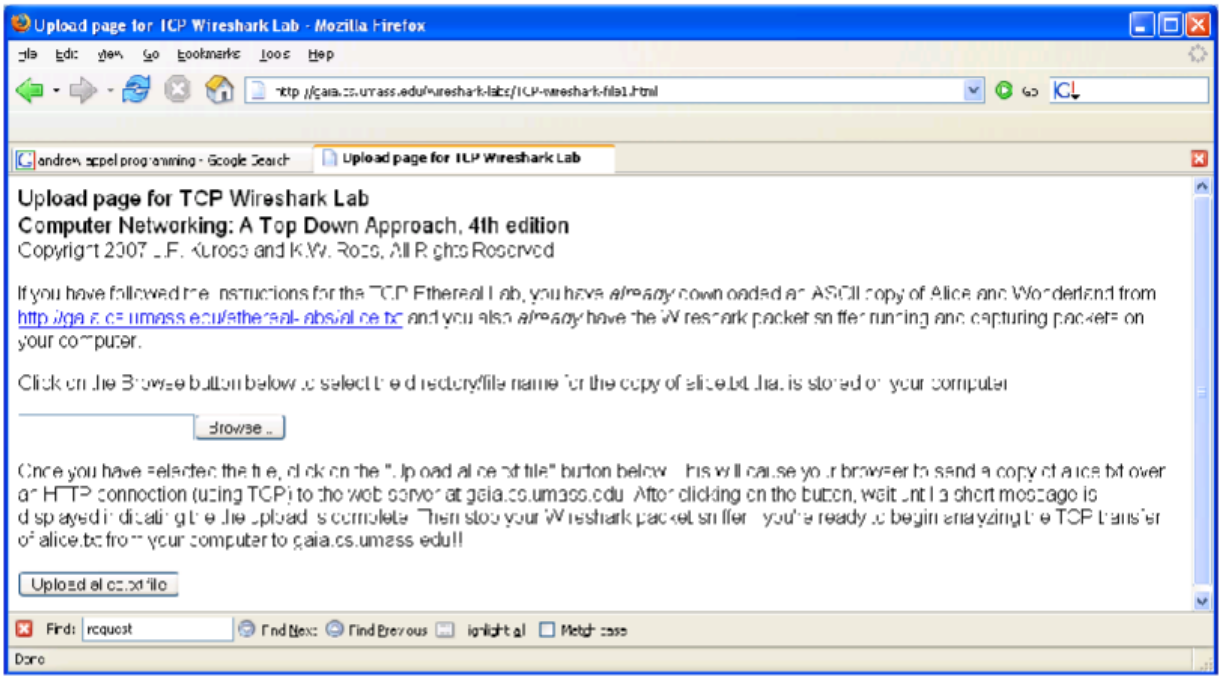
Wireshark Lab - TCP

1. Capturing a bulk TCP transfer from your computer to a remote server

Before beginning our exploration of TCP, we'll need to use Wireshark to obtain a packet trace of the TCP transfer of a file from your computer to a remote server. You'll do so by accessing a Web page that will allow you to enter the name of a file stored on your computer (which contains the ASCII text of Alice in Wonderland), and then transfer the file to a Web server using the HTTP POST method (see section 2.2.3 in the text). We're using the POST method rather than the GET method as we'd like to transfer a large amount of data from your computer to another computer. Of course, we'll be running Wireshark during this time to obtain the trace of the TCP segments sent and received from your computer.

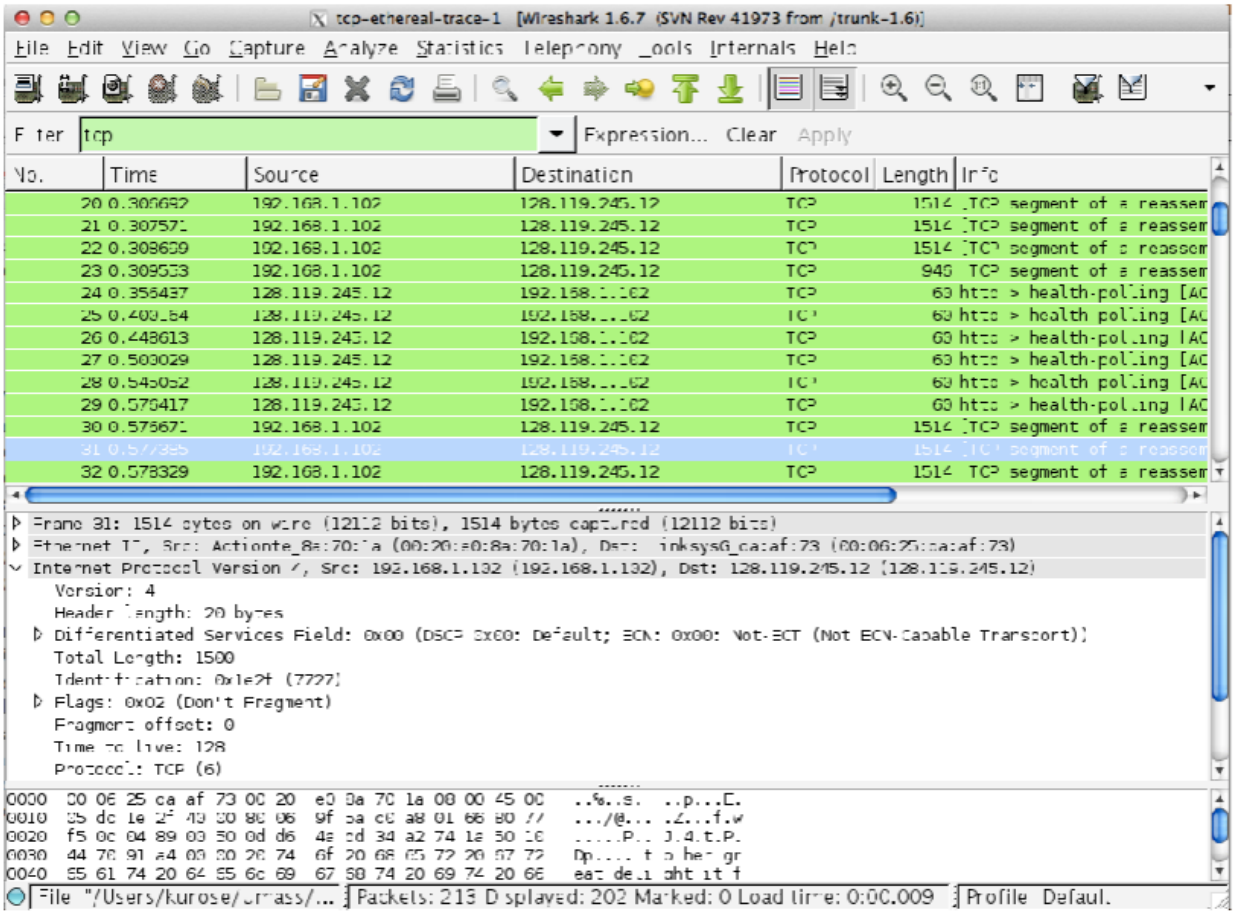
Do the following:

• Start up your web browser. Go the http://gaia.cs.umass.edu/wireshark-labs/alice.txt and retrieve an ASCII copy of Alice in Wonderland. Store this file somewhere on your computer.
• Next go to http://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.html.
• You should see a screen that looks like:



• Use the Browse button in this form to enter the name of the file (full path name)

on your computer containing Alice in Wonderland (or do so manually). Don't yet press the "Upload alice.txt file" button.
• Now start up Wireshark and begin packet capture (Capture->Start) and then press OK on the Wireshark Packet Capture Options screen (we'll not need to select any options here).
• Returning to your browser, press the "Upload alice.txt file" button to upload the file to the gaia.cs.umass.edu server. Once the file has been uploaded, a short congratulations message will be displayed in your browser window.
• Stop Wireshark packet capture. Your Wireshark window should look similar to the window shown below.



If you are unable to run Wireshark on a live network connection, you can download a packet trace file that was captured while following the steps above on one of the author's computers2. You may well find it valuable to download this trace even if you've captured your own trace and use it, as well as your own trace, when you explore the questions below.

2. A first look at the captured trace.

Before analyzing the behavior of the TCP connection in detail, let's take a high level

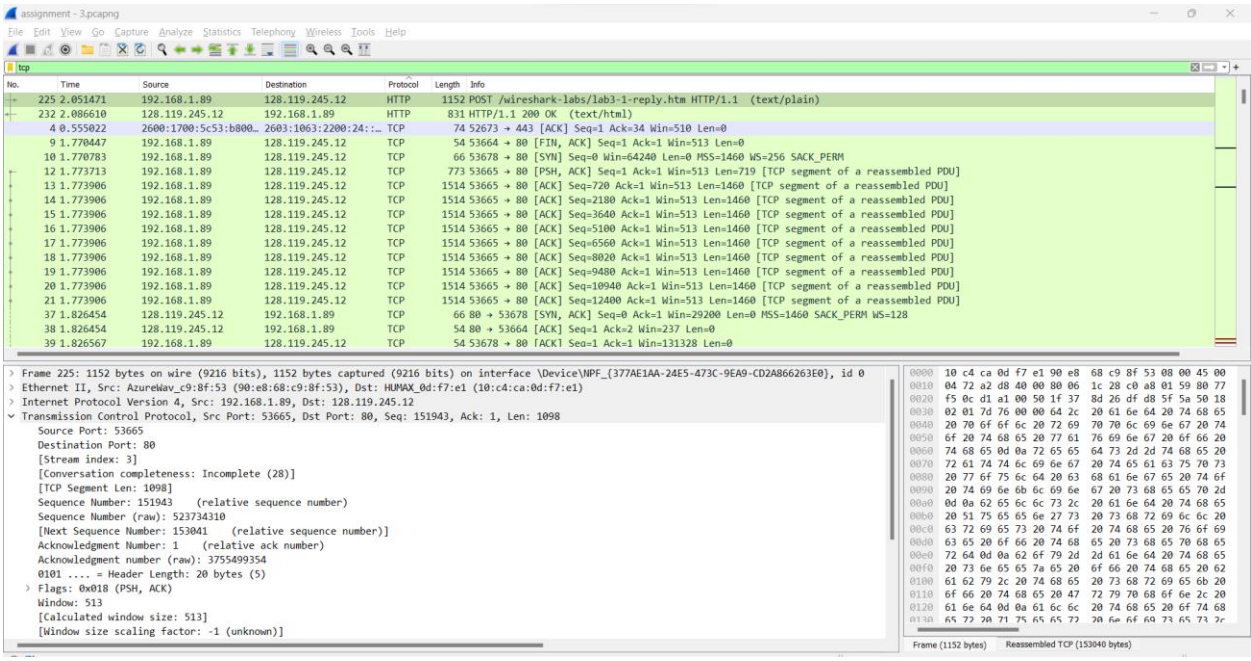
view of the trace.
• First, filter the packets displayed in the Wireshark window by entering "tcp" (lowercase, no quotes, and don't forget to press return after entering!) into the display filter specification window towards the top of the Wireshark window.
What you should see is series of TCP and HTTP messages between your computer and gaia.cs.umass.edu. You should see the initial three-way handshake containing a SYN message. You should see an HTTP POST message. Depending on the version of Wireshark you are using, you might see a series of "HTTP Continuation" messages being sent from your computer to gaia.cs.umass.edu. Recall from our discussion in the earlier HTTP Wireshark lab, that is no such thing as an HTTP Continuation message – this is Wireshark's way of indicating that there are multiple TCP segments being used to carry a single HTTP message. In more recent versions of Wireshark, you'll see "[TCP segment of a reassembled PDU]" in the Info column of the Wireshark display to indicate that this TCP segment contained data that belonged to an upper layer protocol message (in our case here, HTTP). You should also see TCP ACK segments being returned from gaia.cs.umass.edu to your computer.

