Assignment # 2

CTEC12052- Data Communication and Networking

Faculty of Computing & Technology

University of Kelaniya.

CT-2020-054

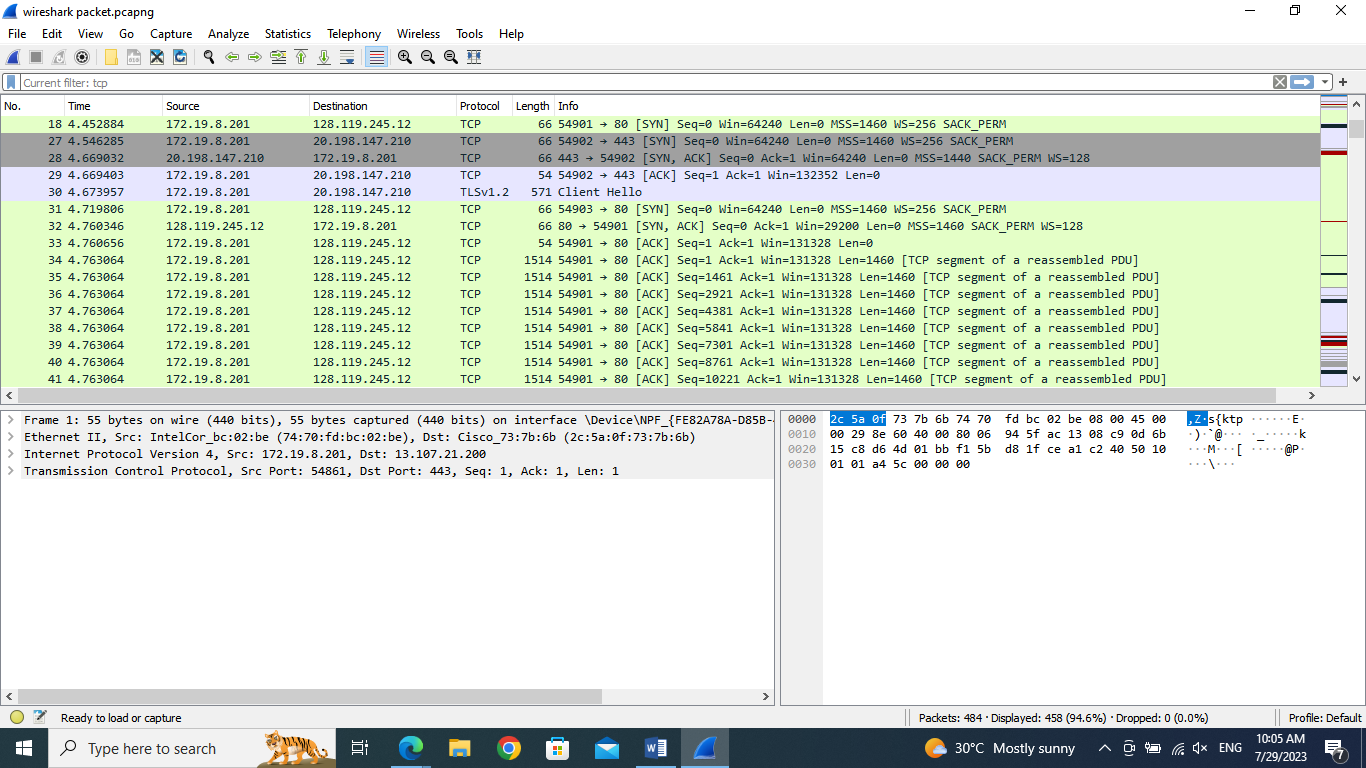
Handed to students: July 16, 2023 **Due Date:** August 04, 2023

**Question # 1: Capturing a bulk TCP transfer from your computer to a remote server**

**Answer the following questions, by opening the Wire shark captured packet file**.

(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.)

