Lab 5: *Introduction to HTTP*

Name1: Muhammad Rizwan Khalid ***Reg. No# 180459***
Name2: Muhammad Roshan Mughees ***Reg. No# 193590***

Lab Title: Wireshark – HTTP (Hypertext Transfer Protocol)

Objective of this lab:

In this lab, we'll explore several aspects of the HTTP protocol: the basic GET/response interaction, and HTTP message formats.

Instructions:

- *Read carefully before starting the lab.*
- *These exercises are to be done individually.*
- *You are supposed to provide the answers to the in-line questions in this document and upload the completed document to your course's LMS site.*
- ***For all questions, you must not only answer the question, but also supply all necessary information regarding how you arrived at the answer (e.g., use screenshots/ accompanying text, etc.) Use red font color to distinguish your replies from the rest of the text.***
- *Avoid plagiarism by copying from the Internet or from your peers. You may refer to source/ text but you must paraphrase the original work.*

Background:

The world's web browsers, servers and related web applications all talk to each other through HTTP, the Hypertext Transfer Protocol. Before proceeding to the experiments, it is recommended that you read introductions to some general terms used in this lab, to avoid any confusion.

1. What is a web page?

A Web page (also called a document) consists of objects. An object is a simple file -- such as a HTML file, a JPEG image, a GIF image, a Java applet, an audio clip, etc. -- that is addressable by a single URL. Most Web pages consist of a base HTML file and several referenced objects. For example, if a Web page contains HTML text and five JPEG images, then the Web page has six objects: the base HTML file plus the five images. The base HTML file references the other objects in the page with the objects' URLs. Each URL has two components: the host name of the server that houses the object and the object's path name. For example, the URL www.someSchool.edu/someDepartment/picture.gif has www.someSchool.edu for a host name and /someDepartment/picture.gif for a path name.

2. What is a web browser?

A browser is a user agent for the Web; it displays to the user the requested Web page and provides numerous navigational and configuration features. Web browsers also implement the client side of HTTP. Thus, in the context of the Web, we will interchangeably use the words "browser" and "client".

Popular Web browsers include Google Chrome, Netscape Communicator, Apple Safari and Microsoft Explorer.

3. What is a web server?

A Web server hosts Web objects, each addressable by a URL. Web servers also implement the server side of HTTP. Popular Web servers include Apache, Microsoft Internet Information Server, and the Netscape Enterprise Server.

4. Introduction to HTTP:

The Hypertext Transfer Protocol (HTTP), the Web's application-layer protocol, is at the heart of the Web. HTTP is implemented in two programs: a client program and server program. The client program and server programs, executing on different end systems, talk to each other by exchanging HTTP messages. HTTP defines the structure of these messages and how the client and server exchange the messages. HTTP defines how Web clients (i.e., browsers) request Web pages from servers (i.e., Web servers) and how servers transfer Web pages to clients. When a user requests a Web page (e.g., clicks on a hyperlink), the browser sends HTTP request messages for the objects in the page to the server. The server receives the requests and responds with HTTP response messages that contain the objects.

Steps for performing this lab:

For all the experiments we will use *Wireshark* packet analyzer.

Exercise 01: The Basic HTTP GET/response interaction

Aim of this exercise: We will now learn about what packets are exchanged during a HTTP conversation---we will learn about the HTTP GET message that is sent from the HTTP client to the HTTP server and the HTTP message that is sent as response to this message.

Follow the steps below to complete this exercise and to provide answers to the questions below

- Start up your web browser.
- Start up the Wireshark packet sniffer (but don't yet begin packet capture). Enter "http" (just the letters, not the quotation marks) in the display-filter-specification window, so that only captured HTTP messages will be displayed later in the packet-listing window. (We're only interested in the HTTP protocol here, and don't want to see the clutter of all captured packets).
- Begin Wireshark packet capture.
- Enter the following to your browser http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html. Your browser should display the very simple, one-line HTML file.
- Stop Wireshark packet capture.

The example in Figure 1 shows in the packet-listing window that two HTTP messages were captured: the GET message (from your browser to the gaia.cs.umass.edu web server) and the response message from the server to your browser. The packet-contents window shows details of the selected message (in this case the HTTP GET message, which is highlighted in the packet-listing window). Recall that since the HTTP message was carried inside a TCP segment, which was carried inside an IP datagram, which was carried within an Ethernet frame, Wireshark displays the Frame, Ethernet, IP, and TCP packet information as well.

