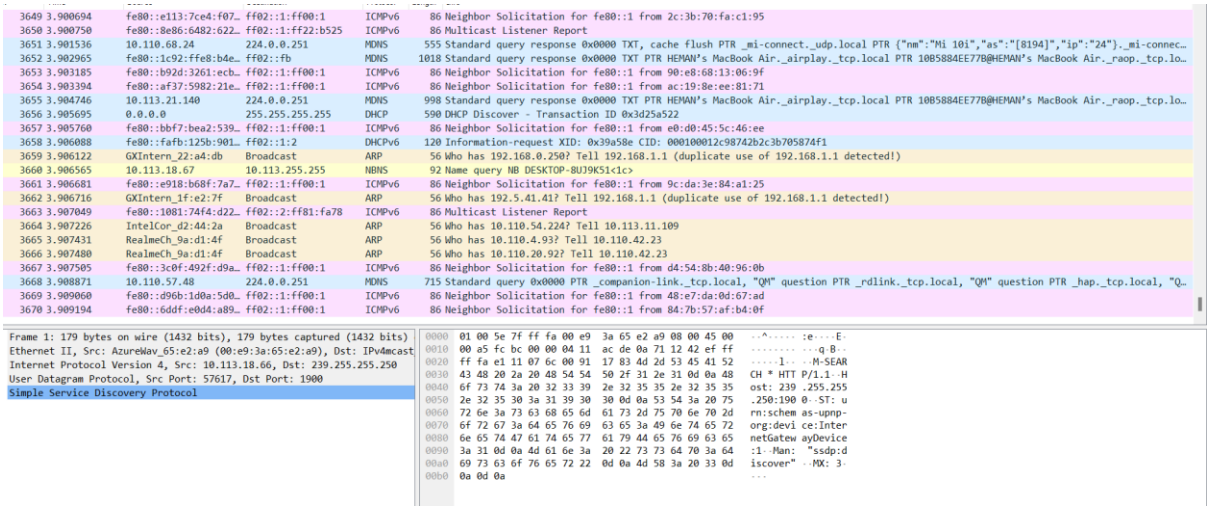
Lab Sheet 1
Analyzing http using Wireshark

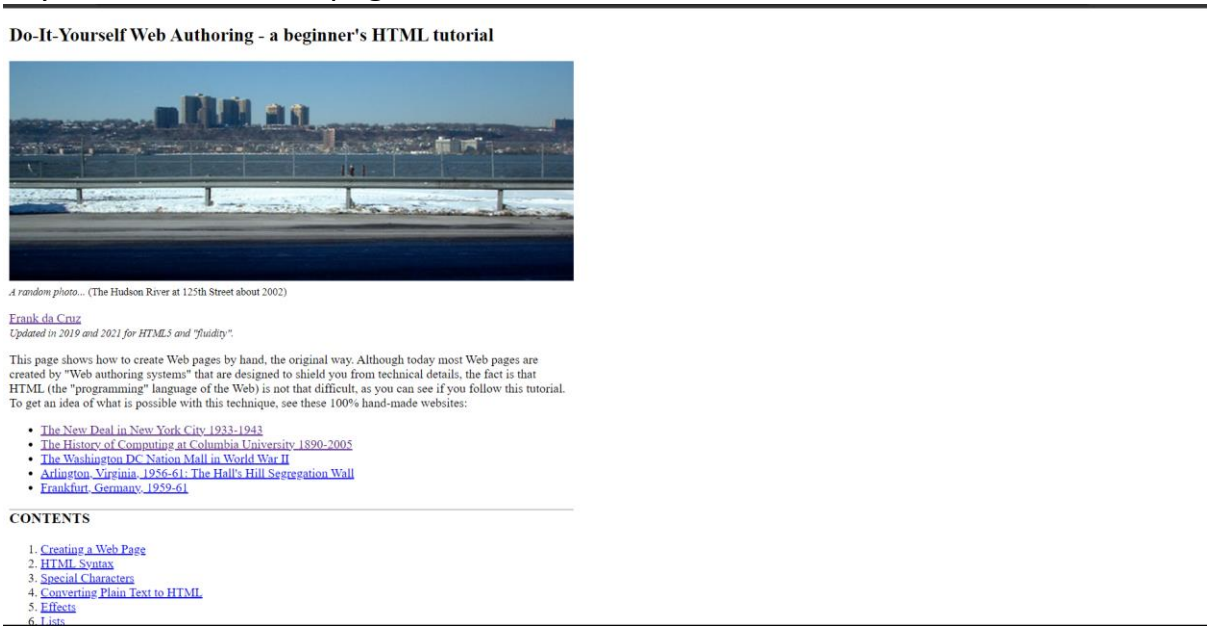
Name: Vinayak V Thayil
Roll No:AM.EN.U4CSE21161

1. Open the Packet sniffer [Wireshark] Application and Capture **YOUR NETWORK** Interface



2. Do this activity capture frames?

a. Request for a web page from amrita.edu and search for some keywords on the webpage



Do-It-Yourself Web Authoring - a beginner's HTML tutorial

A random photo... (The Hudson River at 125th Street about 2002)

Frank da Cruz
Updated in 2019 and 2021 for HTML5 and "fluidity".

