Wireshark Lab

Rajdeep Gill 7934493

ECE 3700 B01

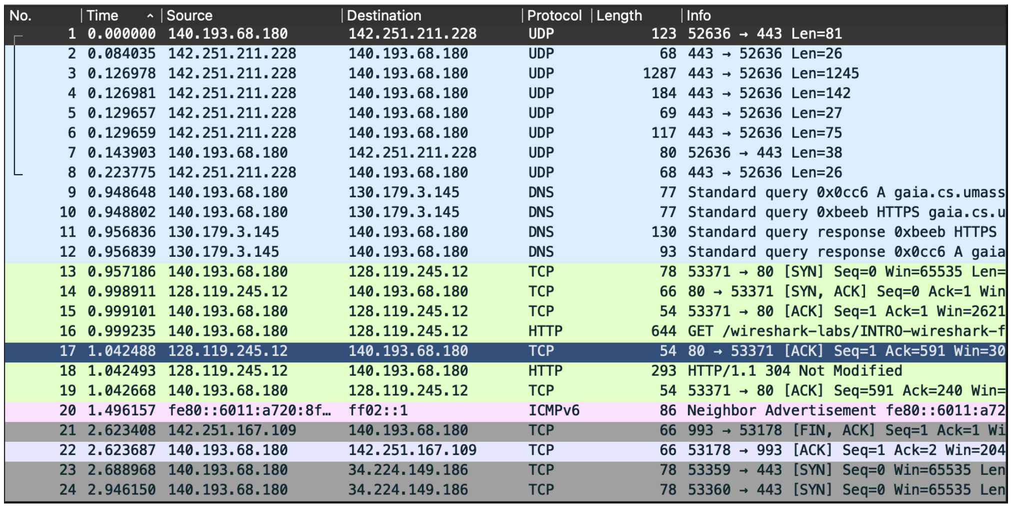
January 24, 2025

Contents

1 Wireshark Lab: Getting Started

1. The different protocols that appear in the protocol column in the unfiltered packet-listing window are:
   - UDP, TCP, DNS, HTTP, ICMPv6, TLSv1.2, LLC, TLSv1.3,

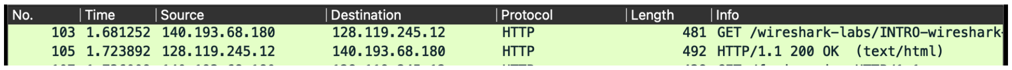
   Some of these packets can be see in Figure 1.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	140.193.68.180	142.251.211.228	UDP	123	52636 → 443 Len=81
2	0.084035	142.251.211.228	140.193.68.180	UDP	68	443 → 52636 Len=26
3	0.126978	142.251.211.228	140.193.68.180	UDP	1287	443 → 52636 Len=1245
4	0.126981	142.251.211.228	140.193.68.180	UDP	184	443 → 52636 Len=142
5	0.129657	142.251.211.228	140.193.68.180	UDP	69	443 → 52636 Len=27
6	0.129659	142.251.211.228	140.193.68.180	UDP	117	443 → 52636 Len=75
7	0.143903	140.193.68.180	142.251.211.228	UDP	80	52636 → 443 Len=38
8	0.223775	142.251.211.228	140.193.68.180	UDP	68	443 → 52636 Len=26
9	0.948648	140.193.68.180	130.179.3.145	DNS	77	Standard query 0x0cc6 A gaia.cs.umass
10	0.948802	140.193.68.180	130.179.3.145	DNS	77	Standard query 0xbeeb HTTPS gaia.cs.u
11	0.956836	130.179.3.145	140.193.68.180	DNS	130	Standard query response 0xbeeb HTTPS
12	0.956839	130.179.3.145	140.193.68.180	DNS	93	Standard query response 0x0cc6 A gaia
13	0.957186	140.193.68.180	128.119.245.12	TCP	78	53371 → 80 [SYN] Seq=0 Win=65535 Len=
14	0.998911	128.119.245.12	140.193.68.180	TCP	66	80 → 53371 [SYN, ACK] Seq=0 Ack=1 Win
15	0.999101	140.193.68.180	128.119.245.12	TCP	54	53371 → 80 [ACK] Seq=1 Ack=1 Win=2621
16	0.999235	140.193.68.180	128.119.245.12	HTTP	644	GET /wireshark-labs/INTRO-wireshark-f
17	1.042488	128.119.245.12	140.193.68.180	TCP	54	80 → 53371 [ACK] Seq=1 Ack=591 Win=30
18	1.042493	128.119.245.12	140.193.68.180	HTTP	293	HTTP/1.1 304 Not Modified
19	1.042668	140.193.68.180	128.119.245.12	TCP	54	53371 → 80 [ACK] Seq=591 Ack=240 Win=
20	1.496157	fe80::6011:a720:8f…	ff02::1	ICMPv6	86	Neighbor Advertisement fe80::6011:a72
21	2.623408	142.251.167.109	140.193.68.180	TCP	66	993 → 53178 [FIN, ACK] Seq=1 Ack=1 Wi
22	2.623687	140.193.68.180	142.251.167.109	TCP	66	53178 → 993 [ACK] Seq=1 Ack=2 Win=204
23	2.688968	140.193.68.180	34.224.149.186	TCP	78	53359 → 443 [SYN] Seq=0 Win=65535 Len
24	2.946150	140.193.68.180	34.224.149.186	TCP	78	53360 → 443 [SYN] Seq=0 Win=65535 Len

