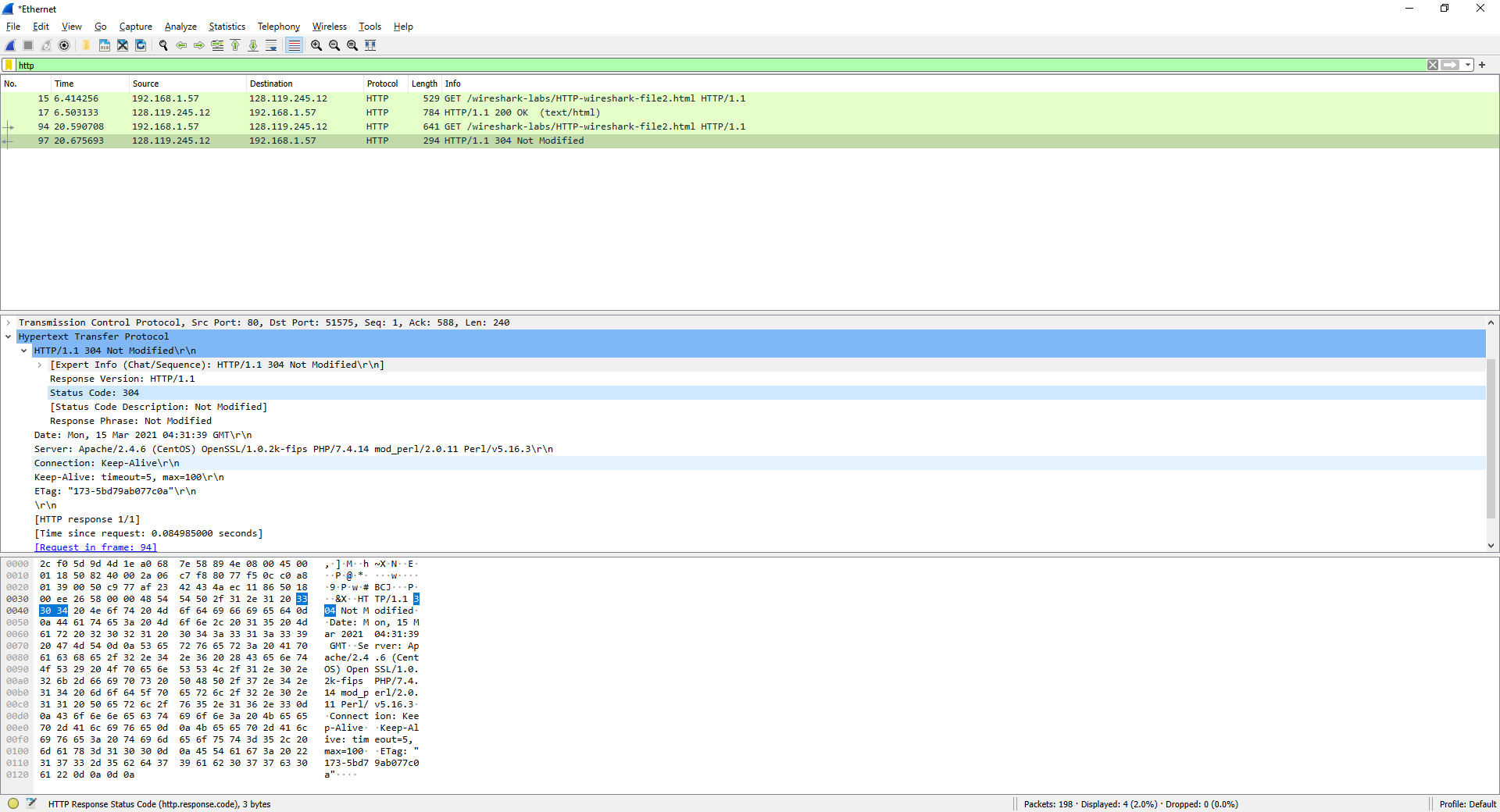
There is no “If-Modified-Since” in the first HTTP GET request.

9.

From the server response, we can see that the server explicitly returns 371 bytes of file data, or 10 lines of “text/html” data from the screenshot highlight.

