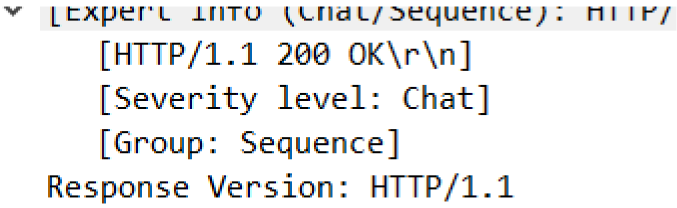
1. is your browser running HTTP version 1.0 or 1.1? What version of HTTP is the server running?

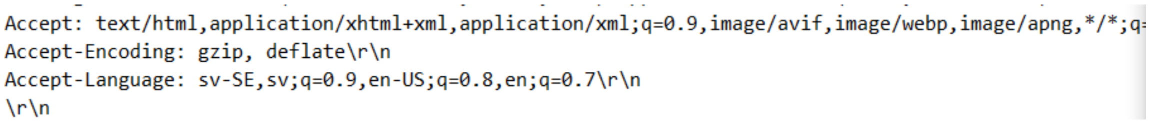
   We are ussing HTTP 1.1 (both source and destination)



```
[Expert Info (Chat/Sequence): HTTP/
    [HTTP/1.1 200 OK\r\n]
    [Severity level: Chat]
    [Group: Sequence]
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?

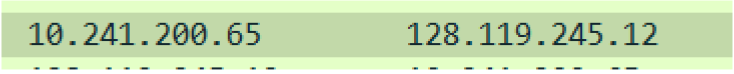
   We are accepting Swedish and English. We are also accepting html, xhtml, images of different file formats



```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=
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
```

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

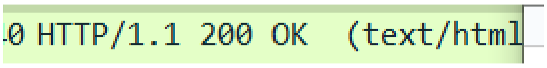
   Our ip address is 10.241.200.65 and the destination is 128.119.245.12.



```
10.241.200.65          128.119.245.12
```

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

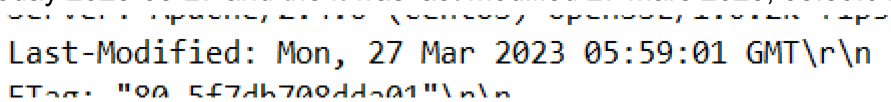
   We got 200 OK. after the TCP packet was delivered



```
0 HTTP/1.1 200 OK  (text/html
```

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

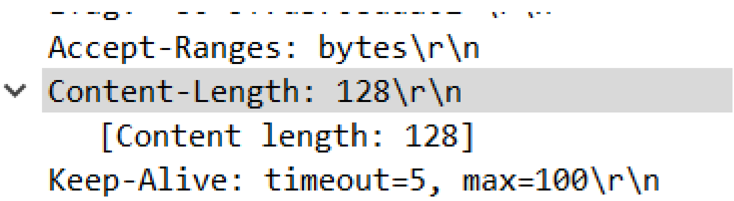
   It is today 2023-03-27 and the it was last modified 27 mars 2023, 05:59:01 GMT



```
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips 
Last-Modified: Mon, 27 Mar 2023 05:59:01 GMT\r\n
ETag: "80-5f7db708dda01"\r\n
```

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

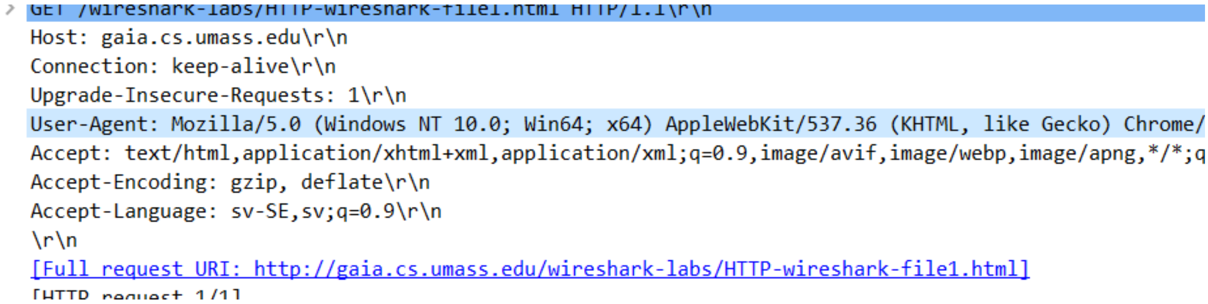
   128