Figure 1: Packet-listing window

2. The time taken for when the HTTP GET message was sent to when the HTTP OK reply was recieved was 0.04264 seconds. This was calculated by subtracting the time the HTTP GET message was sent from the time the HTTP OK reply was recieved. This can be seen in Figure 2.



No.	Time	Source	Destination	Protocol	Length	Info
103	1.681252	140.193.68.180	128.119.245.12	HTTP	481	GET /wireshark-labs/INTRO-wireshark
105	1.723892	128.119.245.12	140.193.68.180	HTTP	492	HTTP/1.1 200 OK (text/html)

Figure 2: Time taken for HTTP GET message to HTTP OK reply

3. The Internet address of the gaia.cs.umass.edu is 128.119.245.12 and the Internet address of my computer is 140.193.68.100. These addresses are sseen in Figure 2.

4. Making 20 different requests and finding the delay between the HTTP GET message and the HTTP OK reply, the average delay was 0.08513 seconds. The plot of the 20 different requests can be seen in Figure 3.

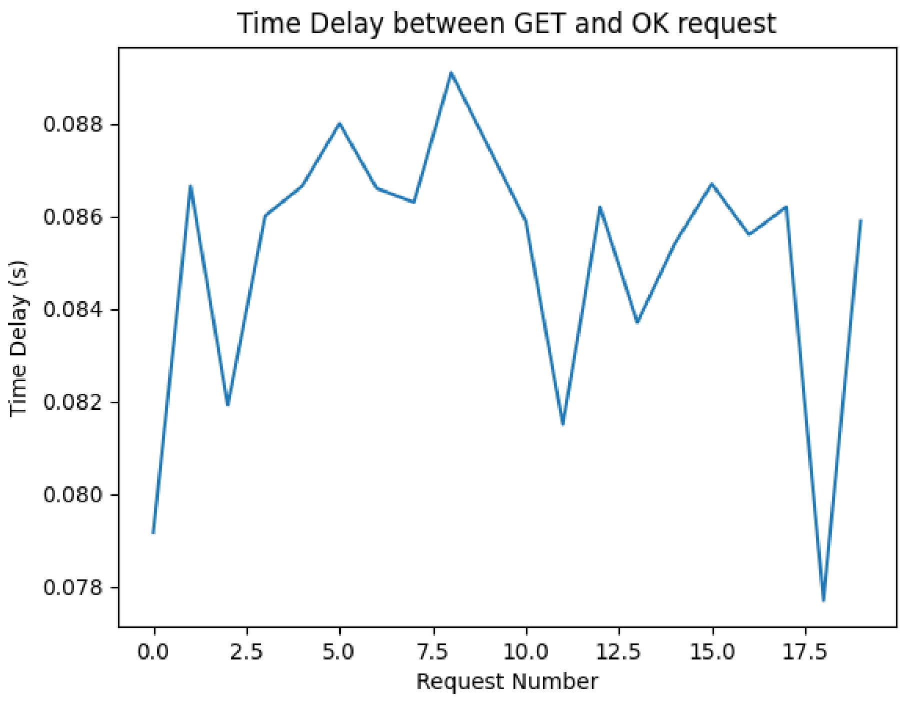
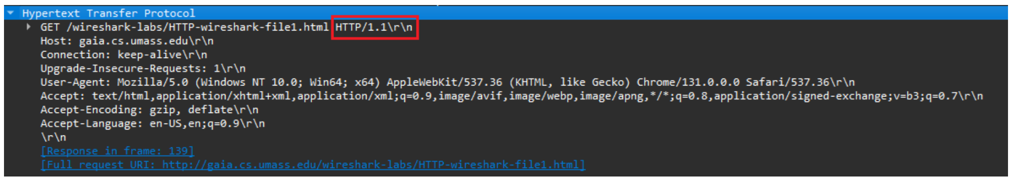