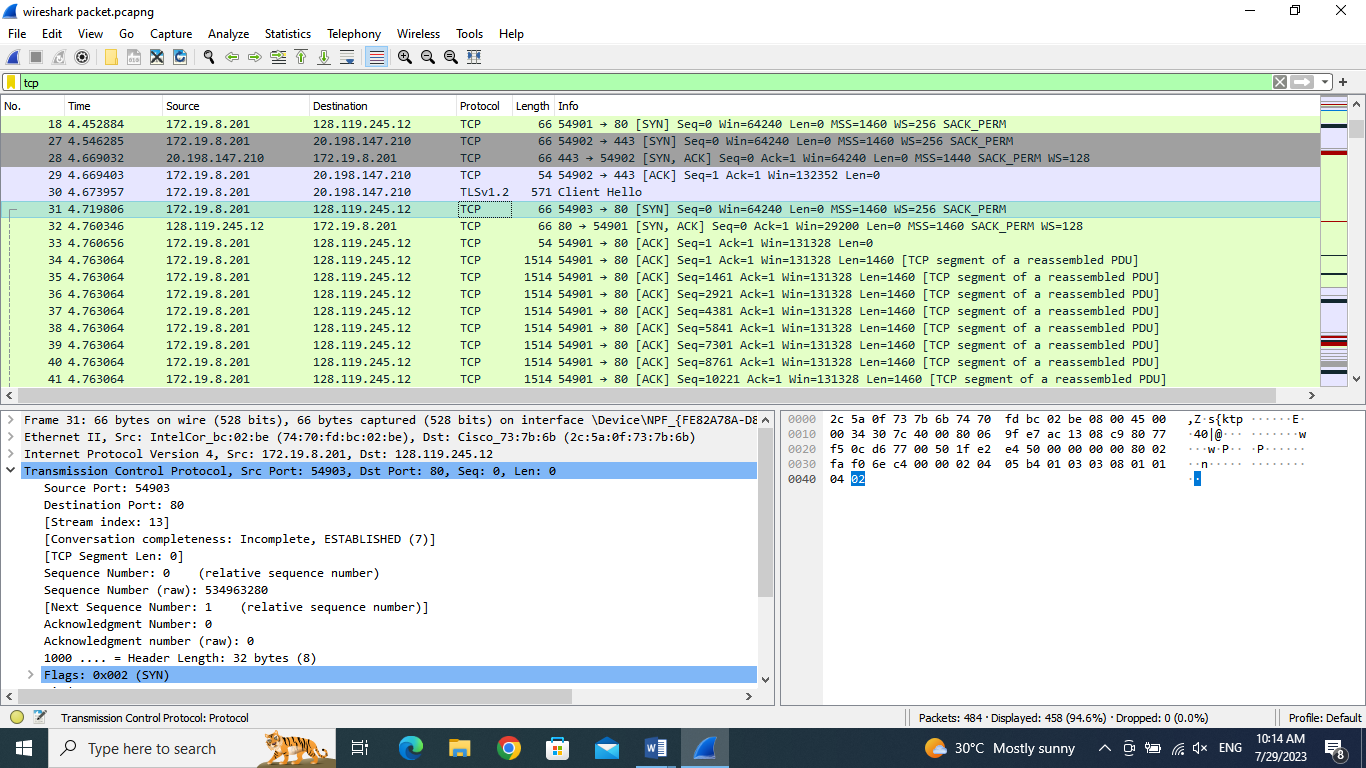
1. What is the IP address and TCP port number used by the client computer (source) that is transferring the file to gaia.cs.umass.edu?



**Solution: Client computer (source)**

**IP address: 172.19.8.201**

**TCP port number: 54903**

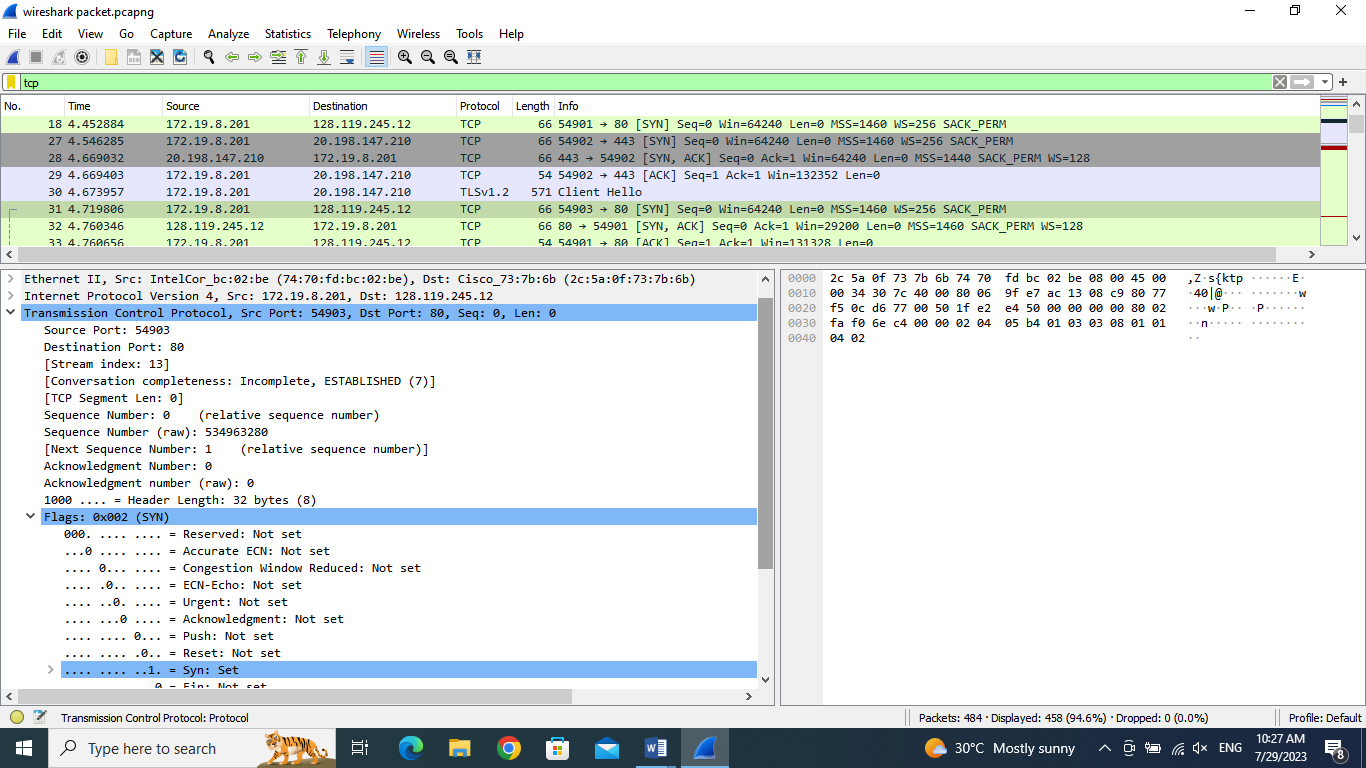
1. What is the IP address of gaia.cs.umass.edu? On what port number is it sending and receiving TCP segments for this connection?

