

# Wireshark HTTP lab quiz

**Due** Mar 20 at 17:00

**Points** 11

**Questions** 11

**Time Limit** None

**Allowed Attempts** Unlimited

[Take the Quiz Again](#)

## Attempt History

	Attempt	Time	Score
LATEST	<a href="#">Attempt 3</a>	1 minute	11 out of 11
	<a href="#">Attempt 2</a>	4 minutes	11 out of 11
	<a href="#">Attempt 1</a>	92 minutes	10 out of 11

Score for this attempt: **11** out of 11

Submitted Mar 20 at 1:47

This attempt took 1 minute.

Now that we are more familiar with Wireshark and the lesson format, let's start exploring Web traffic. This lab will help you with your first practical, so complete it early.

As before you can attempt the quiz multiple times with your score being the average mark of your attempts. All correct answers are worth 1 mark and incorrect answers are worth 0 marks.

You don't have to finish the entire lab in one setting. You can resume the lab at a later time. Feel free to use the forums to discuss approaches to solving the questions or discuss any unclear points.

Now to look at HTTP.....

## **Part 1 of 6**

Adapted from Version: 6.0

(c) 2012 J.F. Kurose, K.W. Ross. All Rights Reserved

Having gotten our feet wet with the Wireshark packet sniffer in the introductory lab, we're now ready to use Wireshark 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, you might want to review Section 2.2 of the text.

### 1. The Basic HTTP GET/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:

1. Start up your web browser.
2. Start up the Wireshark packet sniffer, as described in the Introductory lab (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).
3. Wait a bit more than one minute (we'll see why shortly), and then begin Wireshark packet capture.

1. Enter the following to your browser

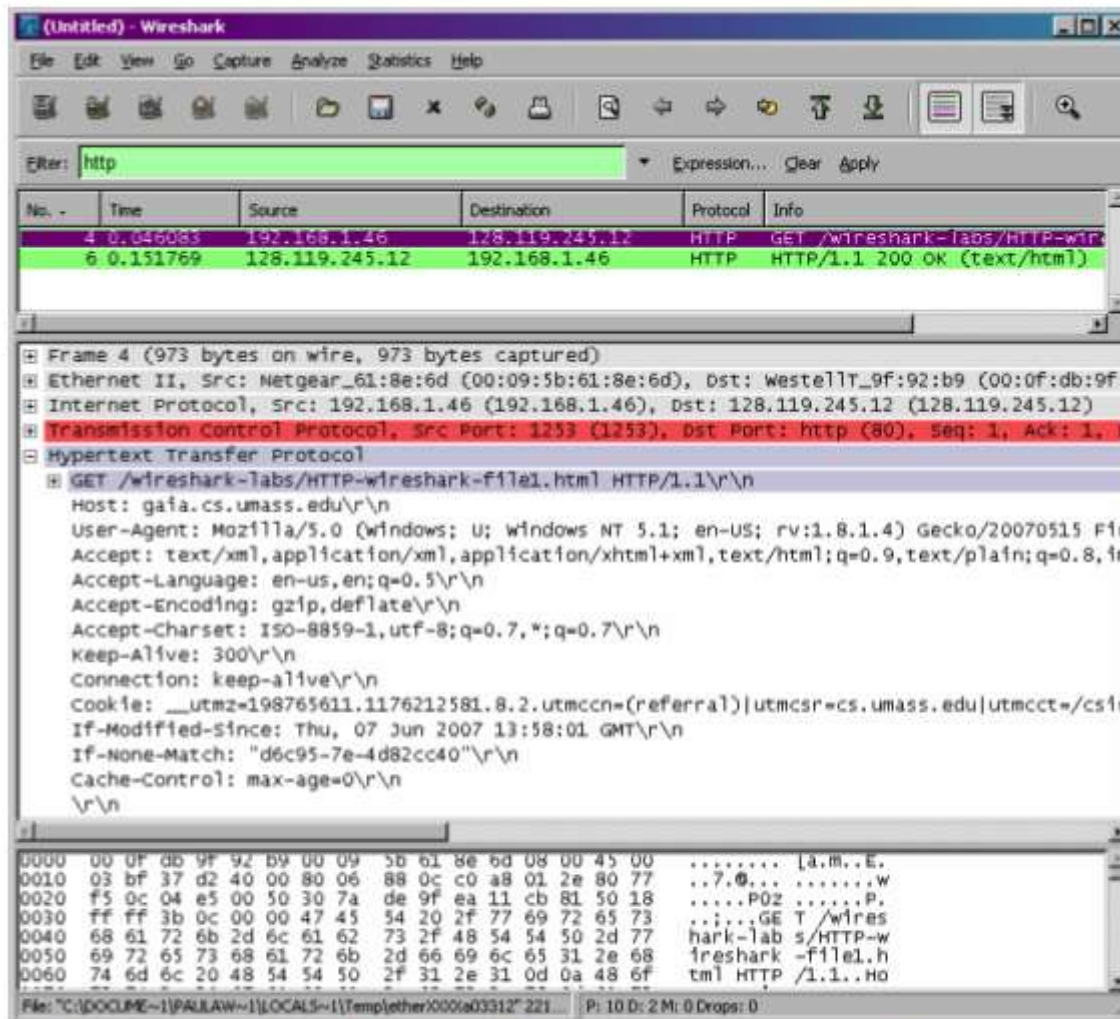
<http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html> ↗

