**IT304**

**Computer Networks**

**Lab 4**

**Analyzing congestion policy, RTT of TCP and working of UDP using**

**NetSim and wireshark.**

**Group ID – TG3**

Prepared by :-

Aditya Nawal (202001402)

Divya Patel (202001420)

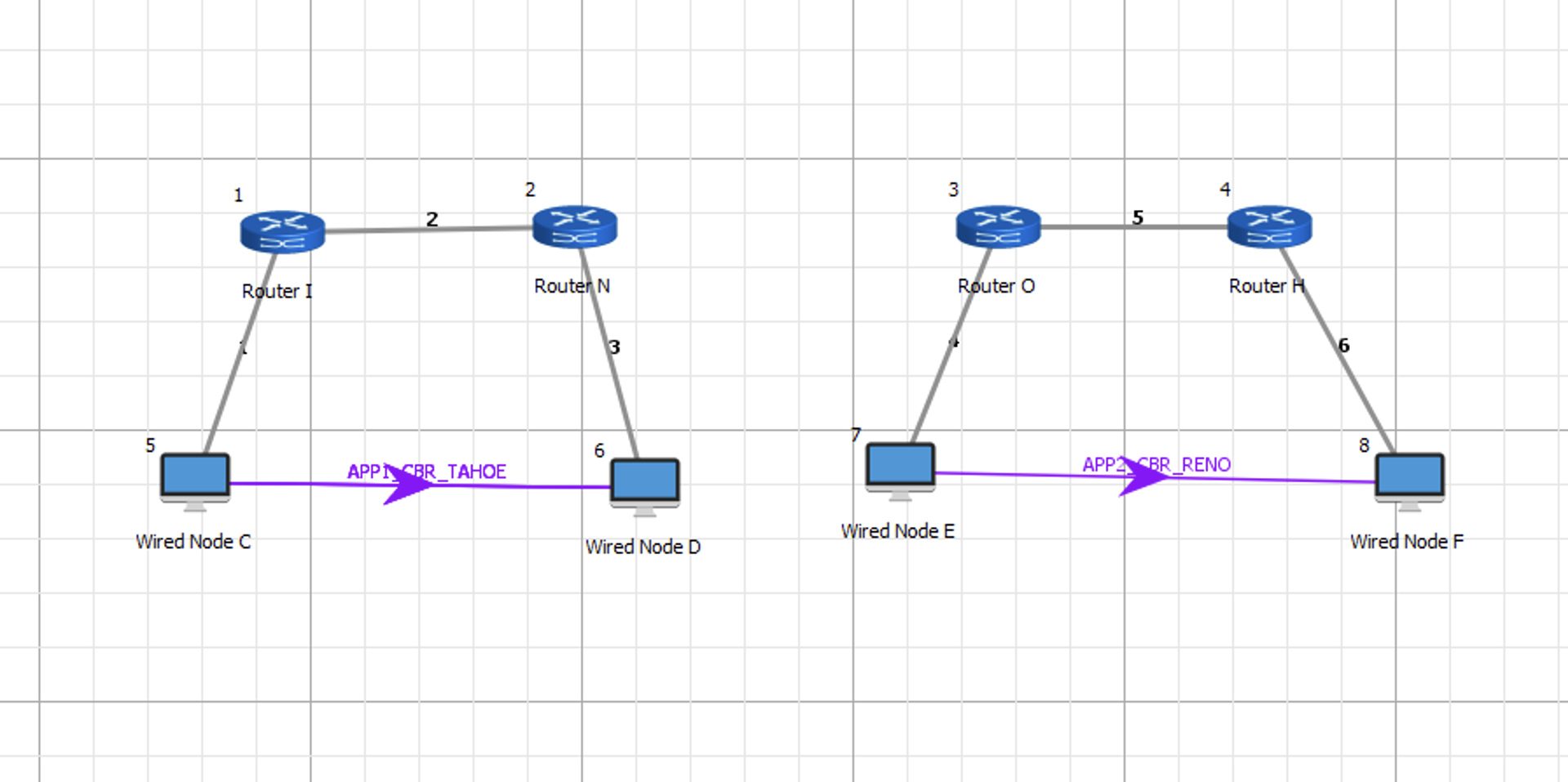
Kunj Patel (202001421)

Aryan Shah (202001430)

Experiment 1 - Introduction of Congestion Policy of TCP

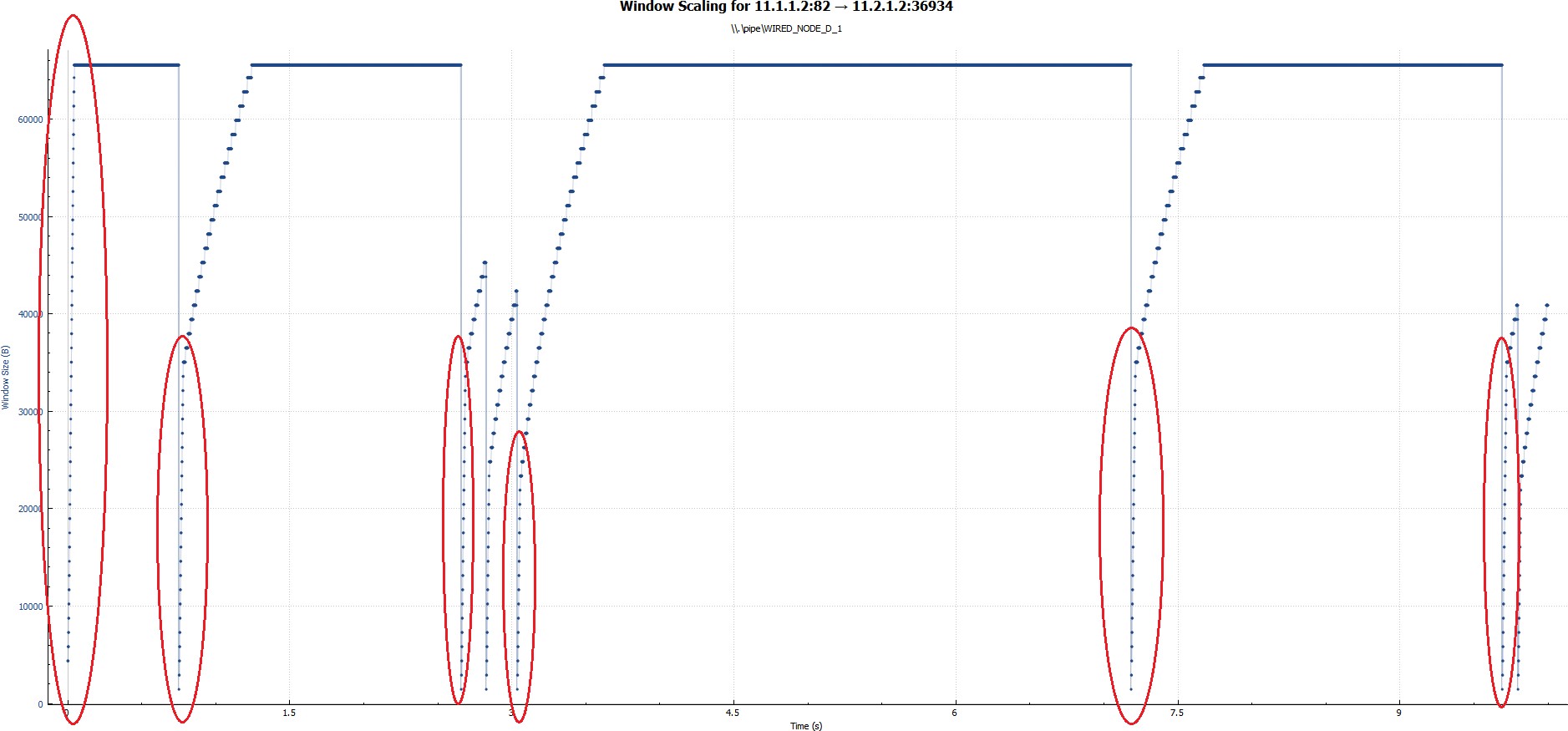
**Ques 1: For both the variant, analyze graph of congestion window, answer the following by marking in the graph.**