**Destination computer: gaia.cs.umass.edu**

**IP address: 128.119.245.12**

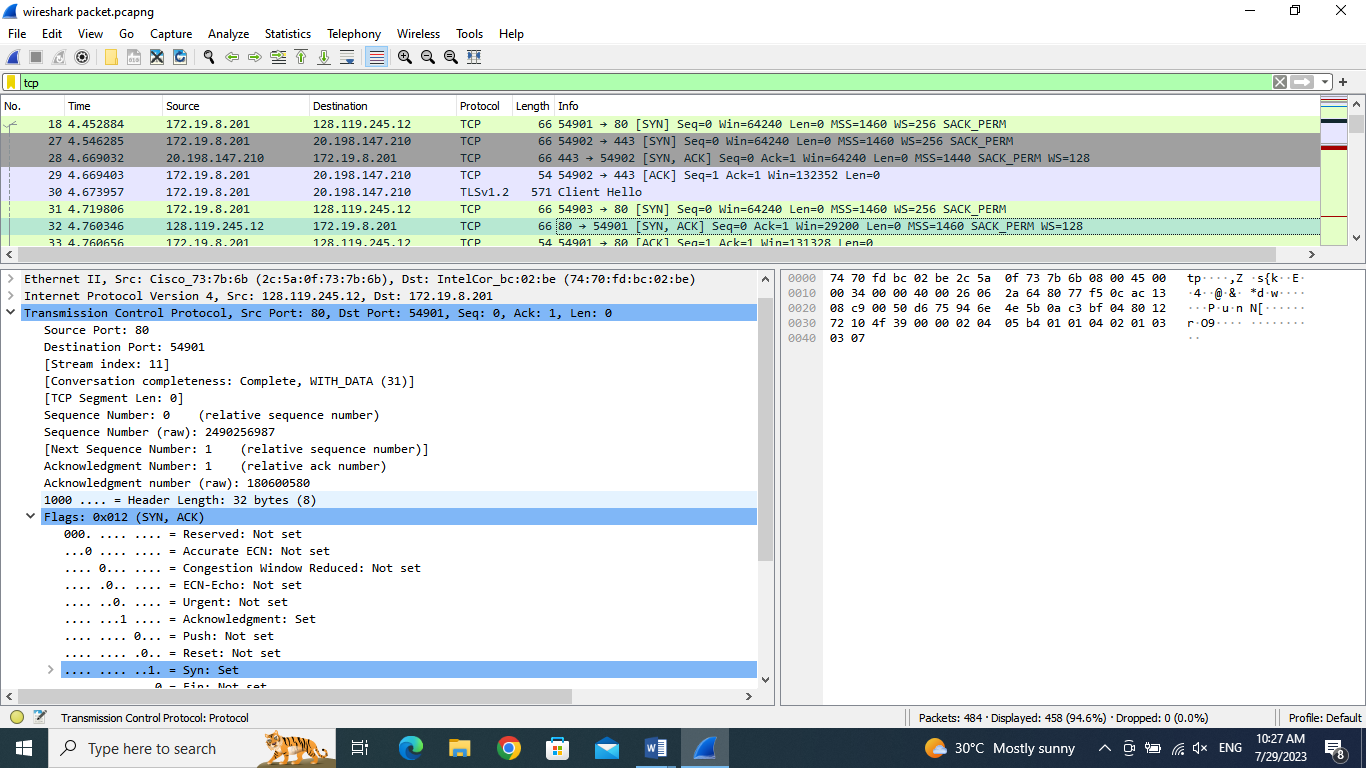
**TCP port number: 80**

Answer the following questions for the TCP segments:

1. 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 a SYN segment?

