**Assignment 3**

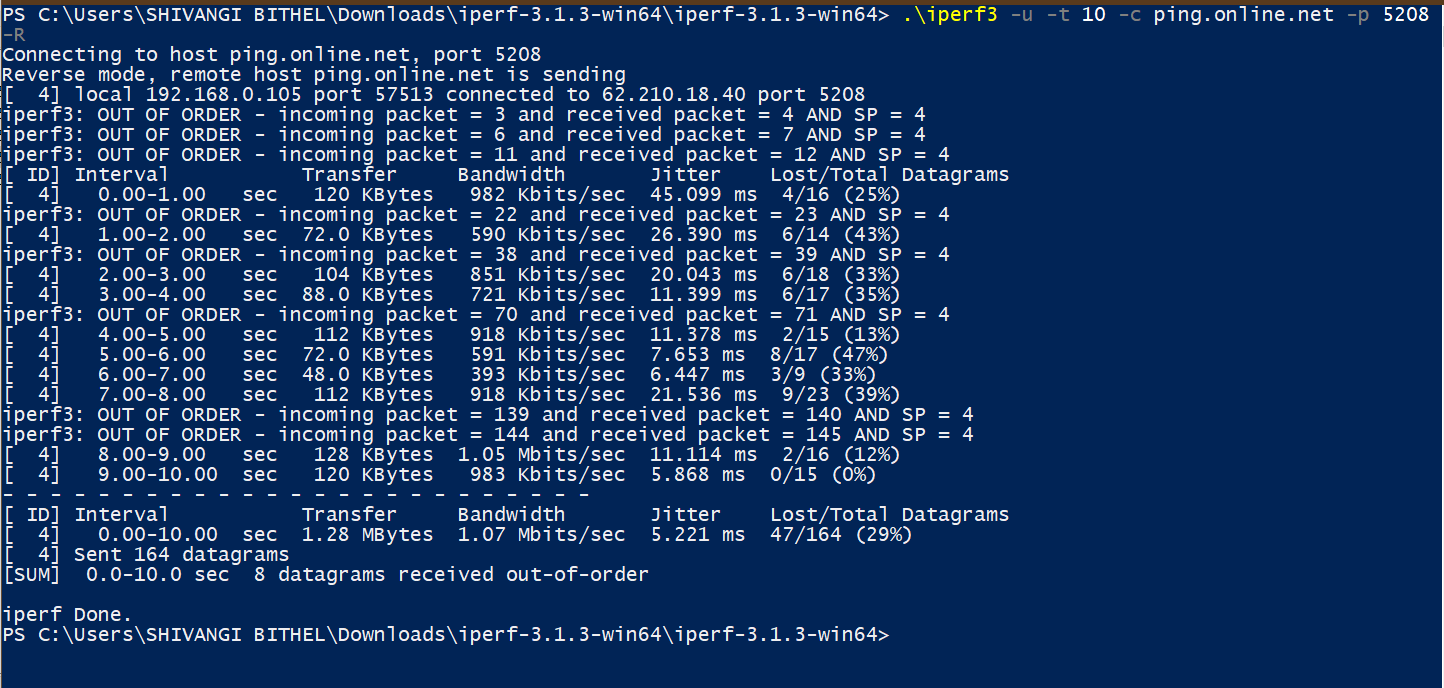
I certify that this assignment/report is my own work, based on my personal study and/or research and that I have acknowledged all material and sources used in its preparation, whether they be books, articles, reports, lecture notes, and any other kind of document, electronic or personal communication. I also certify that this assignment/report has not previously been submitted for assessment in any other course, except where specific permission has been granted from all course instructors involved, or at any other time in this course, and that I have not copied in part or whole or otherwise plagiarised the work of other students and/or persons. I pledge to uphold the principles of honesty and responsibility at CSE@IITH. In addition, I understand my responsibility to report honour violations by other students if I become aware of it.

**Shivangi Bithel**

**CS20MTECH12004**

**Task1**

**Task1: iperf3 to remote server in UDP**



1. How many UDP packets are exchanged in this communication between iperf3 client and remote server?

Ans. 127 packets in wireshark

164 sent in Iperf

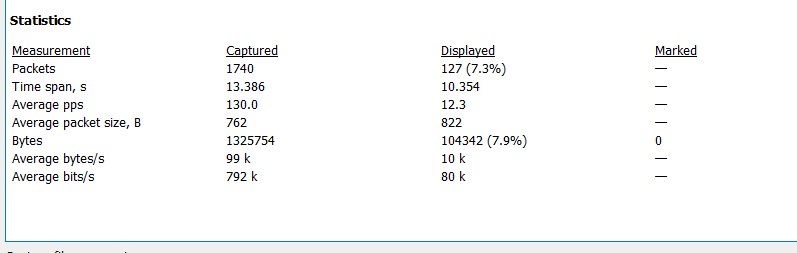
1. Who is sending bulk data to whom? What is the average size of the packet sent?