(<http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html>)

Your browser should display the very simple, one-line HTML file

2. Stop Wireshark packet capture.

Your Wireshark window should look similar to the window shown in Figure 1. Note you may also see a proxy authentication request before the packets shown below.



## **Part 2 of 6**

Wireshark shows in the packet-listing window that two HTTP messages were captured:

1. the GET message (from your browser to the gaia.cs.umass.edu web server) and
2. 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. We want to minimize the amount of non-HTTP data displayed (we're interested in HTTP here, and will be investigating these other protocols in later labs), so make sure the boxes at the far left of the Frame, Ethernet, IP and TCP information have the arrow pointing to the right (which means there is hidden, undisplayed information), and the HTTP line has the arrow pointing down (which means that all information about the HTTP message is displayed).

(Note: You should ignore any HTTP GET and response for favicon.ico. If you see a reference to this file, it is your browser automatically asking the server if it (the server) has a small icon file that should be displayed next to the displayed URL in your browser. We'll ignore references to this pesky file in this lab.).

By looking at the information in the HTTP GET and response messages, answer the following questions.

Decide which packet (GET or OK) is being sent by your browser. Look at that packet's HTTP information. Which version of HTTP is your browser using?

Correct!

1.1

Most modern browsers and servers run version 1.1 to take advantage of persistence and pipelining or 2.0 if they find a 2.0 server.

Correct Answers

HTTP/1.1

1.1

Most modern browsers support HTTP 2.0, but you would need to use the http2 filter to see these in wireshark. The gaia server is a 1.1 server so the browser would use 1.1 to talk to it.

HTTP 2.0 traffic is encrypted, so to view it in wireshark you need to configure wireshark to know where to find the decryption keys.

## Question 2

1 / 1 pts

What [status code](https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html) was returned to your browser for the successful HTTP GET request? (make sure you know what a [status code](https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html) is and what status codes mean before answering.)

Correct!

200

**Question 3**

1 / 1 pts

How many bytes of content were returned to your browser in response to the request?

**Correct!**☒ 128☐ 628☐ 1024☐ 152**Question 4**

1 / 1 pts



**Part 3 of 6**

Have a look at when the HTML file that you are retrieving was last modified at the server? You may be surprised to find that the document you just retrieved was last modified within a minute before you downloaded the document. That's because (for this particular file), the `gaia.cs.umass.edu` server is setting the file's last-modified time to be the current time, and is doing so once per minute. Thus, if you wait a minute between accesses, the file will appear to have been recently modified, and hence

your browser will download a "new" copy of the document.

**Most** web browsers perform object caching and thus perform a **conditional** GET when retrieving an HTTP object. Before performing the steps below, make sure your browser's cache is empty.

Using **Firefox**, open the menu by selecting the menu icon from the top right of your browser. Select Preferences. Select Privacy from the left hand menu. Choose: Clear Recent History; these actions will remove cached files from your browser's cache. Now do the following:

- 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>  [\\_ \(http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file2.html\)](http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file2.html) Your browser should display a very simple five-line HTML file.
- Quickly select the refresh button  on your browser window displaying the HTML file.
- 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.

Answer the following questions...

The first HTTP request include an IF-MODIFIED-SINCE header?

☐ True

☒ False

Correct!

The page was not in the cache for the first request, so the client did not add an IF-MODIFIED-SINCE header.



### Question 5

1 / 1 pts

The **second** HTTP GET contains an "IF-MODIFIED-SINCE:" header.

Note that you may see HTTP GET and replies other than the ones you send/receive from gaia. Look at the sender and receiver address to make sure you are matching the correct request/response.

Correct!

☒ True

This time the page is cached in the browser, so it only wants the page if it has changed.

☐ False

### Question 6

1 / 1 pts

What HTTP **status code** is returned from the server in response to an IF-MODIFIED-SINCE HTTP GET when the file has **not** been modified since the specified time?

If you're not sure what the status code is, look in the RFC.

Correct!

304

## Question 7

1 / 1 pts

The aim of caching is two fold: 1) improving speed to display page and 2) reducing demand on the network and server.

Which of the following are **sufficient on their own** (ie require nothing else to be true about the request or the requested resource) to cause a server to **not** return the web page in response to a client GET request?