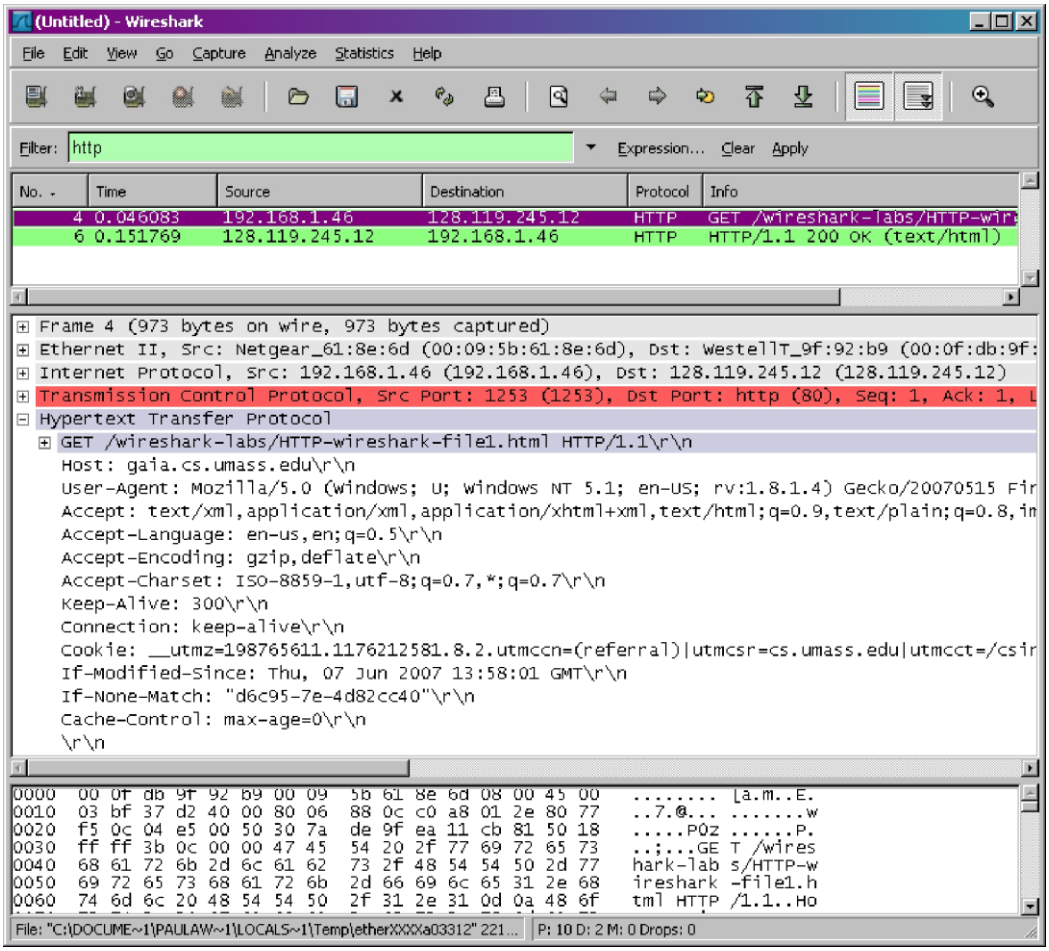
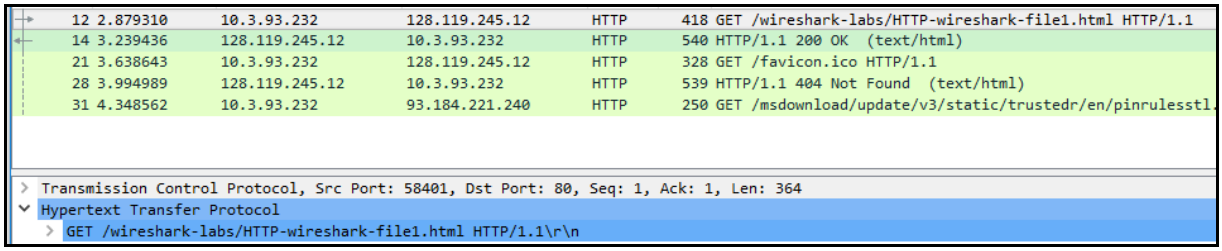


Figure 1: Wireshark display after http://gaia.cs.umass.edu/wireshark-labs/ HTTP-wireshark-file1.html has been retrieved by your browser

