



TAIBAH UNIVERSITY



College of Computer Science and Engineering

Computer Engineering Department

COE332
Computer Networks
Student's Lab Manual
V7

Prepared by:

Dr. Mohamed ZAYED Dr. Ahmed ABDELMONEM
Dr. Abdullah AL BINALI

Lab03

Student Name: Norah Fahad Aloufi

Student ID: 4050772

Section: C8C **Group:**

Session (Fall / Spring / Summer): 14/02/2022

Lab-3: HTTP Protocol

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.⁴

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.
4. 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.
5. Stop Wireshark packet capture.

Your Wireshark window should look similar to the window shown in Figure 1. If you are unable to run Wireshark on a live network connection, you can download a packet trace that was created when the steps above were followed.⁵

⁴ References to figures and sections are for the 8th edition of our text, *Computer Networks, A Top-down Approach*, 8th ed., J.F. Kurose and K.W. Ross, Addison-Wesley/Pearson, 2020.

⁵ Download the zip file <http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip> and extract the file http-ethereal-trace-1. The traces in this zip file were collected by Wireshark running on one of the author's computers, while performing the steps indicated in the Wireshark lab. Once you have downloaded the trace, you can load it into Wireshark and view the trace using the *File* pull down menu, choosing *Open*, and then selecting the http-ethereal-trace-1 trace file. The resulting display should look similar to Figure 1. (The Wireshark user interface displays just a bit differently on different operating systems, and in different versions of Wireshark).

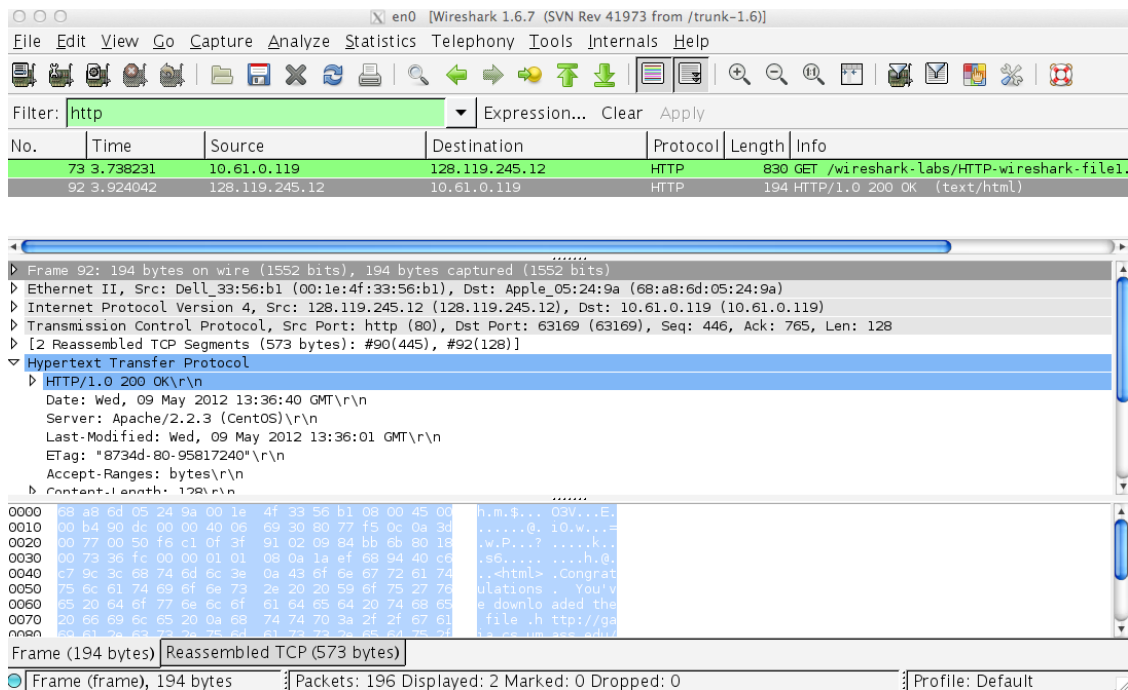


Figure 1: Wireshark Display after `http://gaia.cs.umass.edu/wireshark-labs/ HTTP-wireshark-file1.html` has been retrieved by your browser

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 OK 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 a plus sign or a right-pointing triangle (which means there is hidden, undisplayed information), and the HTTP line has a minus sign or a down-pointing triangle (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. When answering the following questions, you should print out the GET and response messages (see the introductory Wireshark lab for an explanation of how to do this) and indicate where in the message you've found the information that answers the following questions. When you hand in your assignment, annotate the output so that it's clear where in the output you're getting the information for your answer (e.g., for our classes, we ask that students markup paper copies with a pen, or annotate electronic copies with text in a colored font).