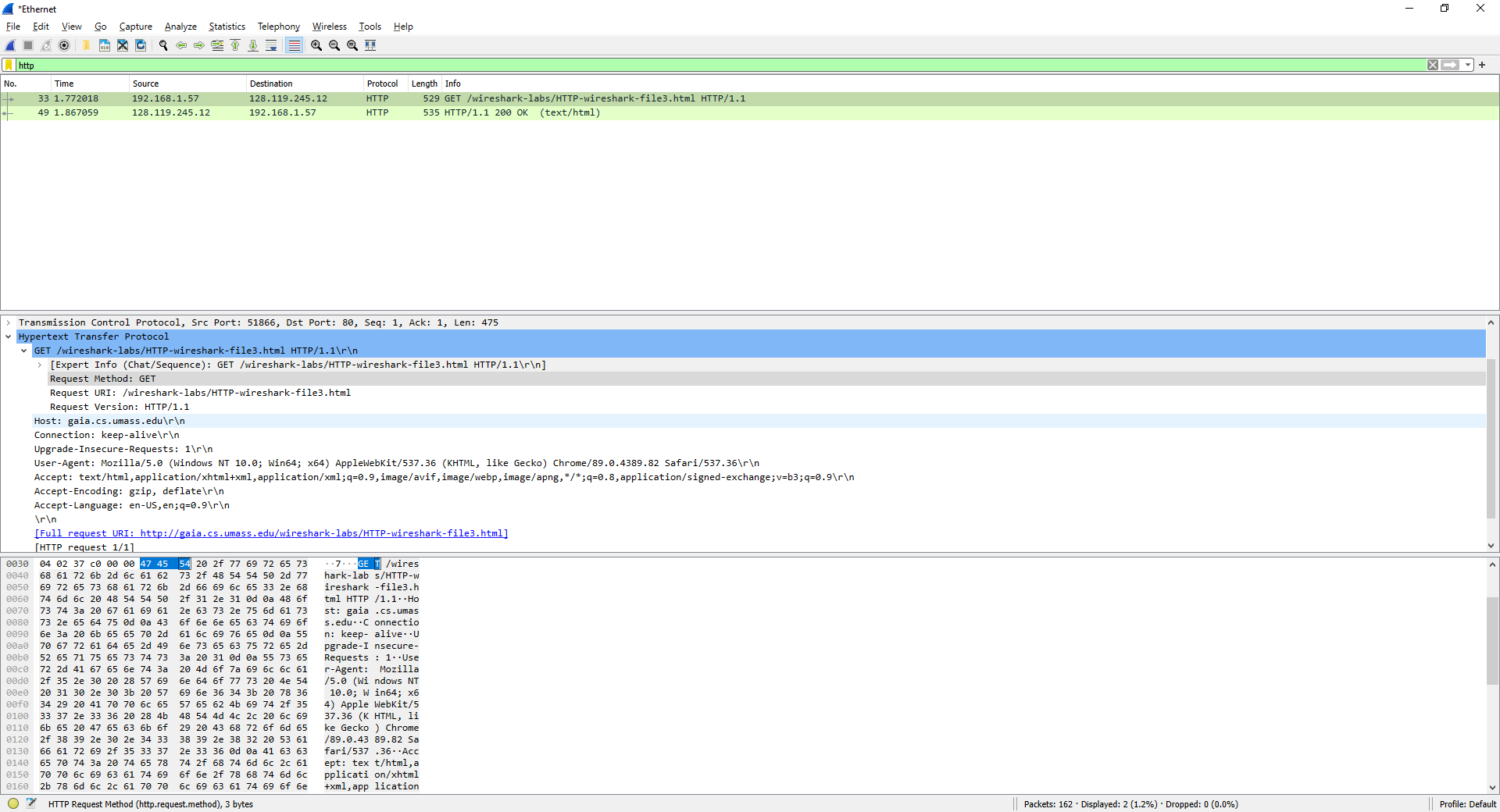
10.

There is an “If-Modified-Since” and the information following it is the exact time and data that it was sent from the server.

