**Lab - 08**

**Analyzing Congestion Policy, RTT Of TCP And Working Of UDP Using NetSim And Wireshark.**

**Program: MScIT**

**Sem-2**

**Group ID : 28**

**Student Name Student ID**

Dev Adnani 202212012

Saif Saiyed 202212083

**(1) Introduction Of Congestion Policy Of TCP**

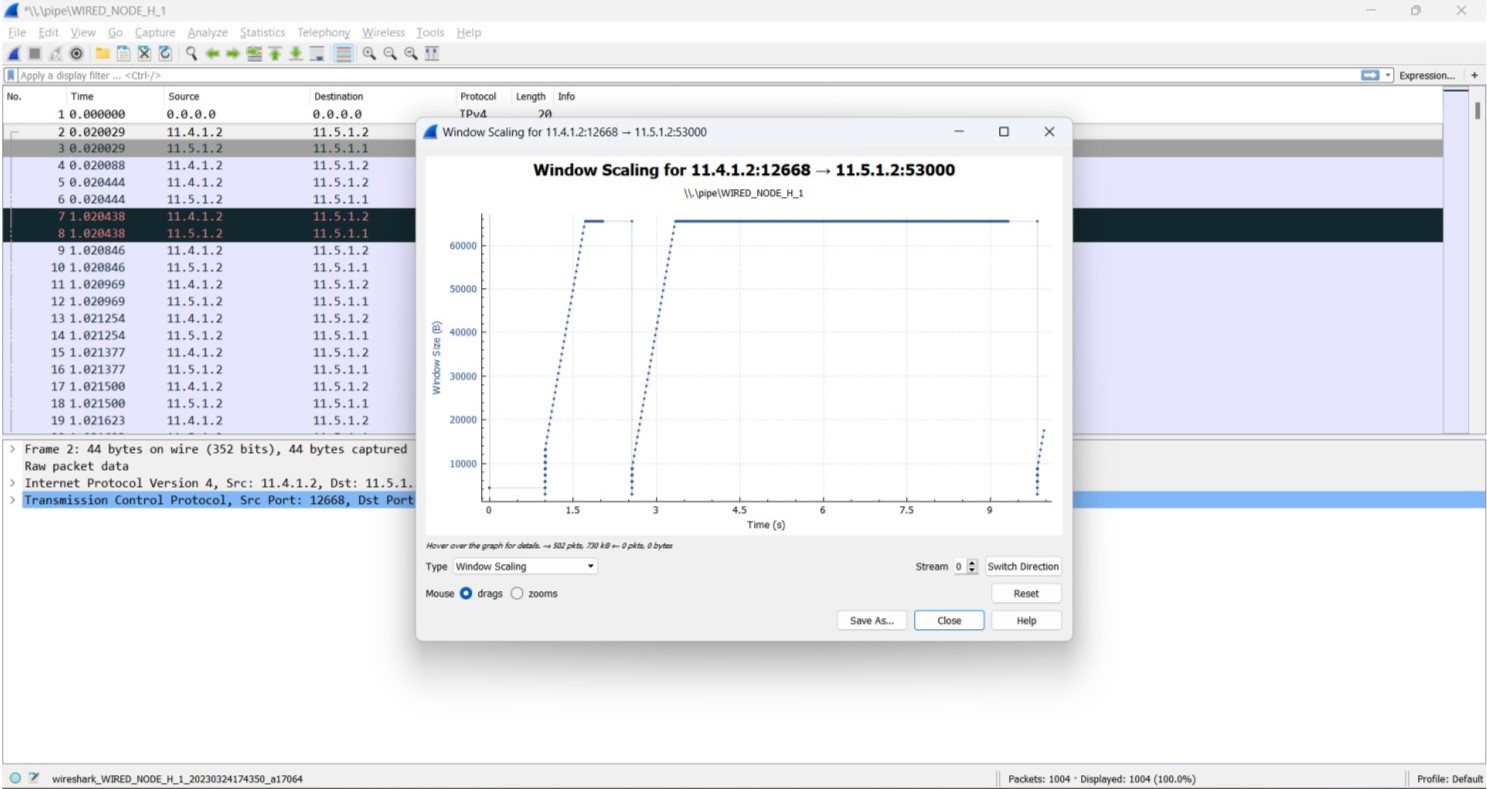
**1.2 Exercise**

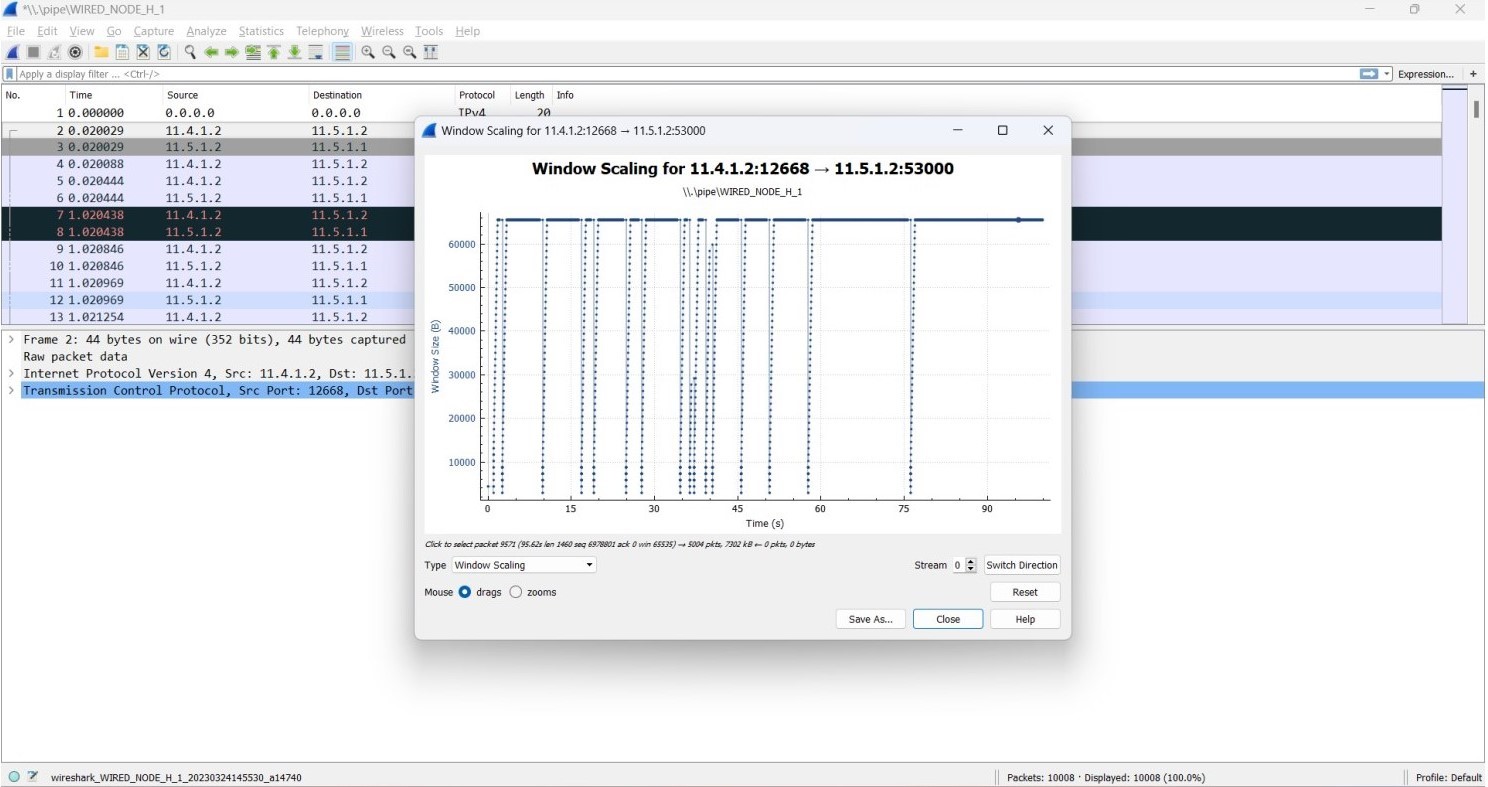
**1. For both the variant, analyze graph of congestion window, answer the following by marking in the graph.**

