Assignment 1

The Basic HTTP GET/response interaction

Task A:

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

   Our browser is running HTTP 1.1, and the server is also running HTTP 1.1

   ```
   Request Version: HTTP/1.1      Response Version: HTTP/1.1
   ```

2. What languages (if any) does your browser indicate that it can accept to the server? In the captured session, what other information (if any) does the browser provide the server with regarding the user/browser?

   It allows swedish, we found the answer under the HTTP-header accept language.

   ```
   Accept-Language: sv-SE,sv;q=0.9\r\n
   ```

   And we get the browsers properties under HTTP-header user agent

   ```
   User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.0.0 Safari/537.36\r\n
   ```

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

   Ours: 10.241.230.90, Server: 128.119.245.12

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

   We got the status code 200

   ```
   HTTP/1.1 200 OK\r\n
   [Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
       [HTTP/1.1 200 OK\r\n]
       [Severity level: Chat]
       [Group: Sequence]
   Response Version: HTTP/1.1
   Status Code: 200
   [Status Code Description: OK]
   Response Phrase: OK
   ```

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

   05:59 todays date

   ```
   Last-Modified: Tue, 02 Apr 2024 05:59:01 GMT\r\n
   ```

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

   128 bytes

   ```
   File Data: 128 bytes
   ```

7. By inspecting the raw data in the packet content pane, do you see any HTTP headers within the data that are not displayed in the packet-listing window? If so, name one.

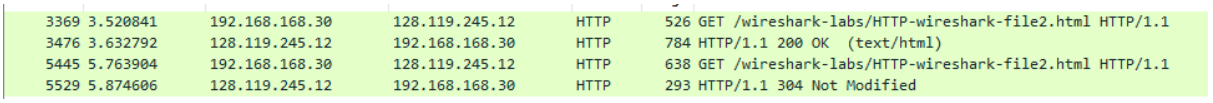
   No, there is no HTTP-header in the raw data that isn't in the packet-listing window.

Observations for Task A:

It is mostly about basic information regarding GET requests and HTTP headers. This section also brings up some good to know things from the headers that we have inspected like languages that are accepted and http versions running.

The HTTP CONDITIONAL GET/response interaction

Task B:



3369	3.520841	192.168.168.30	128.119.245.12	HTTP	526	GET /wireshark-labs/HTTP-wireshark-file2.html HTTP/1.1
3476	3.632792	128.119.245.12	192.168.168.30	HTTP	784	HTTP/1.1 200 OK (text/html)
5445	5.763904	192.168.168.30	128.119.245.12	HTTP	638	GET /wireshark-labs/HTTP-wireshark-file2.html HTTP/1.1
5529	5.874606	128.119.245.12	192.168.168.30	HTTP	293	HTTP/1.1 304 Not Modified

1. Inspect the contents of the first HTTP GET request from your browser to the server. Do you see an "IF-MODIFIED-SINCE" line in the HTTP GET?

   No

2. Inspect the contents of the server response. Did the server explicitly return the contents of the file? How can you tell?

   Yes, we could tell by inspecting the raw data that the first response from the server explicitly returned the contents of the file.

3. 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?

Yes, the information that followed was the date, day of the week and timezone.

```
If-Modified-Since: Tue, 02 Apr 2024 05:59:01 GMT\r\n
```

4. 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.