By looking at the information in the HTTP GET and response messages that you have captured, answer the following questions:

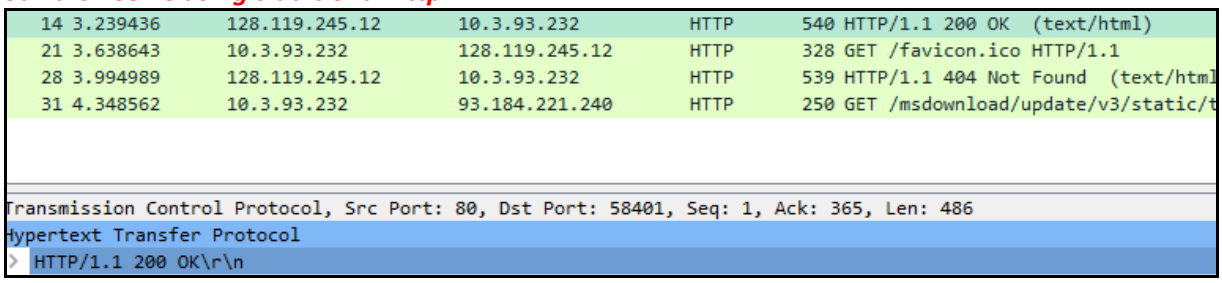
1.1 Which version of HTTP is the browser running 1.0 or 1.1? Which HTTP version is the server running?

Answer:



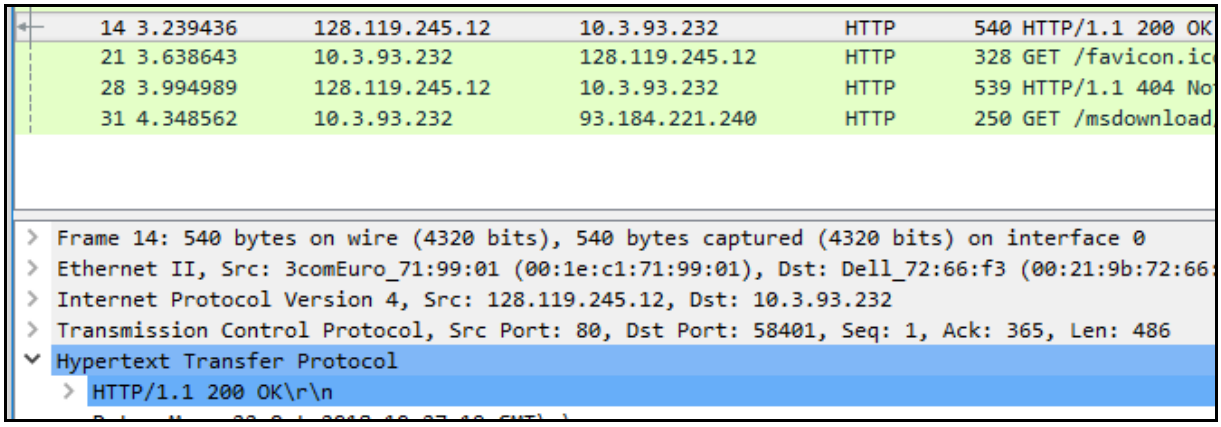
Browser sent the get message to the server for downloading the required files. The get message

contain the version of http. As it is seen in the picture that the request version of http is 1.1. So, our browser is using traditional http 1.1.



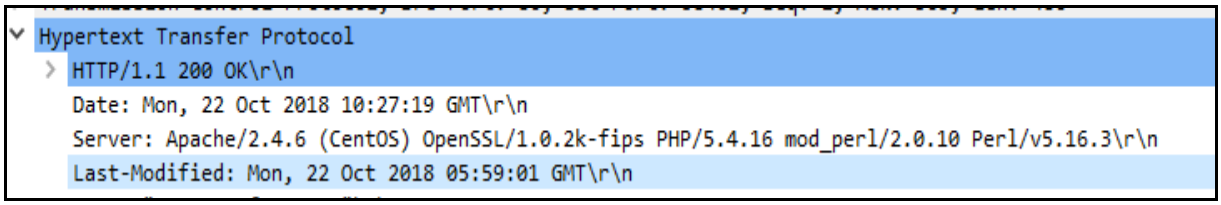
Here as we are receiving the required files. The server sent a response message that file is received and it has redirected to another server for completon of requests. As it is seen in the picture that the version of http 1.1. So, our server is using traditional http 1.1.