```
Accept-Ranges: bytes\r\n
Content-Length: 128\r\n
    [Content length: 128]
Keep-Alive: timeout=5, max=100\r\n
```

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.

   We got several fields that are not included into the content-pane such as content-length, the “real” host name (not the ip, the dns translated), what language we are preferring and much more

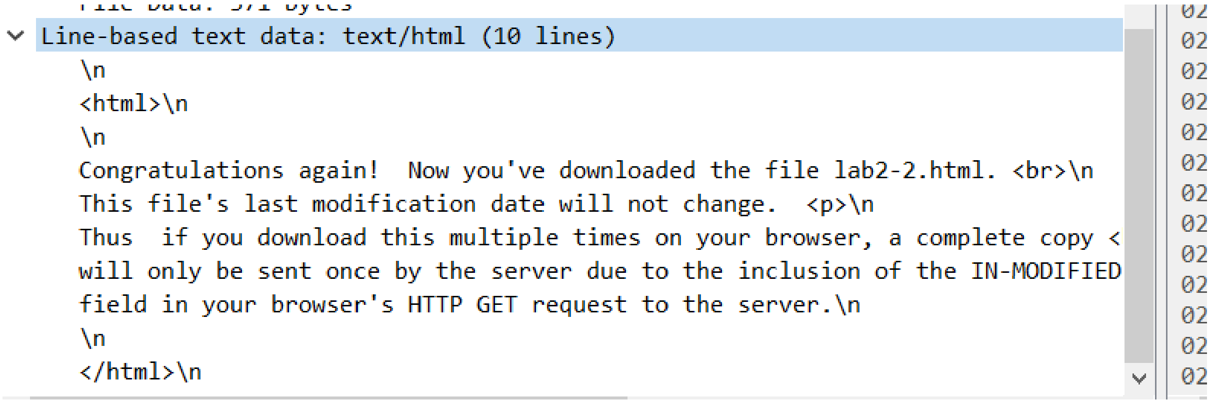


```
GET /wireshark-labs/HTTP-wireshark-file1.html HTTP/1.1\r\n
Host: gaia.cs.umass.edu\r\n
Connection: keep-alive\r\n
Upgrade-Insecure-Requests: 1\r\n
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q
Accept-Encoding: gzip, deflate\r\n
Accept-Language: sv-SE,sv;q=0.9\r\n
\r\n
[Full request URI: http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html]
[HTTP request 1/1]
```

Brief on section A 1-7:
This section is mostly about basic information about headers in requests. We learned how to find basic information about GET requests, find content-length, see when a html was updated but also what languages we prefer and also what HTTP version we use and what version the server uses.

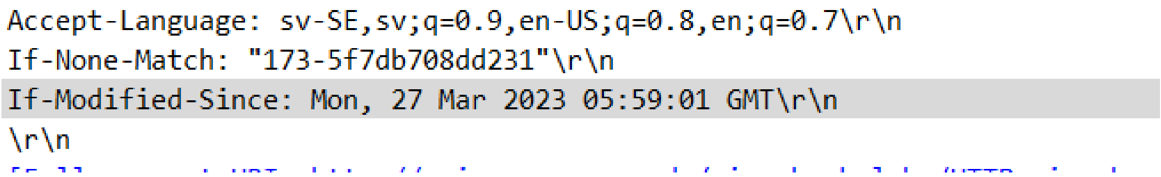
8. Inspect the contents of the first HTTP GET request from your browser to the server. Do you see an “IF-MODIFIED-SINCE” line in the HTTP GET?

We didn’t get it first time:



```
File Data: 371 bytes
Line-based text data: text/html (10 lines)
    \n
    <html>\n
    \n
    Congratulations again!  Now you've downloaded the file lab2-2.html. <br>\n
    This file's last modification date will not change.  <p>\n
    Thus  if you download this multiple times on your browser, a complete copy <
    will only be sent once by the server due to the inclusion of the IN-MODIFIED
    field in your browser's HTTP GET request to the server.\n
    \n
    </html>\n
```

but when we tried again we got the following:



```
Accept-Language: sv-SE,sv;q=0.9,en-US;q=0.8,en;q=0.7\r\n
If-None-Match: "173-5f7db708dd231"\r\n
If-Modified-Since: Mon, 27 Mar 2023 05:59:01 GMT\r\n
\r\n
```

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

   We would have gotten a packet with the html file but we only got a 304 returned to us instead. but the first time we can see that we get the response 200 ok and also a packet with the html file in it.

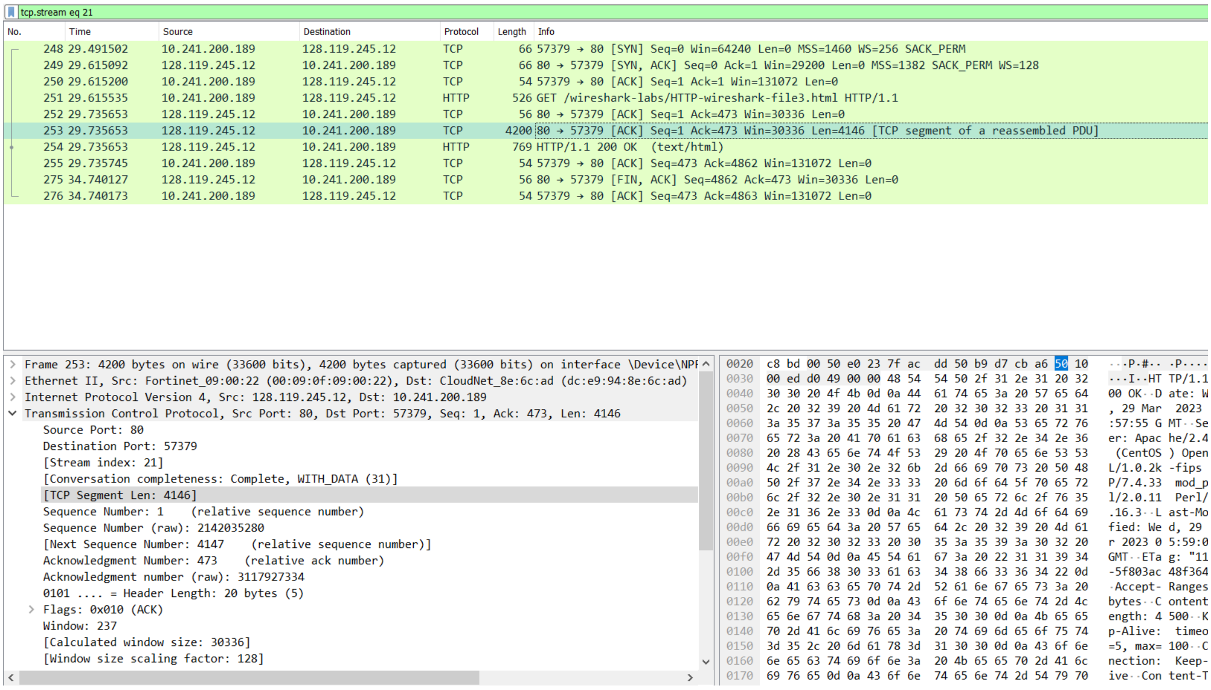
10. Now inspect the contents of the second HTTP GET request from your browser to the server. Do you see an “IF-MODIFIED-SINCE:” line in the HTTP GET? If so, what information follows the “IF-MODIFIED-SINCE:” header?

    check answer 8.

11. 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.

    We got a 304 and we didn’t get a returned html file since we already had it cached on our computer.

Brief:
Here we can see if we have retrieved a html file it stays cached on our computer until we either clean the cache or the server notifies that the html file has changed. So we can see that this saves us bandwidth. If we don’t need to get the updated version we use the cached version instead.