**Topology**

****

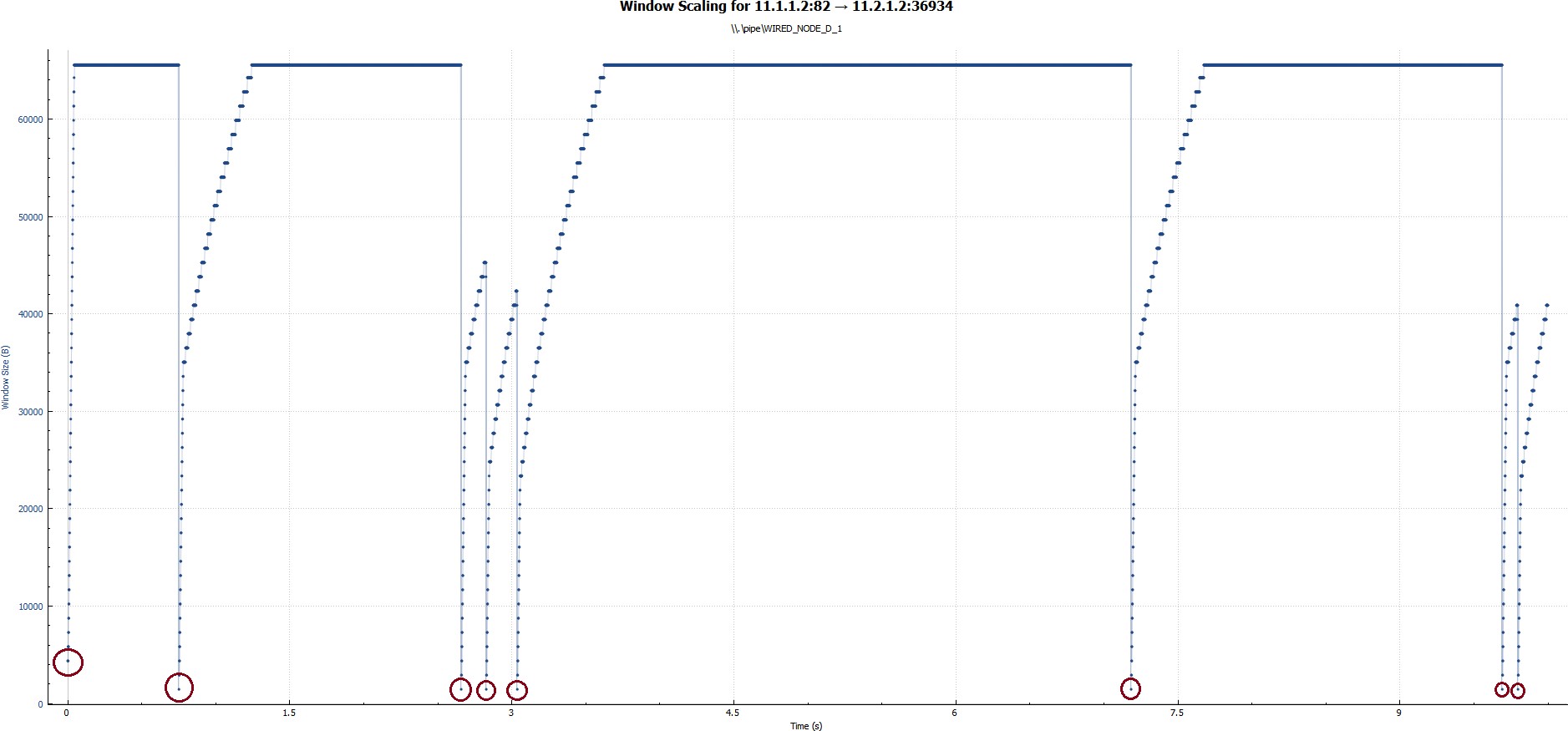
For TCP Tahoe :-

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



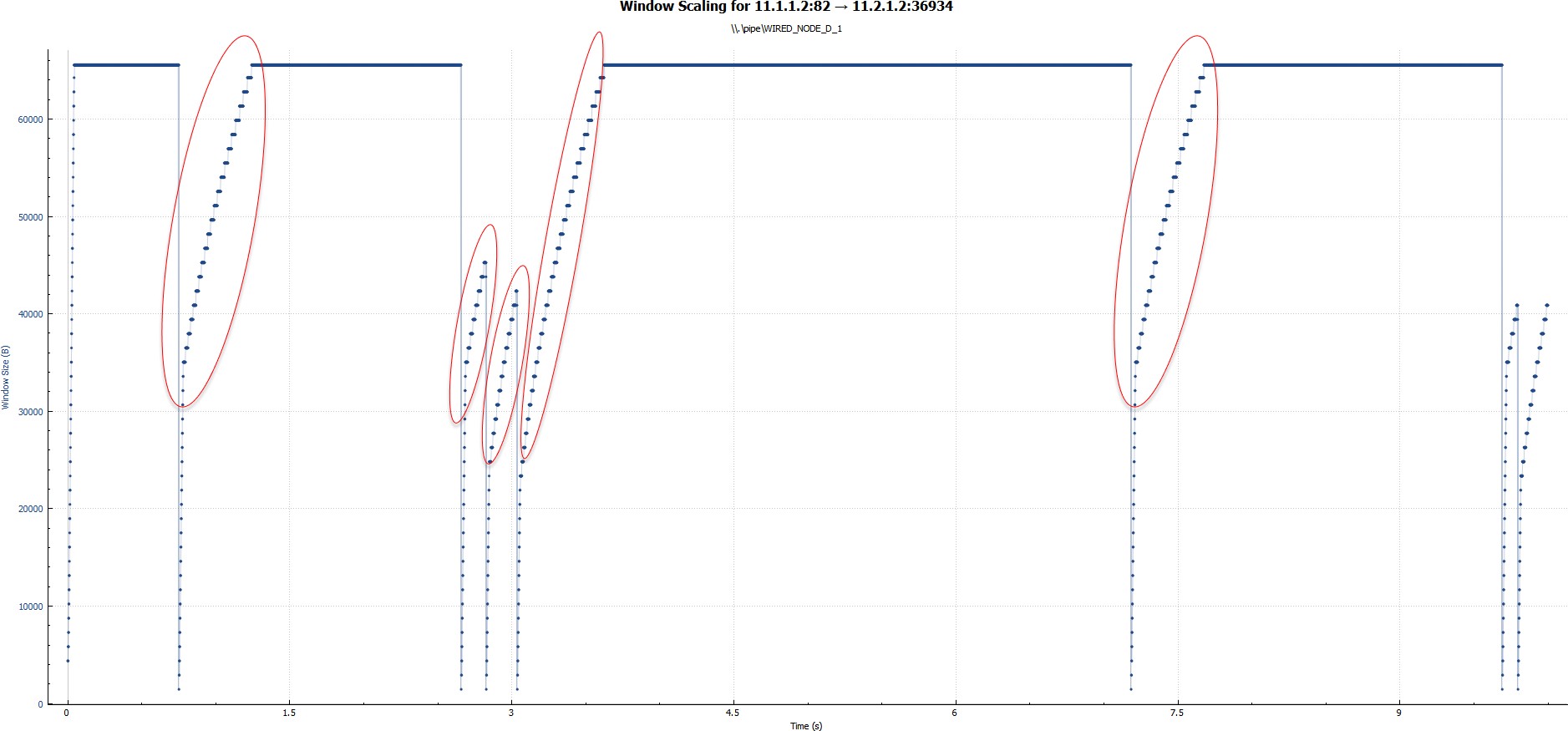
TCP Tahoe