**(a) Identify the event of TCP slow start.**

**(b) Identify the event of packet loss and time out.**

**(c) Identify the intervals of time when TCP congestion avoidance is operating.**

****

****

**2. What is the difference in congestion control policy of Tahoe and Reno, with respect to congestion avoidance and two events of congestion avoidance phase. Explain briefly in your log book.**

Ans:

Reno is more aggressive and adds a fast-recovery phase to avoid resetting the congestion window size to initial size in case of packet loss. Tahoe takes a more conservative strategy to congestion avoidance by resetting to slow-start phase after detecting congestion.

**(2) Analyzing Fairness Of TCP**

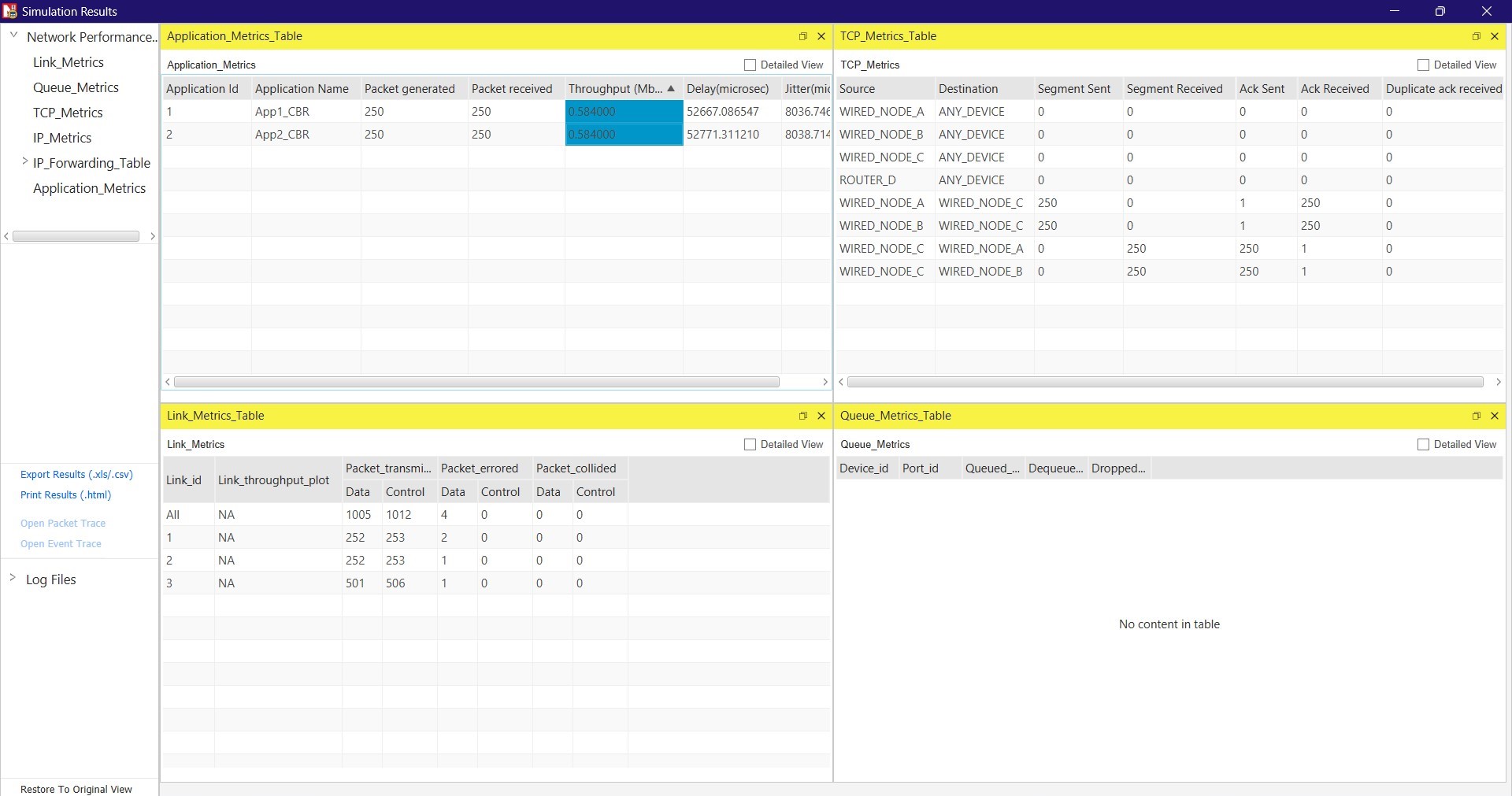
**2.1 Experiment**

1. Take 3 wired nodes and one router, configure 2 identical CBR applications with default app specification between them as shown in the figure4.