Ans. Source/Sender of bulk data: 62.210.18.40 (http://ping.online.net/)

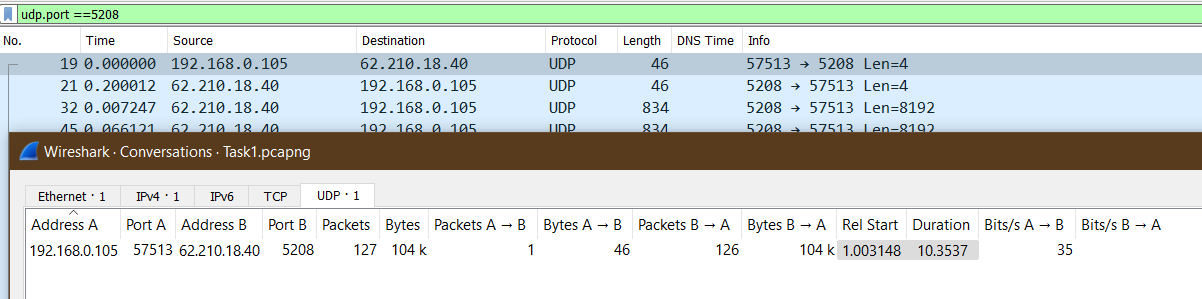
Receiver of bulk data: 192.168.0.105

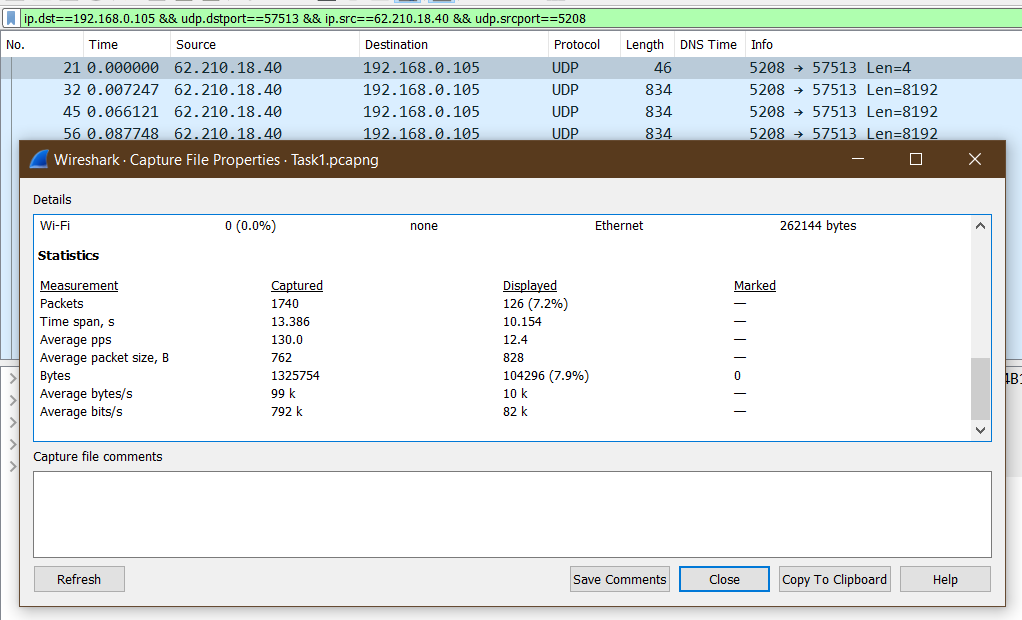
Average packet size = 822 B

This average size can be seen in statistics displayed by wireshark. As we are doing fragmentation of 8200 bytes UDP packet, therefore this value is displayed is less.



3. Calculate the throughput (bytes transferred per unit time) for this UDP conversation using UDP’s length field. Explain how you calculated this value using Wireshark capture in this experiment along with relevant screenshots. Verify your calculation with the one done by Wireshark using “Capture File properties” as well with the one displayed by iperf3 terminal. If you observe the major difference in your calculation and with the other two listed here, comment why and how?