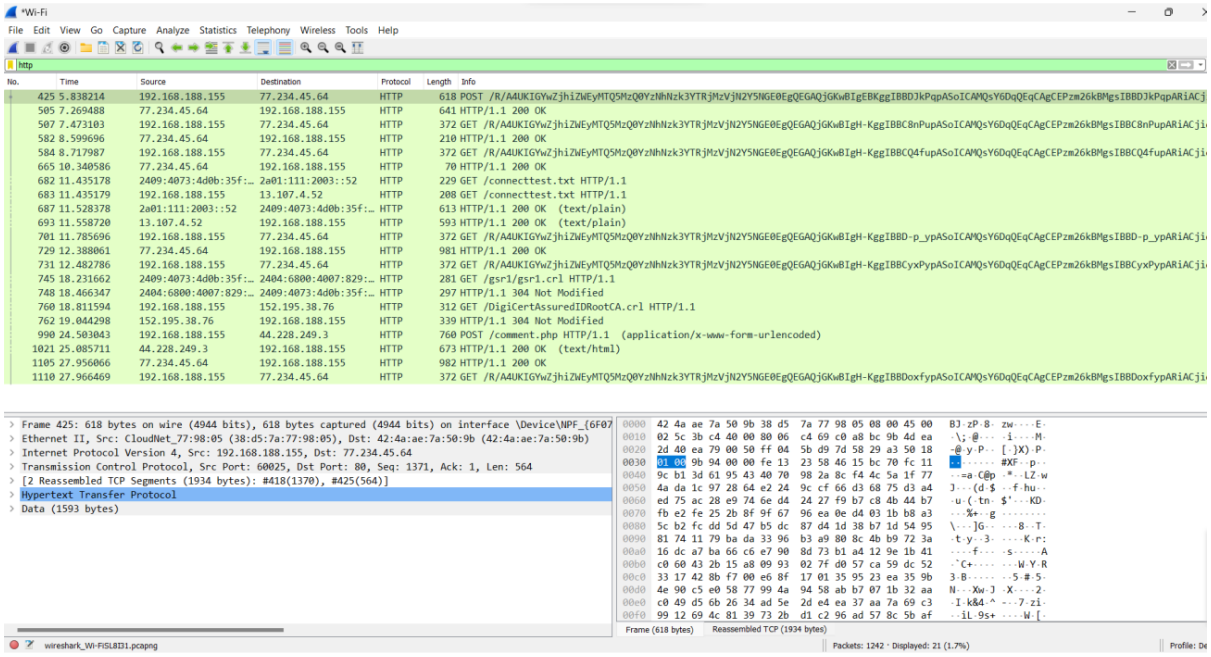
This page shows how to create Web pages by hand, the original way. Although today most Web pages are created by "Web authoring systems" that are designed to shield you from technical details, the fact is that HTML (the "programming" language of the Web) is not that difficult, as you can see if you follow this tutorial. To get an idea of what is possible with this technique, see these 100% hand-made websites:

- The New Deal in New York City 1933-1943
- The History of Computing at Columbia University 1890-2005
- The Washington DC Nation Mall in World War II
- Arlington, Virginia, 1956-61: The Hall's Hill Segregation Wall
- Frankfurt, Germany, 1959-61

CONTENTS

1. Creating a Web Page
2. HTML Syntax
3. Special Characters
4. Converting Plain Text to HTML
5. Effects
6. Lists

b. Open any URL where you can post comments) in your browser and post a comment in it.



3. Based on the above activities, explain the working of the HTTP protocol [HTTP GET and POST message] briefly with typed answers and answer highlighted screenshots for the above activity from Wireshark.

Working:

• The browser sends an HTTP GET request to the web server hosting the URL you entered. This request asks the server to send back the HTML code for the webpage.

• The web server responds to the GET request by sending back the HTML code for the webpage.

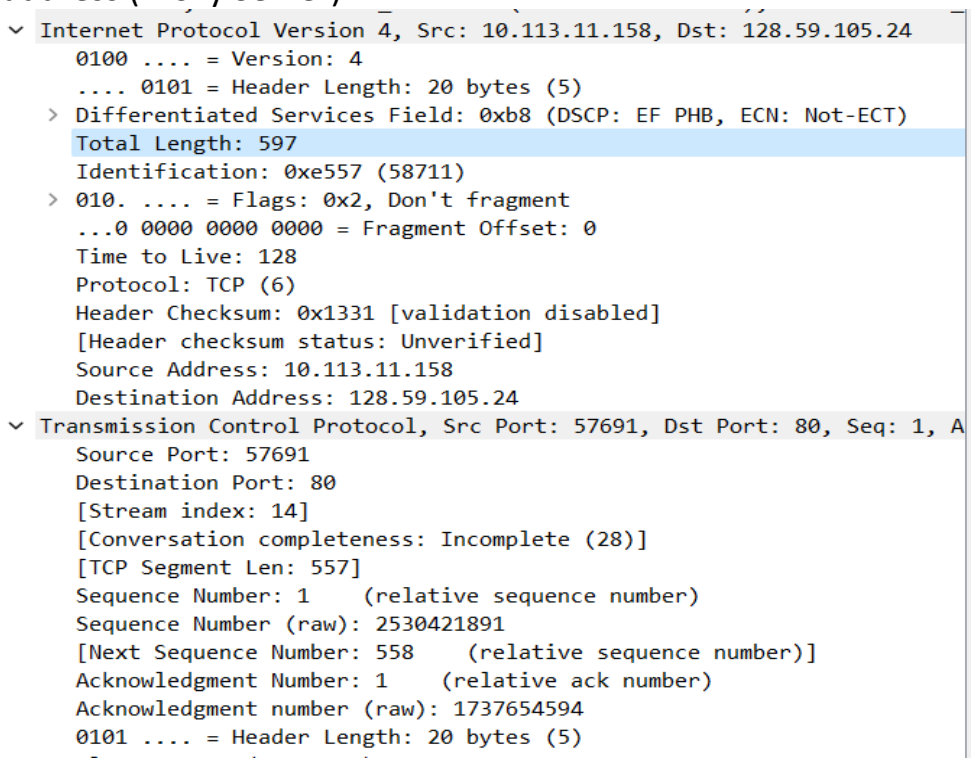
• The browser renders the HTML code and displays the webpage. Depending on the webpage design, there may be a form or text box where you can enter your comment.