2. Keep link properties as default.

3. Run simulation for 5 seconds.

4. Check throughput for both the applications and write down your observation.



**(3) Analyzing Throughput**

**3.1 Experiment**

1. Configure a new network as shown in the figure 5 with 4 wired node, 1 router.

2. Configure two CBR application between node B and F and between C and F with packet size of 500 bytes.

3. Configure two video application between node D and node F and between node E and F with Frame per second 50.

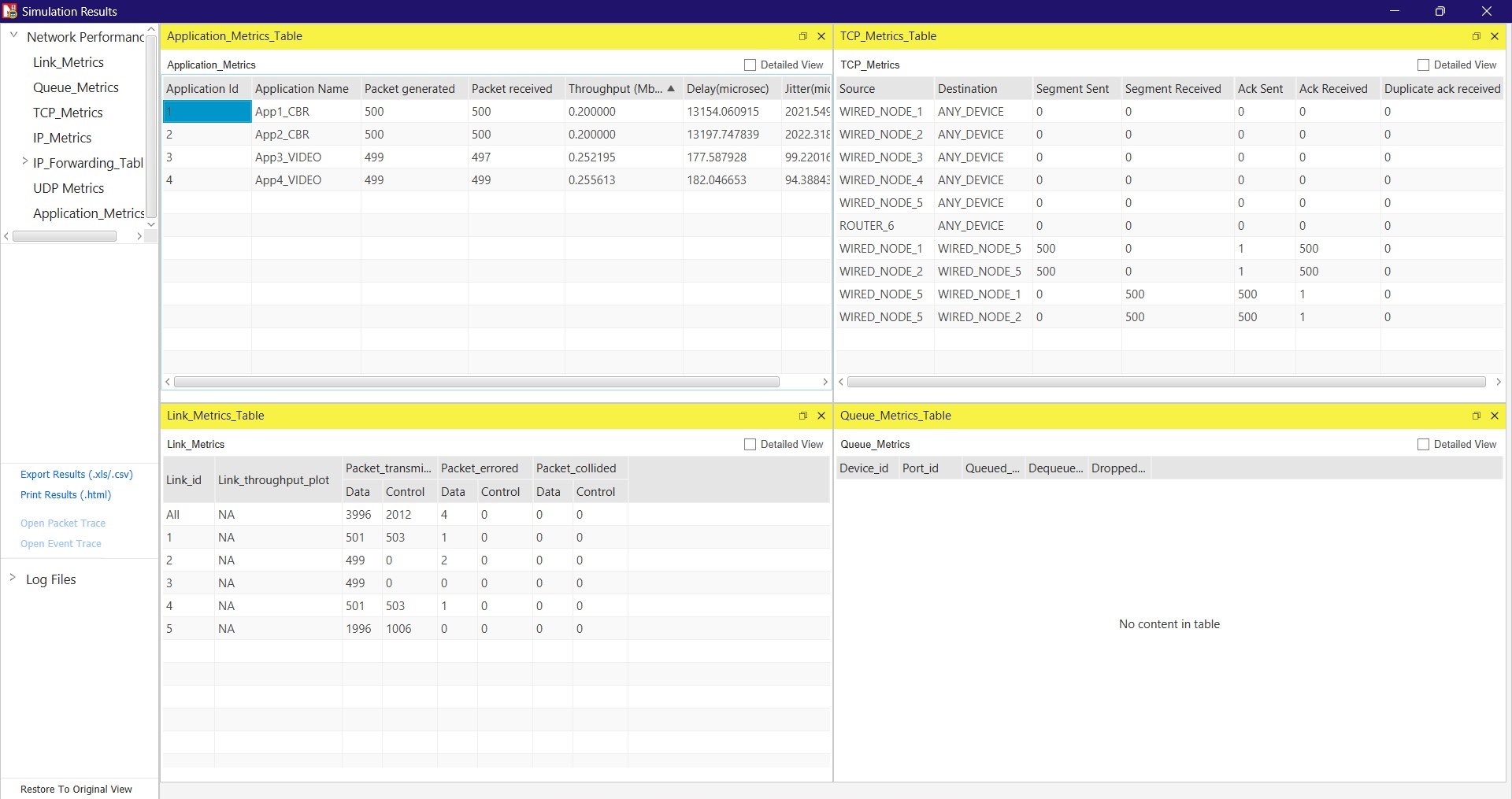
4. Keep all properties of all nodes, router and links as default values.