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



TCP Tahoe

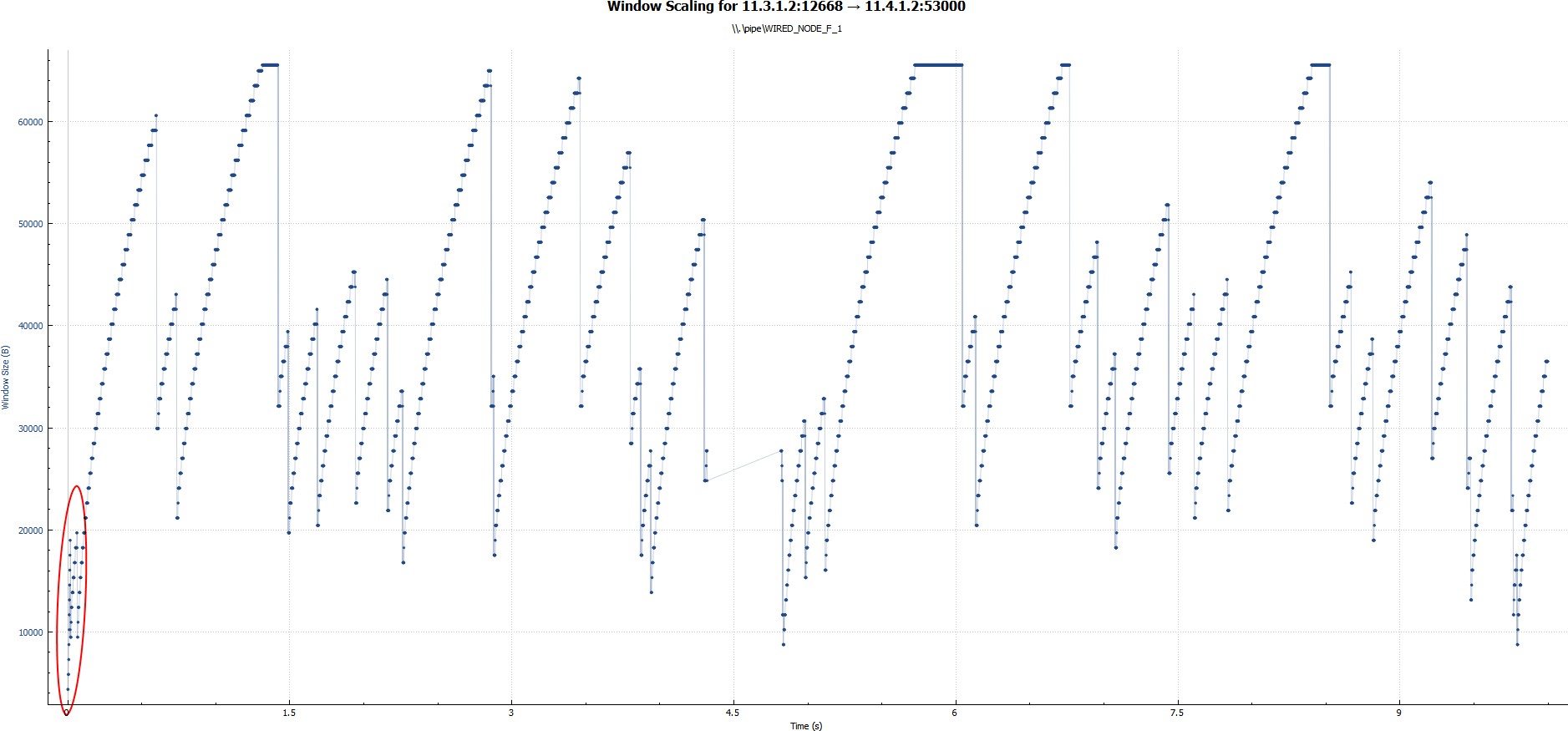
**(c) Identify the intervals of time when TCP congestion avoidance is operating.**