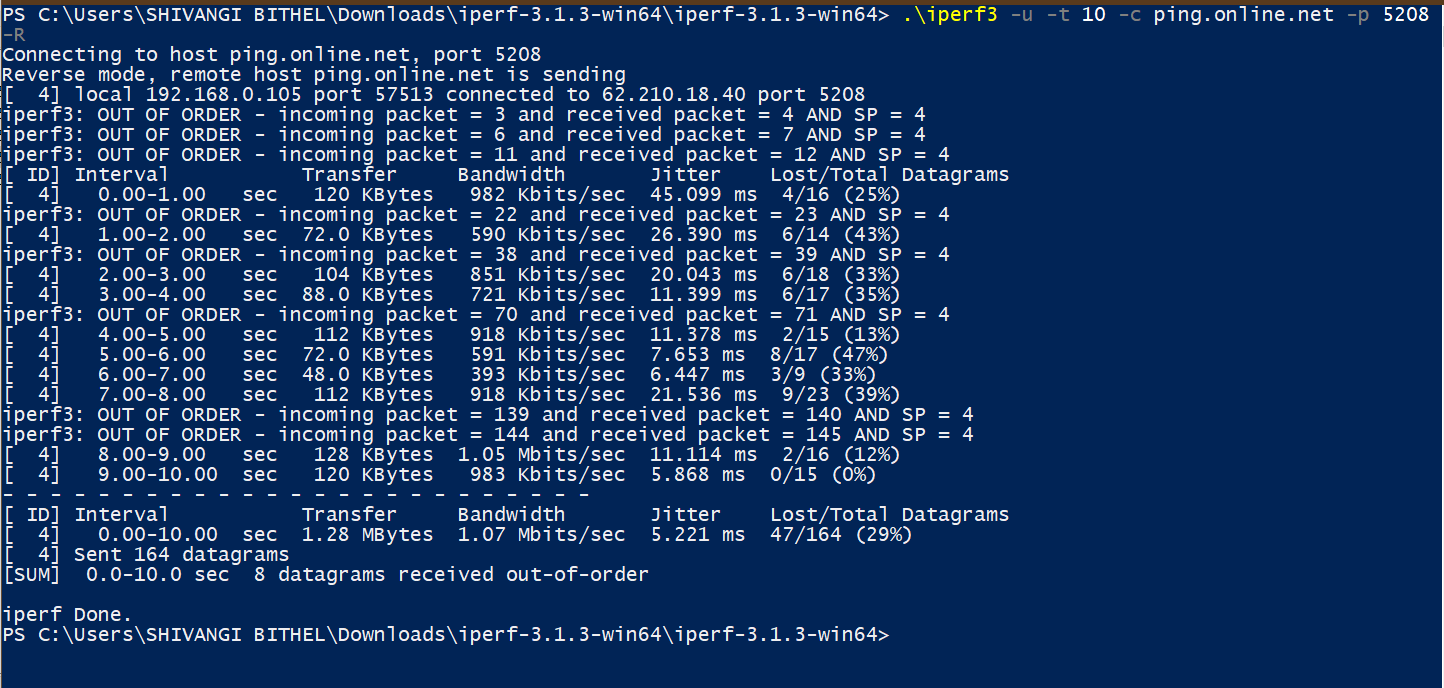
**Using Wireshark “Capture file properties”**

Average bytes/sec = 104296 bytes / 10.15 second = 10275.4 bytes/second

Average bits/sec = 82203.74 bits/sec =82.2K bits/s

Wireshark is using the value 834 bytes as the length of packet which is the length of last fragment of the complete packet of size 8200 bytes. Therefore the result calculated by wireshark has a major difference with what we can calculate using the sum of payload as that will consider packet length as 8192 bytes which is almost 10 times more than what wireshark is using.

**Iperf Terminal data:**



Interval=0.00-10.00 sec Transfer=1.28 MBytes

**Bandwidth= 1.07 Mbits/sec**  Jitter=156209273.097 ms

**Command used: .\iperf3 -u -t 10 -c ping.online.net -p 5208 –R**

-u : use UDP rather than TCP

-t: time in seconds to transmit for (default 10 secs) (here: 10 sec)

-c: run in client mode, connecting to <host> (here: 62.210.18.40 (http://ping.online.net/))

-p: server port to listen on/connect to (here: 5208)

-R: run in reverse mode (server sends, client receives)

**Using length field of UDP packet:**

Sum of length field in data part of UDP packet= (8192\*125) + (1\*4) = 1024004 bytes