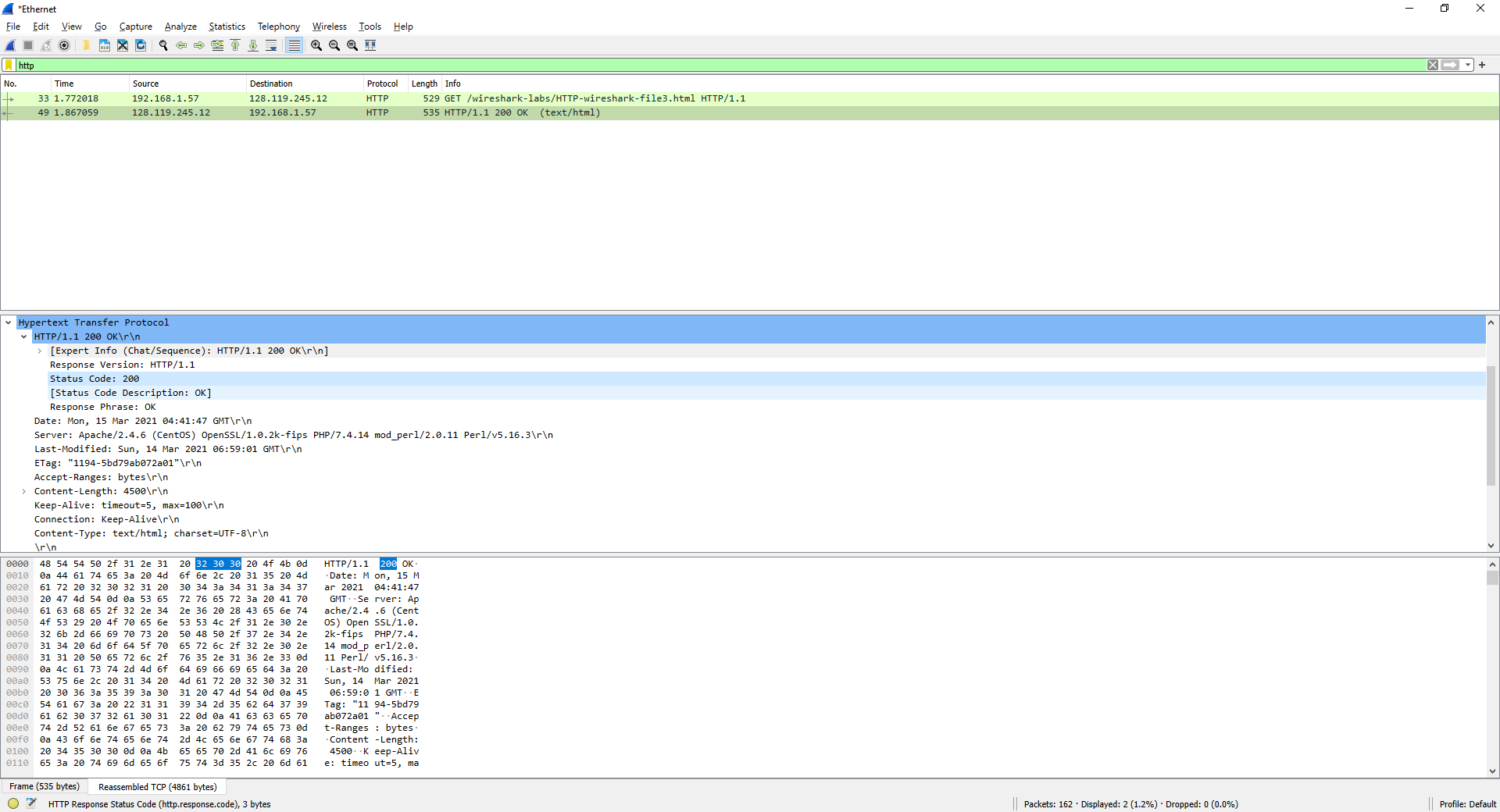
11.

The HTTP status code from the second server response was 304 Not Modified. This time, the server didn’t explicitly return the contents of the file because there is no “Content-Length” header or raw text data sent in this response unlike the first server response.