• When you enter your comment and click the "submit" button, the browser sends an HTTP POST request to the web server. This request contains the data you entered in the form.

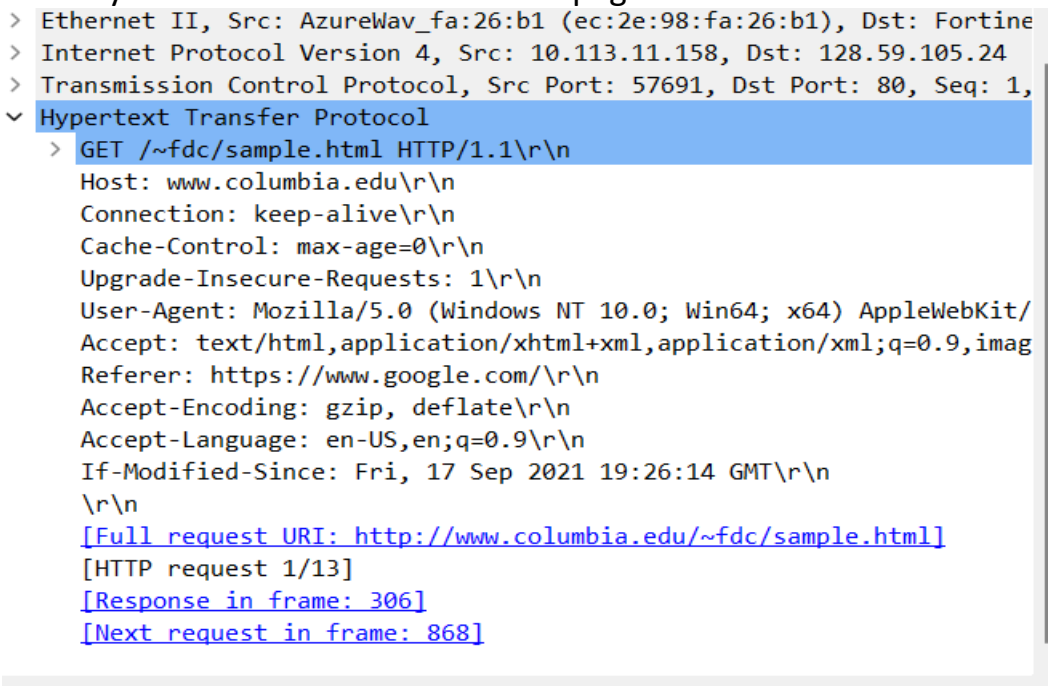
• The web server receives the POST request and processes the data. Depending on how the web application is designed, the data may be stored in a database or sent via email to the website owner.

• The web server sends an HTTP response back to the browser. This response may be a confirmation message, a redirect to a different webpage, or an error message if there was a problem processing the request.

a. Analyze the sender and destination IP address, Port address ,and Physical address (Proxy Server)



b. Analyze the Host machine and webpage name in the file server.



c. Check the acknowledgment frame for successful request and the unsuccessful request.

```
Transmission Control Protocol, Src Port: 57691, Dst Port: 80, Seq: 1,
   Source Port: 57691
   Destination Port: 80
   [Stream index: 14]
   [Conversation completeness: Incomplete (28)]
   [TCP Segment Len: 557]
   Sequence Number: 1    (relative sequence number)
   Sequence Number (raw): 2530421891
   [Next Sequence Number: 558    (relative sequence number)]
   Acknowledgment Number: 1    (relative ack number)
   Acknowledgment number (raw): 1737654594
   0101 .... = Header Length: 20 bytes (5)
```

e. Find the following from frames received for the above activity.