TCP Tahoe

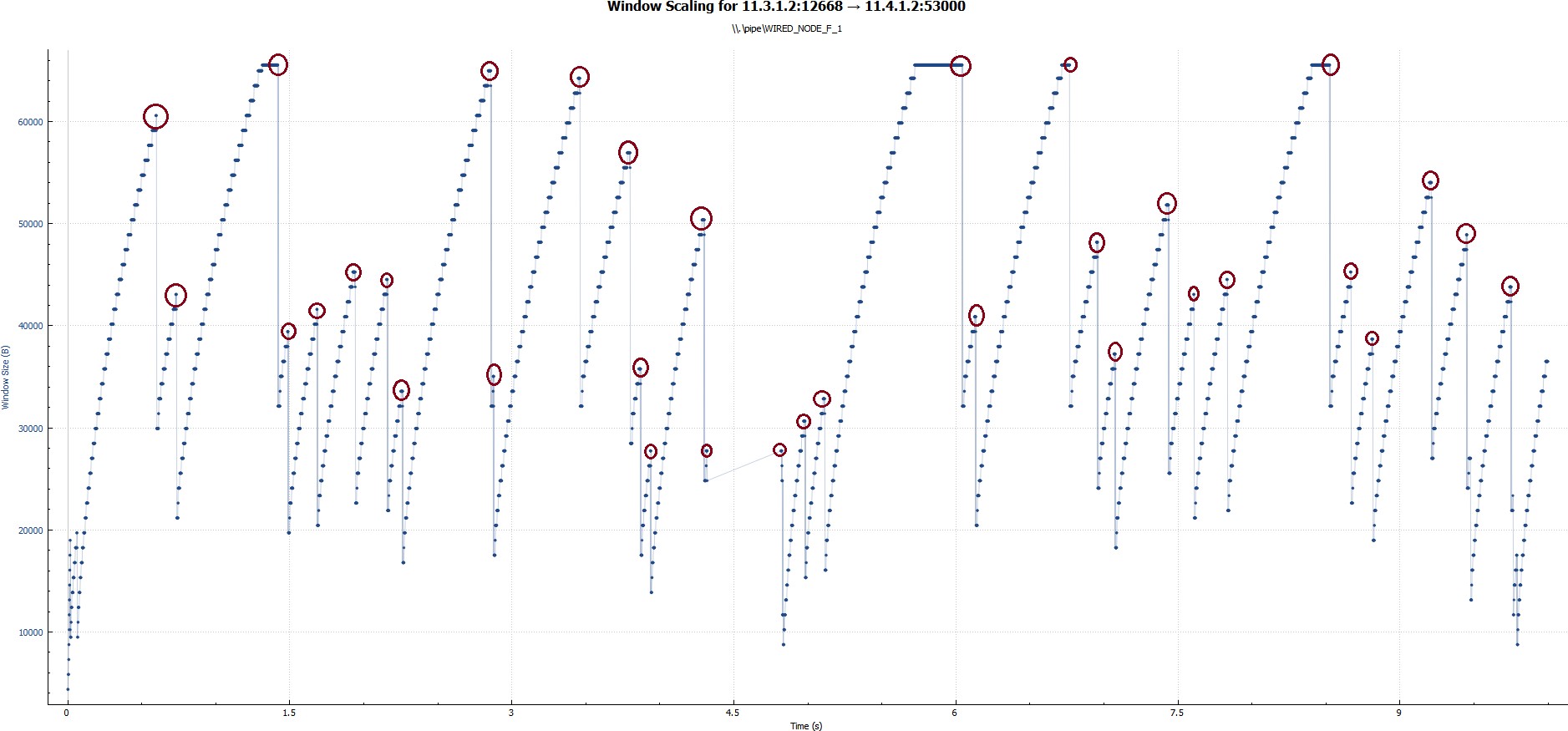
For TCP Reno: -

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



TCP Reno

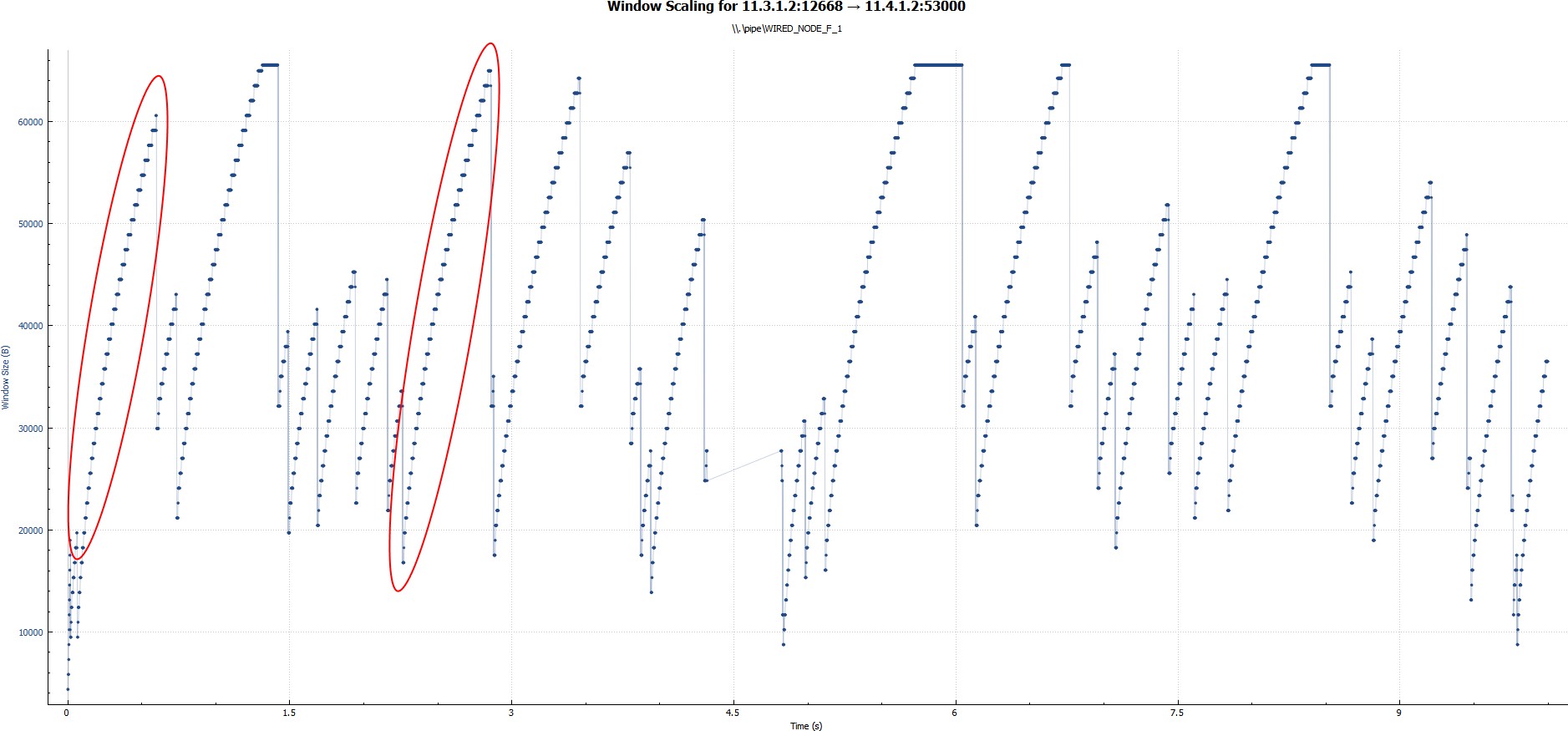
**(b) Identify the event of packet loss and time out.**