5. Run the simulation for 10 second.

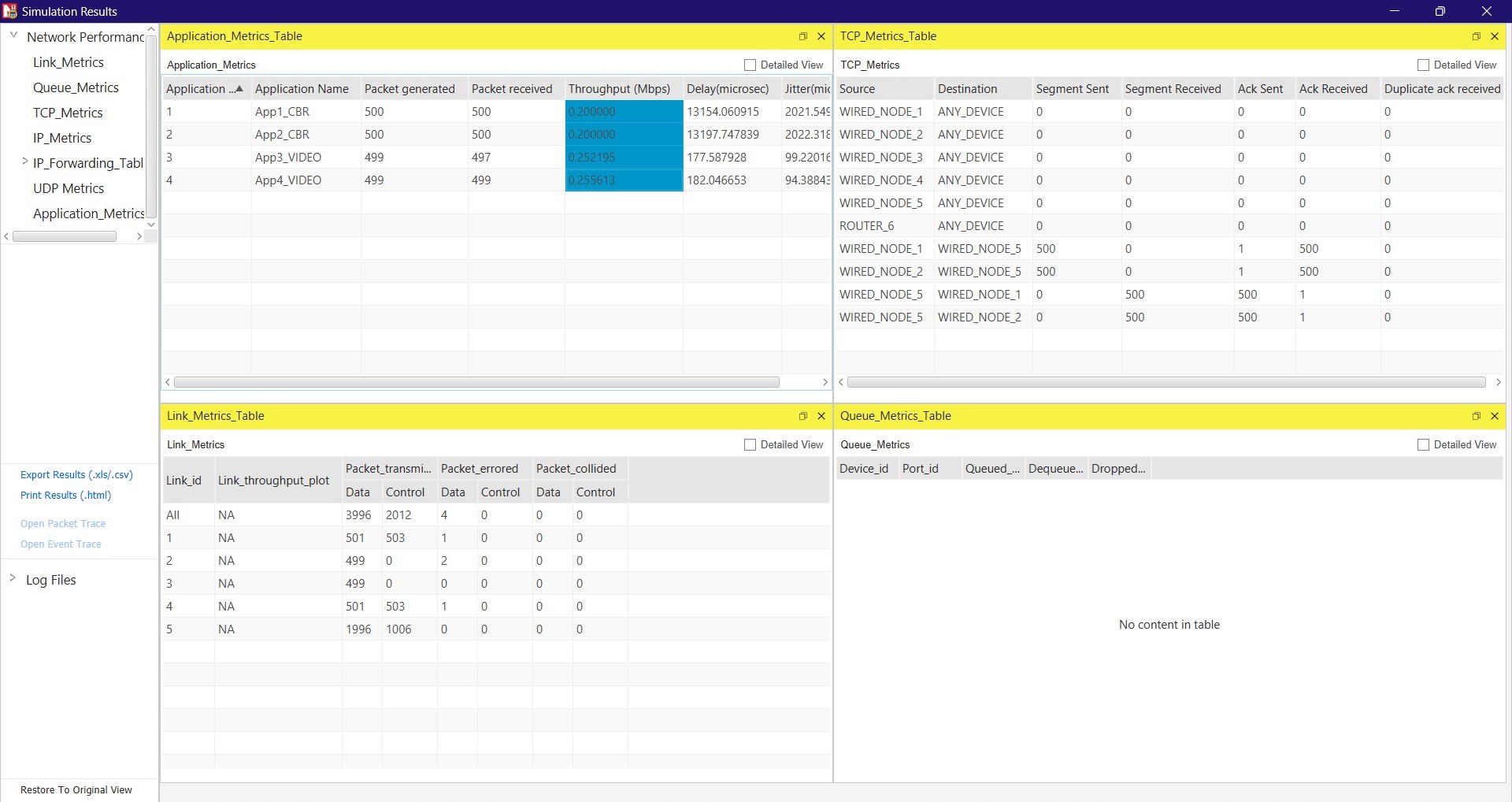
**3.2 Exercise**

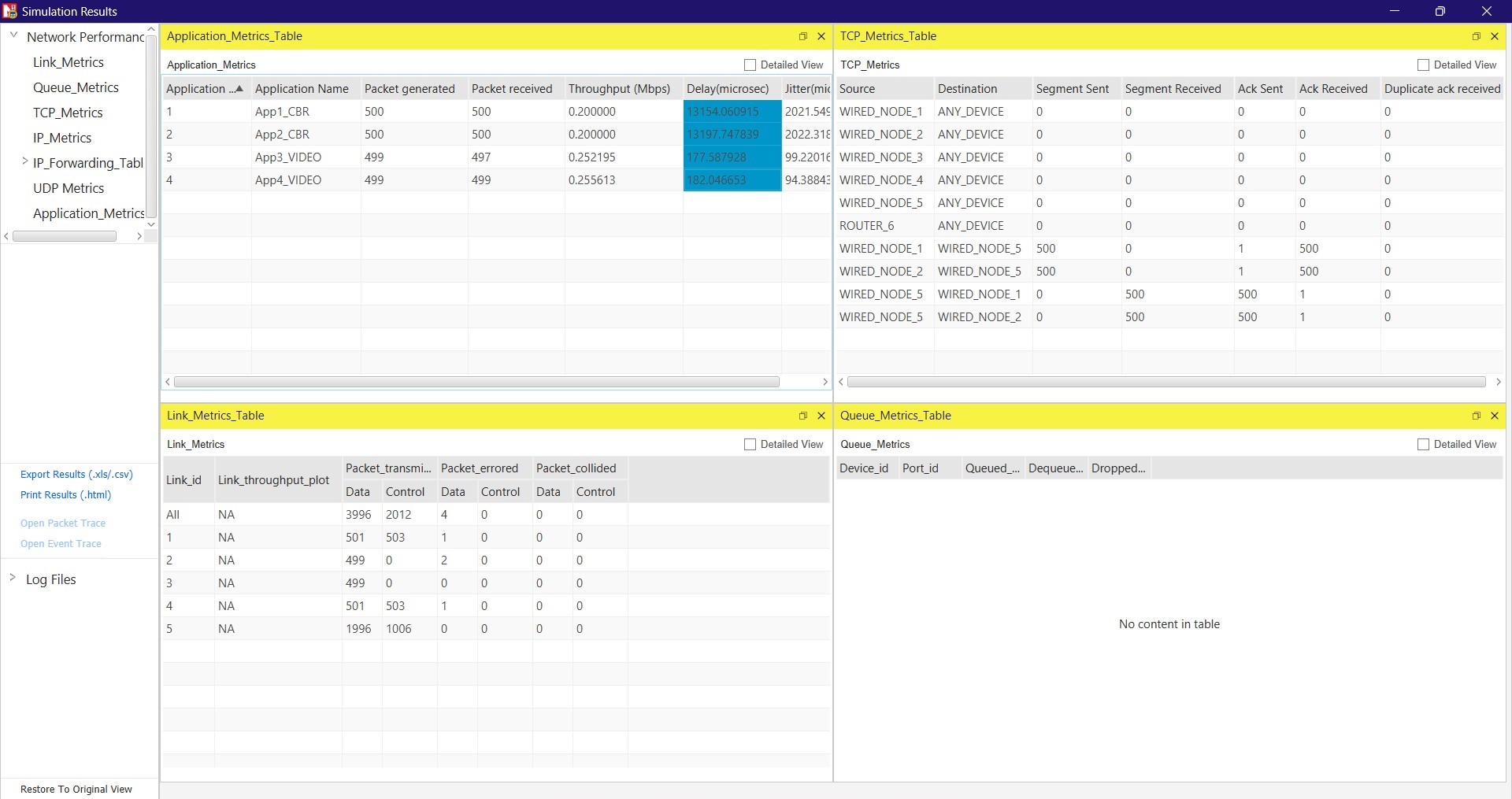
**1. Calculate and Observe average throughput of both the applications (CBR and VIDEO).**

Here are the notations, 1=A; B=2; C=3; D=4; E=5; 6=A.