Figure 3: 20 different requests

2 Wireshark Lab: HTTP

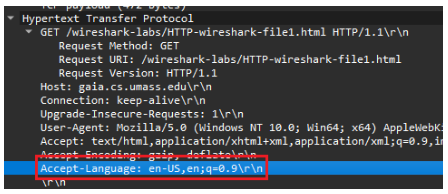
1. The browser is running HTTP version 1.1 and so is the server. This can be deduced from the GET and OK requests both having HTTP/1.1 in the info column. As seen in Figure 4.



```
▾ 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/131.0.0.0 Safari/537.36\r\n
    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=0.7\r\n
    Accept-Encoding: gzip, deflate\r\n
    Accept-Language: en-US,en;q=0.9\r\n
    \r\n
    [Response in frame: 139]
    [Full request URI: http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html]
```

Figure 4: HTTP version

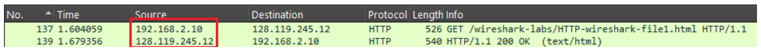
2. The browser indicates it accepts en-US. As seen in Figure 5.



```
▾ Hypertext Transfer Protocol
  ▾ GET /wireshark-labs/HTTP-wireshark-file1.html HTTP/1.1\r\n
      Request Method: GET
      Request URI: /wireshark-labs/HTTP-wireshark-file1.html
      Request Version: HTTP/1.1
    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) AppleWebK
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,in
    Accept-Encoding: gzip, deflate\r\n
    Accept-Language: en-US,en;q=0.9\r\n
    \r\n
```

Figure 5: Accepted languages

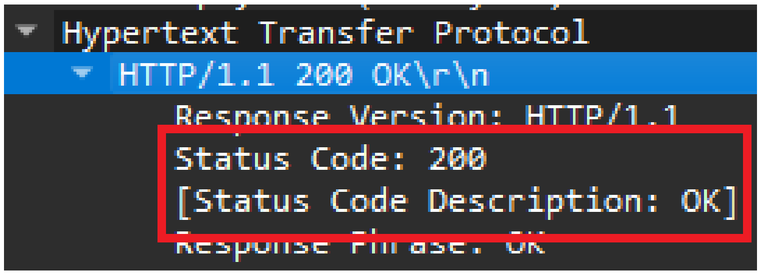
3. The IP address of my computer is 192.168.2.10 and the gaia.cs.umass.edu server is 128.119.245.12. This can be seen in the GET request and OK response. The source of the GET request is my computer and the destination is the server, vice-versa for the OK response. As seen in Figure 6.



```
No.   Time      Source          Destination     Protocol Length Info
  137 1.604059  192.168.2.10    128.119.245.12  HTTP       526 GET /wireshark-labs/HTTP-wireshark-file1.html HTTP/1.1
  139 1.679356  128.119.245.12  192.168.2.10    HTTP       540 HTTP/1.1 200 OK  (text/html)
```

Figure 6: IP addresses

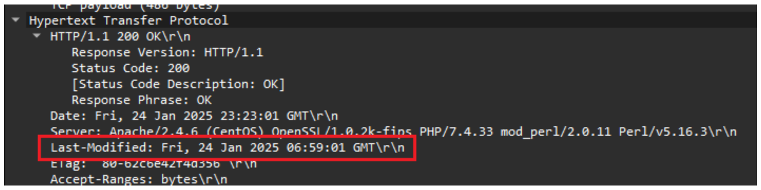
4. The status code returned by the server is 200. This can be seen in the info column of the OK request, or the Hypertext Transfer Protocol section. As seen in Figure 7.



```
Hypertext Transfer Protocol
  HTTP/1.1 200 OK\r\n
    Response Version: HTTP/1.1
    Status Code: 200
    [Status Code Description: OK]
    Response Phrase: OK
```