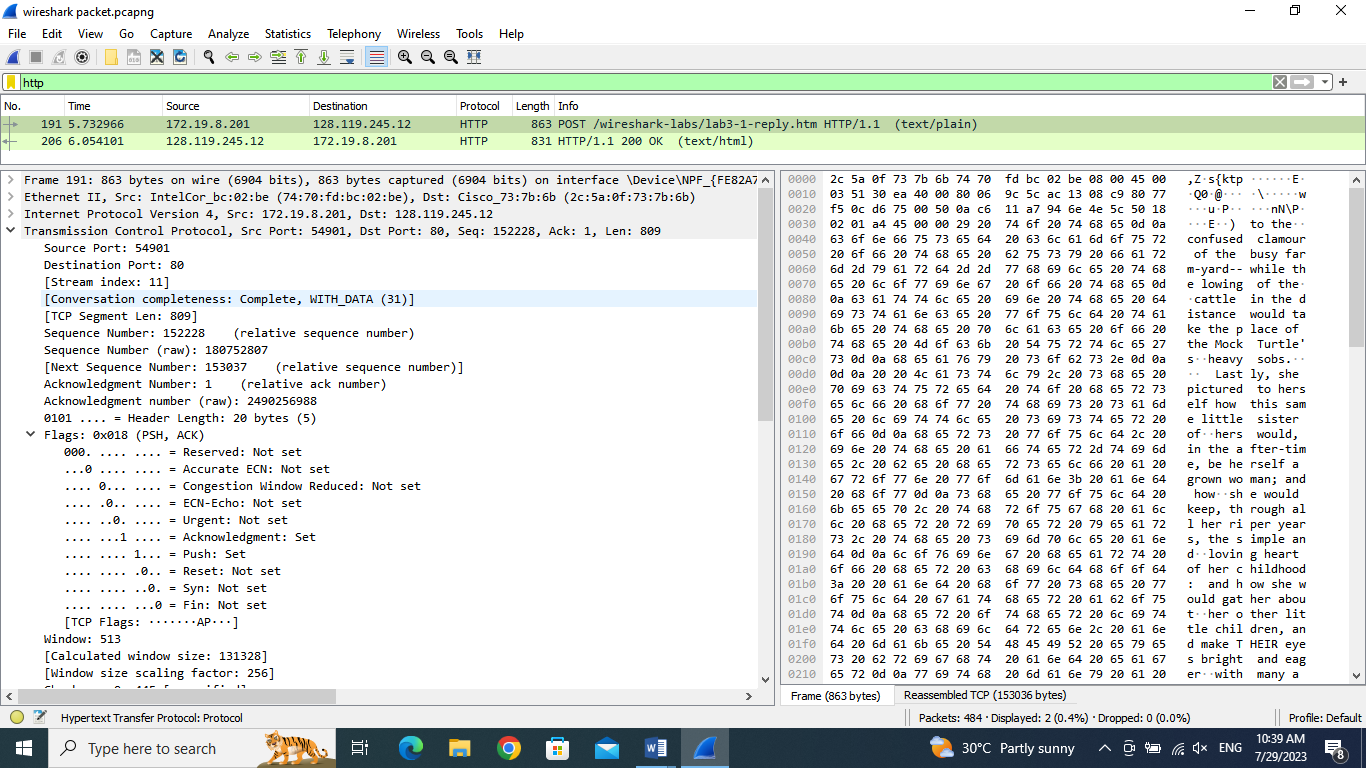
**The sequence number of the TCP SYN segment that is used to initiate the TCP connection between client computer and gaia.cs.umass.edu is 0 in the trace. The segment is identified as a SYN segment since it set up to 1.**

1. 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?

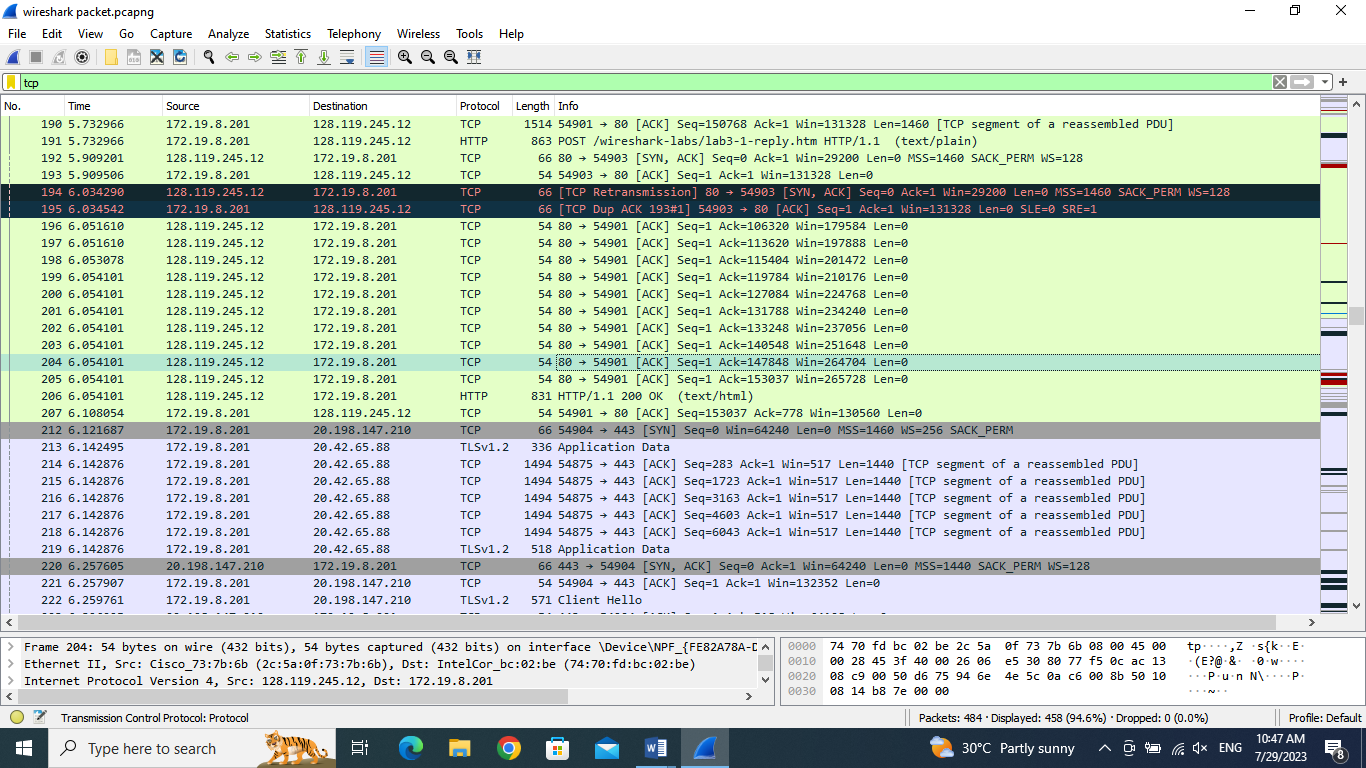
**Solution: Sequence number of the TCP SYN segment is used to initiate the TCP connection between the client computer and gaia.cs.umass.edu. The value is 0 in this trace. The SYN flag is set to 1 and it indicates that this segment is a SYN segment.**