1. Is your browser running HTTP version 1.0 or 1.1? What version of HTTP is the server running?

→ last Pages

2. What languages (if any) does your browser indicate that it can accept to the server?
3. What is the IP address of your computer? Of the `gaia.cs.umass.edu` server?
4. What is the status code returned from the server to your browser?
5. When was the HTML file that you are retrieving last modified at the server?
6. How many bytes of content are being returned to your browser?
7. By inspecting the raw data in the packet content window, do you see any headers within the data that are not displayed in the packet-listing window? If so, name one.

In your answer to question 5 above, you might have been 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.

2. Retrieving Long Documents

In our examples thus far, the documents retrieved have been simple and short HTML files. Let's next see what happens when we download a long HTML file. 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-file3.html>
 Your browser should display the rather lengthy US Bill of Rights.
- Stop Wireshark packet capture, and enter "http" in the display-filter-specification window, so that only captured HTTP messages will be displayed.
- (Note: If you are unable to run Wireshark on a live network connection, you can use the http-ethereal-trace-3 packet trace to answer the questions below; see footnote 1. This trace file was gathered while performing the steps above on one of the author's computers.)

In the packet-listing window, you should see your HTTP GET message, followed by a multiple-packet TCP response to your HTTP GET request. This multiple-packet response deserves a bit of explanation. Recall from Section 2.2 (see Figure 2.9 in the text) that the HTTP response message consists of a status line, followed by header lines, followed by a blank line, followed by the entity body. In the case of our HTTP GET, the entity body in the response is the *entire* requested HTML file. In our case here, the HTML file is rather long, and at 4500 bytes is too large to fit in one TCP packet. The single HTTP response message is thus broken into several pieces by TCP, with each piece being contained within a separate TCP segment (see Figure 1.24 in the text). In recent versions of Wireshark, Wireshark indicates each TCP segment as a separate packet, and the fact that the single HTTP response was fragmented across multiple TCP packets is indicated by the "TCP segment of a reassembled PDU" in the Info column of the Wireshark display. Earlier versions of Wireshark used the "Continuation" phrase to indicate that the entire content of an HTTP message was broken across multiple TCP segments.. We stress here that there is no "Continuation" message in HTTP!

Answer the following questions:

1. How many HTTP GET request messages did your browser send? Which packet number in the trace contains the GET message for the Bill of Rights?
2. Which packet number in the trace contains the status code and phrase associated with the response to the HTTP GET request?

→ Last Pages

3. What is the status code and phrase in the response?
4. How many data-containing TCP segments were needed to carry the single HTTP response and the text of the Bill of Rights?

PART I: -

1. Is your browser running HTTP version 1.0 or 1.1? What version of HTTP is the server running? [HTTP 1.1](#)

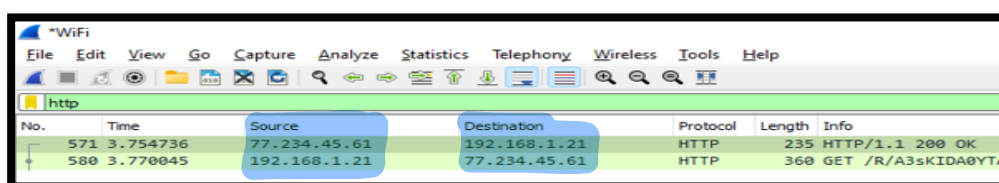
```
√ Hypertext Transfer Protocol
  > GET /R/A3sKIDA0YTAMzE3MG40DRjNTFhODRlMDA5M2UxMjhmWE0EgQAFAlGNIBgECKggIBBC
    > [Expert Info (Chat/Sequence): GET /R/A3sKIDA0YTAMzE3MG40DRjNTFhODRlMDA5M2UxMjhmWE0EgQAFAlGNIBgECKggIBBC]
      Request Method: GET
      Request URI: /R/A3sKIDA0YTAMzE3MG40DRjNTFhODRlMDA5M2UxMjhmWE0EgQAFAlGNIBgECKggIBBC
      Request Version: HTTP/1.1
      Host: su.ff.avast.com\r\n
      Accept: */*\r\n
      Content-Type: application/octet-stream\r\n
      Pragma: no-cache\r\n
```

```
√ 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
      Content-Type: application/octet-stream\r\n
```

2. What languages (if any) does your browser indicate that it can accept to the server? [ar, en-US](#)

```
√ Hypertext Transfer Protocol
  > GET /wireshark-labs/HTTP-wireshark-file1.html 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 (Windows NT 10.0; Win64; x64)
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
    Accept-Encoding: gzip, deflate\r\n
    Accept-Language: ar, en-US;q=0.9, en;q=0.8\r\n
    If-None-Match: "80-5d7f4f38dc3ba"\r\n
    If-Modified-Since: Mon, 14 Feb 2022 06:59:01 GMT\r\n
    \r\n
    [Full request URI: http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html]
    [HTTP request 1/1]
    [Response in frame: 1945]
```