**2. Observe the delay and throughput metrics in the simulation window and write down your observation.**





**(4) Analysing RTT Of TCP Using Wireshark.**

**4.1 Exercise:**

**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**

**(Once you have downloaded the trace, you can load it into Wireshark and view the trace using the File pull down menu, choosing Open, and then selecting the tcp-ethereal-trace-1 trace file.)**

**4.2 Questions:**

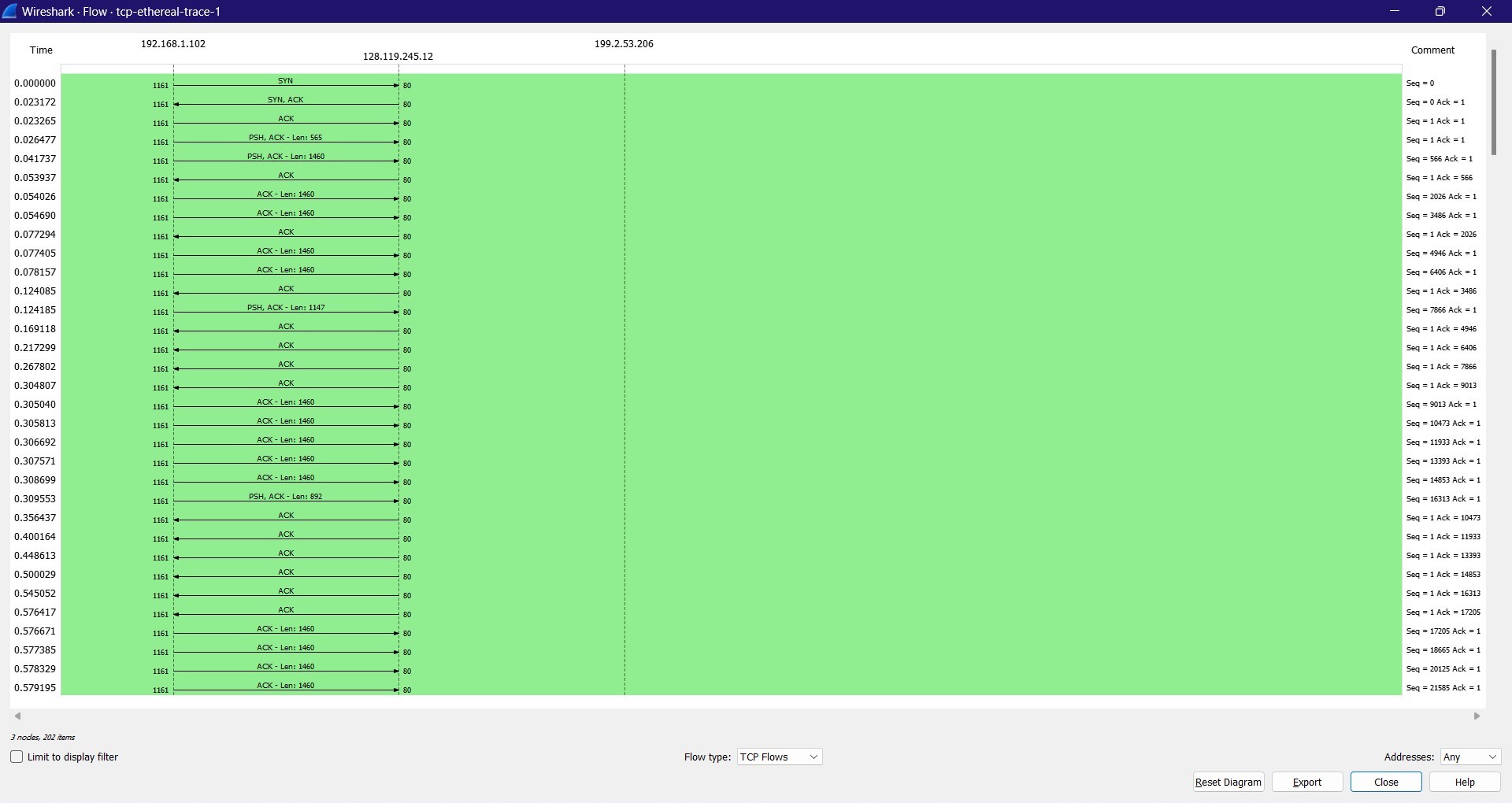
**1. Consider the TCP segment containing the HTTP POST as the first segment in the TCP connection. What are the sequence numbers of the first six segments in the TCP connection (including the segment containing the HTTP POST)? At what time was each segment sent? When was the ACK for each segment received? 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? What is the EstimatedRTT value after the receipt of each ACK?**

**Note: Wireshark has a nice feature that allows you to plot the RTT for each of the TCP segments sent. Select a TCP segment in the listing of captured packets window that is being sent from the client to the gaia.cs.umass.edu server. Then select: Statistics− >TCP Stream Graph− >Round Trip Time Graph.**

Ans:

Sequence Numbers Of First Six Segments:

1. Seq 1 - Sent At 0.023265; ACK Received At 0.053937.
2. Seq 1 - Sent At 0.026477; ACK Received At 0.077294.
3. Seq 1 - Sent At 0.041737; ACK Received At 0.124085.
4. Seq 1 - Sent At 0.054026; ACK Received At 0.169118.
5. Seq 1 - Sent At 0.054690; ACK Received At 0.217299.
6. Seq 1 - Sent At 0.077405; ACK Received At 0.267802.



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

Ans:

1st TCP Segment - 565 bytes.

2nd TCP Segment - 1460 bytes.

3rd TCP Segment - 1460 bytes.

4th TCP Segment - 1460 bytes.