```
tcp.stream eq 21
No.   Time       Source          Destination     Protocol Length Info
  248 29.491502  10.241.200.189  128.119.245.12  TCP        66 57379 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
  249 29.615092  128.119.245.12  10.241.200.189  TCP        66 80 → 57379 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1382 SACK_PERM WS=128
  250 29.615200  10.241.200.189  128.119.245.12  TCP        54 57379 → 80 [ACK] Seq=1 Ack=1 Win=131072 Len=0
  251 29.615535  10.241.200.189  128.119.245.12  HTTP      526 GET /wireshark-labs/HTTP-wireshark-file3.html HTTP/1.1
  252 29.735653  128.119.245.12  10.241.200.189  TCP        56 80 → 57379 [ACK] Seq=1 Ack=473 Win=30336 Len=0
  253 29.735653  128.119.245.12  10.241.200.189  TCP      4200 80 → 57379 [ACK] Seq=1 Ack=473 Win=30336 Len=4146 [TCP segment of a reassembled PDU]
  254 29.735653  128.119.245.12  10.241.200.189  HTTP      769 HTTP/1.1 200 OK  (text/html)
  255 29.735745  10.241.200.189  128.119.245.12  TCP        54 57379 → 80 [ACK] Seq=473 Ack=4862 Win=131072 Len=0
  275 34.740127  128.119.245.12  10.241.200.189  TCP        56 80 → 57379 [FIN, ACK] Seq=4862 Ack=473 Win=30336 Len=0
  276 34.740173  10.241.200.189  128.119.245.12  TCP        54 57379 → 80 [ACK] Seq=473 Ack=4863 Win=131072 Len=0
```

```
> Frame 253: 4200 bytes on wire (33600 bits), 4200 bytes captured (33600 bits) on interface \Device\NPF
> Ethernet II, Src: Fortinet_09:00:22 (00:09:0f:09:00:22), Dst: CloudNet_8e:6c:ad (dc:e9:94:8e:6c:ad)
> Internet Protocol Version 4, Src: 128.119.245.12, Dst: 10.241.200.189
v Transmission Control Protocol, Src Port: 80, Dst Port: 57379, Seq: 1, Ack: 473, Len: 4146
     Source Port: 80
     Destination Port: 57379
     [Stream index: 21]
     [Conversation completeness: Complete, WITH_DATA (31)]
     [TCP Segment Len: 4146]
     Sequence Number: 1    (relative sequence number)
     Sequence Number (raw): 2142035280
     [Next Sequence Number: 4147    (relative sequence number)]
     Acknowledgment Number: 473    (relative ack number)
     Acknowledgment number (raw): 3117927334
     0101 .... = Header Length: 20 bytes (5)
   > Flags: 0x010 (ACK)
     Window: 237
     [Calculated window size: 30336]
     [Window size scaling factor: 128]
```