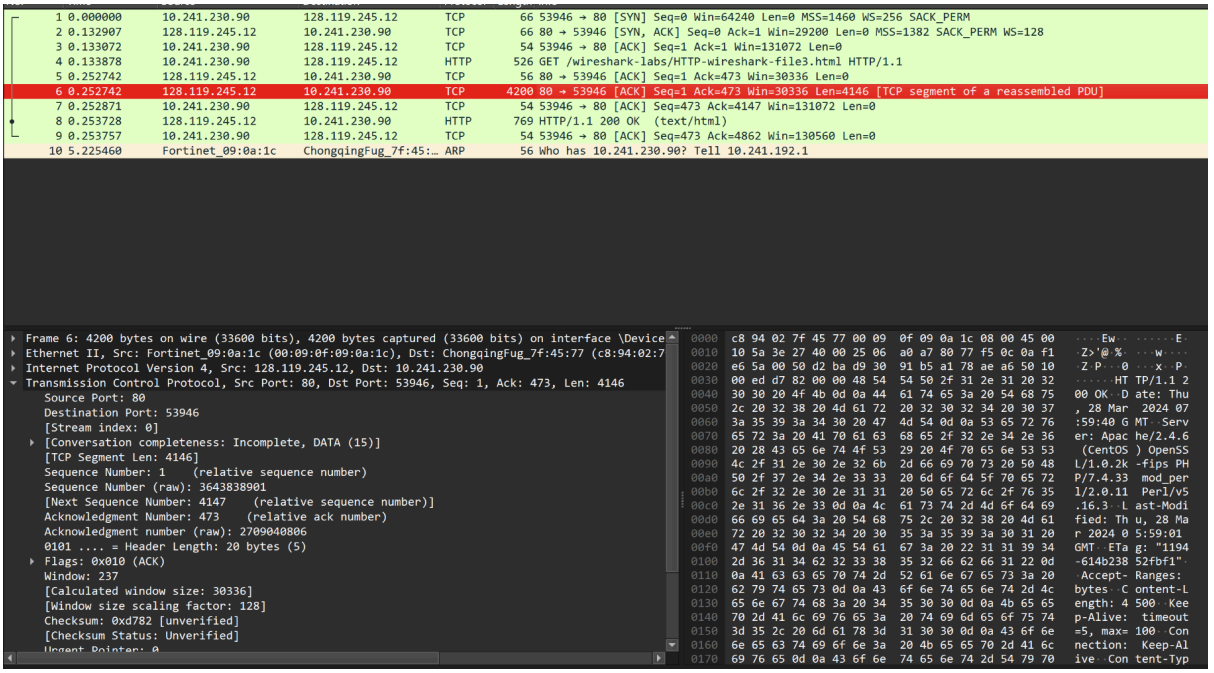
   The status code and phrase returned from the server was in response to the second HTTP GET was 304 and “not modified”. The server did not explicitly return the contents of the file, we could see this by looking at the raw data. This occurred because our computer already had it cached.

Observations for Task B:

We could see from this task what happens when we send another GET request to a server for a html site that hasn't changed since we sent our previous GET request, we get the response 304 not modified and the contents of the file are not sent because our computer already has it cached. We saw that we could also see when the file was last modified or updated by looking at the “IF-MODIFIED-SINCE:” header.

Retrieving Long Documents

Task C:



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?

   The browser sent one HTTP GET message and it was packet number 4

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

   It was packet number eight and the status code and phrase for the response was 200 OK

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

   4 tcp segments

```
[4 Reassembled TCP Segments (4861 bytes): #4947(1420), #4948(1420), #4949(1420), #4950(601)]
    [Frame: 4947, payload: 0-1419 (1420 bytes)]
    [Frame: 4948, payload: 1420-2839 (1420 bytes)]
    [Frame: 4949, payload: 2840-4259 (1420 bytes)]
    [Frame: 4950, payload: 4260-4860 (601 bytes)]
    [Segment count: 4]
    [Reassembled TCP length: 4861]
    [Reassembled TCP Data [truncated]: 485454502f312e3120323030204f4b0d0a446174653a205475652c20303220417072203230323420313333a31383a303
```

4. Is there any HTTP header information in the transmitted data associated with TCP segmentation? For this question you may want to think about at what layer each protocol operates, and how the protocols at the different layers interoperate.

   No there was none. The tcp packets only contain data of the file. This is because HTTP operates in the application layer (High-level protocols) and tcp in the transport layer. Since they are operating in different layers they don't have the same data.

Observations for Task C:

We could see in this task that when we download longer html files we still only need and receive one GET message but the packets are sent in multiple segments and in our case 4 segments. We could also clearly see that the data is handled in separate layers, for instance the transportation of data is handled by the transportation layer and hypertext text transfer protocol operates in the application layer, a higher layer then where TCP operates.