= 8192032 bits

Time = 10.35 sec

**Throughput =8192032/10.3 =795342 bits/sec = 0.81 Mbits/sec**

Three different throughput values are:

Iperf terminal: 1.07 Mbits/sec

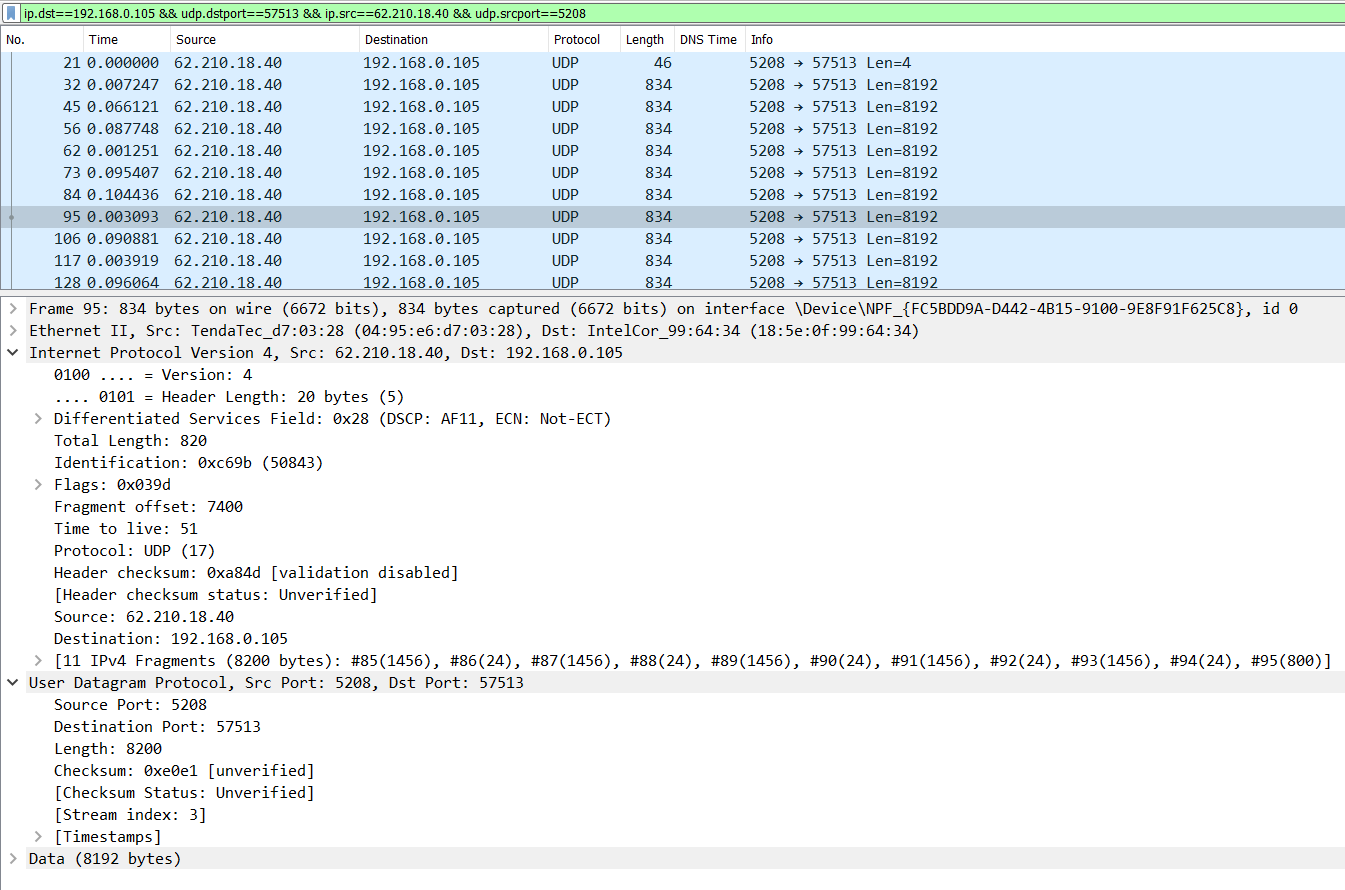
Adding UDP packet payload: 0.81 Mbits/sec

Wireshark value= 82.2 Kbits

Iperf capture the bandwidth using the actual payload and it is 1.07Mbits. Iperf is considering sending 164 packets but as there was packet loss of 47 packets, these packets were not being captured by wireshark. Wireshark shows only 126 packets from sender and this difference of 164-126 = 38 packets which if captured would have contributed around 0.24 Mbits/sec to what wireshark captured. Thus a difference in the value of iperf terminal and UDP payload value calculated is because of packet loss.