☐ The client specifies an IF-MODIFIED-SINCE in the header.

Correct!

☒ None of these

A server does not keep state about what it has previously sent to clients. This is why "cookies" are stored on clients so that browsers can identify them when returning to a site.

So having previously sent the page is not on it's own sufficient to stop the server resending.

Just specifying an IF-MODIFIED-SINCE on its own is not sufficient to stop the server sending the page. If the page has been modified since the specified date, the server will send the page.

HTTP 1.1 accepts HTTP 1.0 commands

☐ The client is using HTTP 1.0 and the server is running HTTP 1.1

- ☐ When the server has previously sent the page to the client.

## Question 8

1 / 1 pts

### Part 5 of 6

Now let's look at what happens when your browser downloads a file with embedded objects, i.e., a file that includes other objects (in the example below, image files) that are stored on another server(s).

Do the following:

- Start up your web browser, and make sure your browser's cache is cleared (selecting History->Clear Recent History from the Firefox menu)
- Start up the Wireshark packet sniffer
- Enter the following URL into your browser

<http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file4.html> 

[\(http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file4.html\)](http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file4.html)

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 logos from the indicated web sites. Our publisher's logo is retrieved from the [www.aw-bc.com](http://www.aw-bc.com) web site. The image of our book's cover is stored at the [manic.cs.umass.edu](http://manic.cs.umass.edu) server.

- Stop Wireshark packet capture, and enter "http" in the display-filter-specification window, so that only captured HTTP messages will be displayed.

How many HTTP GET request messages, ignoring any favorite icon requests, were sent by your browser?

**Ignore** any requests that do not result in a 200 OK (ie proxy authentication, moved resources)

Correct!

3

Correct Answers

3

### Question 9

1 / 1 pts

Review your notes on **pipelining** and **persistence** in HTTP. Look carefully at the sequence of requests and their contents and in particular the **Host** header field in the HTTP requests which indicates where the request is being sent to. Which of the following is true of this particular sequence of requests to download wireshark-file4.html and its embedded contents?

☐ The client is able to take advantage of both pipelining and persistence

Correct!

☒ The client is able to take advantage of persistence but not pipelining

We can see in the header that the Connection: header is set to keep-alive (although the default in HTTP 1.1 is persistent connections even if this is not specified). However, we can only take advantage of persistence and pipelining if the files reside on the same host. In this particular download 2 files requested reside on 1 host and 1 resides on a different host.

We can only take advantage of persistence for the first two files that are on the same host. We can not take advantage of pipelining as we have to download the web page before we know that the images will need to be downloaded. So we can't send the image request until the web page is returned.

- ☐ The client is able to take advantage of pipelining but not persistence
- ☐ The client is able to take advantage of neither pipelining nor persistence

### Question 10

1 / 1 pts

Assume in your wireshark capture that all three files could be accessed from the same server. If the first web page takes 300 ms to download, the first image takes 2 seconds to download and the second image takes 300 milliseconds to download and it takes 300 milliseconds to establish a TCP connection between the client and the server.

How many **seconds** would be saved downloading this web page by using a persistent connection (without pipelining) compared to using neither persistence or pipelining?

Correct!

0.6

There will be 1 TCP connection, followed by three sequential downloads if persistent connections are used compared to 3 TCP connections (1 for each download).

**Correct Answers**

- 0.6
- .6 seconds
- 0.6 seconds
- .6

**Question 11**

1 / 1 pts

Assume in your wireshark capture that again the three files are on the same server and the first web page takes 300 ms to download, the first image takes 2 seconds to download and the second image takes 300 milliseconds to download and it takes 300 milliseconds to establish a TCP connection between the client and the server.

How many **seconds** would be saved downloading this web page by using a persistent connection with pipelining compared to using neither persistence or pipelining?

In order to answer this question, you will need to make some assumptions (which are true in this case):

- The web page itself is very small so the page is fully downloaded by the time the browser reads the embedded object links and that the time between reading the two embedded links is insignificant.
- You should also assume that the queuing delays are negligible - ie propagation delay and transmission delay dominate the time needed to transfer the files.

- Downloading the image files from the same server has a negligible effect on their individual download times
- The time needed to send a request to the web server is 150 milliseconds

Correct!

0.75

There will be one TCP connection (300 ms instead of 900ms for 3 connections), the download times are unchanged. The request for the second image is sent while the first image is downloading so the 150ms to send the second image request, overlaps with the downloading of the first image. Compared to the no pipelining or persistence, this is a saving of .75 seconds (nearly a second)

Correct Answers

0.75 seconds

.75

0.75

.75 seconds

You have seen all questions. If you skipped any questions, go back to those sections and answer them.

Once you click 'Submit Quiz' on this page, the lesson will close and your grade will be recorded.

Quiz Score: **11** out of 11