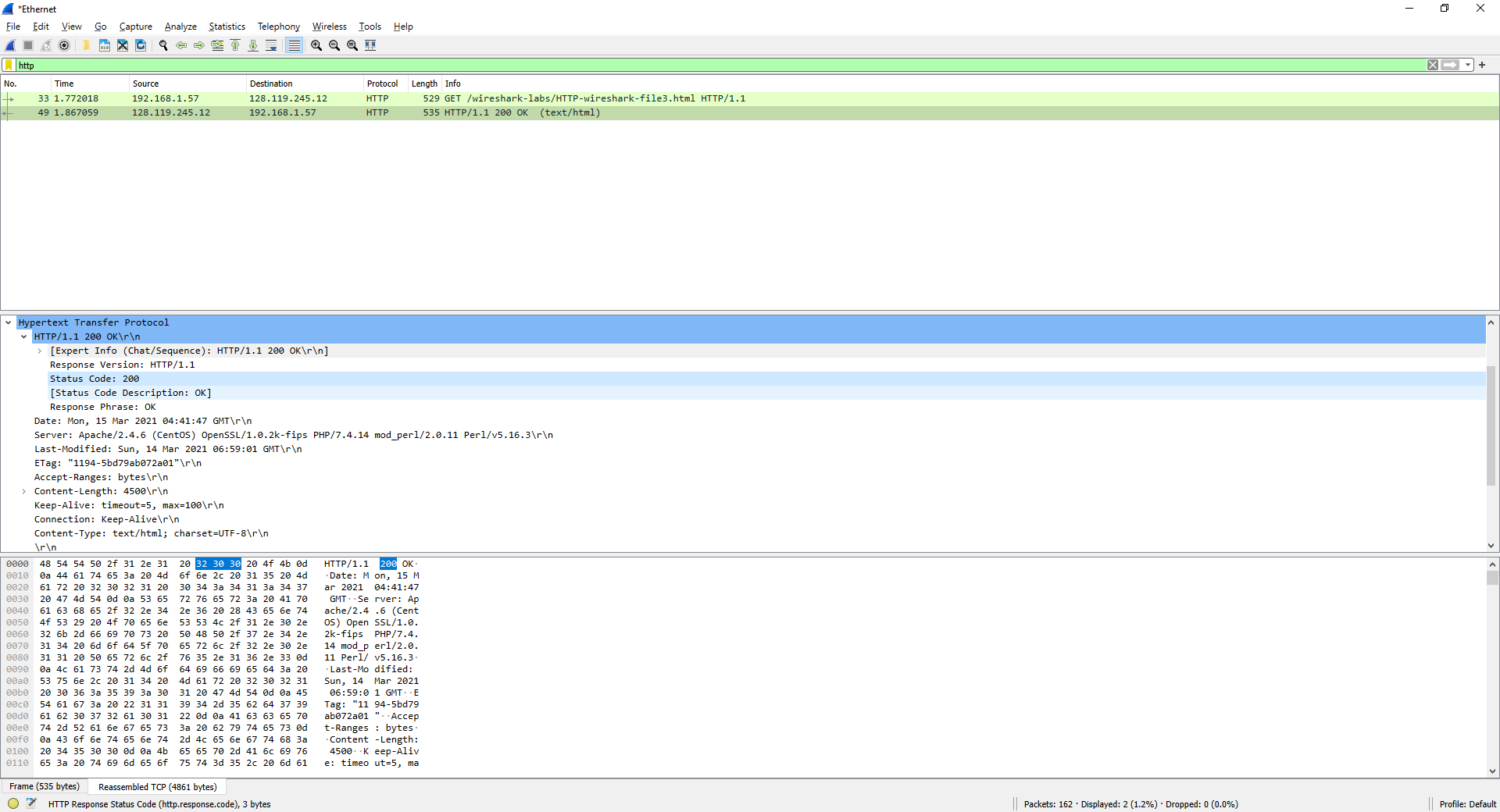
## RETRIEVING LONG DOCUMENTS

12. 

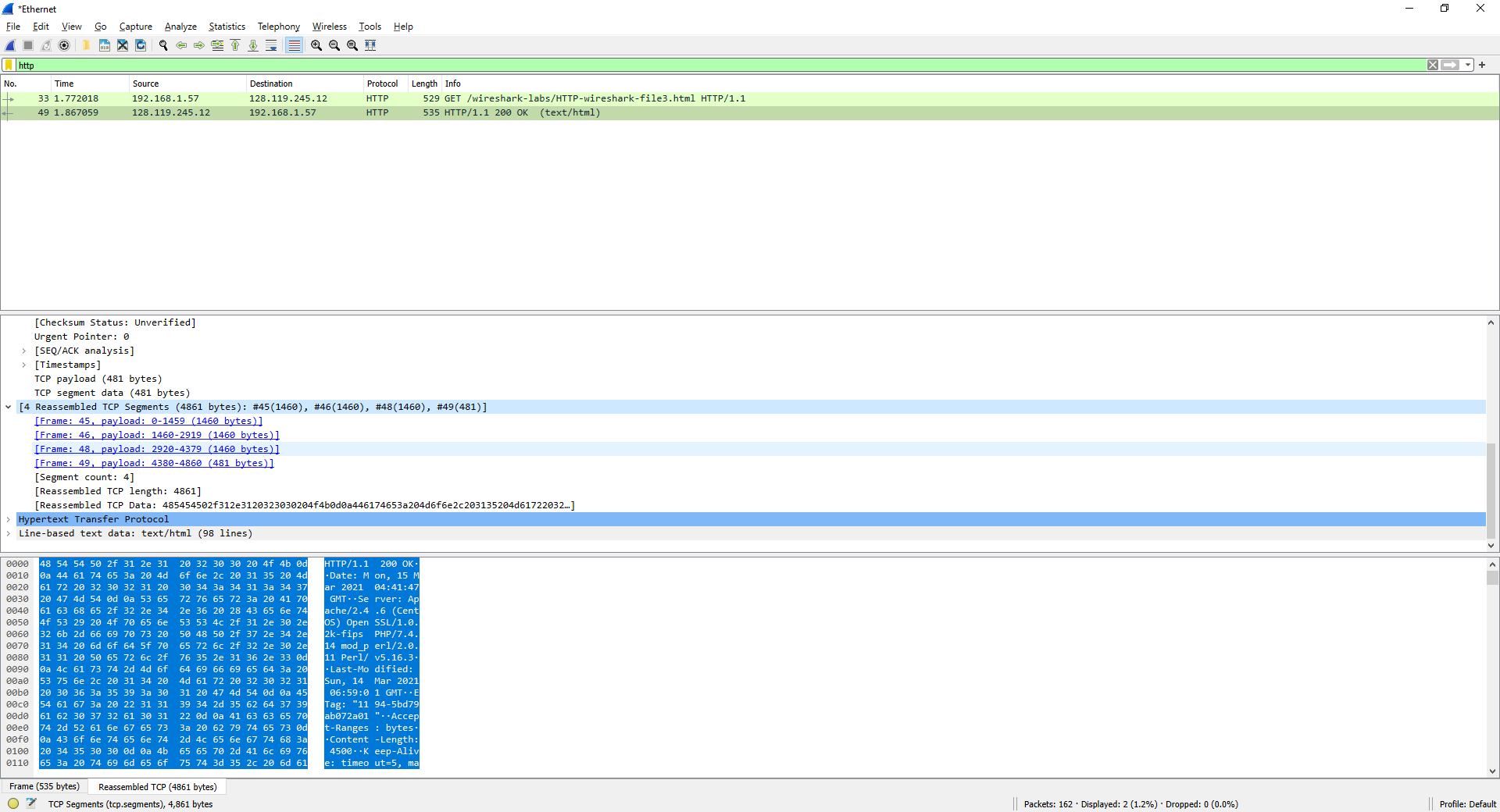
The browser only sent one HTTP GET request message as seen in “[HTTP request 1/1]”. The packet number corresponding to the GET message was “47 54 54” in hexidecimal as seen in the highlighted portions.