```
0020  c8 bd 00 50 e0 23 7f ac  dd 50 b9 d7 cb a6 50 10   ···P·#·· ·P····
0030  00 ed d0 49 00 00 48 54  54 50 2f 31 2e 31 20 32   ···I··HT TP/1.1
0040  30 30 20 4f 4b 0d 0a 44  61 74 65 3a 20 57 65 64   00 OK··D ate: W
0050  2c 20 32 39 20 4d 61 72  20 32 30 32 33 20 31 31   , 29 Mar  2023
0060  3a 35 37 3a 35 35 20 47  4d 54 0d 0a 53 65 72 76   :57:55 G MT··Se
0070  65 72 3a 20 41 70 61 63  68 65 2f 32 2e 34 2e 36   er: Apac he/2.4
0080  20 28 43 65 6e 74 4f 53  29 20 4f 70 65 6e 53 53    (CentOS ) Open
0090  4c 2f 31 2e 30 2e 32 6b  2d 66 69 70 73 20 50 48   L/1.0.2k -fips
00a0  50 2f 37 2e 34 2e 33 33  20 6d 6f 64 5f 70 65 72   P/7.4.33  mod_p
00b0  6c 2f 32 2e 30 2e 31 31  20 50 65 72 6c 2f 76 35   l/2.0.11  Perl/
00c0  2e 31 36 2e 33 0d 0a 4c  61 73 74 2d 4d 6f 64 69   .16.3··L ast-Mo
00d0  66 69 65 64 3a 20 57 65  64 2c 20 32 39 20 4d 61   fied: We d, 29
00e0  72 20 32 30 32 33 20 30  35 3a 35 39 3a 30 32 20   r 2023 0 5:59:0
00f0  47 4d 54 0d 0a 45 54 61  67 3a 20 22 31 31 39 34   GMT··ETa g: "11
0100  2d 35 66 38 30 33 61 63  34 38 66 33 36 34 22 0d   -5f803ac 48f364
0110  0a 41 63 63 65 70 74 2d  52 61 6e 67 65 73 3a 20   ·Accept- Ranges
0120  62 79 74 65 73 0d 0a 43  6f 6e 74 65 6e 74 2d 4c   bytes··C ontent
0130  65 6e 67 74 68 3a 20 34  35 30 30 0d 0a 4b 65 65   ength: 4 500··K
0140  70 2d 41 6c 69 76 65 3a  20 74 69 6d 65 6f 75 74   p-Alive:  timeo
0150  3d 35 2c 20 6d 61 78 3d  31 30 30 0d 0a 43 6f 6e   =5, max= 100··C
0160  6e 65 63 74 69 6f 6e 3a  20 4b 65 65 70 2d 41 6c   nection:  Keep-
0170  69 76 65 0d 0a 43 6f 6e  74 65 6e 74 2d 54 79 70   ive··Con tent-T
```

12. 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?

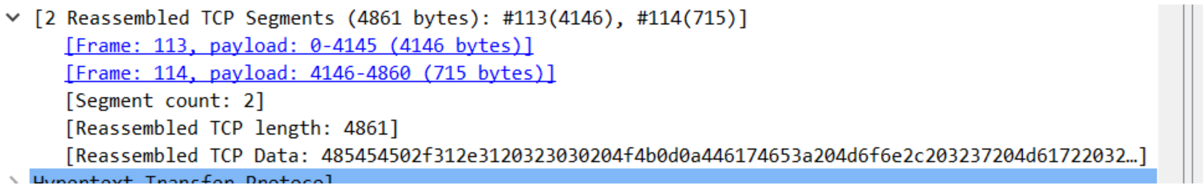
    We sent one get request, the package number is 109

13. 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?

    114 contains the 200 ok.

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

    we needed two packages, the first one contained 4200 bytes and the seconds contained 769 bytes



```
v [2 Reassembled TCP Segments (4861 bytes): #113(4146), #114(715)]
     [Frame: 113, payload: 0-4145 (4146 bytes)]
     [Frame: 114, payload: 4146-4860 (715 bytes)]
     [Segment count: 2]
     [Reassembled TCP length: 4861]
     [Reassembled TCP Data: 485454502f312e3120323030204f4b0d0a446174653a204d6f6e2c203237204d61722032…]
> Hypertext Transfer Protocol
```

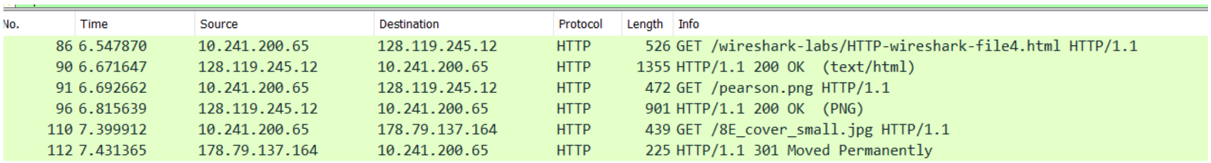
15. How many data-containing TCP segments were needed to carry the single HTTP response and the text of the Bill of Rights?

    No, the package is not segmented hence they have a big packet with the full html file with len=4146. We can see the rest of the files have len=0 so they come with just a header with 0 content. Check the first image of this section.

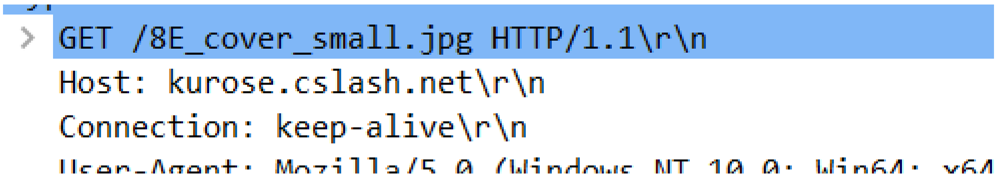
Brief: Here we can see the difference between pipelined requests and serially send get requests. We can also conclude that we can save a lot of time using pipelines if we are trying to get multiple pictures from multiple sites. We can also see, if a package is too large we can see the package getting split up into several packages with the same identifier.

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

    We are doing three GET requests, first we retrieve the html then we try to retrieve two images. The HTML and the first 200 OK image are retrieved from the website we entered. The second picture is retrieved from a new ip address ("178.79.137.164"). We can see the ip address corresponds to kurose.cslash.net with the link /8E_cover_small.jpg there for the link must be: kurose.cslash.net/8E_cover_small.jpg



