Questions Lab1

Task A:

HTTP GET Request

```
No.     Time           Source                Destination           Protocol Length Info
   3555 2.330871       192.168.0.154         128.119.245.12        HTTP     550    GET /wireshark-labs/HTTP-wireshark-file1.html HTTP/
1.1
Frame 3555: 550 bytes on wire (4400 bits), 550 bytes captured (4400 bits) on interface \Device\NPF_{94F5A665-6727-471B-A3A4-
D8513BAB4439}, id 0
Ethernet II, Src: ASUSTekC_98:16:16 (24:4b:fe:98:16:16), Dst: D-LinkIn_4d:07:34 (90:8d:78:4d:07:34)
Internet Protocol Version 4, Src: 192.168.0.154, Dst: 128.119.245.12
Transmission Control Protocol, Src Port: 63108, Dst Port: 80, Seq: 1, Ack: 1, Len: 496
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
    Upgrade-Insecure-Requests: 1\r\n
    User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.90 Safari/537.36\
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-
exchange;v=b3;q=0.9\r\n
    Accept-Encoding: gzip, deflate\r\n
    Accept-Language: sv-SE,sv;q=0.9,en-US;q=0.8,en;q=0.7\r\n
    \r\n
    [Full request URI: http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html]
    [HTTP request 1/2]
    [Response in frame: 3593]
    [Next request in frame: 3624]
```

HTTP Response

```
No.     Time           Source                Destination           Protocol Length Info
   3593 2.450777       128.119.245.12        192.168.0.154         HTTP     540    HTTP/1.1 200 OK  (text/html)
Frame 3593: 540 bytes on wire (4320 bits), 540 bytes captured (4320 bits) on interface \Device\NPF_{94F5A665-6727-471B-A3A4-
D8513BAB4439}, id 0
Ethernet II, Src: D-LinkIn_4d:07:34 (90:8d:78:4d:07:34), Dst: ASUSTekC_98:16:16 (24:4b:fe:98:16:16)
Internet Protocol Version 4, Src: 128.119.245.12, Dst: 192.168.0.154
Transmission Control Protocol, Src Port: 80, Dst Port: 63108, Seq: 1, Ack: 497, Len: 486
Hypertext Transfer Protocol
    HTTP/1.1 200 OK\r\n
    Date: Wed, 31 Mar 2021 15:17:43 GMT\r\n
    Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/7.4.14 mod_perl/2.0.11 Perl/v5.16.3\r\n
    Last-Modified: Wed, 31 Mar 2021 05:59:01 GMT\r\n
    ETag: "80-5bececfbd6ee4"\r\n
    Accept-Ranges: bytes\r\n
    Content-Length: 128\r\n
    Keep-Alive: timeout=5, max=100\r\n
    Connection: Keep-Alive\r\n
    Content-Type: text/html; charset=UTF-8\r\n
    \r\n
    [HTTP response 1/2]
    [Time since request: 0.119906000 seconds]
    [Request in frame: 3555]
    [Next request in frame: 3624]
    [Next response in frame: 3659]
    [Request URI: http://gaia.cs.umass.edu/favicon.ico]
    File Data: 128 bytes
Line-based text data: text/html (4 lines)
```

1: Both server and browser are running HTTP/1.1. (marked with red on both pictures)
2: Our browser accepts sv-SE, en-US and en as languages. The browser also sends user-agent, accept-encoding, and which types of files it accepts from the server. (marked with light blue)
3: Our IP: 192.168.0.154, Server IP: 128.119.245.12. (marked with light green on both pics)
4: Status code: 200 OK (marked with red)
5: Last-Modified: Mon, 29 Mar 2021 05:59:01 GMT (marked with yellow)
6: 128 bytes (marked with purple)
7: No, we couldn't find any differences.

We see in both the request and response headers that both the server and browser use HTTP version 1.1. As for question 2, our browser indicates that it can accept Swedish and English (US/regular) as languages for the website. It also provides the information that it can receive a html page, images and some sort of application, which can be seen in the "Accept" field. It also states that acceptable encoding methods for downloading files are gzip and deflate. We have also observed that if you try to refresh the page within one minute, it gives status code 304 (not modified) instead of the regular 200 (OK), thus it won't give the Last-Modified field if we refresh too soon. Our IP address can be seen in the request message at the internet protocol tab, where our IP is 192.168.0.154 as source IP, and the destination IP of the server is 128.119.245.12. As mentioned before, we got status code 200 at first, but if we refresh the page, we instead get the 304 status code. The last modified field is located in the response message, namely under the HTTP tab. The number of bytes returned to our browser was 128, which can be seen in the field File data, under the HTTP tab. And finally, we couldn't see any differences between the raw data in the packet content pane and the packet-listing window

regarding the HTTP headers. This is because the method was GET, which means that all header parameters will therefore be included in the URL.