13. 

The packet number corresponding the status code 200 in the server response was “32 30 30” in hexidecimal.

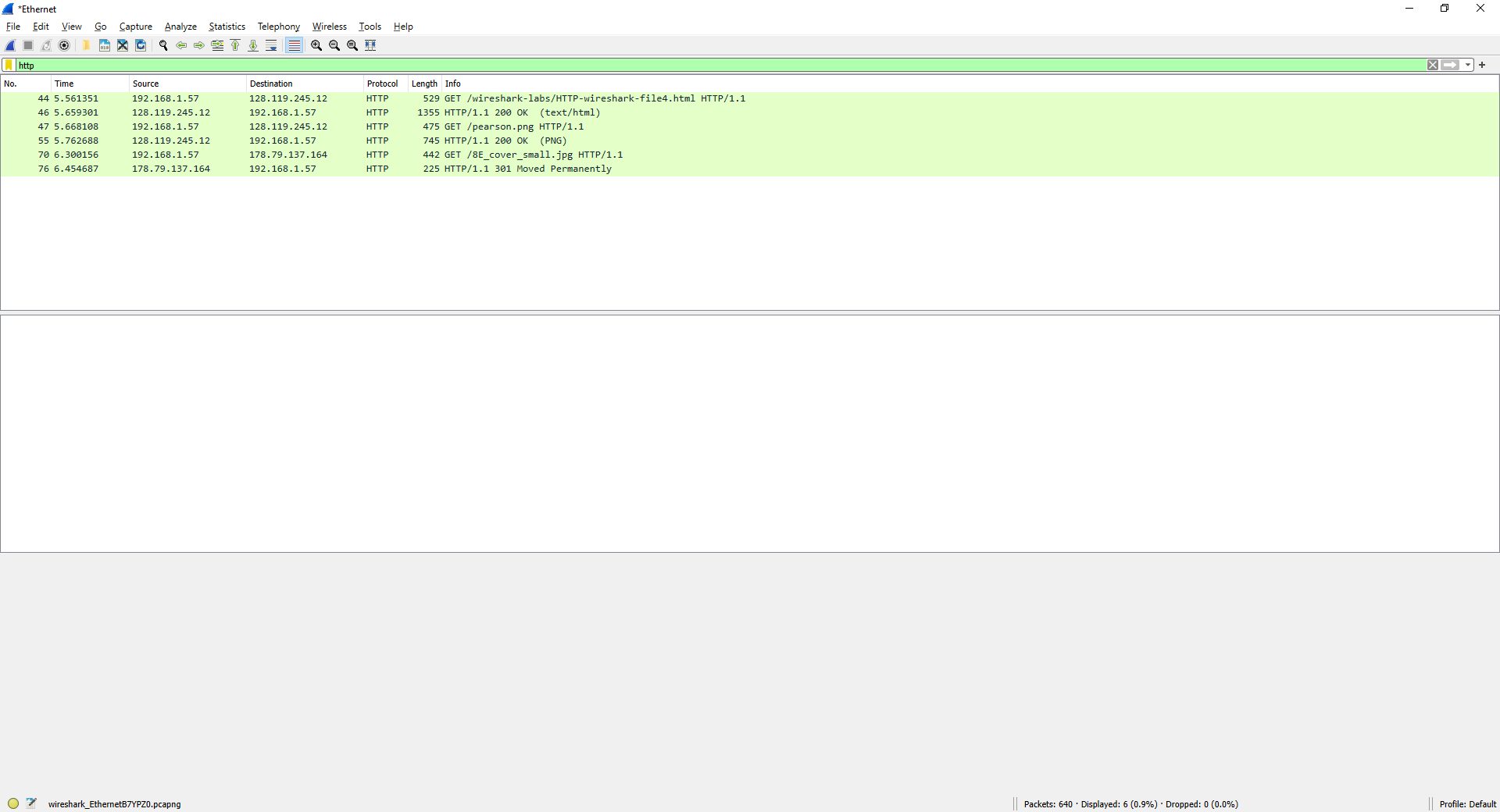
14. 