1.2 What is the status code returned from the server to your browser?



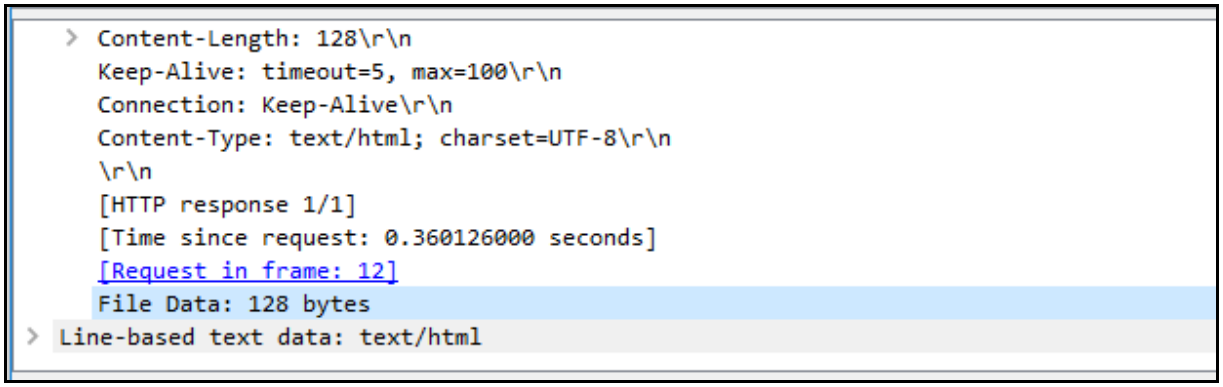
It returns status code 200 OK. This status shows us that there exist a link or file against our requested query and client retrieved the requested file from the server having the file.

1.3 When the HTML file that you are retrieving was last modified at the server?



The wireshark also shows the information about the requested file. It retrieves information from the server and also retrieves the last modification time of the file. As shown in the picture, the requested file was last modified today at 05:59 AM

1.4 How many bytes of content are being returned to your browser?



The wireshark also provides the information of the number of bytes sent by the server. In this case it is "128 bytes"

Exercise 02: *The HTTP CONDITIONAL GET/response interaction*

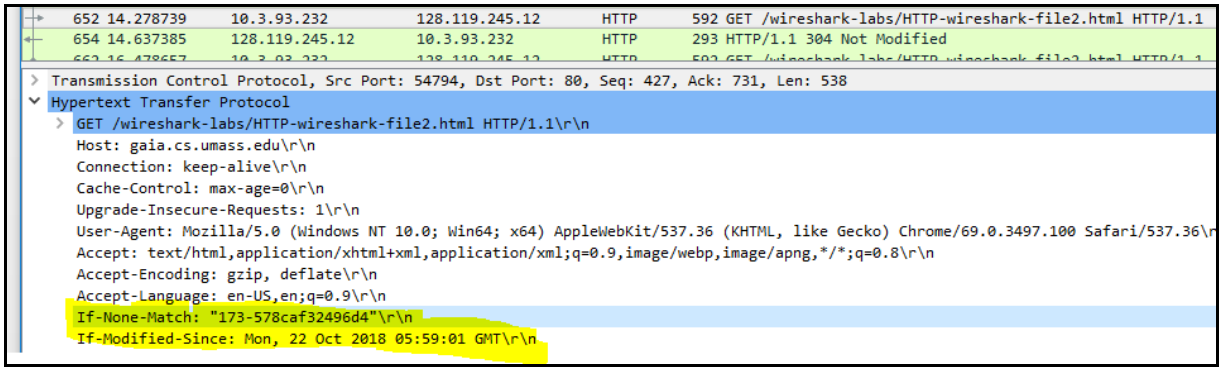
Aim of this exercise: We will now learn about a variant of the HTTP GET request message that we've seen earlier. We will note how the HTTP CONDITIONAL GET request and the reply to such a request differs from a simple HTTP GET request. Before performing the steps below, make sure your browser's cache is empty. (To do this under Firefox, select *Tools->Clear Recent History* and check the Cache box, or for Internet Explorer, select *Tools->Internet Options->Delete File;* these actions will remove cached files from your browser's cache.)