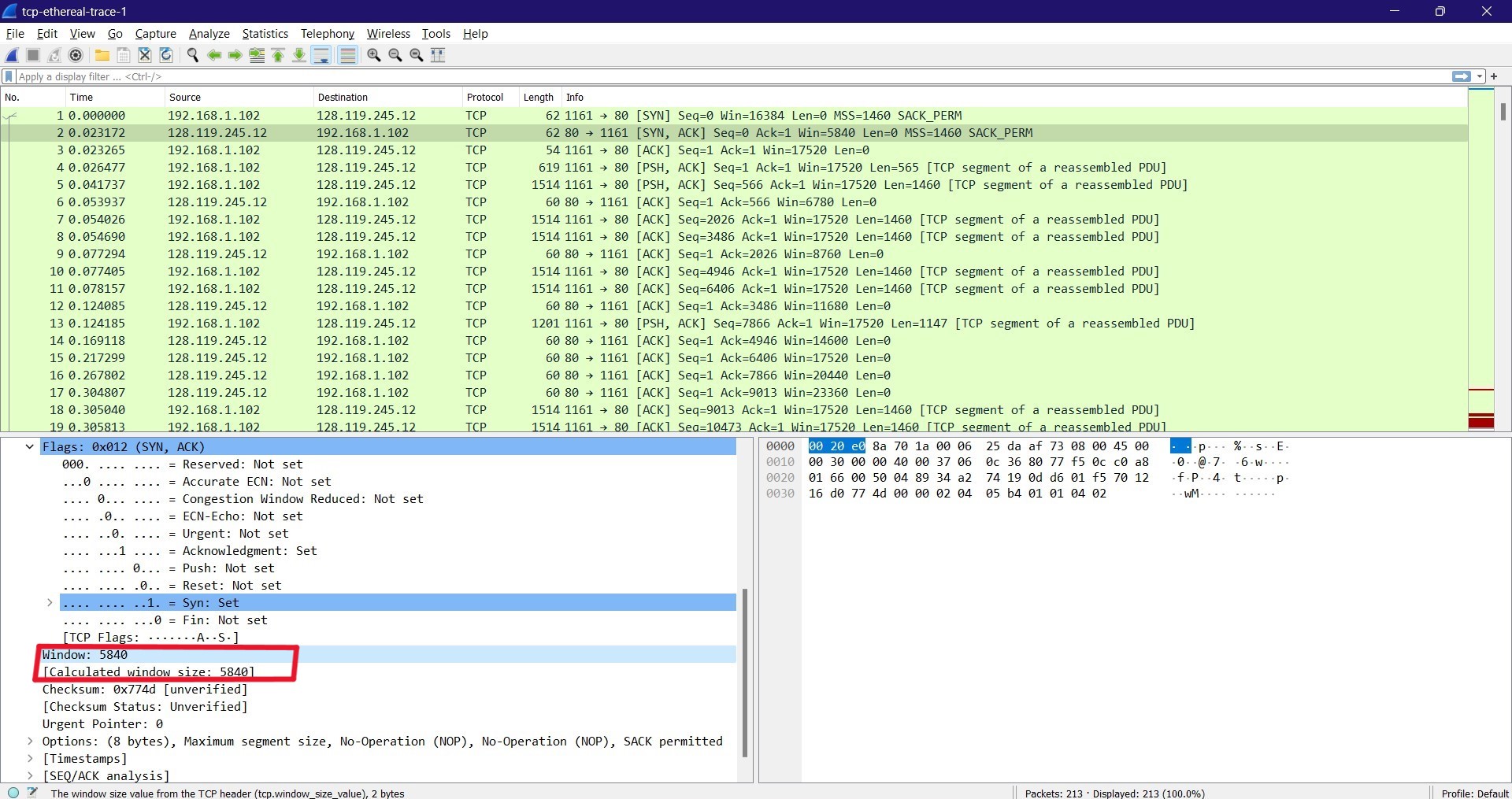
5th TCP Segment - 1460 bytes.

6th TCP Segment - 1460 bytes.

**3. 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?**

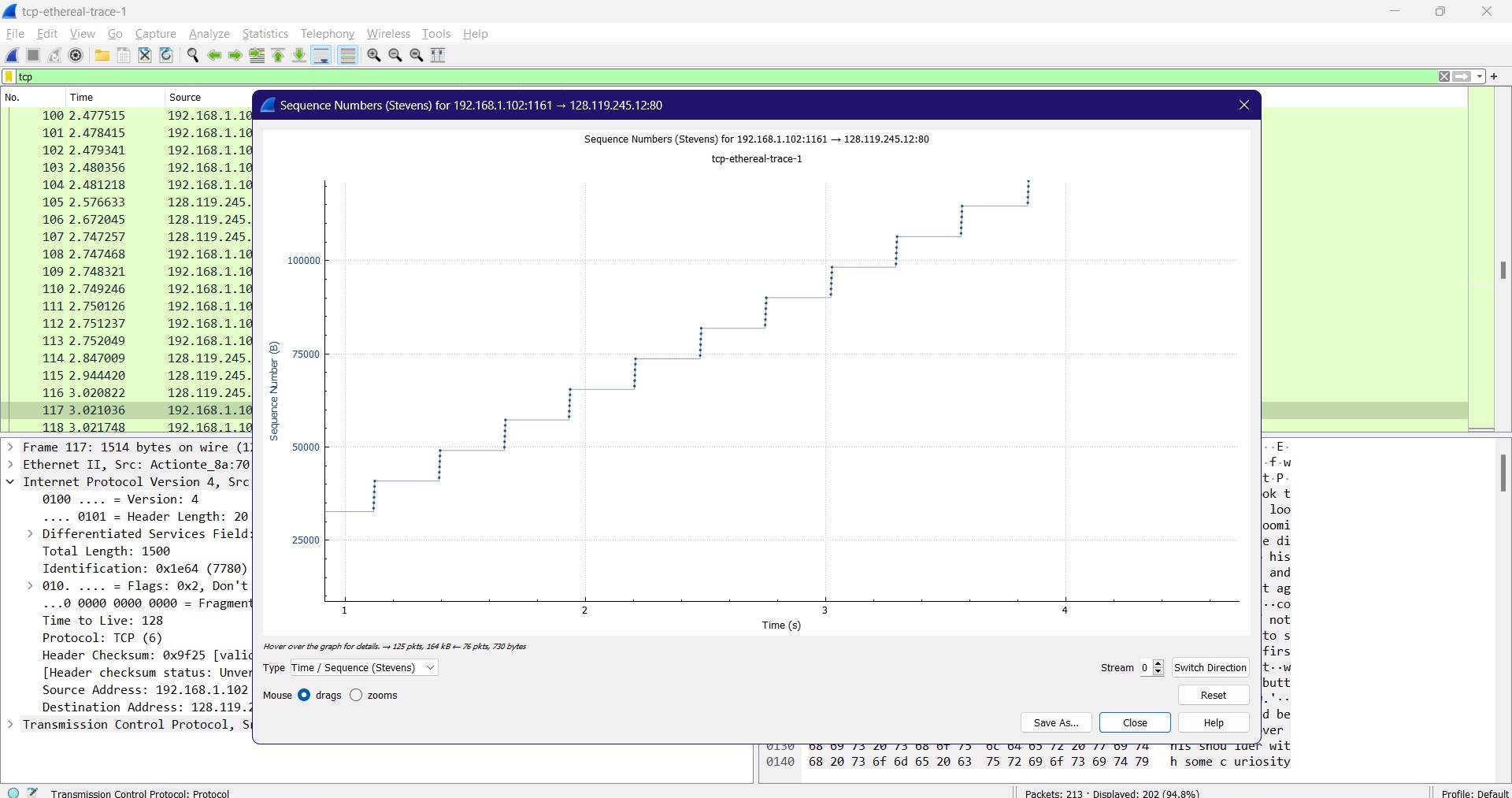
Ans: The minimum amount of available buffer space advertised at the received for the entire trace indicated first ACK from the server, its value is 5840 bytes.

The trace file contains no retransmitted segments. We can check the sequence numbers of the TCP segments in the trace file to confirm this. All sequence numbers from the source to the destination increase monotonically with respect to time in this trace's TimeSequence-Graph (Stevens).