The status code was 200 and the response phrase was OK.

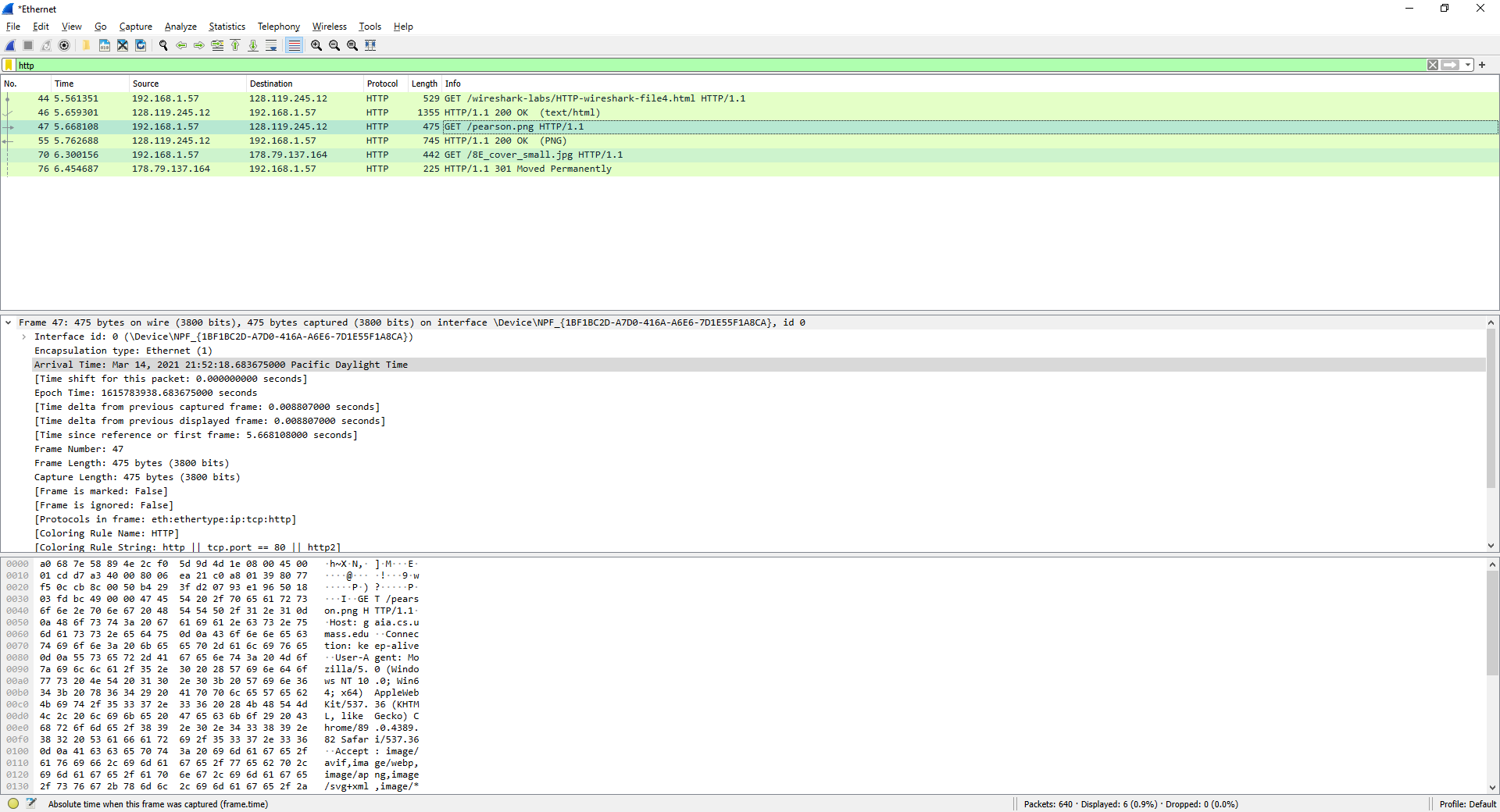
15. 