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

Browser and server are running in HTTP version 1.1

```
Hypertext Transfer Protocol
v GET /cu/computinghistory/ibm-manuals-03-160.jpg HTTP/1.1\r\n
  v [Expert Info (Chat/Sequence): GET /cu/computinghistory/ibm-manu
      [GET /cu/computinghistory/ibm-manuals-03-160.jpg HTTP/1.1\r\
      [Severity level: Chat]
      [Group: Sequence]
    Request Method: GET
    Request URI: /cu/computinghistory/ibm-manuals-03-160.jpg
    Request Version: HTTP/1.1
  Host: www.columbia.edu\r\n
  Connection: keep-alive\r\n
```

ii. What languages (if any) do your browser indicate that it can accept to the server?

Accepted Languages : en-GB, en-US

```
Hypertext Transfer Protocol
> GET /cu/computinghistory/ibm-manuals-03-160.jpg HTTP/1.1\r\n
  Host: www.columbia.edu\r\n
  Connection: keep-alive\r\n
  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/53
  Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q
  Referer: http://www.columbia.edu/cu/computinghistory/\r\n
  Accept-Encoding: gzip, deflate\r\n
  Accept-Language: en-US,en;q=0.9\r\n
> Cookie: BIGipServer~CUIT~www.columbia.edu-80-pool=!eNSD/xqucDRCtaGxk
  \r\n
  [Full request URI: http://www.columbia.edu/cu/computinghistory/ibm-m
  [HTTP request 1/7]
  [Response in frame: 1407]
  [Next request in frame: 1409]
```

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

Status code returned from the server to browser is 200.

```
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
```

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

```
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
  Date: Sat, 25 Nov 2023 07:11:37 GMT\r\n
  Server: Apache\r\n
  Last-Modified: Sun, 27 Feb 2022 11:59:50 GMT\r\n
  Accept-Ranges: bytes\r\n
  Vary: Accept-Encoding,User-Agent\r\n
  Content-Encoding: gzip\r\n
> Content-Length: 2479\r\n
  Keep-Alive: timeout=15, max=95\r\n
  Connection: Keep-Alive\r\n
  Content-Type: text/css\r\n
  Set-Cookie: BIGipServer~CUIT~www.columbia.edu-80-pool=!1mCVNMThVp
  \r\n
  [HTTP response 4/13]
```

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

Capture-Length = 107 bytes

```
Frame 1365: 107 bytes on wire (856 bits), 107 bytes captured (856 bit
  Section number: 1
> Interface id: 0 (\Device\NPF_{9E7F194A-99E6-4917-8027-8738F92DD67B}
  Encapsulation type: Ethernet (1)
  Arrival Time: Nov 25, 2023 12:41:37.478880000 India Standard Time
  [Time shift for this packet: 0.000000000 seconds]
  Epoch Time: 1700896297.478880000 seconds
  [Time delta from previous captured frame: 0.000000000 seconds]
  [Time delta from previous displayed frame: 0.394641000 seconds]
  [Time since reference or first frame: 39.291314000 seconds]
  Frame Number: 1365
  Frame Length: 107 bytes (856 bits)
  Capture Length: 107 bytes (856 bits)
  [Frame is marked: False]
  [Frame is ignored: False]
  [Protocols in frame: eth:ethertype:ip:tcp:http:data-text-lines]
  [Coloring Rule Name: HTTP]
  [Coloring Rule String: http || tcp.port == 80 || http2]
```

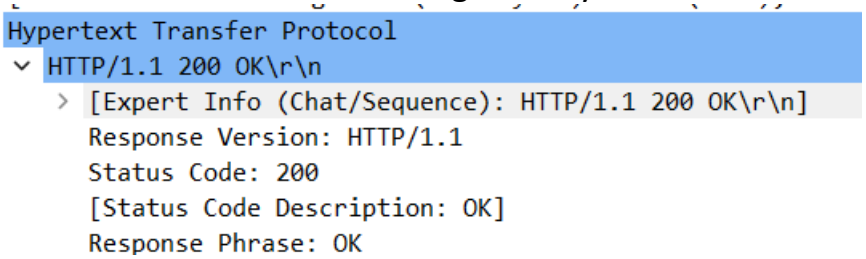
vi. By inspecting the raw data in the packet content window, do you see any headers within the data that are not displayed in the packet-listing window? If so, name one.

Ans. To view the TCP header in the packet content window, select a packet that contains TCP data, expand the "Transmission Control Protocol" section, and look for the "TCP Header" field. You can then view the raw data of the TCP header and any other headers that may be present within the packet.