1. What is the sequence number of the TCP segment containing the HTTP POST command? Note that in order 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 this segment has the value of 152228.**

1. Consider the TCP segment containing the HTTP POST as the first segment in the TCP connection. What are the sequence numbers of the first six segments in the TCP connection (including the segment containing the HTTP POST)? At what time was each segment sent? When was the ACK for each segment received? Given the difference between when each TCP segment was sent, and when its acknowledgement was received, what is the RTT value for each of the six segments?



**The HTTP POST segment is considered as the first segment**

**Segment 1 sequence number: 0**

**Segment 2 sequence number: 1**

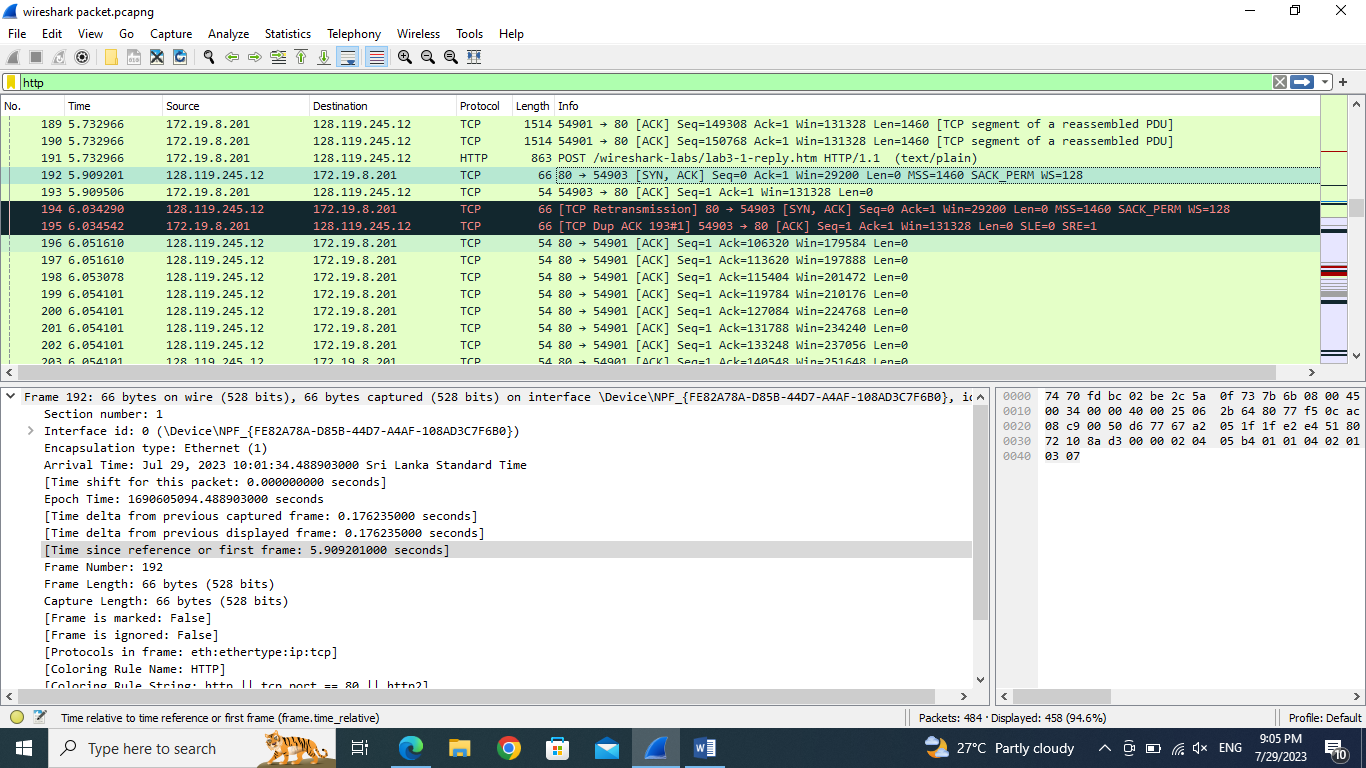
**Segment 3 sequence number: 0**

**Segment 4 sequence number: 1**

**Segment 5 sequence number: 1**

**Segment 6 sequence number: 1**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Sent time** | **ACK received time** | **RTT (seconds)** |
| **Segment 1** | **4.669032** | **5.909201** | **1.240169** |
| **Segment 2** | **4.669403** | **5.909506** | **1.240103** |
| **Segment 3** | **4.760346** | **6.034290** | **1.273944** |
| **Segment 4** | **5.730308** | **6.034542** | **0.304234** |
| **Segment 5** | **5.730433** | **6.051610** | **0.321177** |
| **Segment 6** | **5.730573** | **6.051610** | **0.321037** |