There were 4 TCP segments that were needed to carry the HTTP response and the text.

## HTML DOCUMENTS WITH EMBEDDED OBJECTS

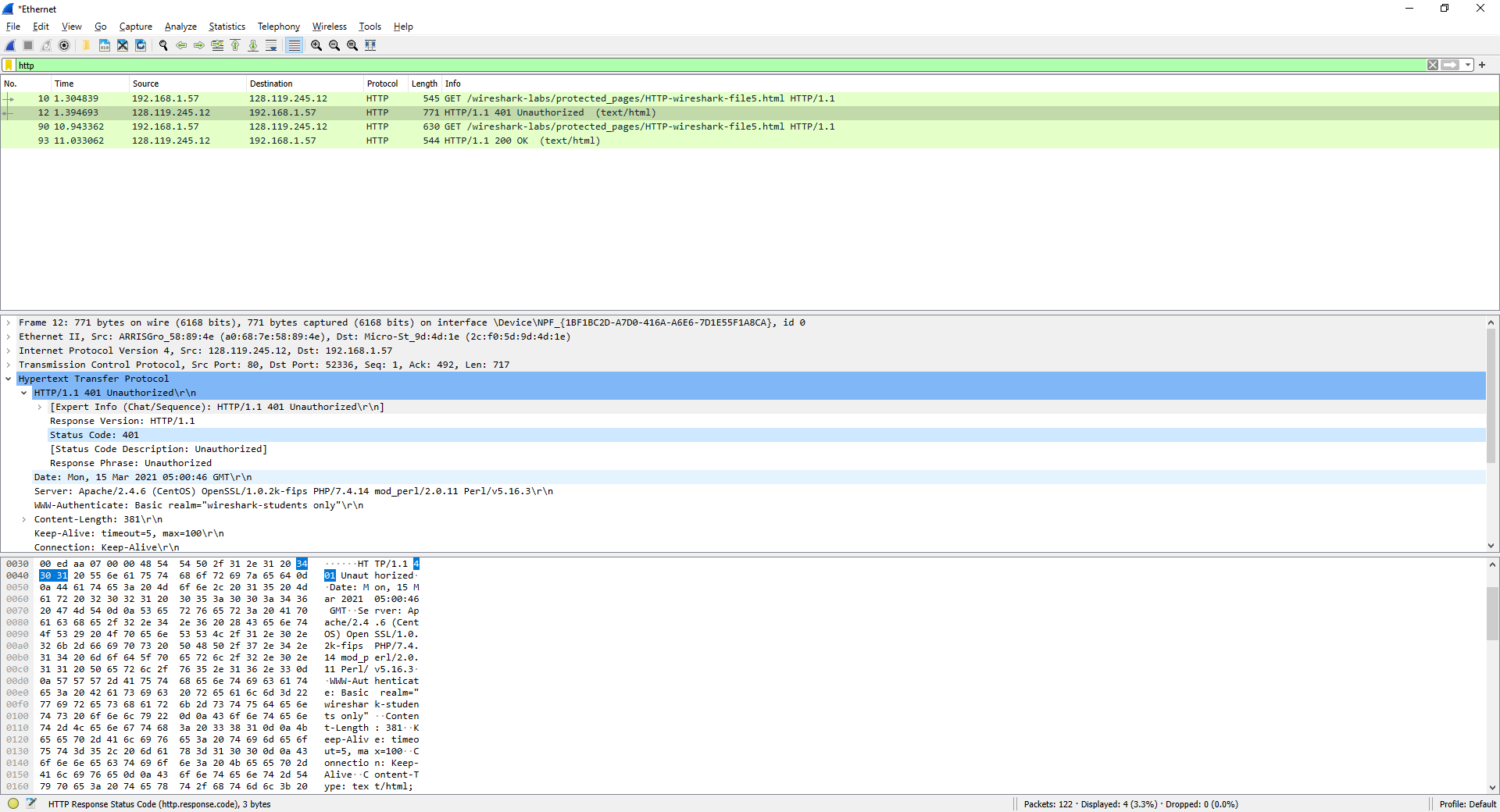
16. 