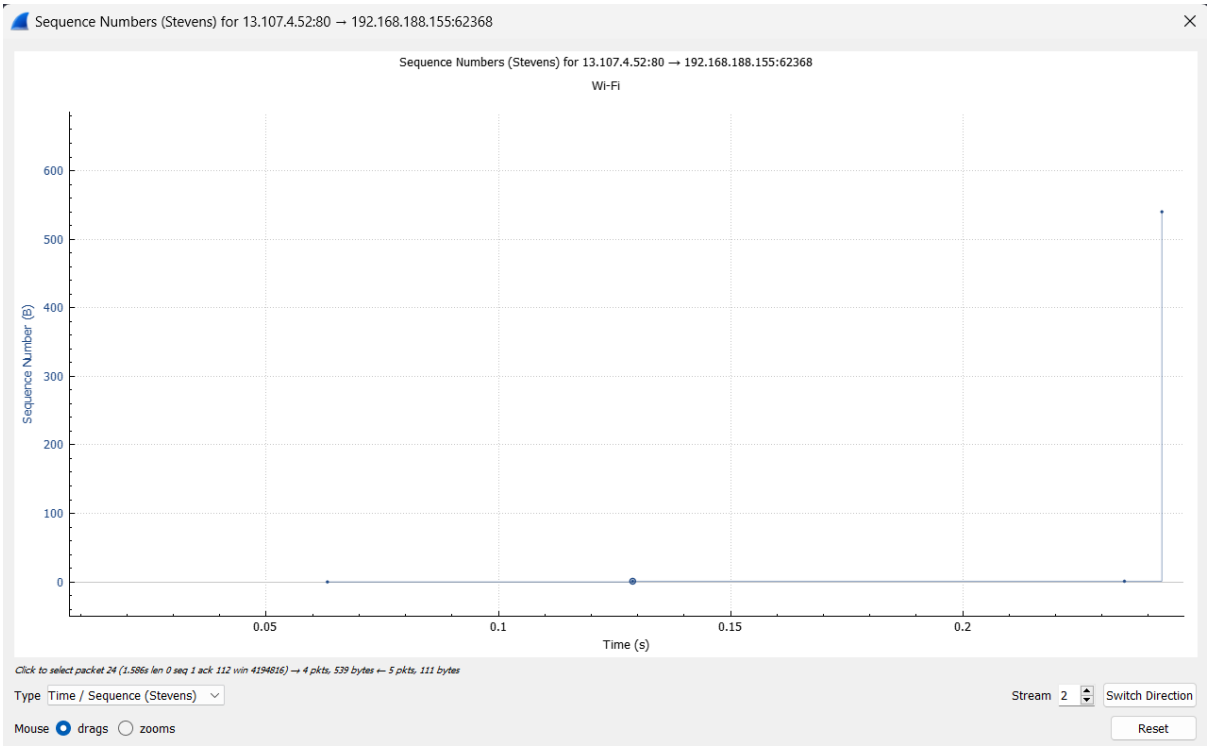
vii. Can you tell whether your browser downloaded the two images serially, or whether they were downloaded from the two websites in parallel? Explain.

Ans. If the packets for both images are interspersed in the packet-listing window and have similar timestamps, it is likely that they were downloaded in parallel. This would suggest that the browser made separate HTTP requests for each image, and the responses were received and downloaded simultaneously. On the other hand, if the packets for one image are all downloaded before the packets for the second image, it is likely that the images were downloaded serially. This would suggest that the browser made a request for one image, received and downloaded the response, and then made a separate request for the second image.
To confirm this, you can also look at the sequence numbers of the TCP packets. If the sequence numbers of the packets for both images are similar, it suggests that they were downloaded in parallel, whereas if the sequence numbers for one image are significantly higher than the other, it suggests that the images were downloaded serially.

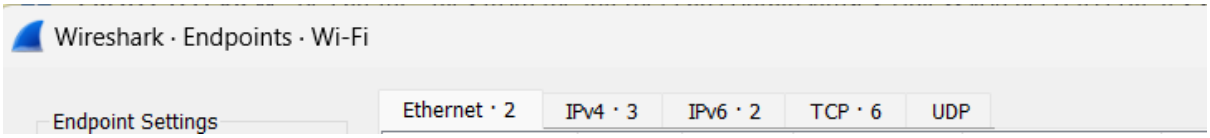
viii. What is the server's response (status code and phrase) in response to the initial HTTP GET message from your browser?