HTML Documents with Embedded Objects

Task D:

1. How many HTTP GET request messages were sent by your browser? To which Internet addresses were these GET requests sent?

Three GET messages were sent by the browser, one for the url of the page and one for each of the images displayed on the page. The GET requests were sent to different internet addresses as you can see below, the IP address for the GET request sent for the jpeg image is different from the website and the png image.

37	6.806579	10.241.230.90	128.119.245.12	HTTP	526 GET /wireshark-labs/HTTP-wireshark-file4.html HTTP/1.1
39	6.928542	128.119.245.12	10.241.230.90	HTTP	1355 HTTP/1.1 200 OK (text/html)
40	6.935454	10.241.230.90	128.119.245.12	HTTP	472 GET /pearson.png HTTP/1.1
50	7.054613	128.119.245.12	10.241.230.90	HTTP	901 HTTP/1.1 200 OK (PNG)
57	7.282692	10.241.230.90	178.79.137.164	HTTP	439 GET /8E_cover_small.jpg HTTP/1.1
59	7.311757	178.79.137.164	10.241.230.90	HTTP	225 HTTP/1.1 301 Moved Permanently
197	7.705434	10.241.230.90	128.119.245.12	HTTP	472 GET /favicon.ico HTTP/1.1
198	7.824629	128.119.245.12	10.241.230.90	HTTP	538 HTTP/1.1 404 Not Found (text/html)

2. Can you tell whether your browser downloaded the two images serially, or whether they were downloaded from the two websites in parallel? Explain.

The browser downloaded the images serially, we could see this by inspecting the HTTP messages. The first image's response message is obtained (200, OK), before the second image's GET message from the browser is sent.

```
HTTP      472 GET /pearson.png HTTP/1.1
HTTP      901 HTTP/1.1 200 OK  (PNG)
HTTP      439 GET /8E_cover_small.jpg HTTP/1.1
```

Observations for Task D:

We can see from task D that it is possible to obtain files from different sites and different servers, everything doesn't need to be in the same place. We also quickly saw in this case that the images were sent serially by looking in the order of the HTTP messages sent for and from the files.

HTTP Authentication

Task E:

1. What is the server's response (status code and phrase) in response to the initial HTTP GET message from your browser?

The server's response code was 401 and the message was "Unauthorized". This occurred because we haven't logged in yet.

No.	Time	Source	Destination	Protocol	Length	Info
36	4.018730	10.241.230.90	128.119.245.12	HTTP	542	GET /wireshark-labs/protected_pages/HTTP-wireshark-file5.html HTTP/1.1
38	4.138915	128.119.245.12	10.241.230.90	HTTP	771	HTTP/1.1 401 Unauthorized (text/html)
51	14.710354	10.241.230.90	128.119.245.12	HTTP	627	GET /wireshark-labs/protected_pages/HTTP-wireshark-file5.html HTTP/1.1
53	14.830485	128.119.245.12	10.241.230.90	HTTP	544	HTTP/1.1 200 OK (text/html)
55	14.878864	10.241.230.90	128.119.245.12	HTTP	488	GET /favicon.ico HTTP/1.1
56	14.998584	128.119.245.12	10.241.230.90	HTTP	538	HTTP/1.1 404 Not Found (text/html)

2. When your browser sends the HTTP GET message for the second time, what new field is included in the HTTP GET message?

The new field that was included was authorization.

Observations for Task E:

In task E we have viewed the HTTP messages that are sent when authorization for a website is necessary, firstly when the first GET message is sent we are not given access and are initially unauthorized. When we then however type in the password and username for the site we send another GET message and we get the HTTP response code 200 OK and we are granted access to the site. We also could see that a new field was included when we sent the GET message for the second time and that field was authorization for the site.

HTTP persistent connection

1. What does the "Connection: close" and "Connection: keep-alive" header field imply in HTTP protocol? When should one be used over the other?

Keep-alive implies that the connection should remain open and close implies that the connection should close after we've got one response. So keep-alive is better for multiple requests over the same connection and close is better when it comes to just a single request and response over a connection.