Figure 7: Status code

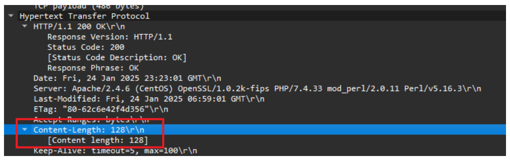
5. The file was last modified at: January 24, 2025 at 6:59:01 GMT. This can be seen in OK response from the server in the Hypertext Transfer Protocol section. As seen in Figure 8.



```
Hypertext Transfer Protocol
  HTTP/1.1 200 OK\r\n
    Response Version: HTTP/1.1
    Status Code: 200
    [Status Code Description: OK]
    Response Phrase: OK
  Date: Fri, 24 Jan 2025 23:23:01 GMT\r\n
  Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/7.4.33 mod_perl/2.0.11 Perl/v5.16.3\r\n
  Last-Modified: Fri, 24 Jan 2025 06:59:01 GMT\r\n
  ETag: "80-62c6e42f4d356"\r\n
  Accept-Ranges: bytes\r\n
```

Figure 8: Last modified date

6. 128 bytes of content was returned to the browser, and can be seen in the Content-Length field in the OK response. As seen in Figure 9.



```
Hypertext Transfer Protocol
  HTTP/1.1 200 OK\r\n
    Response Version: HTTP/1.1
    Status Code: 200
    [Status Code Description: OK]
    Response Phrase: OK
  Date: Fri, 24 Jan 2025 23:23:01 GMT\r\n
  Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/7.4.33 mod_perl/2.0.11 Perl/v5.16.3\r\n
  Last-Modified: Fri, 24 Jan 2025 06:59:01 GMT\r\n
  ETag: "80-62c6e42f4d356"\r\n
  Accept-Ranges: bytes\r\n
  Content-Length: 128\r\n
    [Content length: 128]
  Keep-Alive: timeout=5, max=100\r\n
```

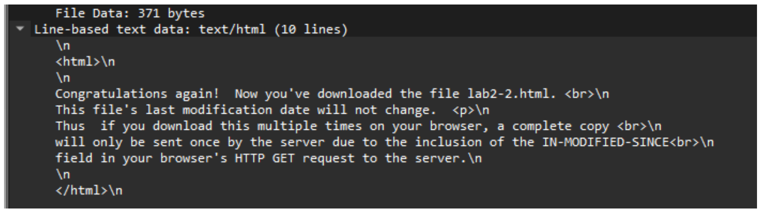
Figure 9: Content length

7. The headings in the two windows are the same.

3 The HTTP CONDITIONAL GET/response interaction

8. The first HTTP GET request does not have an If-Modified-Since header.

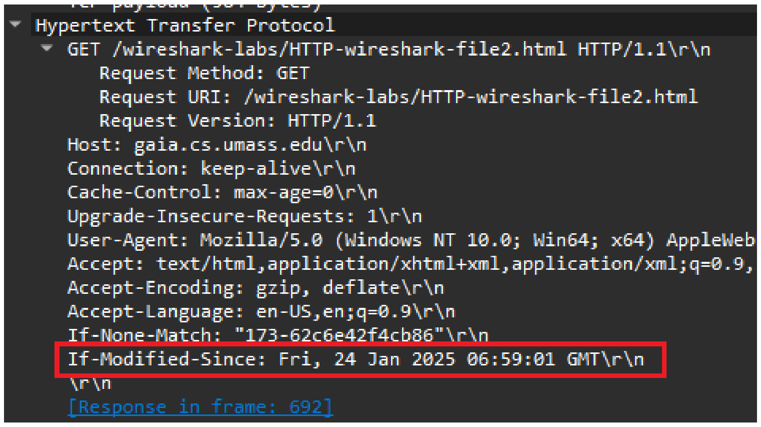
9. The server does explicitly return the contents of the file as we can see the text in the response to the first GET. It is 371 bytes and can be seen in the Hypertext Transfer Protocol section of the response.



```
    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 <br>\n
    will only be sent once by the server due to the inclusion of the IN-MODIFIED-SINCE<br>\n
    field in your browser's HTTP GET request to the server.\n
    \n
    </html>\n
```