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

Ans:

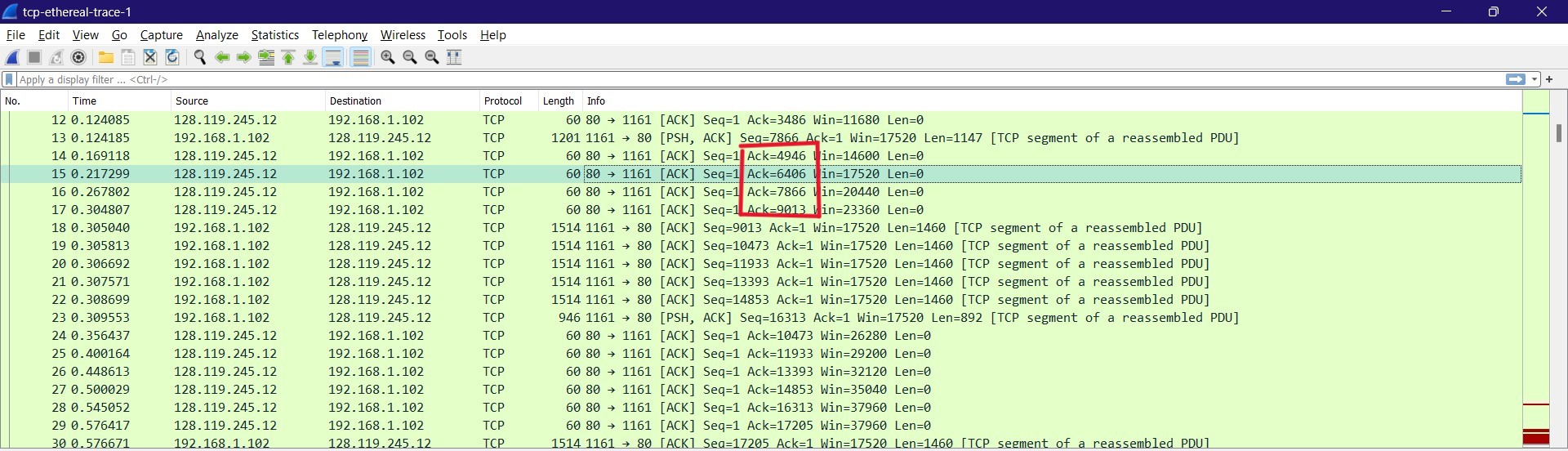


There are no retransmitted segments in the trace file because all sequence numbers in the time sequence graph (Stevens) are monotonically increasing.

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

Ans:

The difference between two consecutive acknowledged sequence numbers indicates the data received by the server between these two ACKs.



**6. What is the throughput (bytes transferred per unit time) for the TCP connection? Explain how you calculated this value.**

Ans:

According to the packet's acknowledgement number, 6406 bytes were acknowledged, as can be seen when looking at it. This message was sent at 0.217299. A rough estimate of the average throughput is 6406 bytes/0.217299 seconds, or 29,480 bytes/seconds.

**(5) Analysing UDP Protocol Using Wireshark**

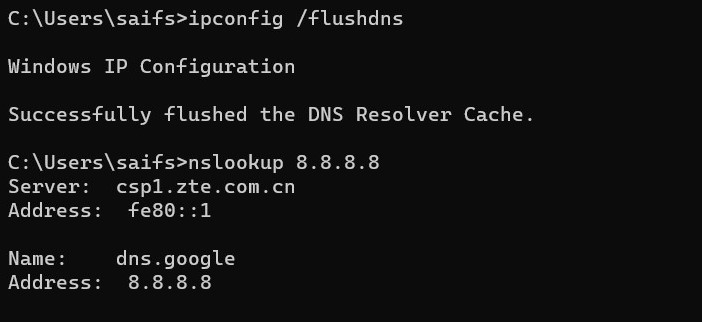
**5.1 Exercise**

1. Start a Wireshark capture.

2. Open a command prompt.

3. Type ipconfig /flushdns and press Enter to clear your DNS name cache.

4. Type nslookup 8.8.8.8 and press Enter to look up the hostname for IP address 8.8.8.8.



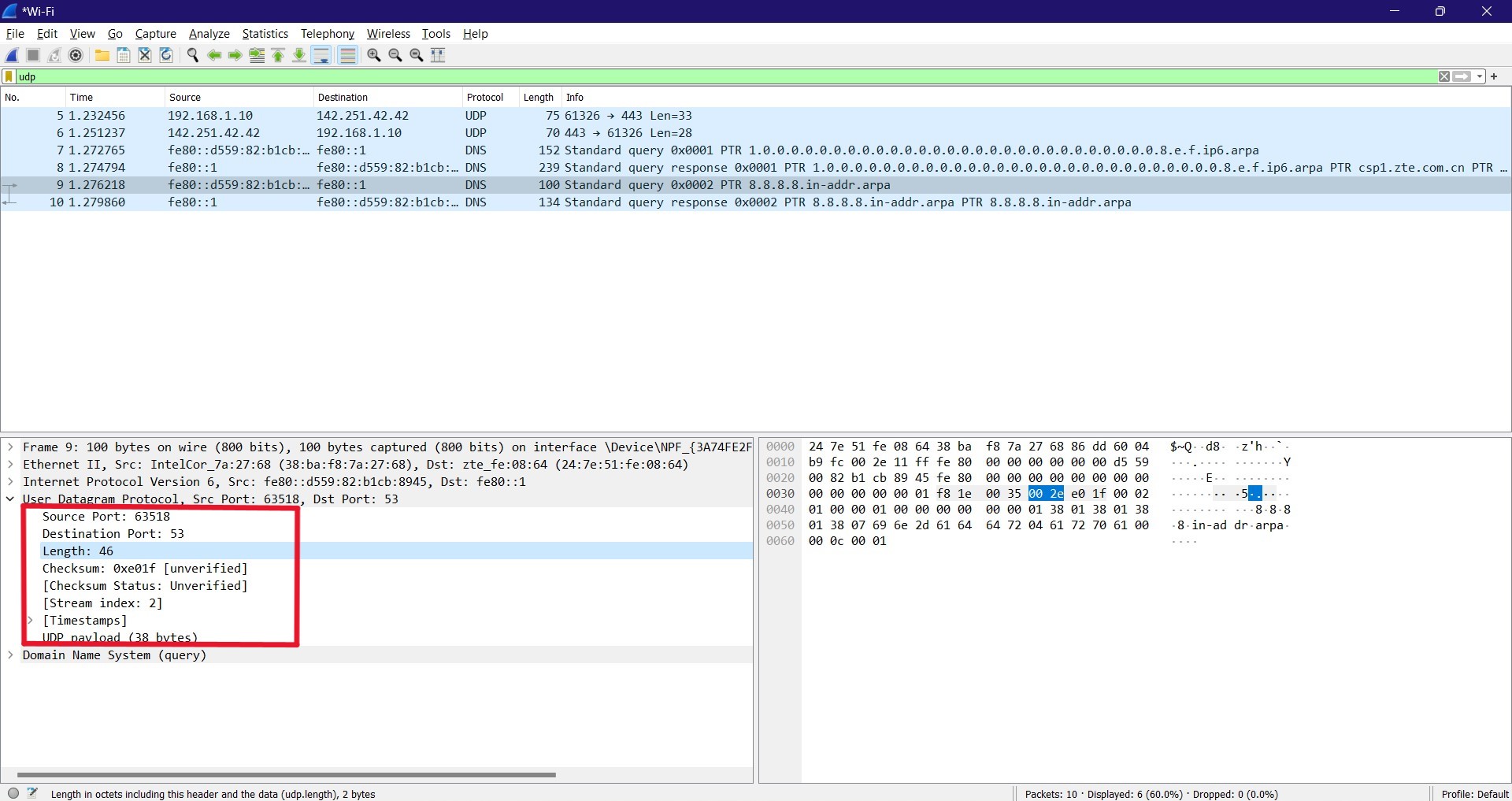
5. Close the command prompt.