1. What is the length of each of the first six TCP segments?

**Not containing http post**

**Segment 1 length = 66 bytes**

**Segment 2 length = 54 bytes**

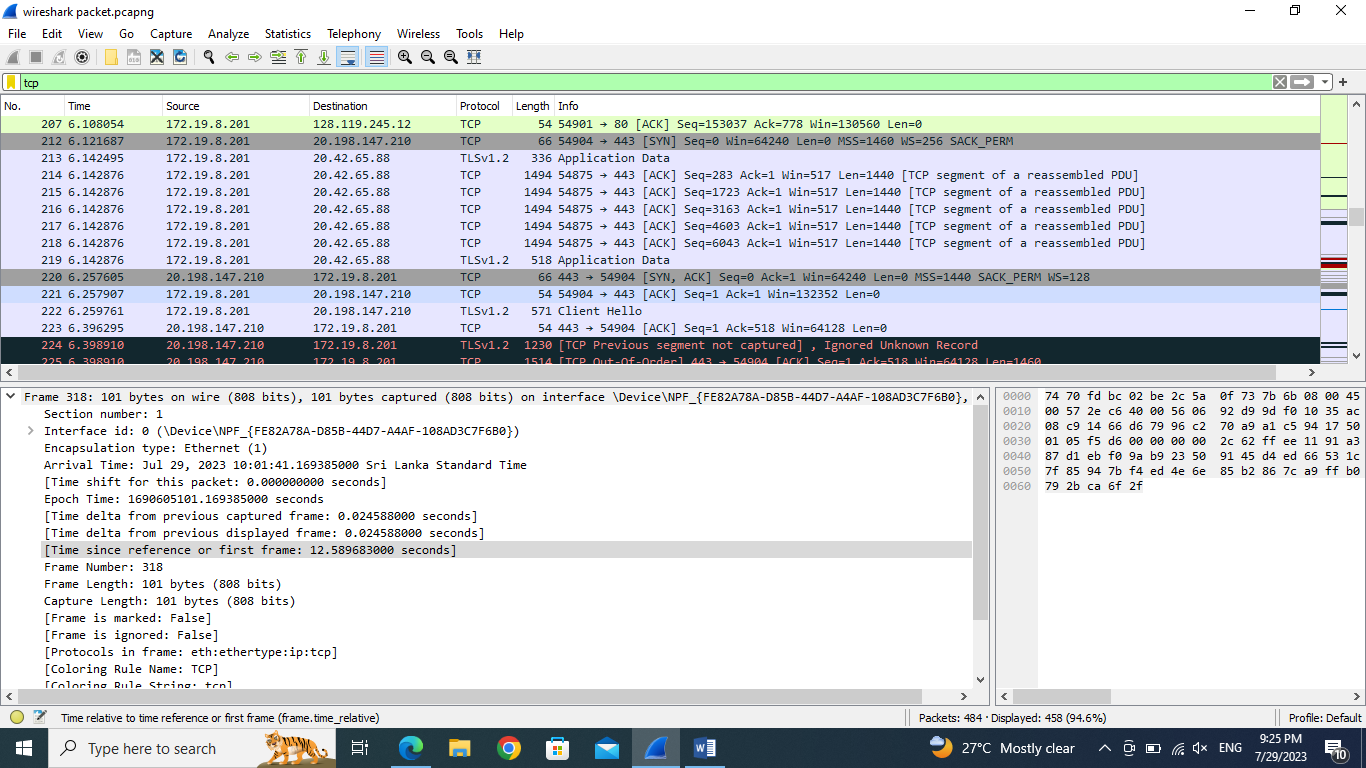
**Segment 3 length = 66 bytes**

**Segment 4 length= 66 bytes**

**Segment 5 length =54bytes**

**Segment 6 length = 54bytes**

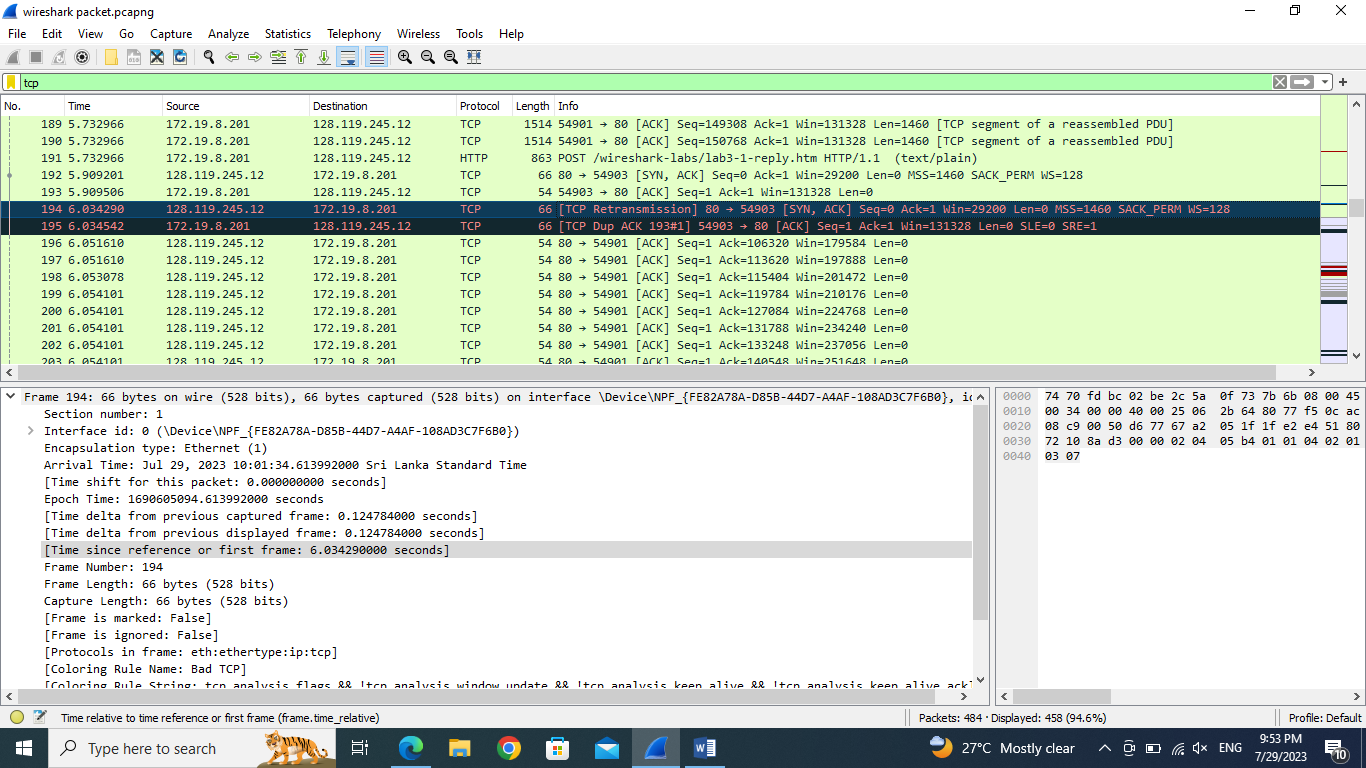
1. What is the minimum amount of available buffer space advertised at the received for the entire trace? Does the lack of receiver buffer space ever throttle the sender?



**Window: 517 bytes**

**Yes , I did.**

1. Are there any retransmitted segments in the trace file? What did you check for (in the trace) in order to answer this question?



**Yes. There are retransmitted segment in the trace file.**

**By paying attention to duplicate sequence numbers, retransmission flags, and using the TCP stream reassembly feature in wire shark.**

1. How much data does the receiver typically acknowledge in an ACK? Can you identify cases where the receiver is ACKing every other received segment?

**Question # 2: Capturing packets from an execution of traceroute**

In order to generate a trace of IP datagrams for this lab, we’ll use the traceroute program to send datagrams of different sizes towards some destination, X. Recall that traceroute operates by first sending one or more datagrams with the time-to-live (TTL) field in the IP header set to 1; it then sends a series of one or more datagrams towards the same destination with a TTL value of 2; it then sends a series of datagrams towards the same destination with a TTL value of 3; and so on. Recall that a router must decrement the TTL in each received datagram by 1. If the TTL reaches 0, the router returns an ICMP message to the sending host. As a result of this behavior, a datagram with a TTL of 1 will cause the router one hop away from the sender to send an ICMP TTL-exceeded message back to the sender; the datagram sent with a TTL of 2 will cause the router two hops away to send an ICMP message back to the sender; the datagram sent with a TTL of 3 will cause the router three hops away to send an ICMP message back to the sender; and so on. In this manner, the host executing traceroute can learn the identities of the routers between itself and destination X by looking at the source IP addresses in the datagrams containing the ICMP TTL-exceeded messages.