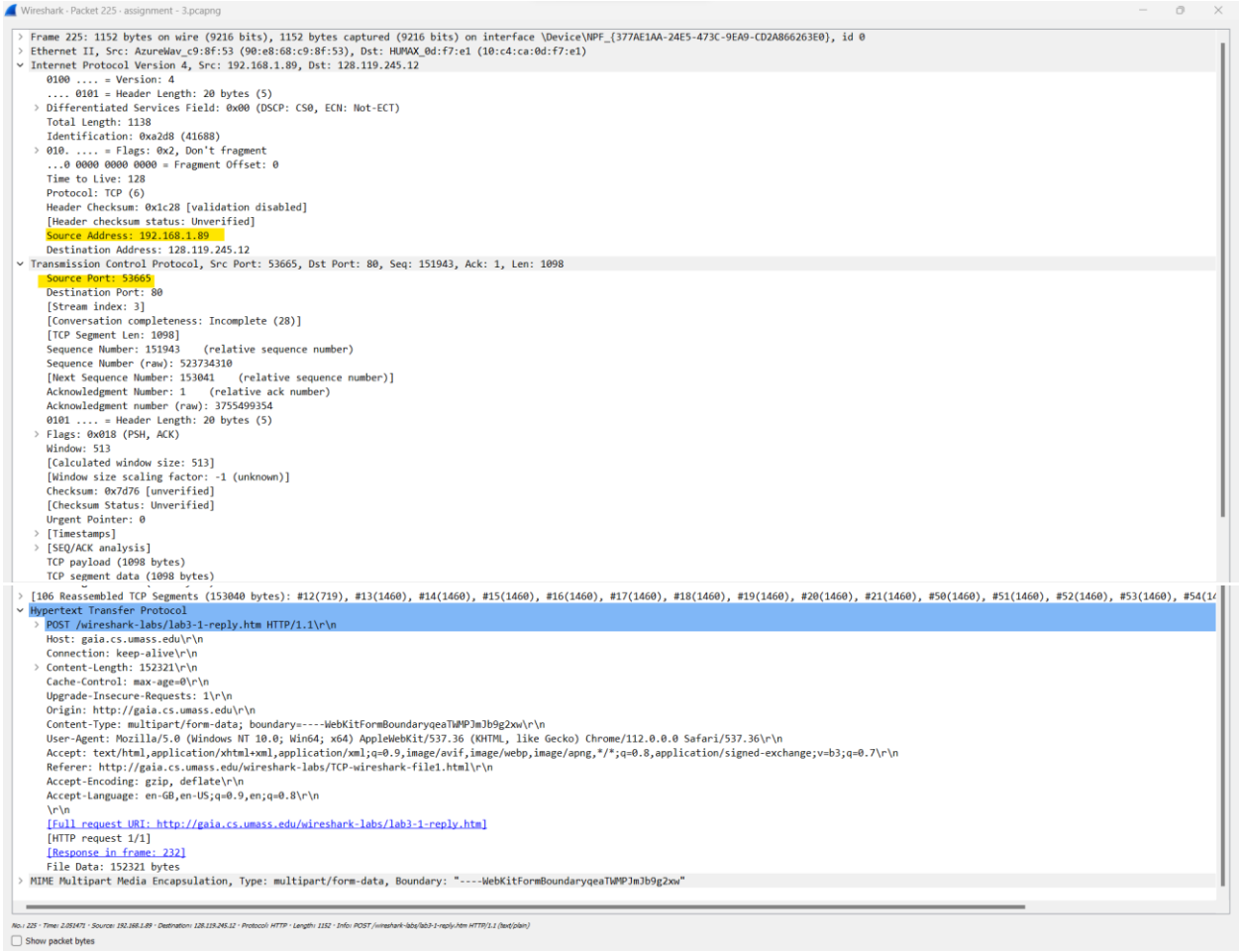
Answer the following questions, by opening the Wireshark captured packet file tcp-ethereal-trace-1 in http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip (that is download the trace and open that trace in Wireshark; see footnote 2). Whenever possible, when answering a question, you should hand in a printout of the packet(s) within the trace that you used to answer the question asked. Annotate the printout3 to explain your answer. To print a packet, use File->Print, choose Selected packet only, choose Packet summary line, and select the minimum amount of packet detail that you need to answer the question.

1. **What is the IP address and TCP port number used by the client computer (source) that is transferring the file to gaia.cs.umass.edu? To answer this question, it's probably easiest to select an HTTP message and explore the details of the TCP packet used to carry this HTTP message, using the "details of the selected packet header window" (refer to Figure 2 in the "Getting Started with Wireshark" Lab if you're uncertain about the Wireshark windows.**

   IP address: 192.168.1.89 (source IP)

   TCP port number: 53665 (source), 80 (destination)


assignment - 3.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp
No. Time Source Destination Protocol Length Info
225 2.051471 192.168.1.89 128.119.245.12 HTTP 1152 POST /wireshark-labs/lab3-1-reply.htm HTTP/1.1 (text/plain)
232 2.086610 128.119.245.12 192.168.1.89 HTTP 831 HTTP/1.1 200 OK (text/html)
4 0.555022 2600:1700:5c53:b800… 2603:1063:2200:24::… TCP 74 52673 → 443 [ACK] Seq=1 Ack=34 Win=510 Len=0
9 1.770447 192.168.1.89 128.119.245.12 TCP 54 53664 → 80 [FIN, ACK] Seq=1 Ack=1 Win=513 Len=0
10 1.770783 192.168.1.89 128.119.245.12 TCP 66 53678 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
12 1.773713 192.168.1.89 128.119.245.12 TCP 773 53665 → 80 [PSH, ACK] Seq=1 Ack=1 Win=513 Len=719 [TCP segment of a reassembled PDU]
13 1.773906 192.168.1.89 128.119.245.12 TCP 1514 53665 → 80 [ACK] Seq=720 Ack=1 Win=513 Len=1460 [TCP segment of a reassembled PDU]
14 1.773906 192.168.1.89 128.119.245.12 TCP 1514 53665 → 80 [ACK] Seq=2180 Ack=1 Win=513 Len=1460 [TCP segment of a reassembled PDU]
15 1.773906 192.168.1.89 128.119.245.12 TCP 1514 53665 → 80 [ACK] Seq=3640 Ack=1 Win=513 Len=1460 [TCP segment of a reassembled PDU]
16 1.773906 192.168.1.89 128.119.245.12 TCP 1514 53665 → 80 [ACK] Seq=5100 Ack=1 Win=513 Len=1460 [TCP segment of a reassembled PDU]
17 1.773906 192.168.1.89 128.119.245.12 TCP 1514 53665 → 80 [ACK] Seq=6560 Ack=1 Win=513 Len=1460 [TCP segment of a reassembled PDU]
18 1.773906 192.168.1.89 128.119.245.12 TCP 1514 53665 → 80 [ACK] Seq=8020 Ack=1 Win=513 Len=1460 [TCP segment of a reassembled PDU]
19 1.773906 192.168.1.89 128.119.245.12 TCP 1514 53665 → 80 [ACK] Seq=9480 Ack=1 Win=513 Len=1460 [TCP segment of a reassembled PDU]
20 1.773906 192.168.1.89 128.119.245.12 TCP 1514 53665 → 80 [ACK] Seq=10940 Ack=1 Win=513 Len=1460 [TCP segment of a reassembled PDU]
21 1.773906 192.168.1.89 128.119.245.12 TCP 1514 53665 → 80 [ACK] Seq=12400 Ack=1 Win=513 Len=1460 [TCP segment of a reassembled PDU]
37 1.826454 128.119.245.12 192.168.1.89 TCP 66 80 → 53678 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM WS=128
38 1.826454 128.119.245.12 192.168.1.89 TCP 54 80 → 53664 [ACK] Seq=1 Ack=2 Win=237 Len=0
39 1.826567 192.168.1.89 128.119.245.12 TCP 54 53678 → 80 [ACK] Seq=1 Ack=1 Win=131328 Len=0
Frame 225: 1152 bytes on wire (9216 bits), 1152 bytes captured (9216 bits) on interface \Device\NPF_{377AE1AA-24E5-473C-9EA9-CD2A866263E0}, id 0
Ethernet II, Src: AzureWav_c9:8f:53 (90:e8:68:c9:8f:53), Dst: HUMAX_0d:f7:e1 (10:c4:ca:0d:f7:e1)
Internet Protocol Version 4, Src: 192.168.1.89, Dst: 128.119.245.12
Transmission Control Protocol, Src Port: 53665, Dst Port: 80, Seq: 151943, Ack: 1, Len: 1098
Source Port: 53665
Destination Port: 80
[Stream index: 3]
[Conversation completeness: Incomplete (28)]
[TCP Segment Len: 1098]
Sequence Number: 151943 (relative sequence number)
Sequence Number (raw): 523734310
[Next Sequence Number: 153041 (relative sequence number)]
Acknowledgment Number: 1 (relative ack number)
Acknowledgment number (raw): 3755499354
0101 .... = Header Length: 20 bytes (5)
Flags: 0x018 (PSH, ACK)
Window: 513
[Calculated window size: 513]
[Window size scaling factor: -1 (unknown)]
0000 10 c4 ca 0d f7 e1 90 e8 68 c9 8f 53 08 00 45 00
0010 04 72 a2 d8 40 00 80 06 1c 28 c0 a8 01 59 80 77
0020 f5 0c d1 a1 00 50 1f 37 8d 26 df d8 5f 5a 50 18
0030 02 01 7d 76 00 00 64 2c 20 61 6e 64 20 74 68 65
0040 20 70 6f 6f 6c 20 72 69 70 70 6c 69 6e 67 20 74
0050 6f 20 74 68 65 20 77 61 76 69 6e 67 20 6f 66 20
0060 74 68 65 0d 0a 72 65 65 64 73 2d 2d 74 68 65 20
0070 72 61 74 74 6c 69 6e 67 20 74 65 61 63 75 70 73
0080 20 77 6f 75 6c 64 20 63 68 61 6e 67 65 20 74 6f
0090 20 74 69 6e 6b 6c 69 6e 67 20 73 68 65 65 70 2d
00a0 0d 0a 62 65 6c 6c 73 2c 20 61 6e 64 20 74 68 65
00b0 20 51 75 65 65 6e 27 73 20 73 68 72 69 6c 6c 20
00c0 63 72 69 65 73 20 74 6f 20 74 68 65 20 76 6f 69
00d0 63 65 20 6f 66 20 74 68 65 20 73 68 65 70 68 65
00e0 72 64 0d 0a 62 6f 79 2d 2d 61 6e 64 20 74 68 65
00f0 20 73 6e 65 65 7a 65 20 6f 66 20 74 68 65 20 62
0100 61 62 79 2c 20 74 68 65 20 73 68 72 69 65 6b 20
0110 6f 66 20 74 68 65 20 47 72 79 70 68 6f 6e 2c 20
0120 61 6e 64 0d 0a 61 6c 6c 20 74 68 65 20 6f 74 68
0130 65 72 20 71 75 65 65 72 20 6e 6f 69 73 65 73 2c
Frame (1152 bytes)
Reassembled TCP (153040 bytes)