6. Stop the Wireshark capture.

**5.2 Questions**

**1. Select one UDP packet from your trace. From this packet, determine how many fields there are in the UDP header. (You shouldnt look in the textbook! Answer these questions directly from what you observe in the packet trace.) Name these fields.**

Ans: UDP Header contains 4 fields. The six fields are Source Port, Destination Port, Length & Checksum.



**2. By consulting the displayed information in Wiresharks packet content field for this packet, determine the length (in bytes) of the UDP header fields.**

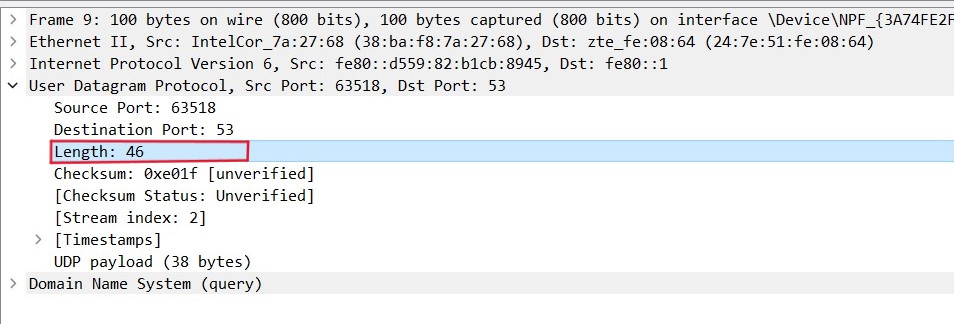
Ans: Length Of UDP Header: 8 bytes.

Header length = Length field bytes - UDP payload bytes = 46 - 38 = 8 bytes.

**3. The value in the Length field is the length of what? (You can consult the text for this answer). Verify your claim with your captured UDP packet.**

Ans: Value in the Length field is the length of the UDP segment (header + data).

Here the length is 46 bytes which is 8 header bytes + 38 UDP payload bytes.



**4. What is the maximum number of bytes that can be included in a UDP payload? (Hint: the answer to this question can be determined by your answer to 2. above)**

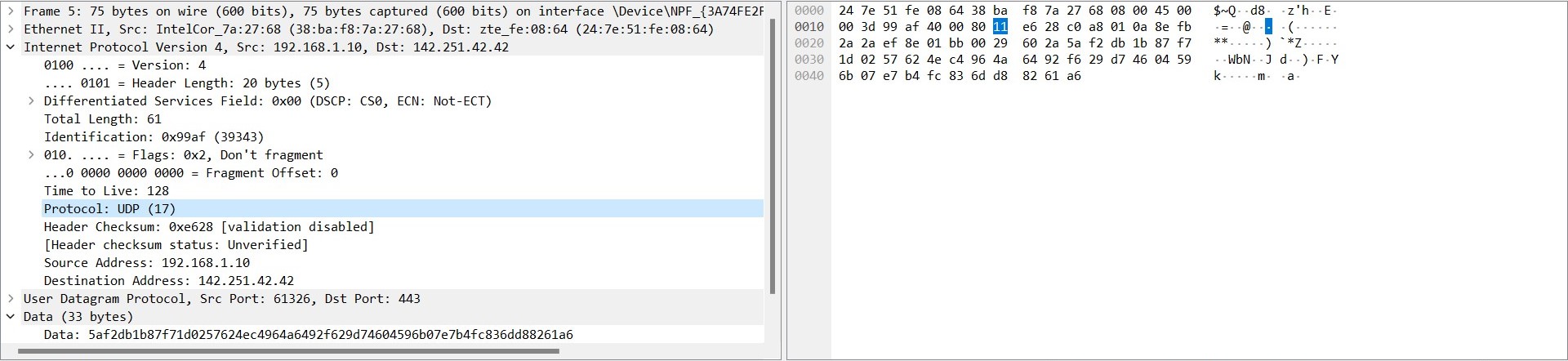
Ans: A UDP payload can contain a maximum of (2^16 - 1) bytes plus the header bytes. This results in 65535 bytes - 8 bytes = 65527 bytes.

**5. What is the largest possible source port number? (Hint: see the hint in 4.)**

Ans: The largest possible Source Port Number is (2^16 – 1) = 65535.

**6. What is the protocol number for UDP? Give your answer in both hexadecimal and decimal notation. To answer this question, youll need to look into the Protocol field of the IP datagram containing this UDP segment.**

Ans: The protocol number for UDP is 0x11 hex and it is 17 in decimal value.



**7. Why we have used DNS commands to capture UDP packets? Do you know any-other method to generate UDP traffic using wireshark? Write your answer in detail.**

Ans: We use DNS commands to capture UDP packets because DNS uses UDP as the transport protocol for the majority of its queries and responses.

Other ways to generate UDP traffic are:

-Using a basic UDP client-server application: We can write a basic UDP client-server application that sends and receives UDP packets over a specific port. Wireshark can then be used to capture the traffic and analyze the packets.