TCP Reno

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

TCP Reno



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

In the case of Tahoe,

At Timeout: Slow start,

Packet loss: Slow start

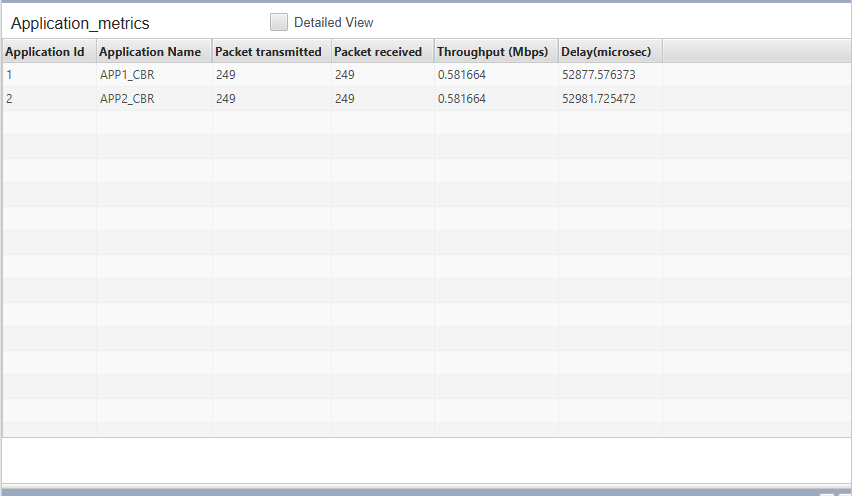
Whereas in the case of Reno,

At Timeout: Slow start,

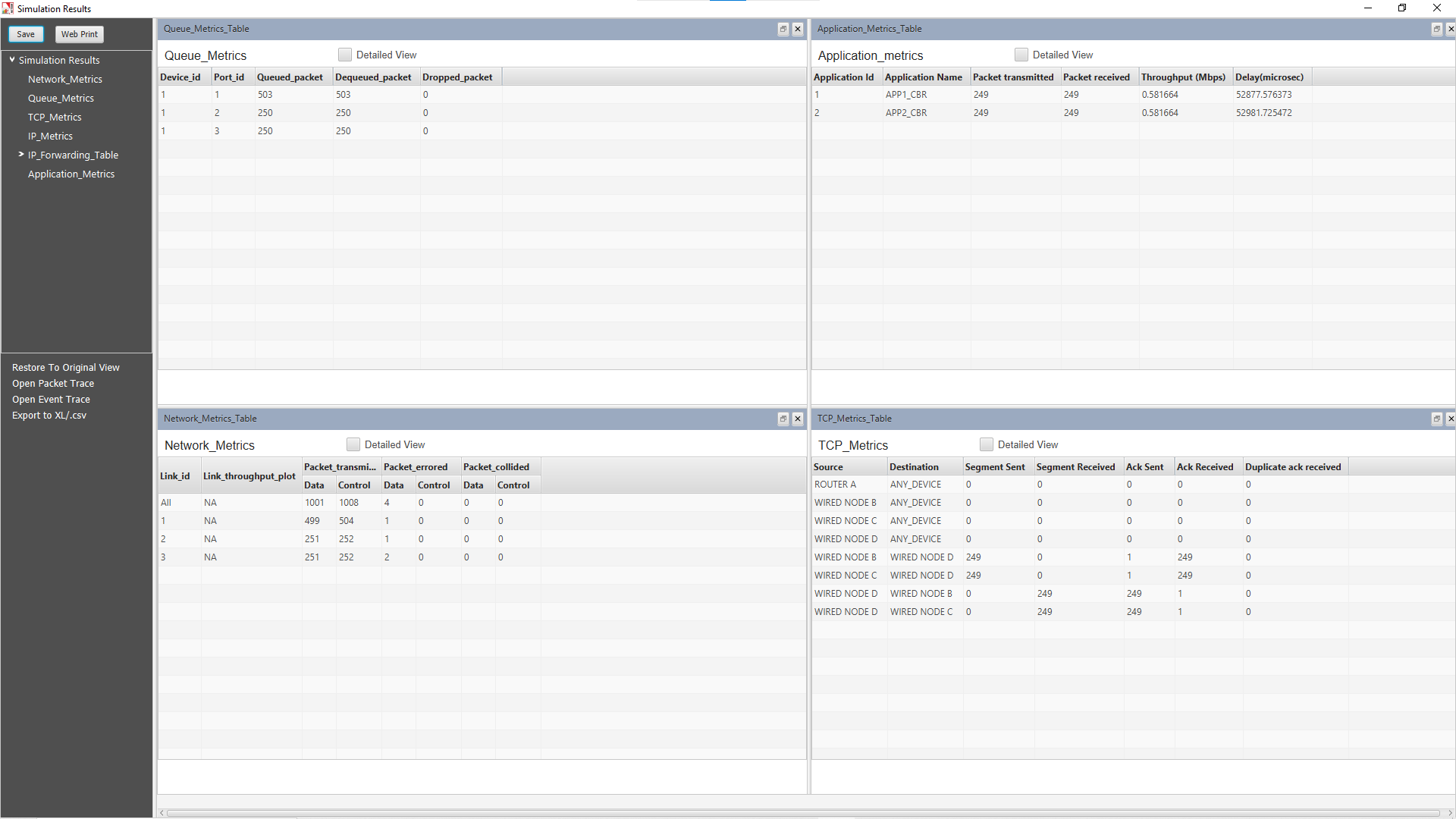
Packet loss: Starts from threshold (congestion avoidance)

Experiment 2 - Analyzing the fairness of TCP

As you can see in the screenshots, the throughput for both links is equal to **0.581664 Mbps.**



Throughput of both applications remain same, this indicates that TCP protocol is fair.



Experiment 3 - Analyzing throughput

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

Average throughput for CBR: 0.199600 Mbps

Average throughput for Video: 0.258985 Mbps

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

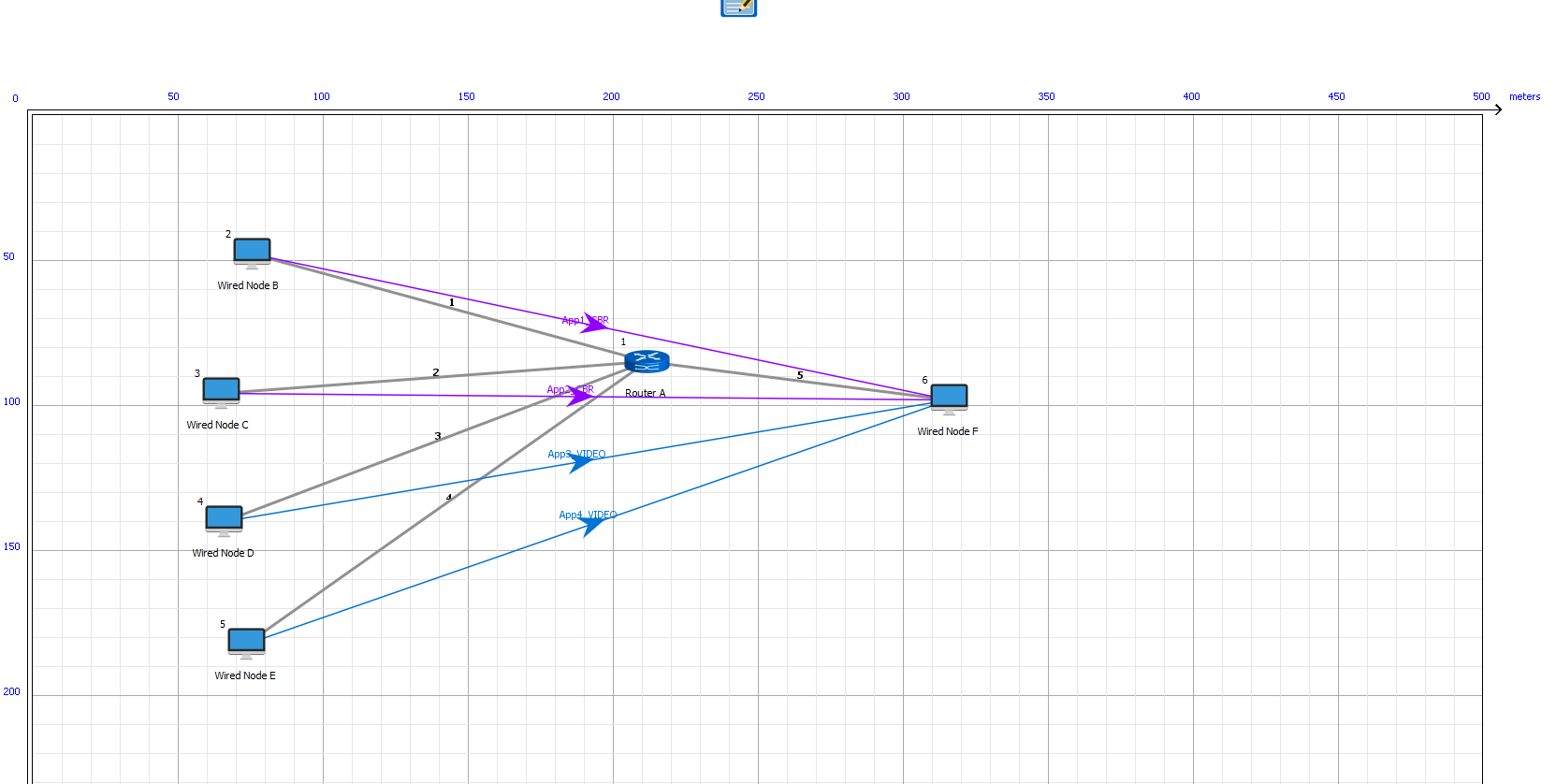
Application Name Throughput (Mbps) Delay(microsec)