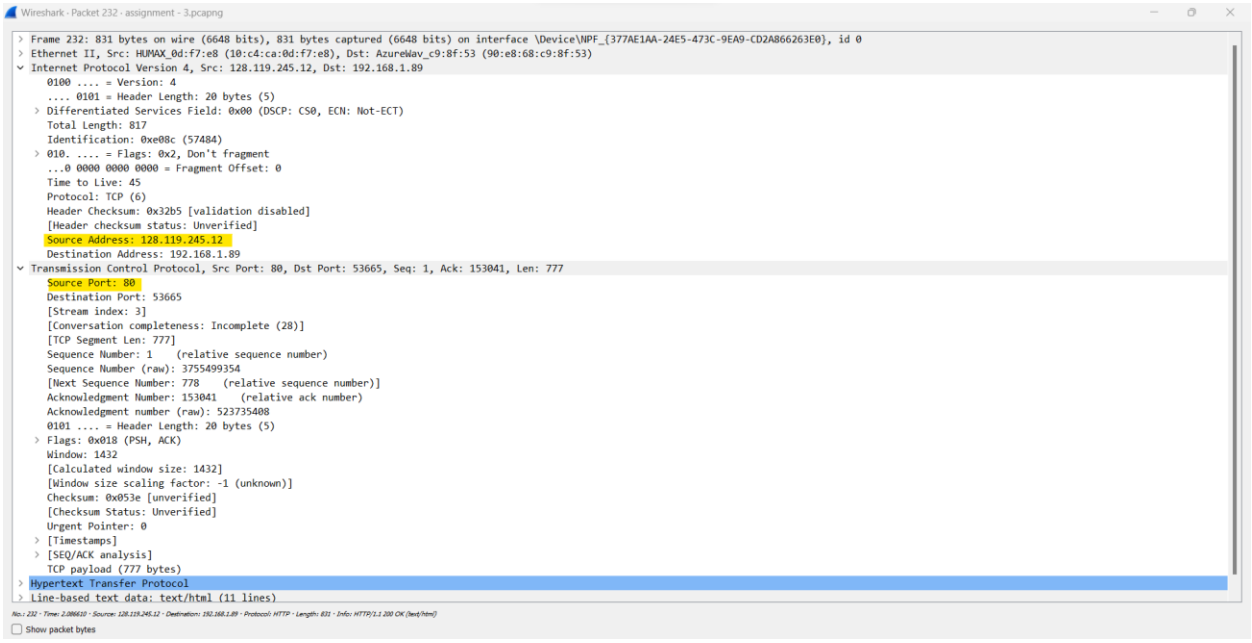

Wireshark · Packet 225 · assignment - 3.pcapng
Frame 225: 1152 bytes on wire (9216 bits), 1152 bytes captured (9216 bits) on interface \Device\NPF_{377AE1AA-24E5-473C-9EA9-CD2A866263E0}, id 0
Ethernet II, Src: AzureWav_c9:8f:53 (90:e8:68:c9:8f:53), Dst: HUMAX_0d:f7:e1 (10:c4:ca:0d:f7:e1)
Internet Protocol Version 4, Src: 192.168.1.89, Dst: 128.119.245.12
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 1138
Identification: 0xa2d8 (41688)
010. .... = Flags: 0x2, Don't fragment
...0 0000 0000 0000 = Fragment Offset: 0
Time to Live: 128
Protocol: TCP (6)
Header Checksum: 0x1c28 [validation disabled]
[Header checksum status: Unverified]
Source Address: 192.168.1.89
Destination Address: 128.119.245.12
Transmission Control Protocol, Src Port: 53665, Dst Port: 80, Seq: 151943, Ack: 1, Len: 1098
Source Port: 53665
Destination Port: 80
[Stream index: 3]
[Conversation completeness: Incomplete (28)]
[TCP Segment Len: 1098]
Sequence Number: 151943 (relative sequence number)
Sequence Number (raw): 523734310
[Next Sequence Number: 153041 (relative sequence number)]
Acknowledgment Number: 1 (relative ack number)
Acknowledgment number (raw): 3755499354
0101 .... = Header Length: 20 bytes (5)
Flags: 0x018 (PSH, ACK)
Window: 513
[Calculated window size: 513]
[Window size scaling factor: -1 (unknown)]
Checksum: 0x7d76 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
[Timestamps]
[SEQ/ACK analysis]
TCP payload (1098 bytes)
TCP segment data (1098 bytes)
[106 Reassembled TCP Segments (153040 bytes): #12(719), #13(1460), #14(1460), #15(1460), #16(1460), #17(1460), #18(1460), #19(1460), #20(1460), #21(1460), #50(1460), #51(1460), #52(1460), #53(1460), #54(1
Hypertext Transfer Protocol
POST /wireshark-labs/lab3-1-reply.htm HTTP/1.1\r\n
Host: gaia.cs.umass.edu\r\n
Connection: keep-alive\r\n
Content-Length: 152321\r\n
Cache-Control: max-age=0\r\n
Upgrade-Insecure-Requests: 1\r\n
Origin: http://gaia.cs.umass.edu\r\n
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryqeaTWMPJmJb9g2xw\r\n
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/112.0.0.0 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
Referer: http://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.html\r\n
Accept-Encoding: gzip, deflate\r\n
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8\r\n
\r\n
[Full request URI: http://gaia.cs.umass.edu/wireshark-labs/lab3-1-reply.htm]
[HTTP request 1/1]
[Response in frame: 232]
File Data: 152321 bytes
MIME Multipart Media Encapsulation, Type: multipart/form-data, Boundary: "----WebKitFormBoundaryqeaTWMPJmJb9g2xw"
Show packet bytes

2. **What is the IP address of gaia.cs.umass.edu? On what port number is it sending and receiving TCP segments for this connection?**

gaia.cs.umass.edu IP address: 128.119.245.12

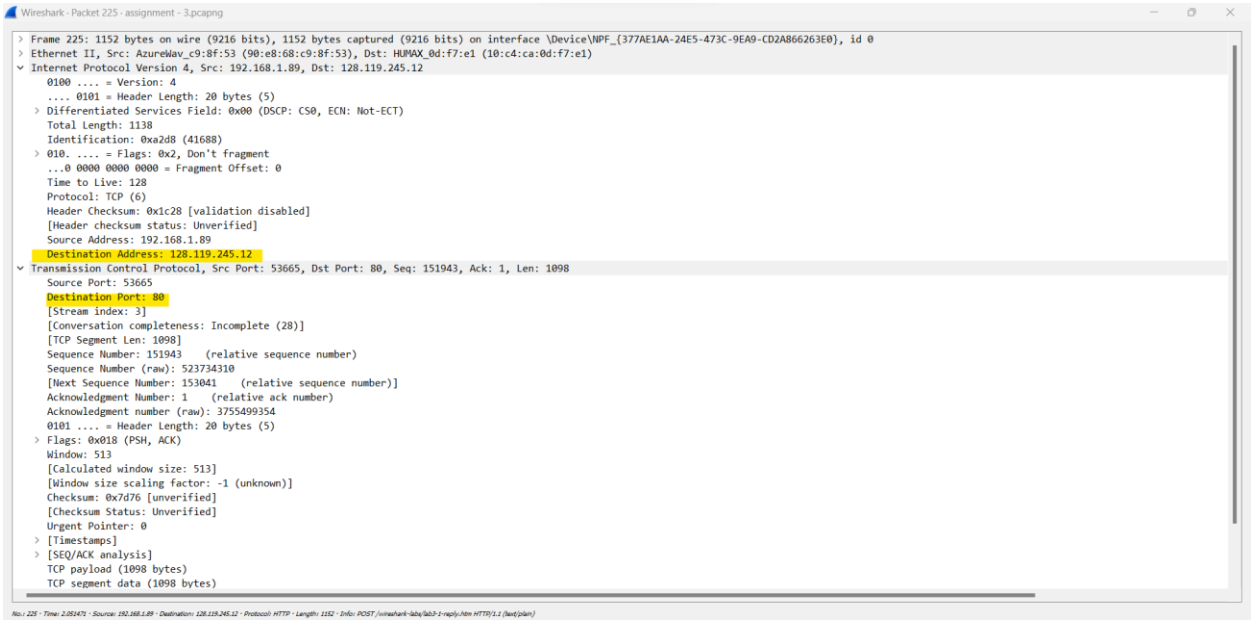
Sending data on the below port:

TCP port number: 80



Receiving data on below address and port:

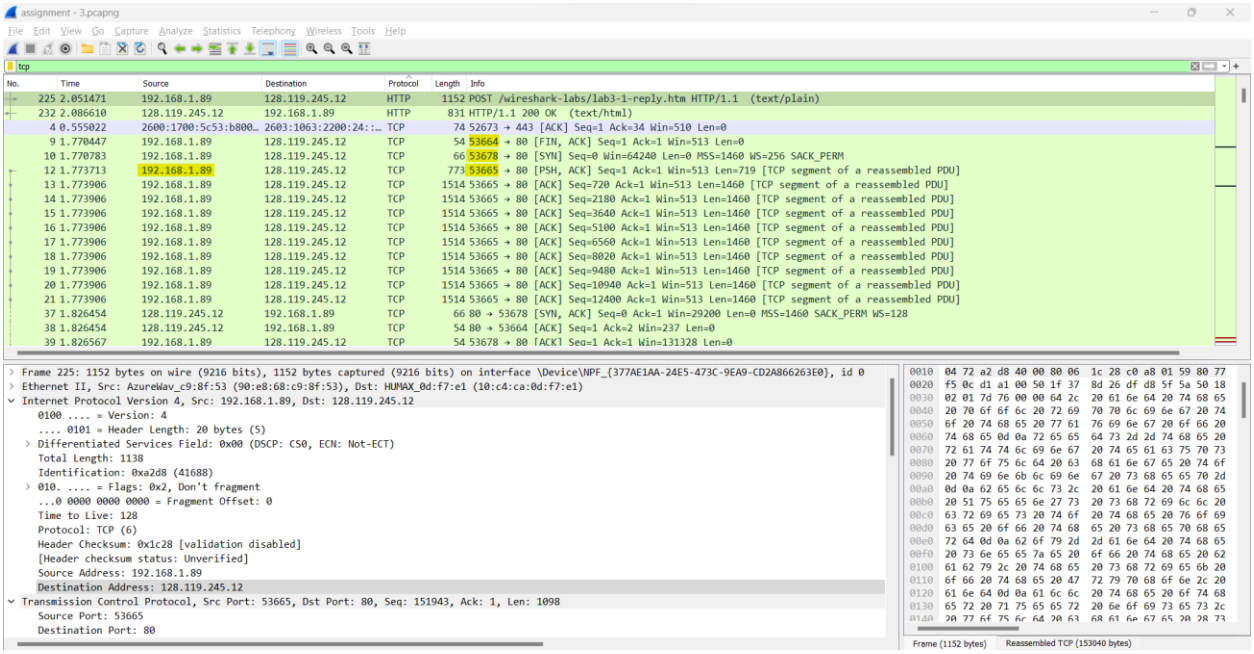
TCP port number: 80



3. **What is the IP address and TCP port number used by your client's computer (source) to transfer the file to gaia.cs.umass.edu?**

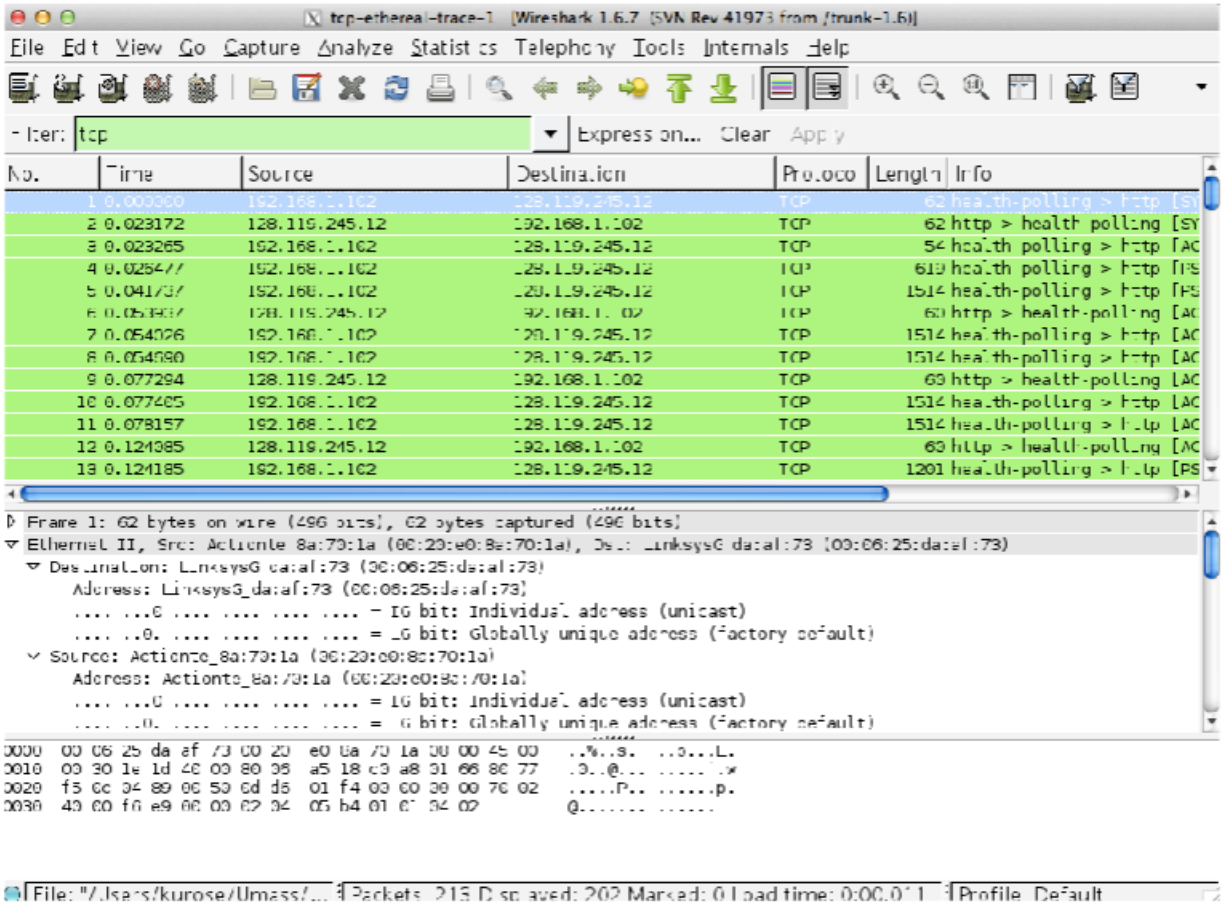
IP address: 192.168.1.89

TCP port number: 53665 is being used for the transfer of files (It is also using 53664 for FIN, ACK, and 53678 for SYN)



Since this lab is about TCP rather than HTTP, let's change Wireshark's "listing of captured packets" window so that it shows information about the TCP segments containing the HTTP messages, rather than about the HTTP messages. To have Wireshark do this, select Analyze->Enabled Protocols. Then uncheck the HTTP box and

select OK. You should now see a Wireshark window that looks like:



This is what we're looking for - a series of TCP segments sent between your computer and gaia.cs.umass.edu. We will use the packet trace that you have captured (and/or the packet trace tcp-ethereal-trace-1 in http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip; see earlier footnote) to study TCP behavior in the rest of this lab.

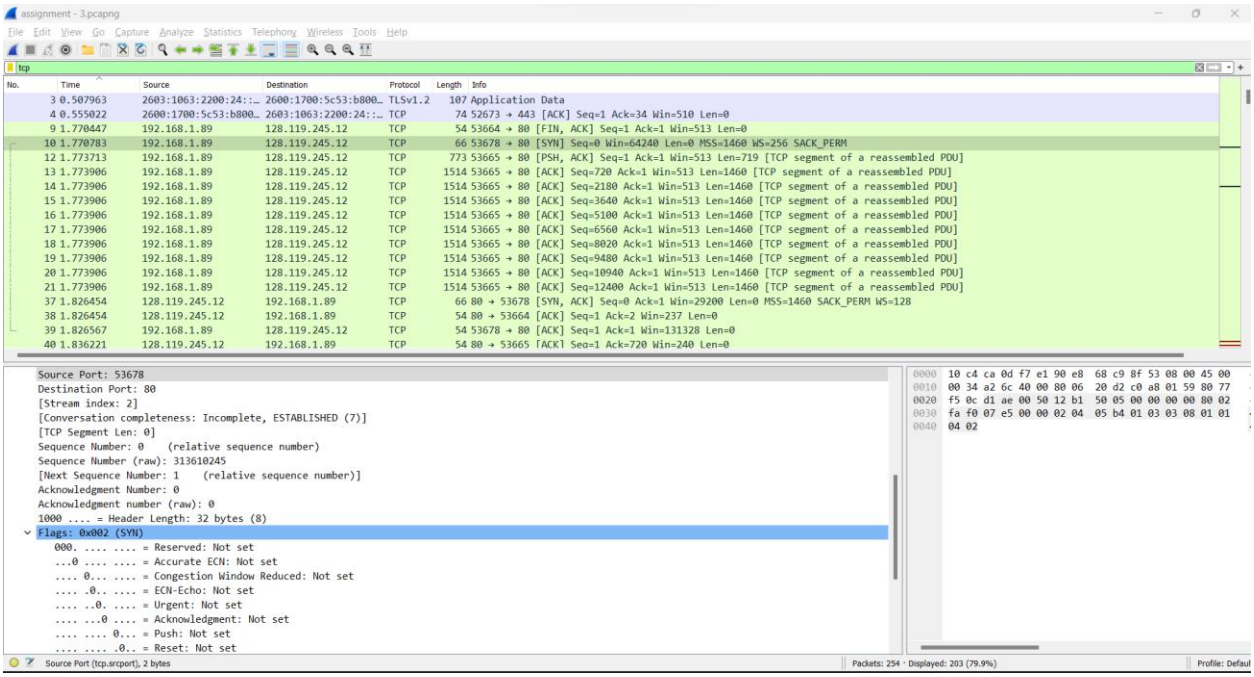
3. TCP Basics

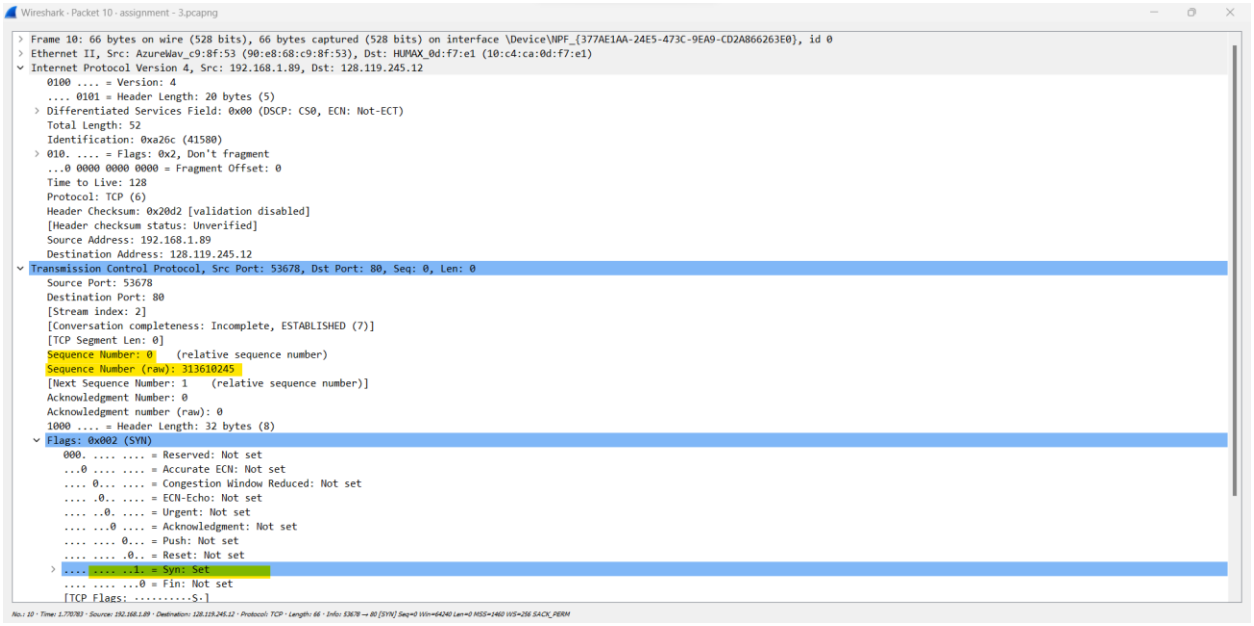
Answer the following questions for the TCP segments:

Note: Close the tab in which you've received the congratulation.