Figure 10: First HTTP GET request

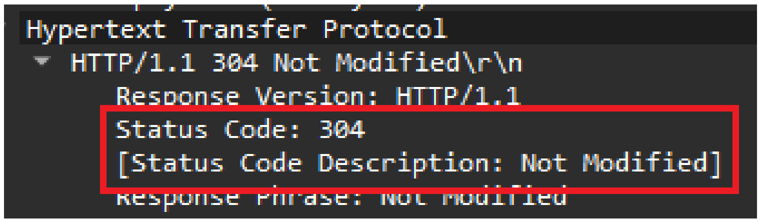
10. Inspecting the second HTTP GET request, we do see an If-Modified-Since header. The info followed by the header is "If-Modified-Since: Tue, 23 Sep 2003 05:35:00 GMT\r\n". This can be seen in the Hypertext Transfer Protocol section of the GET request.



```
Hypertext Transfer Protocol
  GET /wireshark-labs/HTTP-wireshark-file2.html HTTP/1.1\r\n
      Request Method: GET
      Request URI: /wireshark-labs/HTTP-wireshark-file2.html
      Request Version: 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) AppleWeb
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,
  Accept-Encoding: gzip, deflate\r\n
  Accept-Language: en-US,en;q=0.9\r\n
  If-None-Match: "173-62c6e42f4cb86"\r\n
  If-Modified-Since: Fri, 24 Jan 2025 06:59:01 GMT\r\n
  \r\n
  [Response in frame: 692]
```

Figure 11: Second HTTP GET request

11. The HTTP status code of is 304 with a response phrase of Not Modified. The server did not explicitly return the contents of the file as the browser already had the file cached. There is also no content length in the response

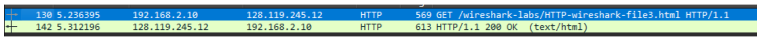


```
Hypertext Transfer Protocol
  HTTP/1.1 304 Not Modified\r\n
      Response Version: HTTP/1.1
      Status Code: 304
      [Status Code Description: Not Modified]
      Response Phrase: Not Modified
```

Figure 12: HTTP response

4 Retrieving Long Documents

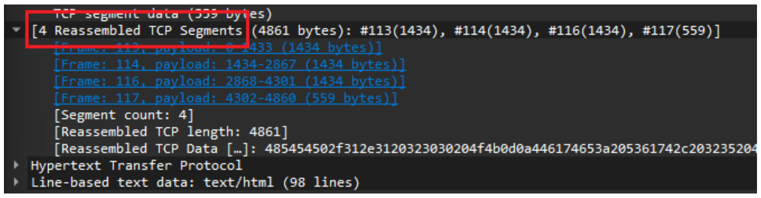
12. One HTTP GET request was sent by the browser.



```
130 5.236395   192.168.2.10     128.119.245.12   HTTP   569 GET /wireshark-labs/HTTP-wireshark-file3.html HTTP/1.1
142 5.312196   128.119.245.12   192.168.2.10     HTTP   613 HTTP/1.1 200 OK  (text/html)
```

Figure 13: HTTP GET request

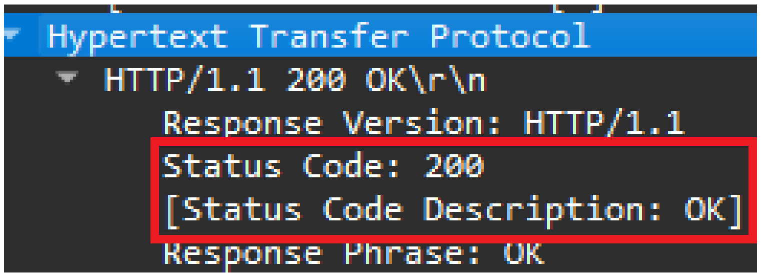
13. Four TCP segments were needed to carry the single HTTP response message. In the response from the server, there is an entry stating 4 reassembled TCP segments.



```
    TCP segment data (559 bytes)
[4 Reassembled TCP Segments (4861 bytes): #113(1434), #114(1434), #116(1434), #117(559)]
    [Frame: 113, payload: 0-1433 (1434 bytes)]
    [Frame: 114, payload: 1434-2867 (1434 bytes)]
    [Frame: 116, payload: 2868-4301 (1434 bytes)]
    [Frame: 117, payload: 4302-4860 (559 bytes)]
    [Segment count: 4]
    [Reassembled TCP length: 4861]
    [Reassembled TCP Data […]: 485454502f312e3120323030204f4b0d0a446174653a205361742c203235204
Hypertext Transfer Protocol
Line-based text data: text/html (98 lines)
```

Figure 14: Reassembled TCP segments

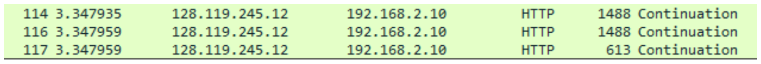
14. The status code and phrase associated with the response to the HTTP GET request is 200 OK.



```
Hypertext Transfer Protocol
  HTTP/1.1 200 OK\r\n
    Response Version: HTTP/1.1
    Status Code: 200
    [Status Code Description: OK]
    Response Phrase: OK
```