The following indicate the steps for this experiment:

- Start up your web browser, and make sure your browser's cache is cleared, as discussed above.
- Start up the Wireshark packet sniffer
- Enter the following URL into your browser http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file2.html

  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" in the display-filter-specification window, so that only captured HTTP messages will be displayed later in the packet-listing window.
- Filter out all the non-HTTP packets and focus on the HTTP header information in the packet-header detail window.
- By looking at the information in the HTTP GET and response messages, answer the following questions:

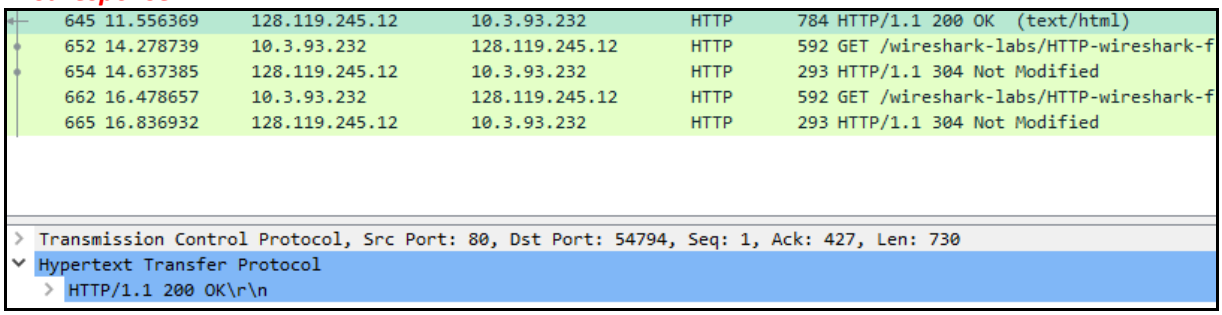
2.1 Inspect the contents of the first and 2nd HTTP GET requests from the browser to the server. Do you see "IF-MODIFIED-SINCE" and "IF-NONE-MATCH" lines in these HTTP GET message? Why?



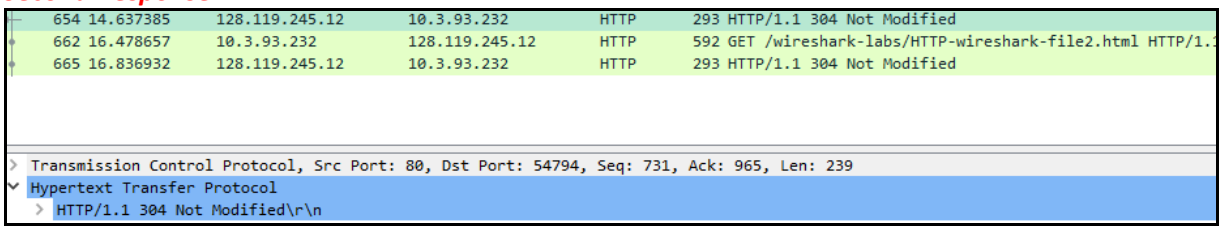
Conditional Get is sent to server to check if the content available in the cache is same as the contents availble in the host server. It sends conditional requests to check if there is need to retrieve data or not. "If-no-match" tells us that whether the data required is present in the server or not. Whereas, "If-modified-since" retrieves the time since the data was changed or modified.

2.2 What is the difference in first and second response received? What is the last modified time in the first response message?

First response:



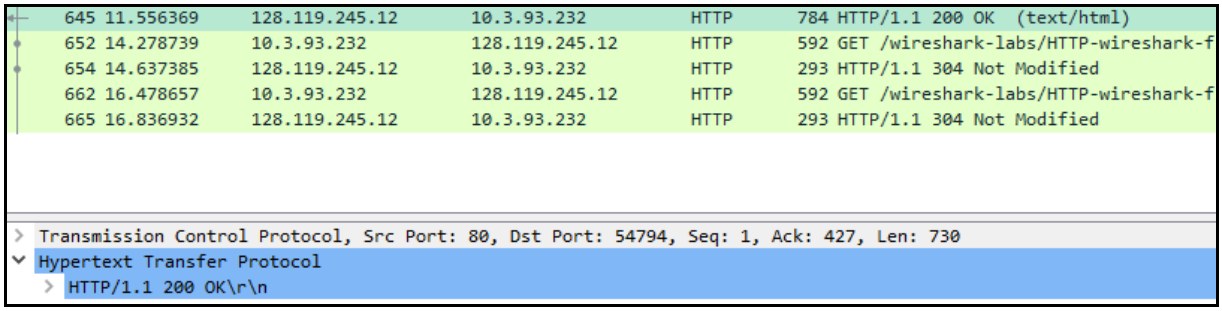
Second Response:



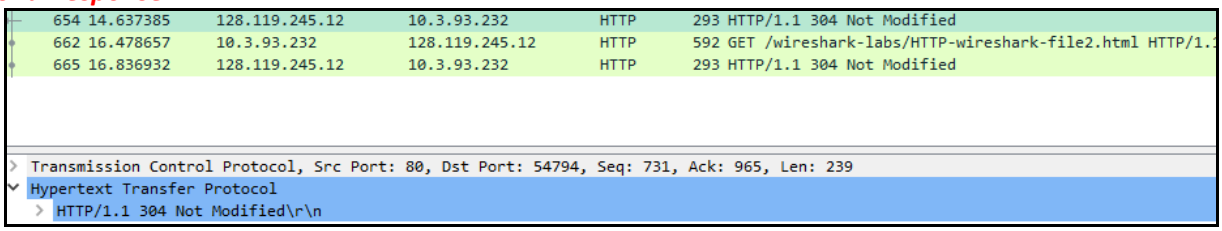
In first response received, client retrieved the entire data from the server because it was not present and it was a simple querry returning status code 200. Whereas, in second request, the response received was Not Modified instead of retrieving the entire data from server. And here we requested conditional get instead of data with status code to be 304.

2.3 What is the HTTP status code and phrase returned from the server in response to the first and second HTTP GET? Did the server explicitly return the contents of the file? Explain.

First response:

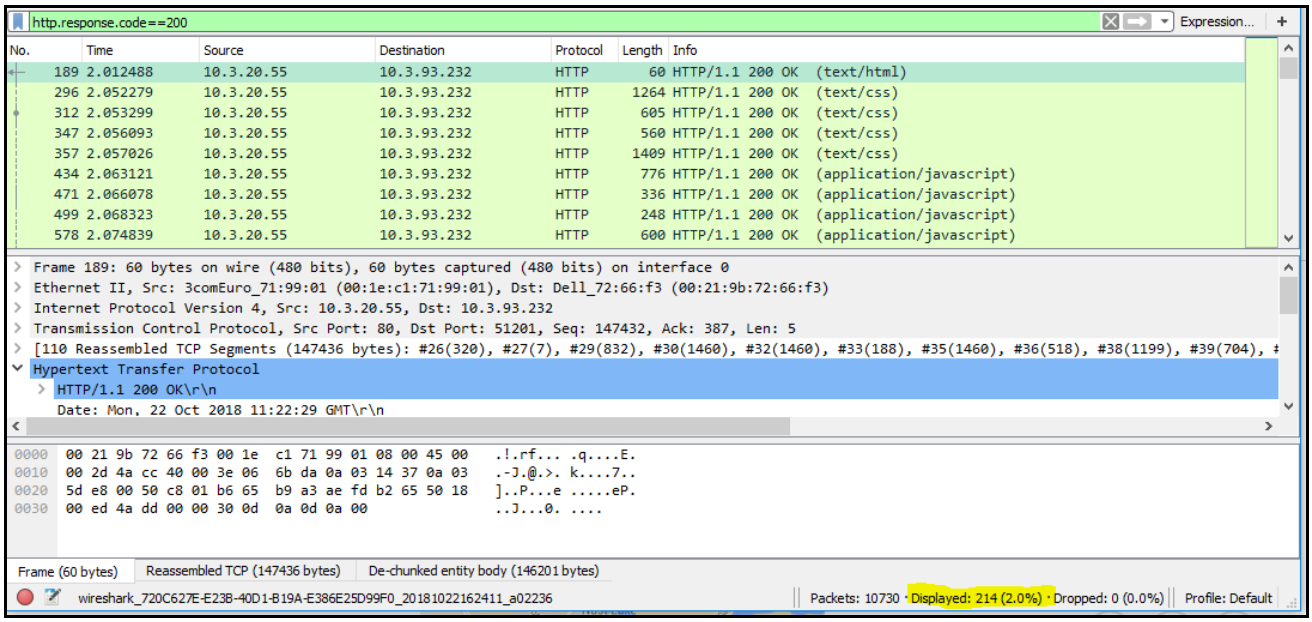


Second Response:



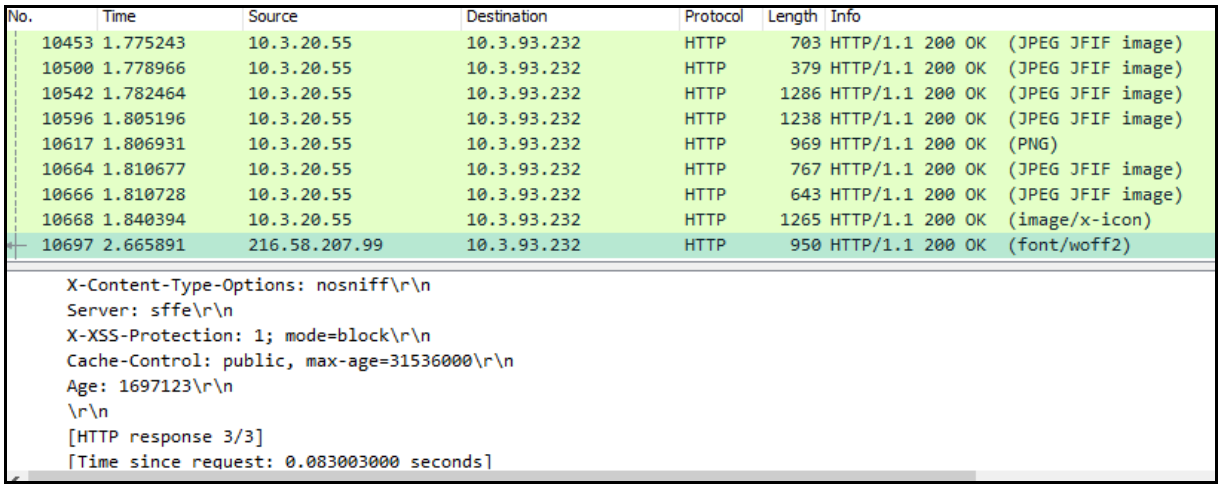
The first response returned from proxy server with status code 200 and phrase OK. Here, 200 states that the response was received because of a simple querry request. Whereas, In second response, our client sends conditional get to server instead of query to check the last modified date of the data which we have in cache. It returns status code 304 with phrase Not Modified. Browser did not explicitly received datat from server because it was not modified in the cache and the file was returned from the cache.

2.4 Empty your browser cache again and open the webpage www.seecs.edu.pk and capture the GET and OK response messages. How many total objects does the server return?



We used the wireshark filter to get the total number of get and response messages. Server returned 214 response messages showing that it sent 214 objects in our capture time.

2.5 What is the page load time (PLT) for the interaction in 2.4?



No.	Time	Source	Destination	Protocol	Length	Info
10453	1.775243	10.3.20.55	10.3.93.232	HTTP	703	HTTP/1.1 200 OK (JPEG JFIF image)
10500	1.778966	10.3.20.55	10.3.93.232	HTTP	379	HTTP/1.1 200 OK (JPEG JFIF image)
10542	1.782464	10.3.20.55	10.3.93.232	HTTP	1286	HTTP/1.1 200 OK (JPEG JFIF image)
10596	1.805196	10.3.20.55	10.3.93.232	HTTP	1238	HTTP/1.1 200 OK (JPEG JFIF image)
10617	1.806931	10.3.20.55	10.3.93.232	HTTP	969	HTTP/1.1 200 OK (PNG)
10664	1.810677	10.3.20.55	10.3.93.232	HTTP	767	HTTP/1.1 200 OK (JPEG JFIF image)
10666	1.810728	10.3.20.55	10.3.93.232	HTTP	643	HTTP/1.1 200 OK (JPEG JFIF image)
10668	1.840394	10.3.20.55	10.3.93.232	HTTP	1265	HTTP/1.1 200 OK (image/x-icon)
10697	2.665891	216.58.207.99	10.3.93.232	HTTP	950	HTTP/1.1 200 OK (font/woff2)

```
X-Content-Type-Options: nosniff\r\n
Server: sffe\r\n
X-XSS-Protection: 1; mode=block\r\n
Cache-Control: public, max-age=31536000\r\n
Age: 1697123\r\n
\r\n
[HTTP response 3/3]
[Time since request: 0.083003000 seconds]
```

Here we used wiresharks functionality of sorting by pressing ctrl+t. Through this every response time is added in consecutive query displaying the total time in the last response. As seen from the picture, the total time for page loading or PLT of the website was 2.6 seconds.