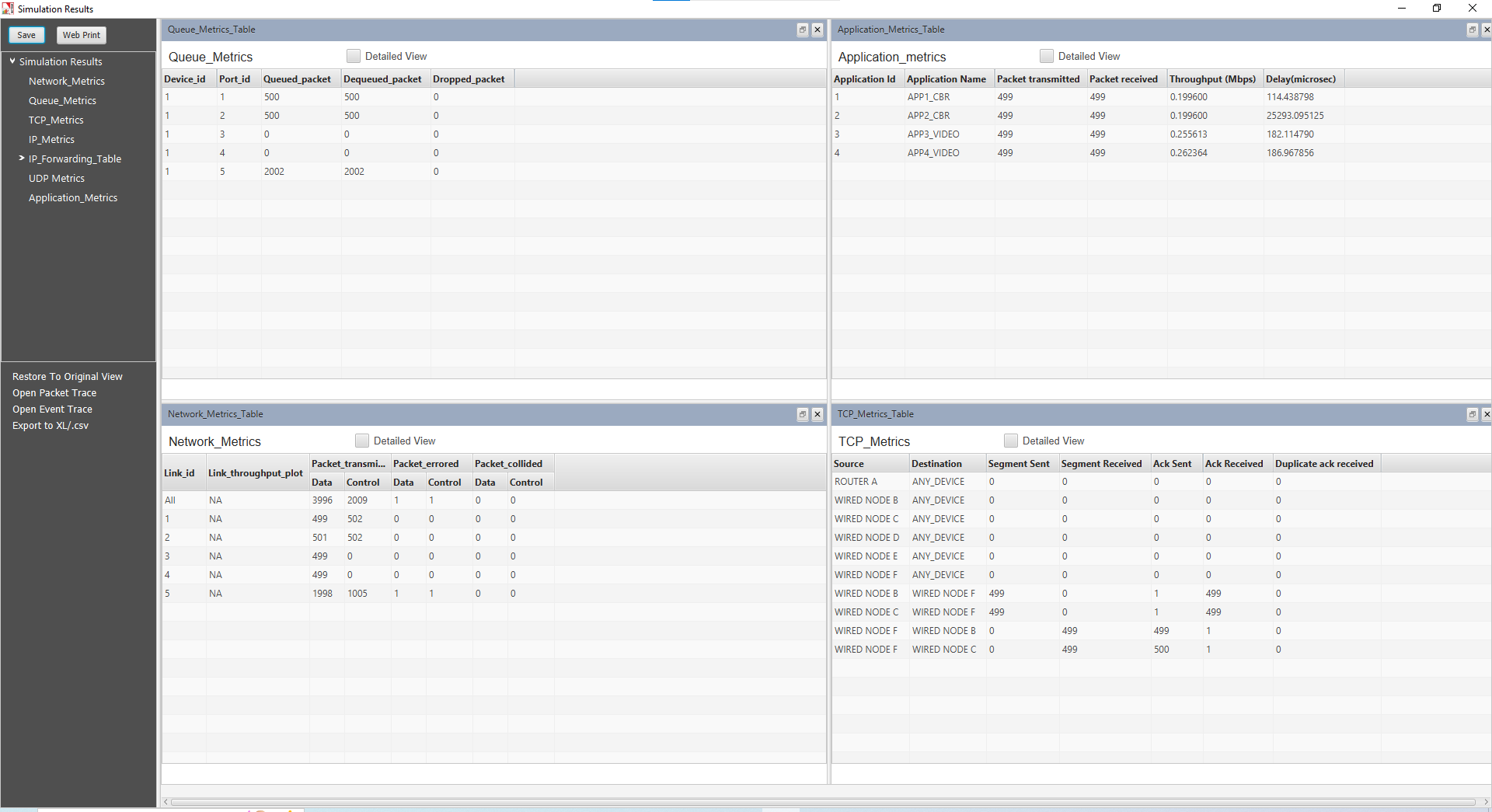
APP1\_CBR 0.199600 114.438798

APP2\_CBR 0.199600 25293.095125

APP3\_VIDEO 0.255613 182.114790

APP4\_VIDEO 0.262364 186.967856

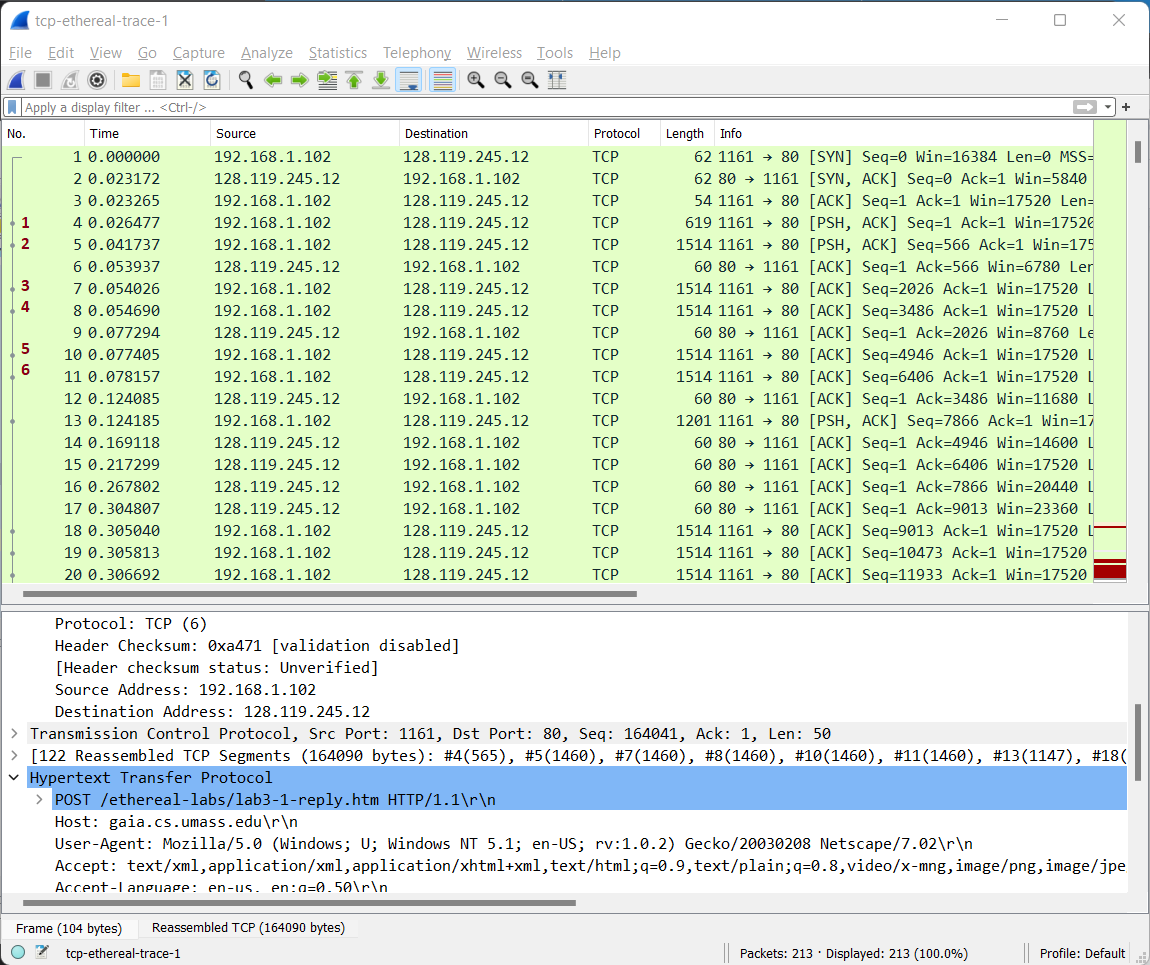




Experiment 4 - Analysing RTT of TCP using Wireshark

**Ques 1: Consider the TCP segment containing the HTTP POST as the rst segment in the TCP connection. What are the sequence numbers of the rst 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?**

Alpha = 0.125



Consider labelled 6 packets

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Sequence number | Sending time | Receiving time | RTT | Estimated RTT |
| 1st packet | 1 | 0.026477 | 0.053937 | 0.02746 | 0.02746 |
| 2nd packet | 566 | 0.041737 | 0.077294 | 0.035557 | 0.0285 |
| 3rd packet | 2026 | 0.054026 | 0.124085 | 0.070059 | 0.0337 |
| 4th packet | 3486 | 0.054690 | 0.169118 | 0.11443 | 0.0438 |
| 5th packet | 4946 | 0.077405 | 0.217299 | 0.13989 | 0.0508 |
| 6th packet | 6406 | 0.078157 | 0.267802 | 0.18964 | 0.0725 |

Consider labelled 6 packets

EstimatedRTT = 0.875\*EstimatedRTT + 0.125\*SampleRTT

EstimatedRTT for 1st segment:

EstimatedRTT= RTT for segment 1 = 0.02746

EstimatedRTT for 2nd segment:

EstimatedRTT = 0.875\*0.02746 + 0.125\*0.035557 = 0.0285

EstimatedRTT for 3rd segment:

EstimatedRTT= 0.875\*0.0285 + 0.125\*0.070059 = 0.0337