Figure 15: Status code and phrase

15. There are three packets with status lines stating a continuation.

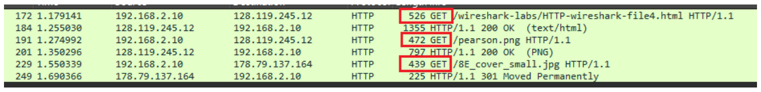


```
114 3.347935   128.119.245.12   192.168.2.10   HTTP   1488 Continuation
116 3.347959   128.119.245.12   192.168.2.10   HTTP   1488 Continuation
117 3.347959   128.119.245.12   192.168.2.10   HTTP    613 Continuation
```

Figure 16: Continuation packets

5 HTML Documents with Embedded Objects

16. 3 HTTP GET request messages were sent by the browser. One for the html and the other two for the image. The internet address of the first and second requests was: 128.119.245.12, and the third request was: 178.79.137.164.



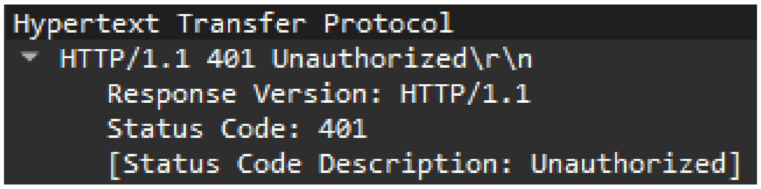
```
172 1.179141   192.168.2.10     128.119.245.12   HTTP    526 GET /wireshark-labs/HTTP-wireshark-file4.html HTTP/1.1
184 1.255030   128.119.245.12   192.168.2.10     HTTP   1355 HTTP/1.1 200 OK  (text/html)
191 1.274992   192.168.2.10     128.119.245.12   HTTP    472 GET /pearson.png HTTP/1.1
201 1.350296   128.119.245.12   192.168.2.10     HTTP    797 HTTP/1.1 200 OK  (PNG)
229 1.550339   192.168.2.10     178.79.137.164   HTTP    439 GET /8E_cover_small.jpg HTTP/1.1
249 1.690366   178.79.137.164   192.168.2.10     HTTP    225 HTTP/1.1 301 Moved Permanently
```

Figure 17: HTTP GET requests

17. The requests for images are done in parallel. However in the packet capture the requests are done sequentially. The first request is for the html file, then we get an OK response. Then the browser sends a request for the first image, then we get an OK response. Then the browser sends a request for the second image, then we get an OK response. This can be seen in Figure 17. This could be due to the second image being larger than the first image, we are also getting a 301 response for the second image which could be causing the delay. In this case it seems the requests are done sequentially, but in a real world scenario the requests would be done in parallel.

6 HTTP Authentication

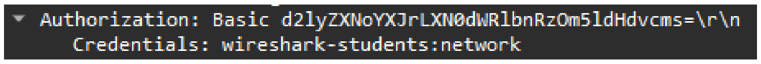
18. In the initial HTTP GET request, the server responds with a 401 code and a message of Unauthorized.



```
Hypertext Transfer Protocol
 ▾ HTTP/1.1 401 Unauthorized\r\n
      Response Version: HTTP/1.1
      Status Code: 401
      [Status Code Description: Unauthorized]
```

Figure 18: Initial HTTP GET request

19. The new field in the second HTTP GET request is the Authorization field. It is a basic authentication field with the value encoded in base64.



```
 ▾ Authorization: Basic d2lyZXNoYXJrLXN0dWRlbnRzOm5ldHdvcms=\r\n
      Credentials: wireshark-students:network
```

Figure 19: Second HTTP GET request

7 Additional Questions

1. In the TCP/IP stack, HTTP belongs in the application layer.
2. The underlying transport layer protocol used by TCP is IP.
3. The HTTP response for a successful request is 200 OK.
4. When a file exceeds the payload size of a single packet, the file is split into multiple TCP segments and are sent individually. The segments are then reassembled at the destination.
5. The components of the HTTP status line are the status code and the status phrase.
6. The encoding method used in HTTP authentication is base64.
7. Basic authentication is not secure as base64 can be easily decoded, allowing the information in the header to be read. If it contains sensitive information, it can be easily stolen. As well as, base64 is not an encryption method, it is an encoding method.