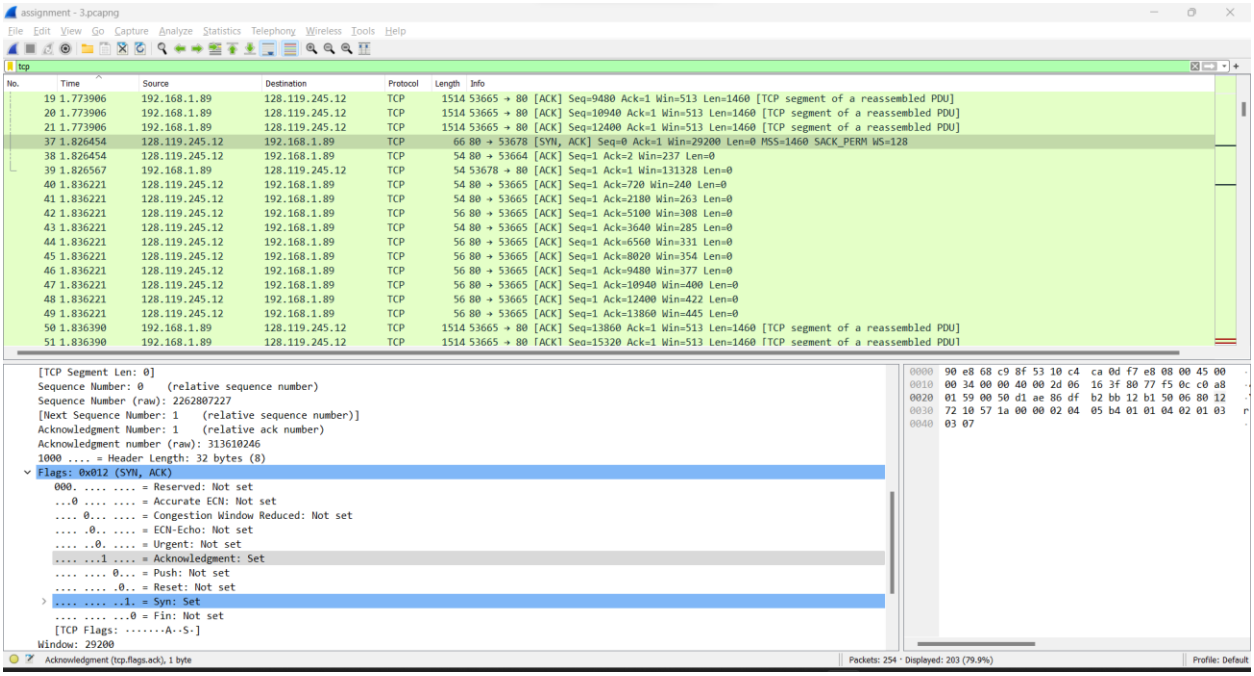
4. **What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client computer and gaia.cs.umass.edu? What is it in the segment that identifies the segment as an SYN segment?**

The sequence number of the TCP SYN segment used to initiate the TCP connection between the client computer and gaia.cs.umass.edu is 0. The SYN flag is set to 1 and it indicates that this segment is a SYN segment.

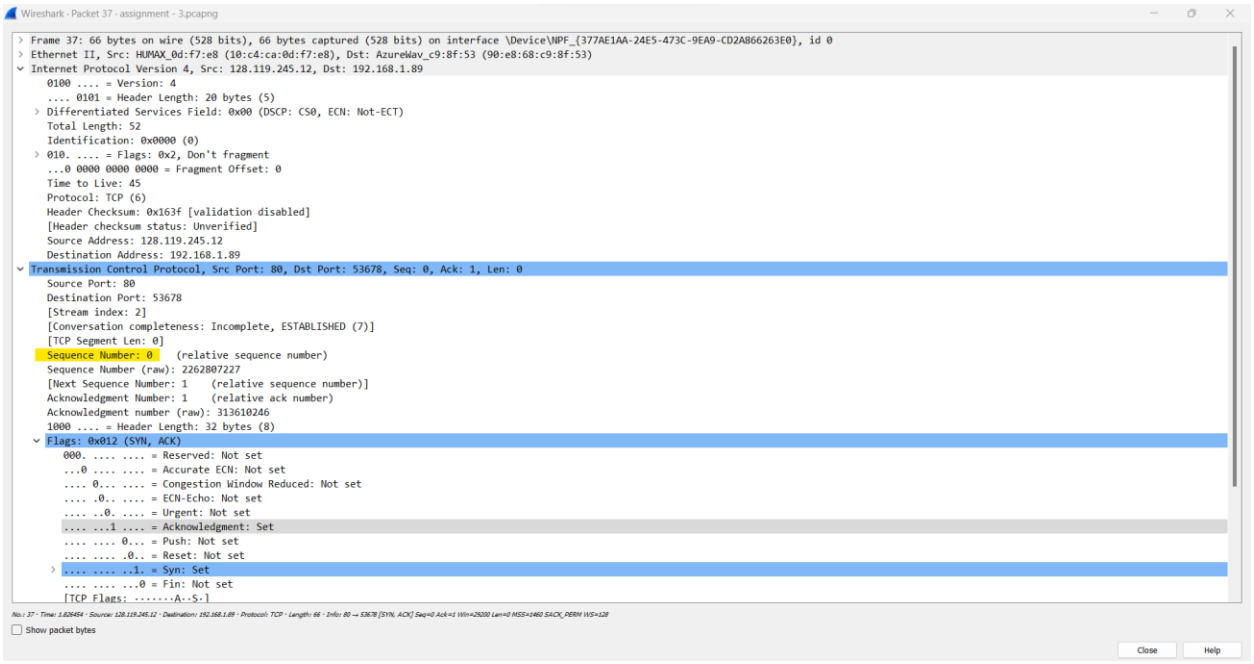




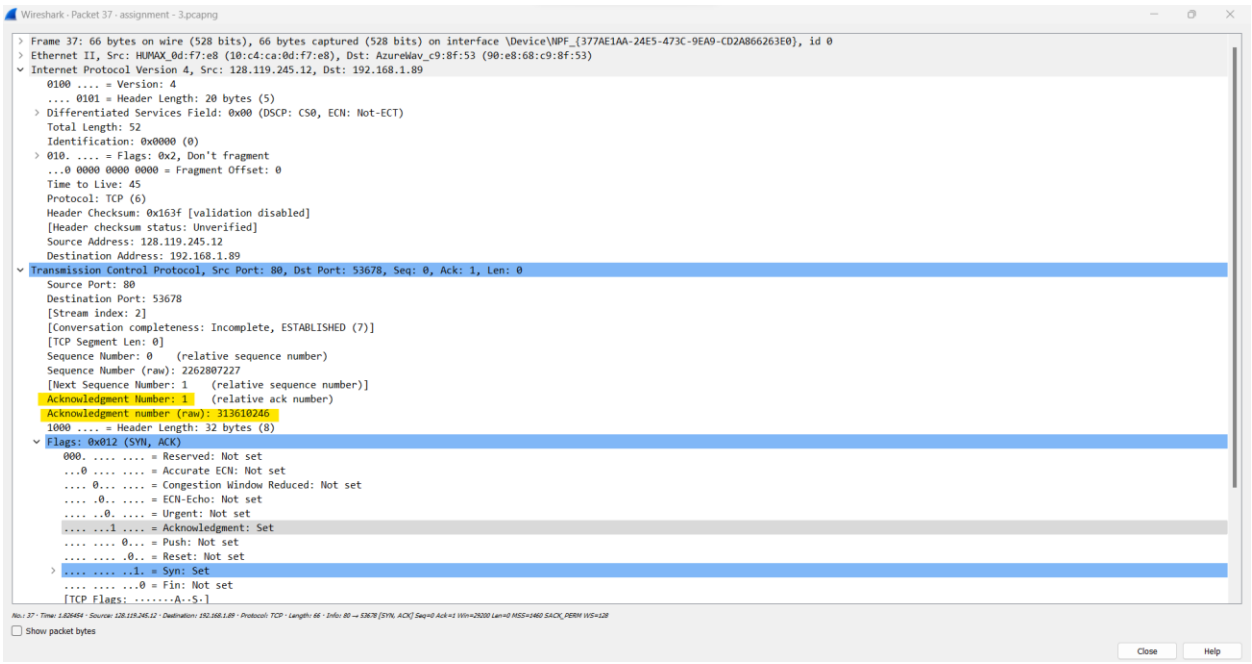
5. **What is the sequence number of the SYNACK segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN? What is the value of the Acknowledgement field in the SYNACK segment? How did gaia.cs.umass.edu determine that value? What is it in the segment that identifies the segment as a SYNACK segment?**



The Sequence number of the SYNACK segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN is 0. The raw value is 2262807227.