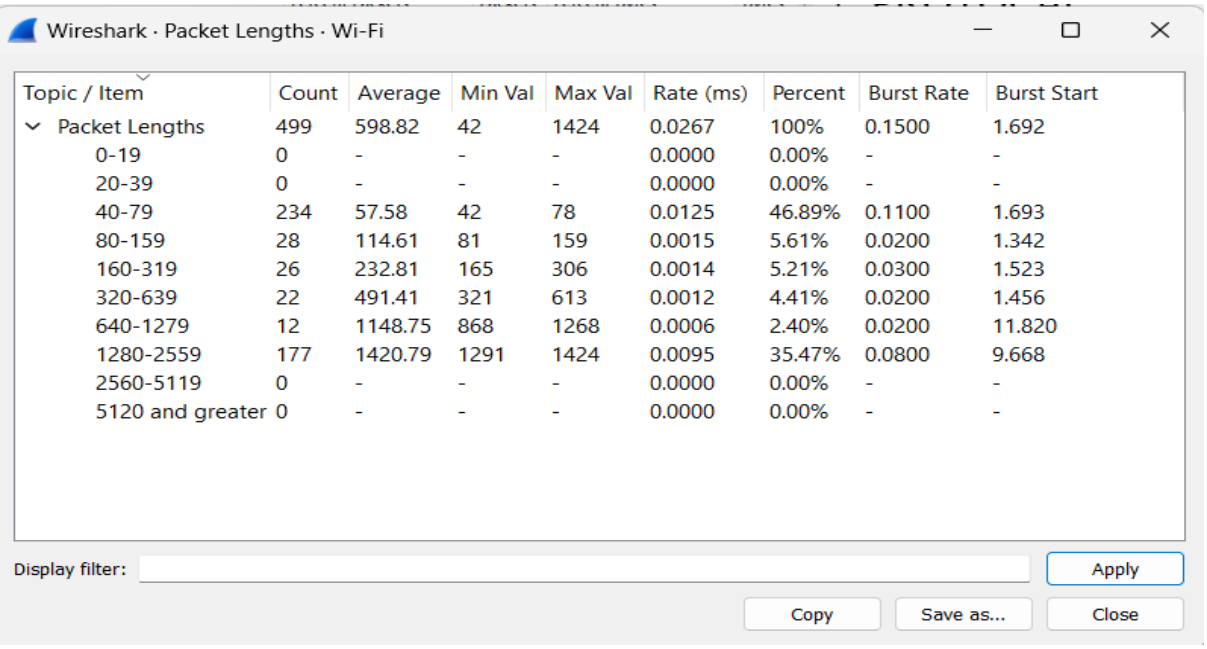


4. Open AUMS and download a file. Stop capture. Using statistics tools find the time taken for downloading the file.

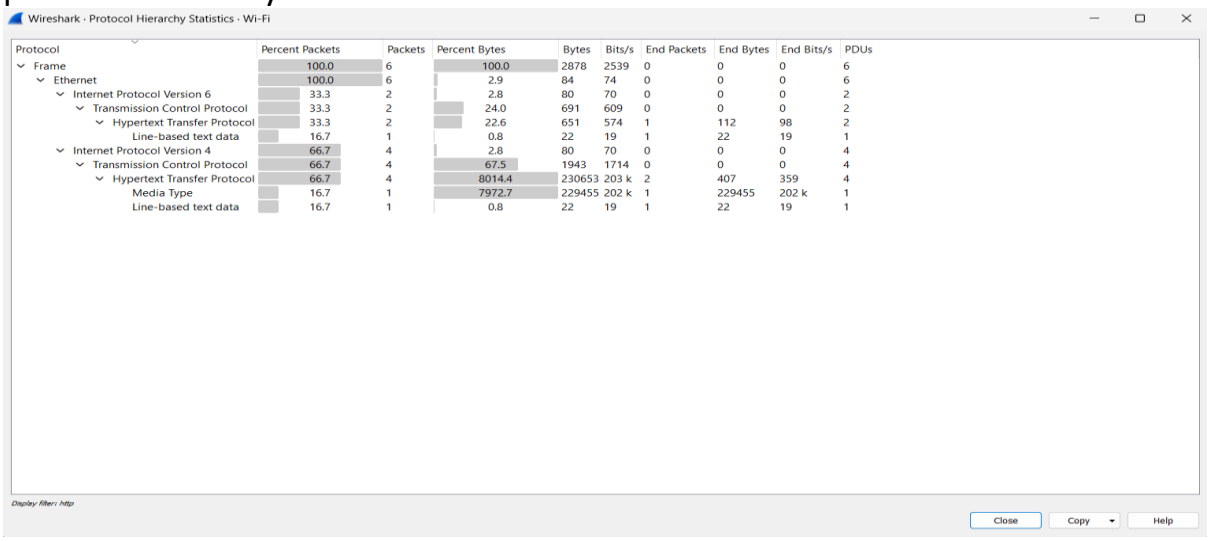


5. Open any SharePoint page and download a video file. Find the number of routing endpoints visited and the switches visited. Find the packet length transmitted and received, give the protocol hierarchy and the Flowgraph.