EstimatedRTT for 4th segment:

EstimatedRTT= 0.875\*0.0337 + 0.125\*0.11443 = 0.0438

EstimatedRTT for 5th segment:

EStimatedRTT = 0.875\*0.0438 + 0.125\*0.13989 = 0.0508

EstimatedRTT for 6th segment:

EstimatedRTT = 0.875\*0.0558 + 0.125\*0.18964 = 0.0725

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

The length of 1st segment: 566

The length of 2nd segment: 1460

The length of 3rd segment: 1460

The length of 4th segment: 1460

The length of 5th segment: 1460

The length of 6th segment: 1460

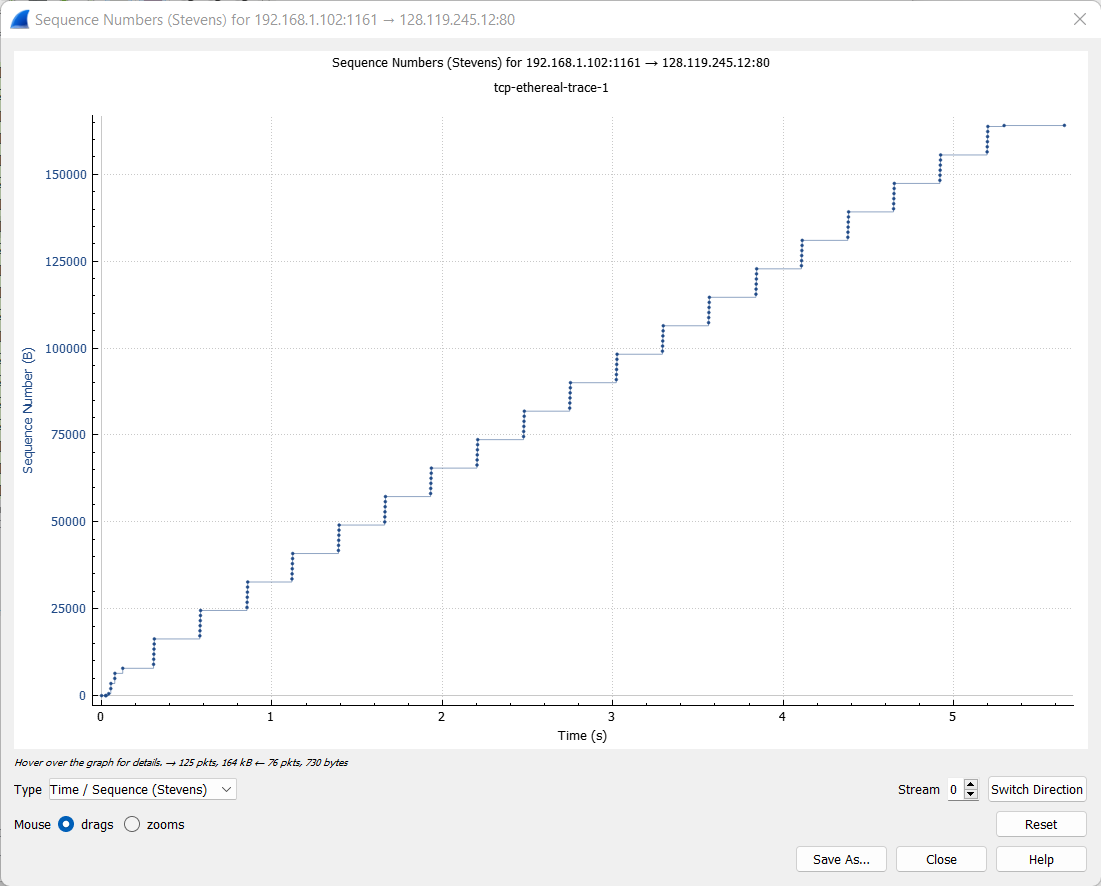
**Ques 3: What is the minimum amount of available buffer space advertised at the received for the entire trace? Does the lack of receiver buffer space ever throttle the sender?**

Minimum calculated window size (Buffer Space): 5840 bytes.

Maximum length of the segment transmitted (MSS): 1460 bytes.

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

In this packet there is no retransmission. We can verify with the use of sequence numbers in a trace file. In the time-sequence graph (stevens) of this trace, all sequence numbers from the source(192.168.1.102) to destination(128.119.245.12) are increasing monotonically with time. If there is a retransmitted segment, the sequence number of this retransmitted segment should be smaller than those of its neighboring segments.