```
No.    Time        Source           Destination      Protocol  Length  Info
    86 6.547870    10.241.200.65    128.119.245.12   HTTP       526 GET /wireshark-labs/HTTP-wireshark-file4.html HTTP/1.1
    90 6.671647    128.119.245.12   10.241.200.65    HTTP      1355 HTTP/1.1 200 OK  (text/html)
    91 6.692662    10.241.200.65    128.119.245.12   HTTP       472 GET /pearson.png HTTP/1.1
    96 6.815639    128.119.245.12   10.241.200.65    HTTP       901 HTTP/1.1 200 OK  (PNG)
   110 7.399912    10.241.200.65    178.79.137.164   HTTP       439 GET /8E_cover_small.jpg HTTP/1.1
   112 7.431365    178.79.137.164   10.241.200.65    HTTP       225 HTTP/1.1 301 Moved Permanently
```

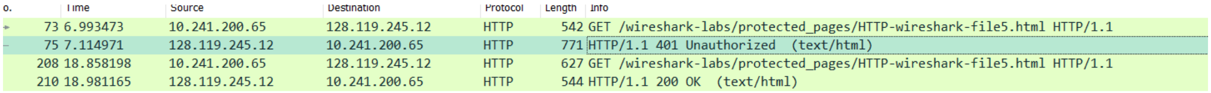


```
> GET /8E_cover_small.jpg HTTP/1.1\r\n
  Host: kurose.cslash.net\r\n
  Connection: keep-alive\r\n
  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64
```

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

    We can see this was retrieved serially since we first get a 200ok for the html, then we make a new request. We get again 200 Ok then we do the last Get request and get the 200 ok. If this was made in parallel we would do two get requests at the same time. We can also see on the time elapsed that this can't be made in parallel since there is around 0,4 seconds between the requests which is a lot of time keeping in mind we are talking about computers.

Brief: Here we can see that we can retrieve different files from different sites. Everything doesn't need to be on the same site. This is why proxies exist and can save us loads of time if we need to travel for a long distance. (getting an image from a proxy in Europe is way faster than trying to get it from the original server in Canada).



```
o.     Time         Source           Destination      Protocol  Length  Info
    73 6.993473     10.241.200.65    128.119.245.12   HTTP       542 GET /wireshark-labs/protected_pages/HTTP-wireshark-file5.html HTTP/1.1
    75 7.114971     128.119.245.12   10.241.200.65    HTTP       771 HTTP/1.1 401 Unauthorized  (text/html)
   208 18.858198    10.241.200.65    128.119.245.12   HTTP       627 GET /wireshark-labs/protected_pages/HTTP-wireshark-file5.html HTTP/1.1
   210 18.981165    128.119.245.12   10.241.200.65    HTTP       544 HTTP/1.1 200 OK  (text/html)
```

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

    We get a 401 unauthorized since we haven't logged in.

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

The new fields is the login form we now retrieve since the server knows that we are not logged in.

Brief: Here we can see the protected site returning a 401 unauthorized firstly since the server assumes we are logged in. When we receive the 401 we send a new package asking for the login form, which we get and after that we can send our information and later get a 200 OK.

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

Connection: close means the client wants the server to close the connection after sending the response, while Connection: keep-alive means the client wants the connection to remain open for multiple requests. Choosing between the two depends on the specific use case. Connection: close is more efficient for a single request, while Connection: keep-alive is better for multiple requests over the same connection.