We’ll want to run traceroute and have it send datagrams of various lengths.

* **Windows**. The tracert program (used for our ICMP Wireshark lab) provided with Windows does not allow one to change the size of the ICMP echo request (ping) message sent by the tracert program. A nicer Windows traceroute program is pingplotter, available both in free version and shareware versions at http://www.pingplotter.com. Download and install pingplotter, and test it out by performing a few traceroutes to your favorite sites. The size of the ICMP echo request message can be explicitly set in pingplotter by selecting the menu item Edit-> Options->Default Settings->Engine then filling in the Packet Size field. The default packet size is 56 bytes. Once pingplotter has sent a series of packets with the increasing TTL values, it restarts the sending process again with a TTL of 1, after waiting Trace Interval amount of time. The value of Trace Interval and the number of intervals can be explicitly set in pingplotter.
* **Linux/Unix/MacOS**. With the Unix/MacOS traceroute command, the size of the UDP datagram sent towards the destination can be explicitly set by indicating the number of bytes in the datagram; this value is entered in the traceroute command line immediately after the name or address of the destination. For example, to send traceroute datagrams of 2000 bytes towards gaia.cs.umass.edu, the command would be:

%traceroute gaia.cs.umass.edu 2000

**Do the following:**

* 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).
* If you are using a Windows platform, start up pingplotter and enter the name of a target destination in the “Address Window.” Select the menu item Edit>Options-> Default Settings ->Engine and enter a value of 56 in the Packet Size field and then press OK. Then press the Trace button.

Next, send a set of datagrams with a longer length, by selecting Edit->Options->Default Settings-> Engine and enter a value of 2000 in the Packet Size field and then press OK. Then press the Resume button.

Finally, send a set of datagrams with a longer length, by selecting Edit->Options->Default Settings-> Engine and enter a value of 3500 in the Packet Size field and then press OK. Then press the Resume button.

Stop Wireshark tracing.

* If you are using a Unix or Mac platform, enter three traceroute commands, one with a length of 56 bytes, one with a length of 2000 bytes, and one with a length of 3500 bytes.

Stop Wireshark tracing.

**A look at the captured trace**

In your trace, you should be able to see the series of ICMP Echo Request (in the case of Windows machine) or the UDP segment (in the case of Unix) sent by your computer and the ICMP TTL-exceeded messages returned to your computer by the intermediate routers. Whenever possible, when answering a question below you should hand in a printout of the packet(s) within the trace that you used to answer the question asked.

1. Select the first ICMP Echo Request message sent by your computer, and expand the Internet Protocol part of the packet in the packet details window. What is the IP address of your computer?
2. Within the IP packet header, what is the value in the upper layer protocol field?
3. How many bytes are in the IP header? How many bytes are in the payload of the IP datagram? Explain how you determined the number of payload bytes.
4. Has this IP datagram been fragmented? Explain how you determined whether or not the datagram has been fragmented.

Next, sort the traced packets according to IP source address by clicking on the Source column header; a small downward pointing arrow should appear next to the word Source. If the arrow points up, click on the Source column header again. Select the first ICMP Echo Request message sent by your computer, and expand the Internet Protocol portion in the “details of selected packet header” window. In the “listing of captured packets” window, you should see all of the subsequent ICMP messages (perhaps with additional interspersed packets sent by other protocols running on your computer) below this first ICMP. Use the down arrow to move through the ICMP messages sent by your computer.

1. Which fields in the IP datagram always change from one datagram to the next within this series of ICMP messages sent by your computer?
2. Which fields stay constant? Which of the fields must stay constant? Which fields must change? Why?
3. Describe the pattern you see in the values in the Identification field of the IP datagram

Next (with the packets still sorted by source address) find the series of ICMP TTLexceeded replies sent to your computer by the nearest (first hop) router.

1. What is the value in the Identification field and the TTL field?
2. Do these values remain unchanged for all of the ICMP TTL-exceeded replies sent to your computer by the nearest (first hop) router? Why?

**Fragmentation**

Sort the packet listing according to time again by clicking on the Time column.

1. Find the first ICMP Echo Request message that was sent by your computer after you changed the Packet Size in pingplotter to be 2000. Has that message been fragmented across more than one IP datagram?
2. Print out the first fragment of the fragmented IP datagram. What information in the IP header indicates that the datagram been fragmented? What information in the IP header indicates whether this is the first fragment versus a latter fragment? How long is this IP datagram?
3. Print out the second fragment of the fragmented IP datagram. What information in the IP header indicates that this is not the first datagram fragment? Are the more fragments? How can you tell?
4. What fields change in the IP header between the first and second fragment?

Now find the first ICMP Echo Request message that was sent by your computer after you changed the Packet Size in pingplotter to be 3500.

1. How many fragments were created from the original datagram? 15. What fields change in the IP header among the fragments?

---------------------------------END-------------------------------------