3. What is the IP address of your computer? Of the gaia.cs.umass.edu server?



No.	Time	Source	Destination	Protocol	Length	Info
571	3.754736	77.234.45.61	192.168.1.21	HTTP	235	HTTP/1.1 200 OK
580	3.770045	192.168.1.21	77.234.45.61	HTTP	360	GET /R/A3sKIDA0YTAMzE3MG40DRjNTFhODRlMDA5M2UxMjhmWE0EgQAFAlGNIBgECKggIBBC

4. What is the status code returned from the server to your browser? [200](#)

```
> Frame 571: 235 bytes on wire (1880 bits), 235 bytes captured (1880 bits) on interface \Device\NPF{...}
> Ethernet II, Src: HuaweiTe_8c:26:c2 (2c:97:b1:8c:26:c2), Dst: IntelCor_78:63:29 (04:d3:b7:78:63:29)
> Internet Protocol Version 4, Src: 77.234.45.61, Dst: 192.168.1.21
> Transmission Control Protocol, Src Port: 80, Dst Port: 50748, Seq: 1, Ack: 1, Len: 181
√ 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
      Content-Type: application/octet-stream\r\n
    > Content-Length: 35\r\n
      Pragma: no-cache\r\n
      Cache-control: no-cache\r\n
      Connection: keep-alive\r\n
```

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

```
▼ Hypertext Transfer Protocol
  > GET /wireshark-labs/HTTP-wireshark-file1.html HTTP/1.1\r\n
    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 (Windows NT 10.0; Win64; x64) App
    Accept: text/html,application/xhtml+xml,application/xml;q
    Accept-Encoding: gzip, deflate\r\n
    Accept-Language: ar,en-US;q=0.9,en;q=0.8\r\n
    If-None-Match: "80-5d7f4f38dc3ba"\r\n
    If-Modified-Since: Mon, 14 Feb 2022 06:59:01 GMT\r\n
    \r\n
    [Full request URI: http://gaia.cs.umass.edu/wireshark-lab
    HTTP/1.1 200 OK]
```

6. How many bytes of content are being returned to your browser? 35 bytes

```
▼ 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
      Content-Type: application/octet-stream\r\n
      ▼ Content-Length: 35\r\n
        [Content length: 35]
      Pragma: no-cache\r\n
      Cache-control: no-cache\r\n
      Connection: keep-alive\r\n
      \r\n
      [HTTP response 1/2]
      [Next request in frame: 580]
```

7. By inspecting the raw data in the packet content window, do you see any headers within the data that are not displayed in the packet-listing window? If so, name one. NO

PART II: -

1. How many HTTP GET request messages did your browser send? Which packet number in the trace contains the GET message for the Bill or Rights?

1 HTTP GET request to the server. The Packet number 629.

No.	Time	Source	Destination	Protocol	Length	Info
629	7.392105	192.168.1.21	128.119.245.12	HTTP	538	GET /wireshark-labs/HTTP-wireshark-file3.html HTTP/1.1
654	7.557588	128.119.245.12	192.168.1.21	HTTP	679	HTTP/1.1 200 OK (text/html)

2. Which packet number in the trace contains the status code and phrase associated with the response to the HTTP GET request?

The Packet number 654.

No.	Time	Source	Destination	Protocol	Length	Info
629	7.392105	192.168.1.21	128.119.245.12	HTTP	538	GET /wireshark-labs/HTTP-wireshark-file3.html HTTP/1.1
654	7.557588	128.119.245.12	192.168.1.21	HTTP	679	HTTP/1.1 200 OK (text/html)

3. What is the status code and phrase in the response?

```
▼ 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
      [Status Code Description: OK]
      Response Phrase: OK → Phrase
```

4. How many data-containing TCP segments were needed to carry the single HTTP response and the text of the Bill of Rights?

```
> Frame 654: 679 bytes on wire (5432 bits), 679 bytes captured (5432 bits) on interface \Device\NPF_{9888...}
> Ethernet II, Src: HuaweiTe_8c:26:c2 (2c:97:b1:8c:26:c2), Dst: IntelCor_78:63:29 (04:d3:b0:78:63:29)
> Internet Protocol Version 4, Src: 128.119.245.12, Dst: 192.168.1.21
> Transmission Control Protocol, Src Port: 80, Dst Port: 59341, Seq: 4237, Ack: 485, Len: 625
▼ [4 Reassembled TCP Segments (4861 bytes): #649(1412), #650(1412), #653(1412), #654(625)]
  [Frame: 649, payload: 0-1411 (1412 bytes)]
  [Frame: 650, payload: 1412-2823 (1412 bytes)]
  [Frame: 653, payload: 2824-4235 (1412 bytes)]
  [Frame: 654, payload: 4236-4860 (625 bytes)]
  [Segment count: 4]
  [Reassembled TCP length: 4861]
  [Reassembled TCP Data: 485454502f312e3120323030204f4b0d0a4461746553a204d6f6e2c203134204665622032...]
> Hypertext Transfer Protocol
```