**Ques 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?**

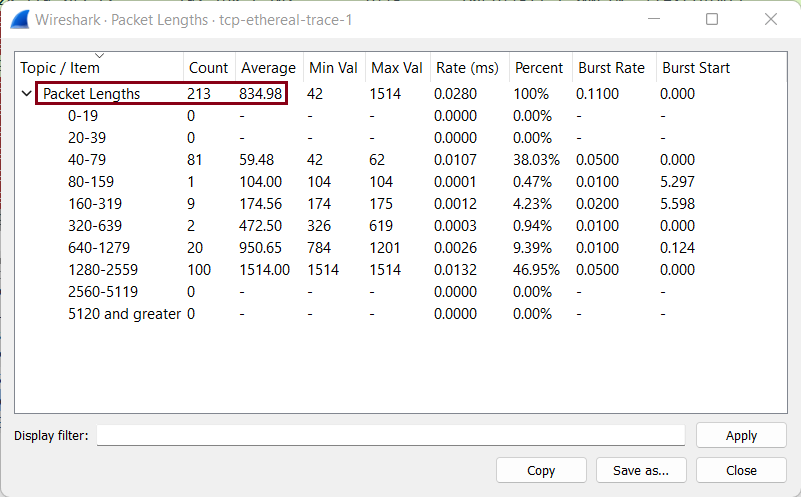
Receiver is typically acknowledging 1460 bytes in an ACK.

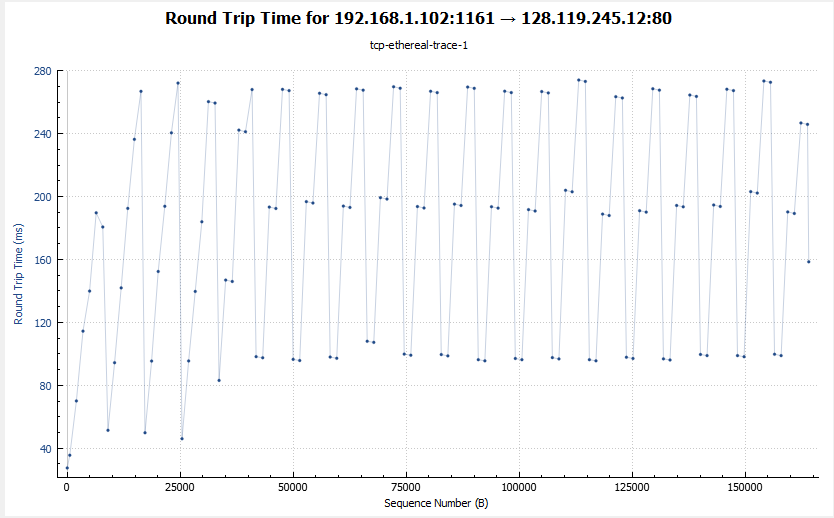
**Ques 6: What is the throughput (bytes transferred per unit time) for the TCP connection? Explain how you calculated this value.**

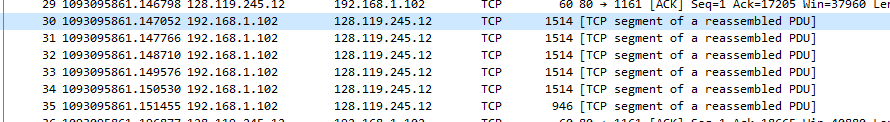
We can calculate the throughput as:

(total transferred data)/(require time) = 177850.74/7.595557

= 23415.06 Bytes/sec **~=22.87KB/sec**

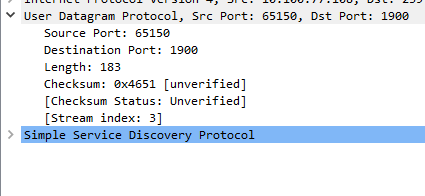
Count \*average packet length = 213\* 834.98 = 177,850.74





Experiment 5 - Analysing UDP protocol using Wireshark

**Ques 1: Select one UDP packet from your trace. From this packet, determine how many fields there are in the UDP header. (You shouldn't look in the textbook! Answer these questions directly from what you observe in the packet trace.) Name these fields.**



There are 4 fields in the UDP header: source port, destination port, Length, and Checksum.

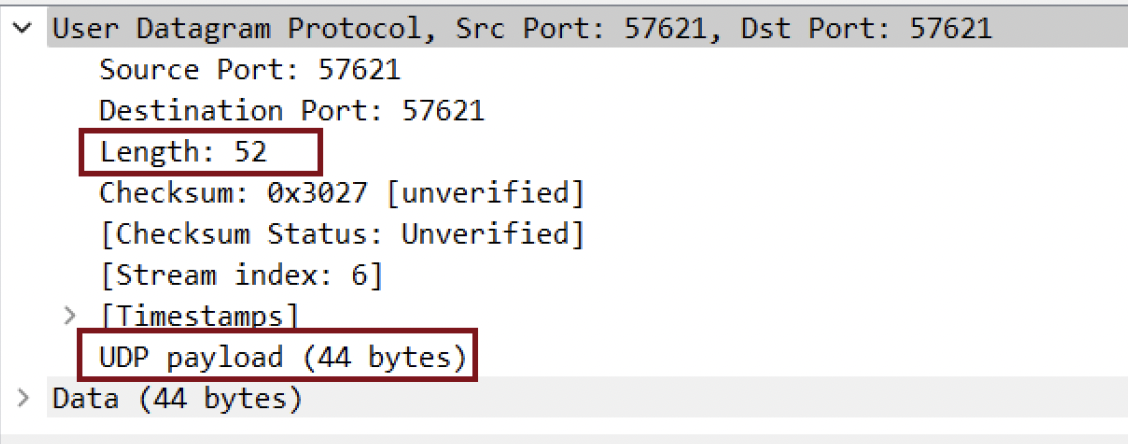
**Ques 2: By consulting the displayed information in Wireshark's packet content field for this packet, determine the length (in bytes) of the UDP header fields.**

Length of UDP header = Length of UDP Length of UDP Payload

= 52 - 44 = 8

Length of UDP header field is 8 bytes.

Each of these 4 header fields is 2 bytes long.



**Ques 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.**

From screenshot below we can see that length of UDP payload is 44 bytes. Whereas total length of UDP packet is 52 bytes. These extra 8 bytes are from UDP header fields.

Length Field = 52 bytes

UDP Header = 8 bytes

So, Payload = 44 bytes

**Ques 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)**

A UDP port is a 16-bit number, meaning there are theoretically 65,535 possible values it can have. Out of these 8 bytes are required for UDP header fields.  
  
Maximum number of bytes included = (2^16 - 1) - 8 (for header) = 65527 bytes.

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

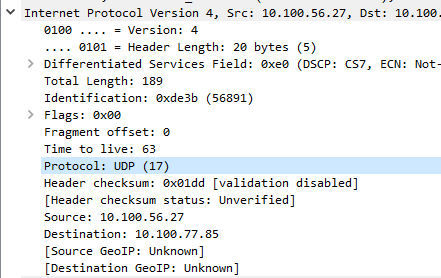
Largest Possible source port number = 2^16-1 = **65535**

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

UDP protocol in :

Hexadecimal = 0x11

Decimal = 17



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

DNS commands are used since:

1. DNS servers don’t have to keep the connection established.
2. DNS requests can be easily fit in UDP segments due to their small size.