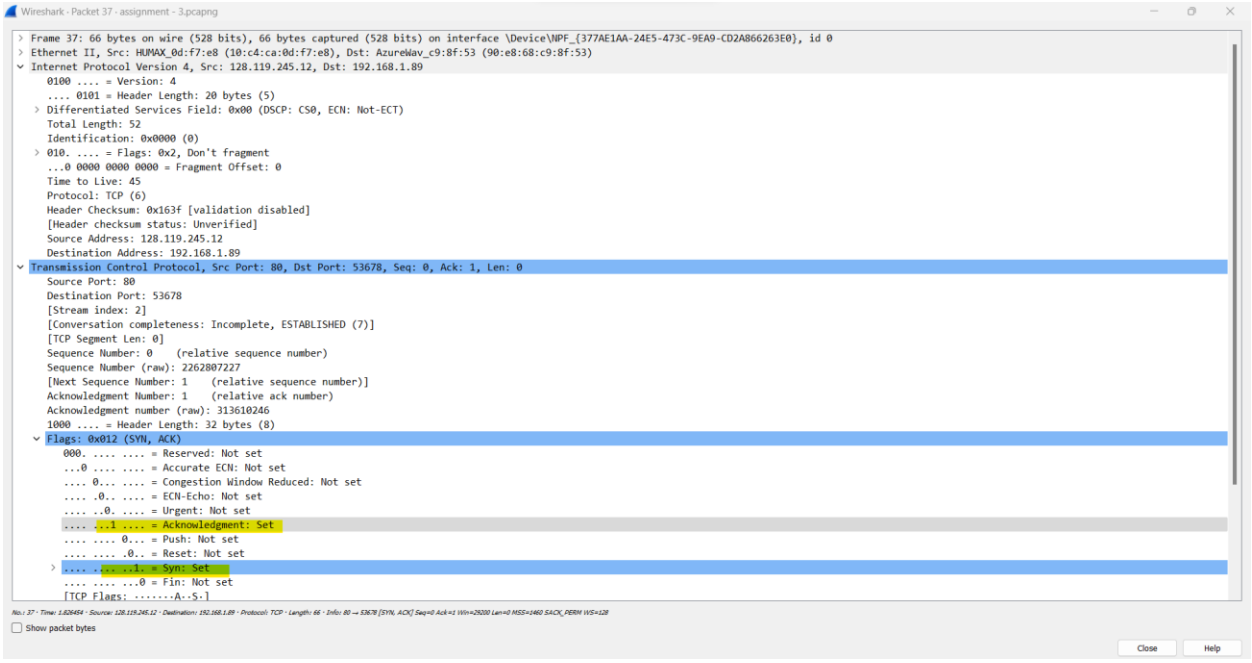


The value of the Acknowledgement field in the SYNACK segment is 1. The raw value is 313610246.



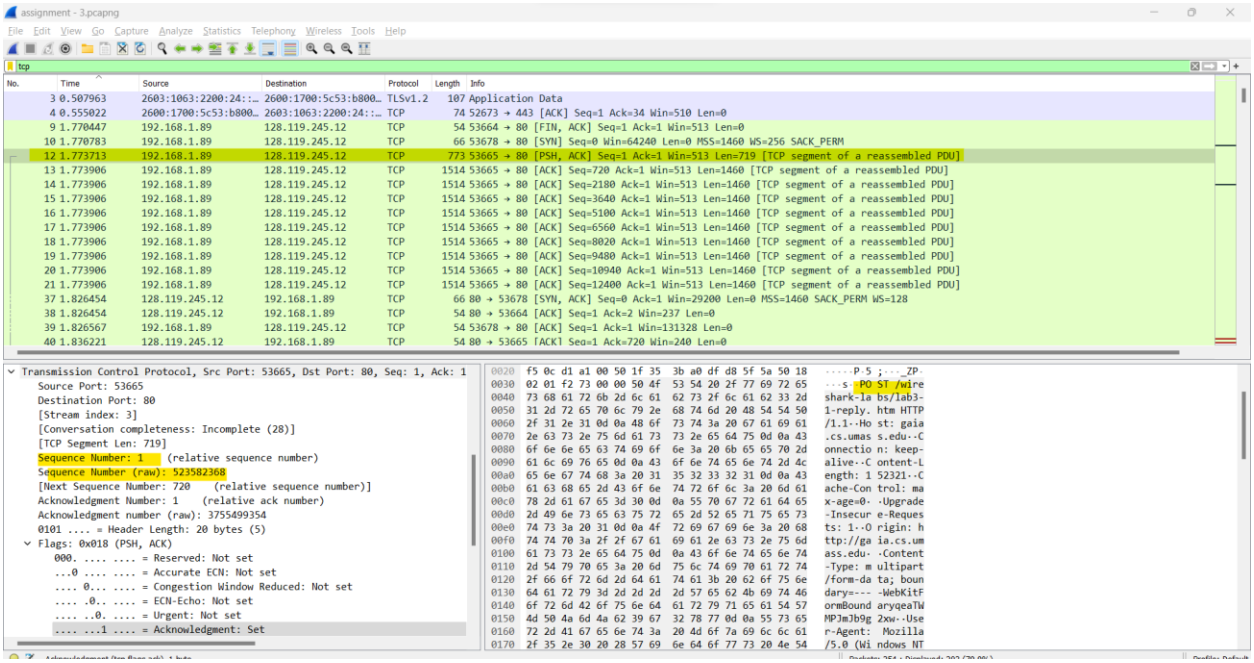
The value of the Acknowledgement field in the SYNACK segment is determined by gaia.cs.umass.edu by adding 1 to the initial sequence number of the SYN segment that was received from the client computer. Also, the acknowledgment number of SYN, ACK is equal to the sequence number of the next ACK segment.

The SYN flag and Acknowledgement flag in the segment are set to 1 and they indicate that this segment is a SYNACK segment.



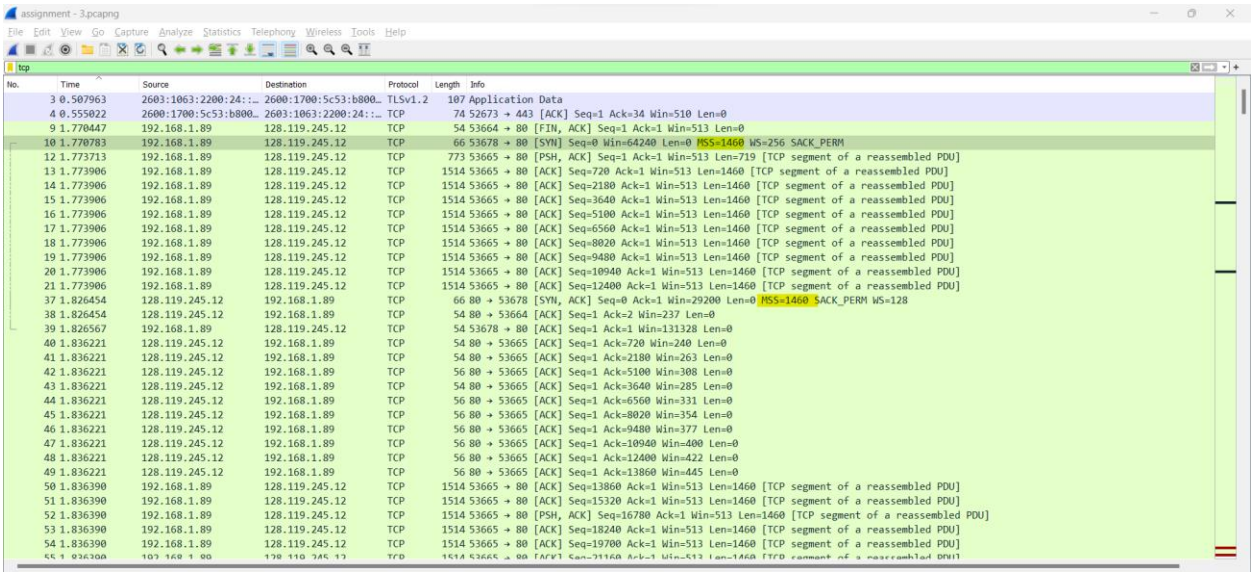
6. **What is the sequence number of the TCP segment containing the HTTP POST command? Note that to find the POST command, you'll need to dig into the packet content field at the bottom of the Wireshark window, looking for a segment with a "POST" within its DATA field.**

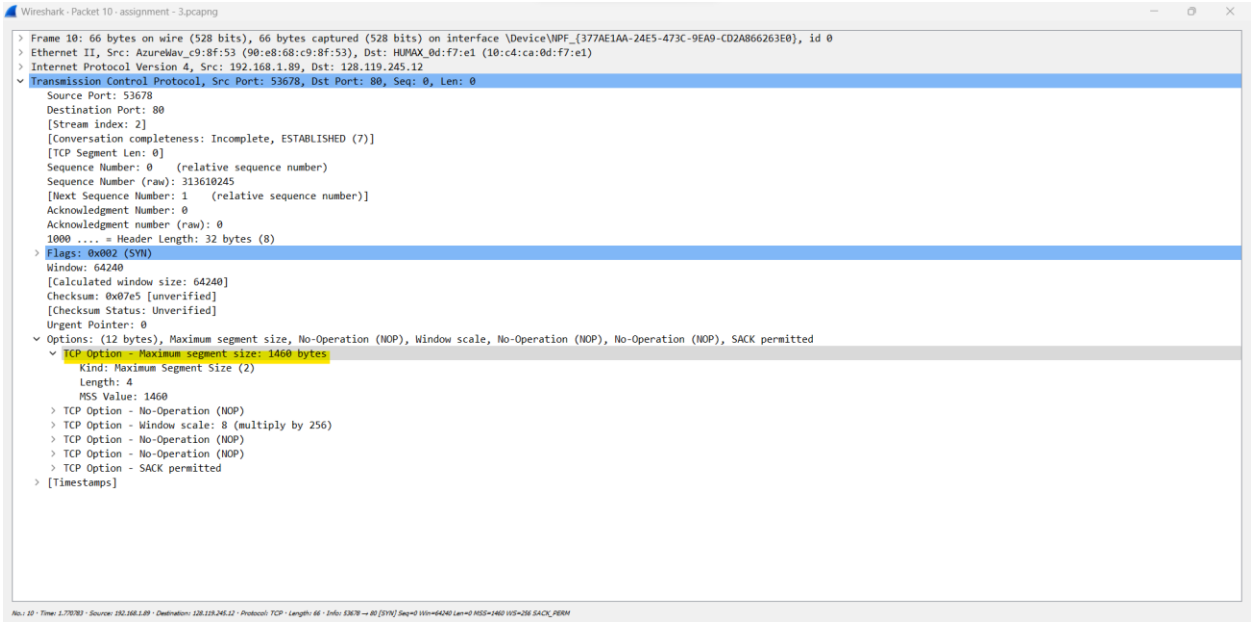
The sequence number of the TCP segment containing the HTTP POST the command is 1. The raw value is 523582368.



7. **What is the maximum segment size in the first ACK by the server?**

MSS = 1460 (It is found in SYN or SYN, ACK segments, in other segments its indicated using the length field)





8. What is the round-trip time(RRT) for the ACK received by the server?

Round trip time by taking the time difference between the SYN and the SYN/ACK packets.