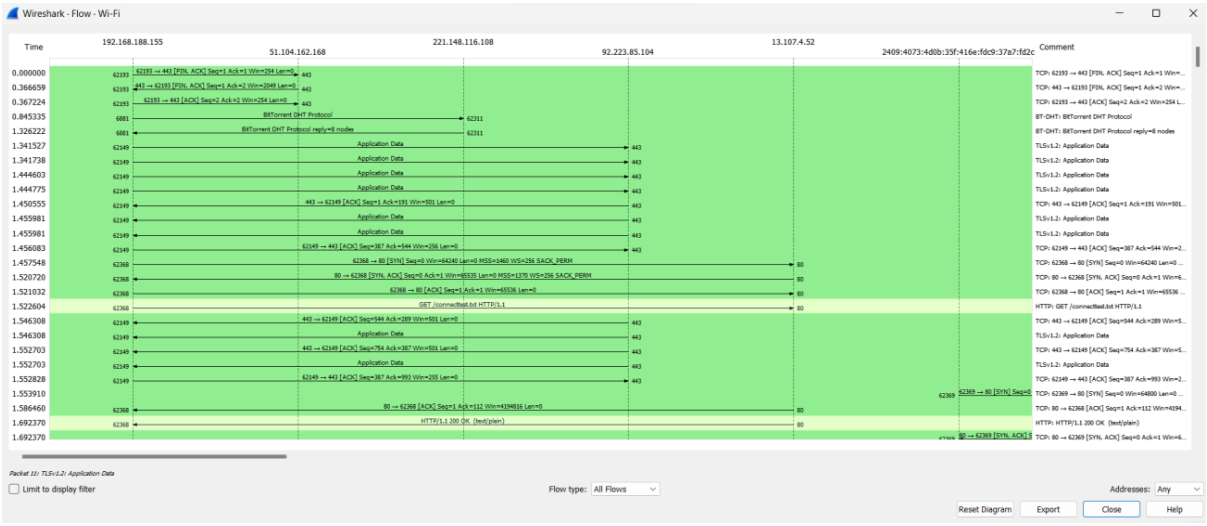


protocol hierarchy



Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s	PDUs
Frame	100.0	6	100.0	2878	2539	0	0	0	6
Ethernet	100.0	6	2.9	84	74	0	0	0	6
Internet Protocol Version 6	33.3	2	2.8	80	70	0	0	0	2
Transmission Control Protocol	33.3	2	24.0	691	609	0	0	0	2
Hypertext Transfer Protocol	33.3	2	22.6	651	574	1	112	98	2
Line-based text data	16.7	1	0.8	22	19	1	22	19	1
Internet Protocol Version 4	66.7	4	2.8	80	70	0	0	0	4
Transmission Control Protocol	66.7	4	67.5	1943	1714	0	0	0	4
Hypertext Transfer Protocol	66.7	4	8014.4	230653	203 k	2	407	359	4
Media Type	16.7	1	7972.7	229455	202 k	1	229455	202 k	1
Line-based text data	16.7	1	0.8	22	19	1	22	19	1

Flowgraph.



6. Capture the frames for the following commands in the command prompt.
print “GET /HTTP/1.0\r\n" | nc ac.amrita.ac.in 80

Start Up your web browser, and make sure your browser’s cache is cleared. Do this activity and capture frames.

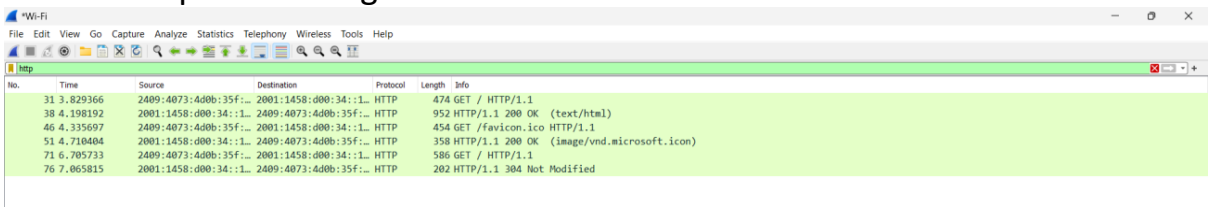
No.	Time	Source	Destination	Protocol	Length	Info
245	287.411997039	10.0.2.15	185.125.190.48	HTTP	141	GET / HTTP/1.1
247	287.869821340	185.125.190.48	10.0.2.15	HTTP	243	HTTP/1.1 204 No Content
267	587.436503275	10.0.2.15	91.189.91.49	HTTP	141	GET / HTTP/1.1
269	587.932213218	91.189.91.49	10.0.2.15	HTTP	243	HTTP/1.1 204 No Content
297	887.431082277	10.0.2.15	185.125.190.18	HTTP	141	GET / HTTP/1.1
299	887.807218501	185.125.190.18	10.0.2.15	HTTP	243	HTTP/1.1 204 No Content

7. Open your web browser, and make sure your browser’s cache is cleared. Do this activity and capture frames.

a. Enter the following URL into your browser “http://ac.amrita.ac.in”

b. Quickly enter the same URL into your browser again (or simply select the refresh button on your browser)

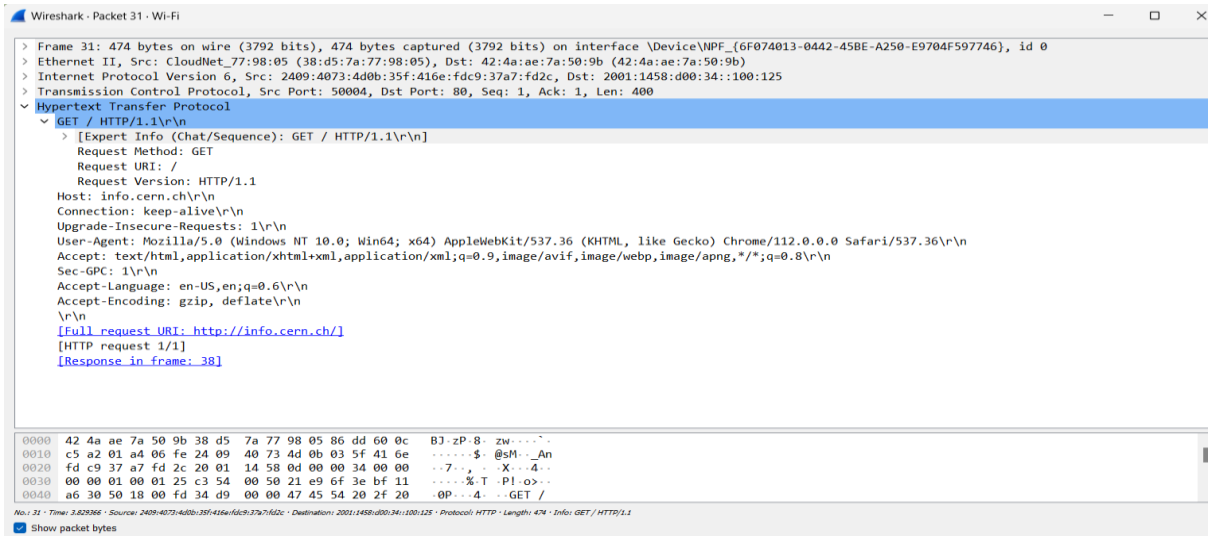
c. Stop Wireshark packet capture and enter “HTTP” in the display-filter-specification window, so that only captured HTTP messages will be displayed later in the packet listing window.