Task B:
8: No, no such line exists in the first GET request.
9: Yes, the server explicitly returned the contents of the file. This can be seen in the field "Line-Based Text Data" which is the same text as written on the webpage.
10: Yes, the second HTTP request message includes a IF-MODIFIED-SINCE line, which is the date and time of the first time we accessed the page.
11: The HTTP status code is 304 (not modified). The server didn't explicitly return the contents of the file since the browser already has accessed the webpage before. Therefore, it can retrieve the page from its cache since it hasn't been modified since we last got the contents of the page.

Since the cache is cleared in the beginning, the first GET request will not contain an IF-MODIFIED-SINCE line, because we haven't accessed the page before, therefore we can't compare with an earlier GET request regarding the contents of the page (if it has been changed or not). We can also see that the first response message contains the whole content of the file, since we don't have it stored on the cache, and we need to retrieve the full content of the web page. The second GET request, however, do have the IF-MODIFIED-SINCE line, this is because we have the page in the cache already, and we need to see if the contents have been changed since we last fetched the content. If it would've changed, we would have needed to get the full content of the page again, but since it wasn't modified, we can just retrieve it from the cache. The status code from the second response message was 304 (not modified) which tells us that since we last got the page content, the content hasn't been modified and we therefore can fetch the information from the cache instead of the server returning the contents of the page.

Task C:
12: One GET request, packet number 3442.
13: Packet number 3475, with the status code 200 and the phrase OK.
14: It took 4 TCP segments to carry the HTTP response and the text.
15: The information is present in the TCP field, associated with the transport layer, with source port 80 and destination port 54029.

Task C paragraph
Our browser sent one GET request but since the HTML page is too big for just one TCP segment it instead sends 4 segments that contain the page. The GET request is placed in packet number 3442. The packet that contains the status code and phrase regarding the GET request has number 3475, and states that the status code is 200, which tells us that everything went well, thus giving the phrase "OK" in the response. It took 4 TCP segments to carry the whole HTTP response and the text, which can be seen in the packet-listing window since there is, between the GET request and the response message, 4 TCP segments sent to the browser. Lastly, the information associated with TCP segmentation is present in the TCP field of the HTTP header information. This TCP field is associated with the transport layer, which states that the packets should travel from source port 80 to destination port 54029.

You can see in the packet-listing window that the first TCP segment has Seq = 1 and Len = 0, the second has also Seq = 1, since the second TCP segment has Seq = Seq+Len at the previous TCP segment. This segment has Len = 1460, and the third has Seq = 1461 and Len =

1460. Finally, the last segment has Seq = 2921 and Len = 1460. As we can see here, they are connected and carrying the HTML content.

Task D:
16: 3 GET requests, one for the HTML page, and one for each image. HTML: 128.119.245.12, pearson.png: 128.119.245.12, 8E_cover_small: 178.79.137.164.
17: The images were downloaded serially.

Task D paragraph
Our browser sent three GET requests; one for the HTML page and one for each image. We can see that the HTML page and one of the images (the first one) are located on the same server, and the second image is on another server, therefore the addresses differ. Since one of the images were located on another server, the status message, instead of 200, was 301 (permanently moved) which creates a Location field in the response with the full URL. Lastly, we can tell that the images were downloaded serially. You can see that they were downloaded serially on the time stamps (arrival time) of the GET-requests and the response messages. The following timestamps mentioned will only be denoted with the 100^{th} of a second, since every message happened on the same second. The first request message arrived at .22, and the response for that request message arrived at .34. Furthermore, the GET request for the second image arrived at .46 and the following GET response arrived at .50. This means that the time windows, between the request and response, was NOT overlapping, thus they were done serially (not parallell).

Task E:
18: Status code: 401, phrase: Unauthorized.
19: Authorization, with Credentials under.

Task E paragraph
The first response has the status code 401, which is associated with the phrase Unauthorized. This is because we haven't filled in the credentials yet, so the page is still protected. This changes at the second response, when we have filled in the correct credentials and reached the "secret page". In the second response, we get HTTP status code 200, OK, which also comes with a new field; namely Authorization. The Authorization field tells which encoding format (basic) and the resulting encoded string. The subfield to Authorization is Credentials, which is the login credentials in plain text.

HTTP Persistent Connection
20: The "Connection: close" header field in HTTP protocol means that after the current communication sequence, the connection will be terminated. "Connection: keep-alive" on the other hand the connection will be persistent, and therefore allowing several requests to the same server.