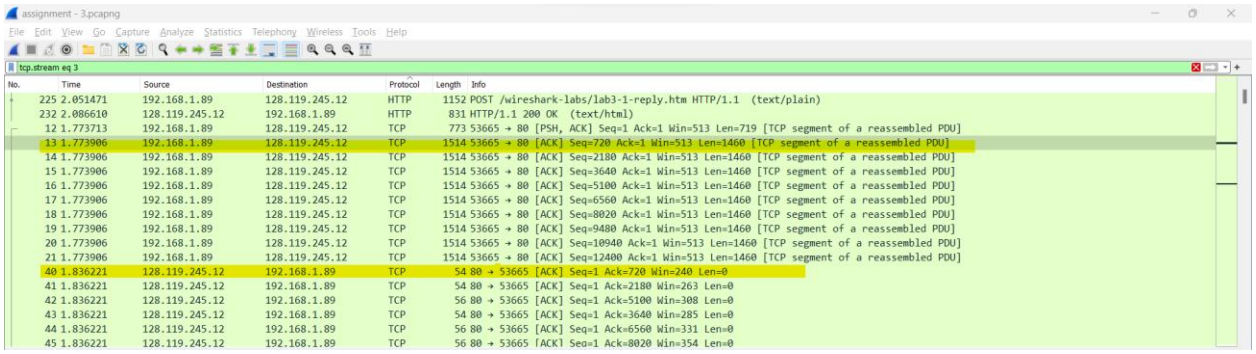
We can Calculate the RTT by subtracting the timestamp of the original data packet from the client and the timestamp of the ACK packet from the server.

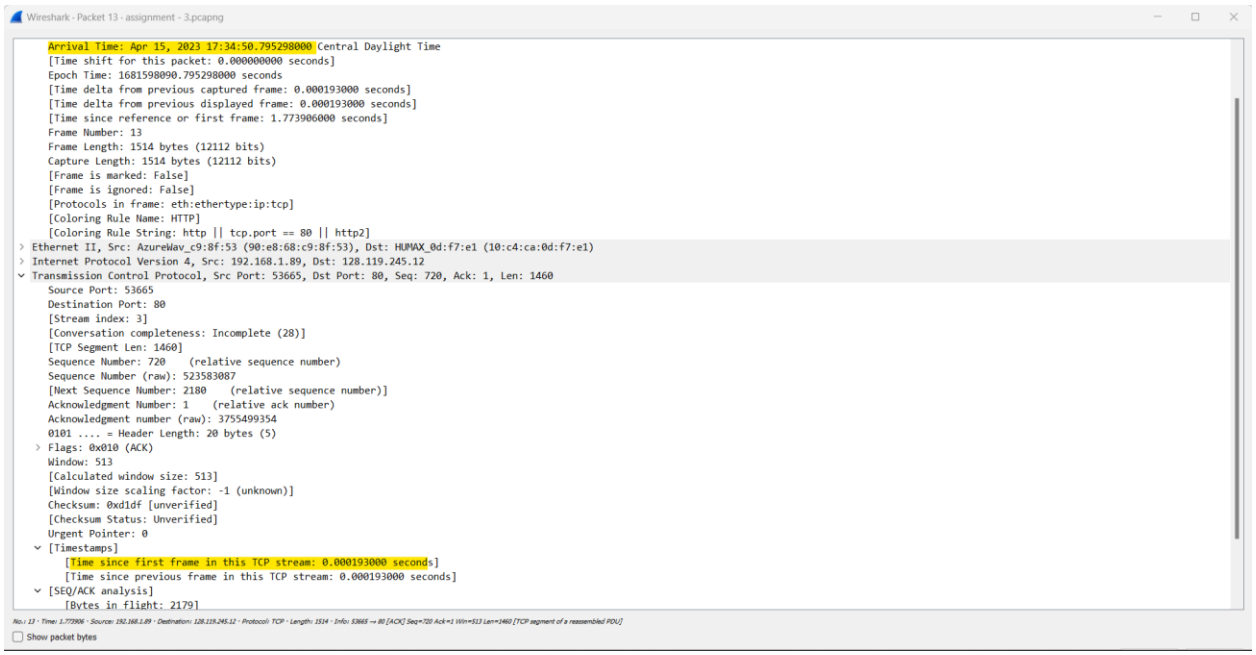
RTT = 17:34:50.795298000 – 17:34:50.85761300 = 0.062315

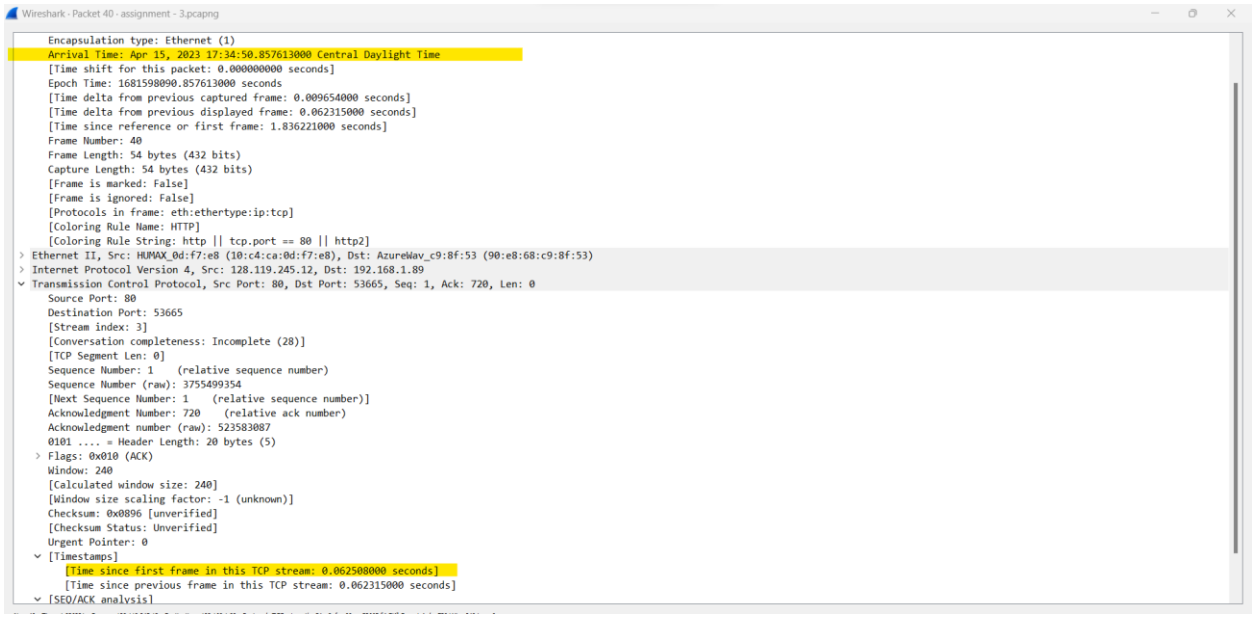
Or

RTT = 0.062508000-0.000193000 = 0.062315

The packets are highlighted below:



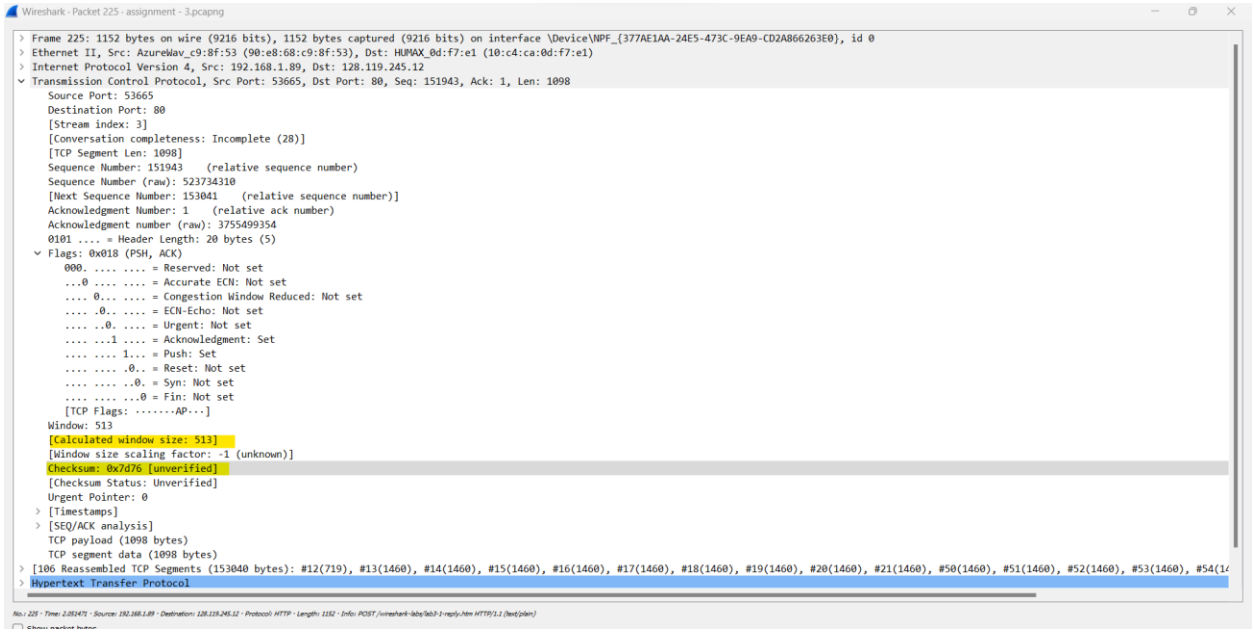




9. What is the calculated window size and checksum in the HTTP first packet?

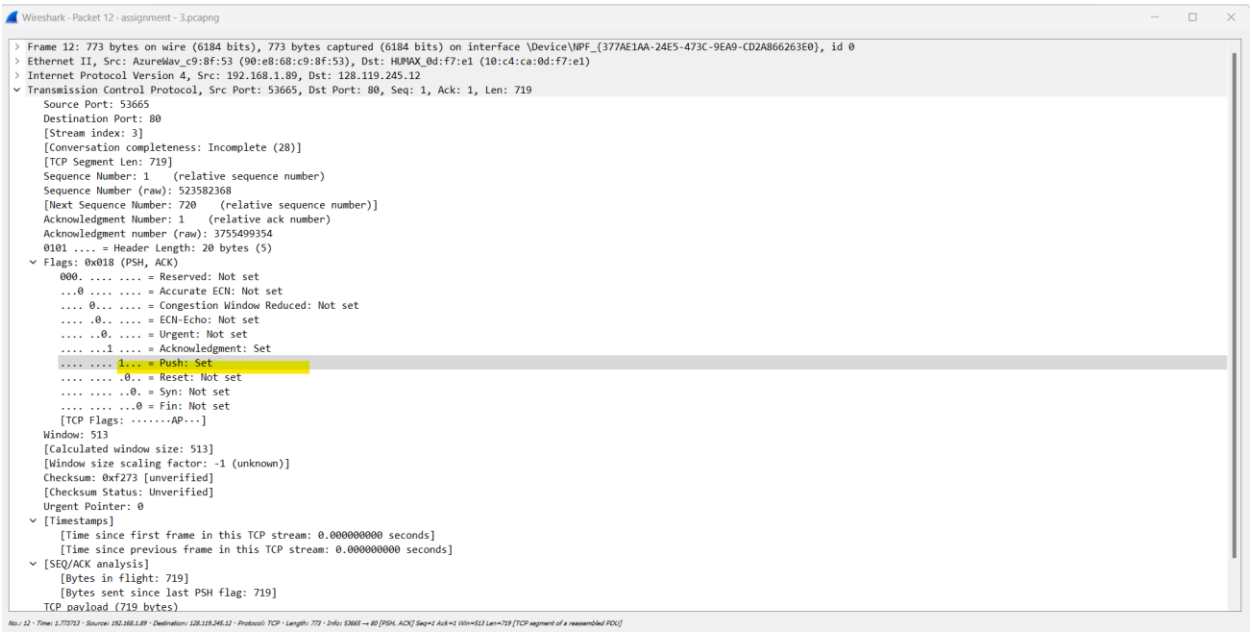
The window field controls the flow of data. It is a new window size that has been requested.