There were 3 total GET requests from the browser, where the first 2 were sent to IP address 128.119.245.12 and the last was sent to IP address 178.79.137.164.

17. 

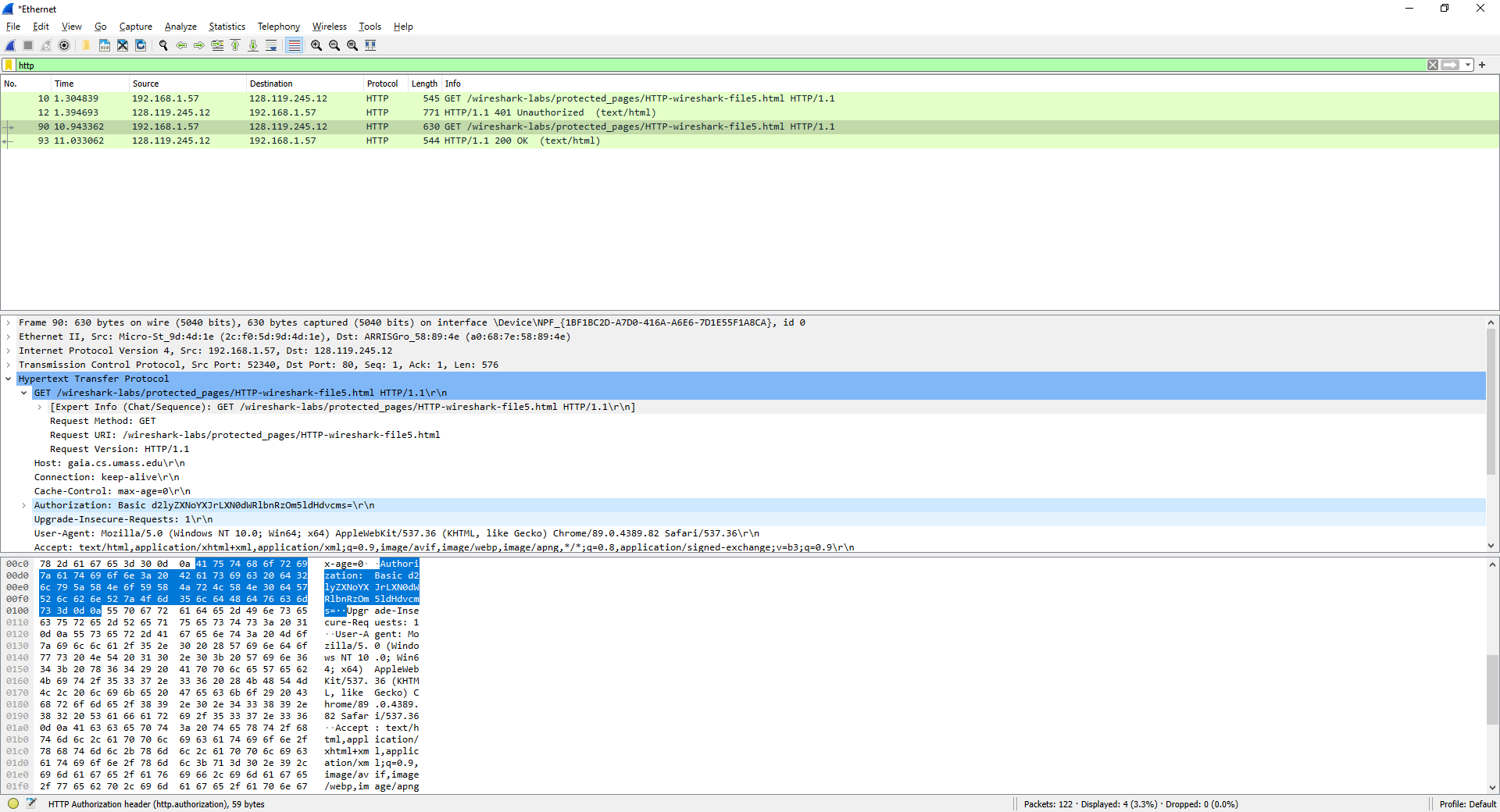
We can tell that these two images were downloaded at different times due to the Time column in the packet listing window. The second GET request was done after the first one got its response, so we can say that the images were downloaded serially.

## HTTP AUTHENTICATION

18. 

The server’s intial response the HTTP GET message was 401 Unauthorized

19.



The new field included in the second HTTP GET message is “Authorization”