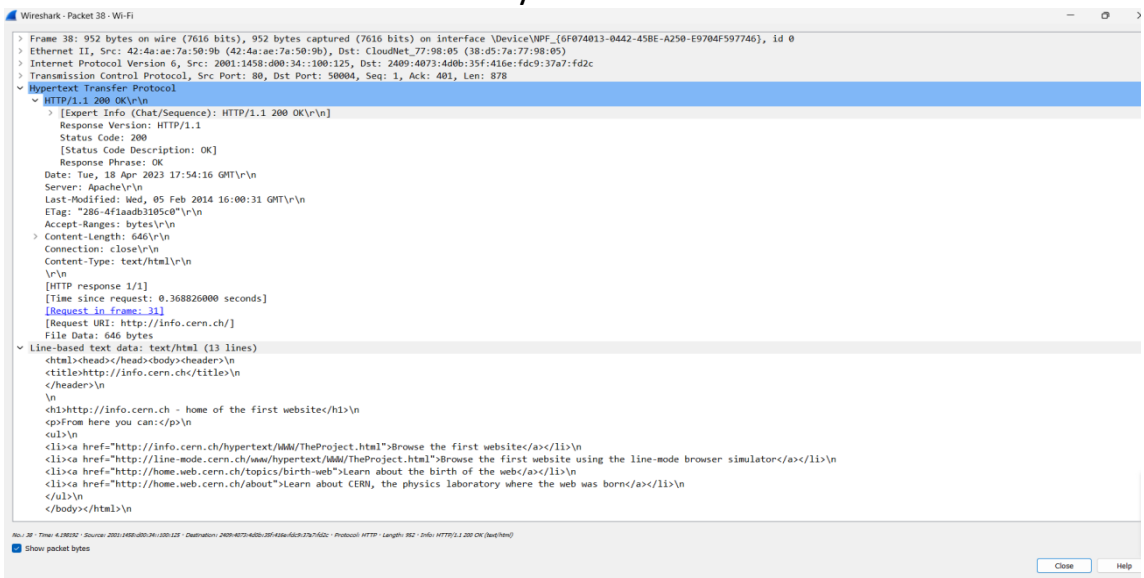
d. Answer the following questions:

i. 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 request?

No.



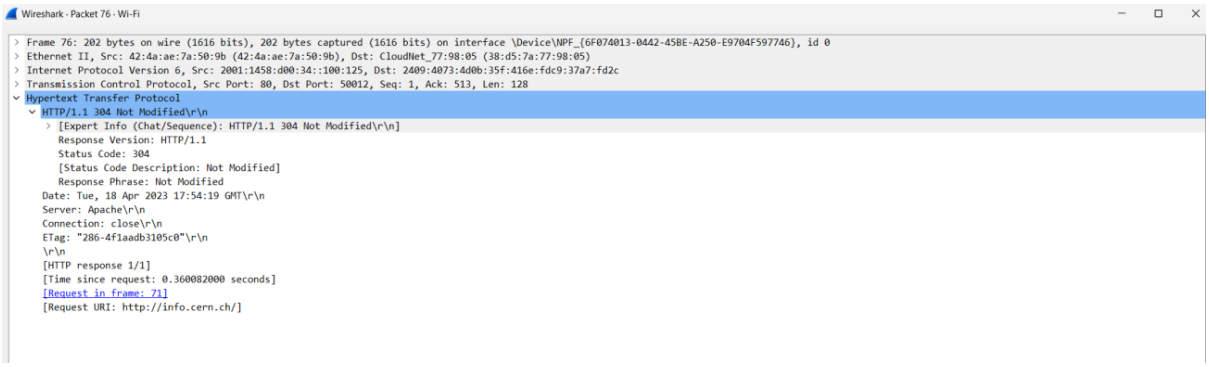
ii. Inspect the contents of the server response. Did the server explicitly return the contents of the file? How can you tell?



iii. 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.

iv. 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.



v. How many HTTP GET request messages were sent by your browser?

The number of HTTP GET request messages sent by the browser depends on the number of files requested. Here it is 3 HTTP GET request messages

vi. How many data-containing TCP segments were needed to carry the single HTTP response? What is the size for each of the segments?

The number of data-containing TCP segments needed to carry the single HTTP response depends on the size of the file and the maximum segment size (MSS)

of the TCP connection. The size of each segment will be less than or equal to the MSS.

vii. What is the status code and phrase associated with the response to the HTTP GET request?

The status code and phrase associated with the response to the HTTP GET request will be either HTTP 200 "OK" or HTTP 304 "Not Modified", depending on whether the file has been modified since the last time the browser requested it.