The checksum field is to ensure that the TCP segment reaches the destination without any errors. It is used for error detection.



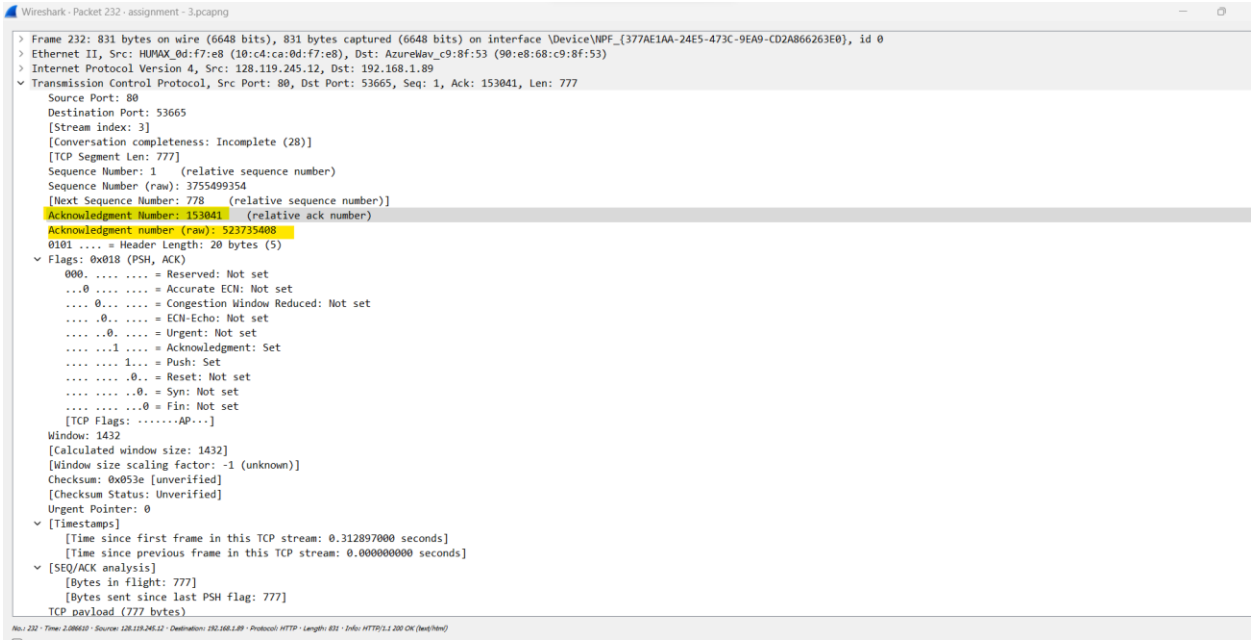
10. Is the PUSH flag set? If yes? What is its significance?

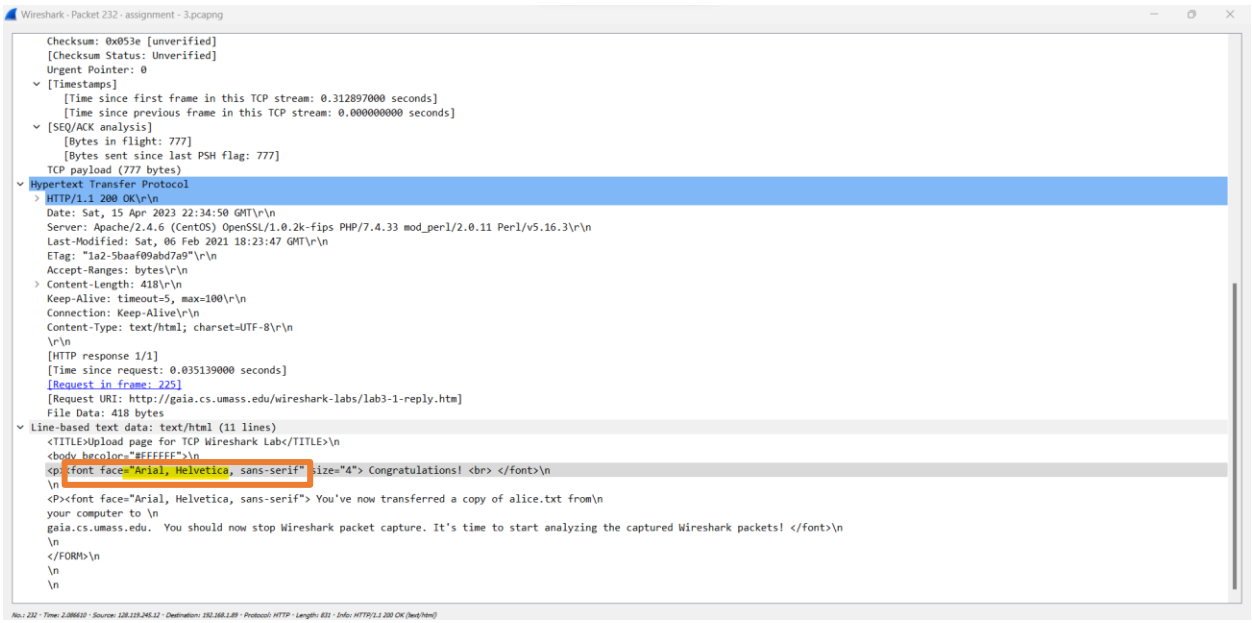
Yes, the PUSH flag is set. The PUSH flag instructs the receiving application to process the data as soon as possible rather than waiting for additional data to arrive, which can enhance the program's overall performance. Additionally, it can serve as a reminder for the sender to transmit data as soon as feasible.



11. What are the font face and acknowledgment number of the HTTP packet which has the status of 200 OK?

The font face is Arial, Helvetica, and Sans-serif. The acknowledgment number is 153041.





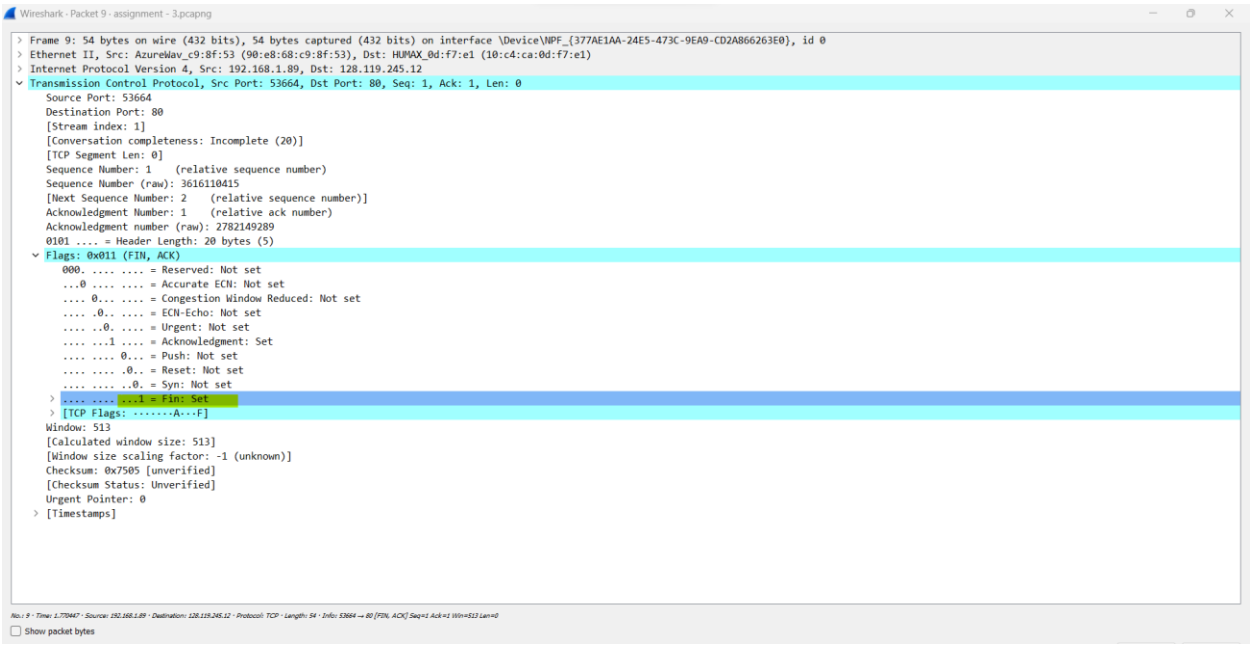
12. How is the connection closed? (Hint: FIN flag) Attach the screenshot of the flags which are set at the end?

The FIN flag is used to indicate that the sender has no more data to send.

There are three ways a TCP connection is closed:

- **By sending a FIN packet to the server, the client takes the initiative to cut off the connection.**
- **By sending a FIN packet to the client, the server starts the connection from being closed.**
- **The connection is cut off by both the client and the server.**

The ACK and FIN flags are set at the end.



13. Who has set the FIN bit? Is it server or client? Attach the screenshot.

The client set the FIN flag first.

The FIN flag is set in two packets:

1. **Frame 9: Source IP - 192.168.1.89 (Client), Destination IP - 128.119.245.12 (Server)**
2. **Frame 251: Source IP - 128.119.245.12 (Server), Destination IP - 192.168.1.89 (Client)**

14. What is the acknowledgment number of the last TCP ACK?

The acknowledgment number of the last TCP ACK is 779.