The difference of the other two calculated values to what Wireshark is calculating is because wireshark is using the value 834 bytes as the length of packet which is the length of last fragment of the complete packet of size 8200 bytes. Therefore the result calculated by wireshark has a major difference with what we can calculate using the sum of payload as that will consider packet length as 8192 bytes which is almost 10 times more than what wireshark is using. In the figure below we can see that in frame 95: 834 bytes are being reported and used as len filed by wireshark.

Whereas the UDP below shows data as 8192 bytes which I used to calculate throughput and also used by IPERF.



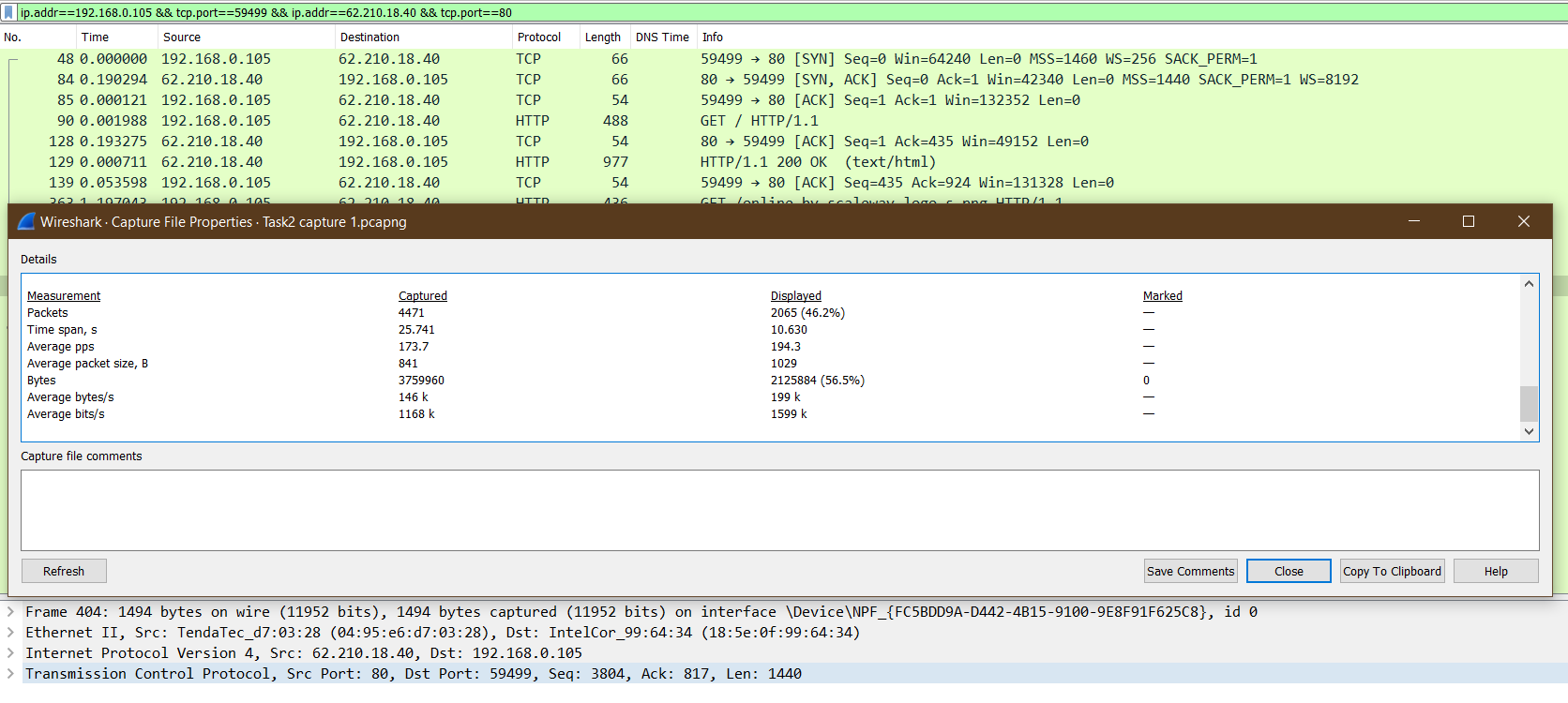
As can been seen in the above screenshot, the packet is being fragmented and thus the value of 834 bytes on wire gives an average packet size value of 822 B.

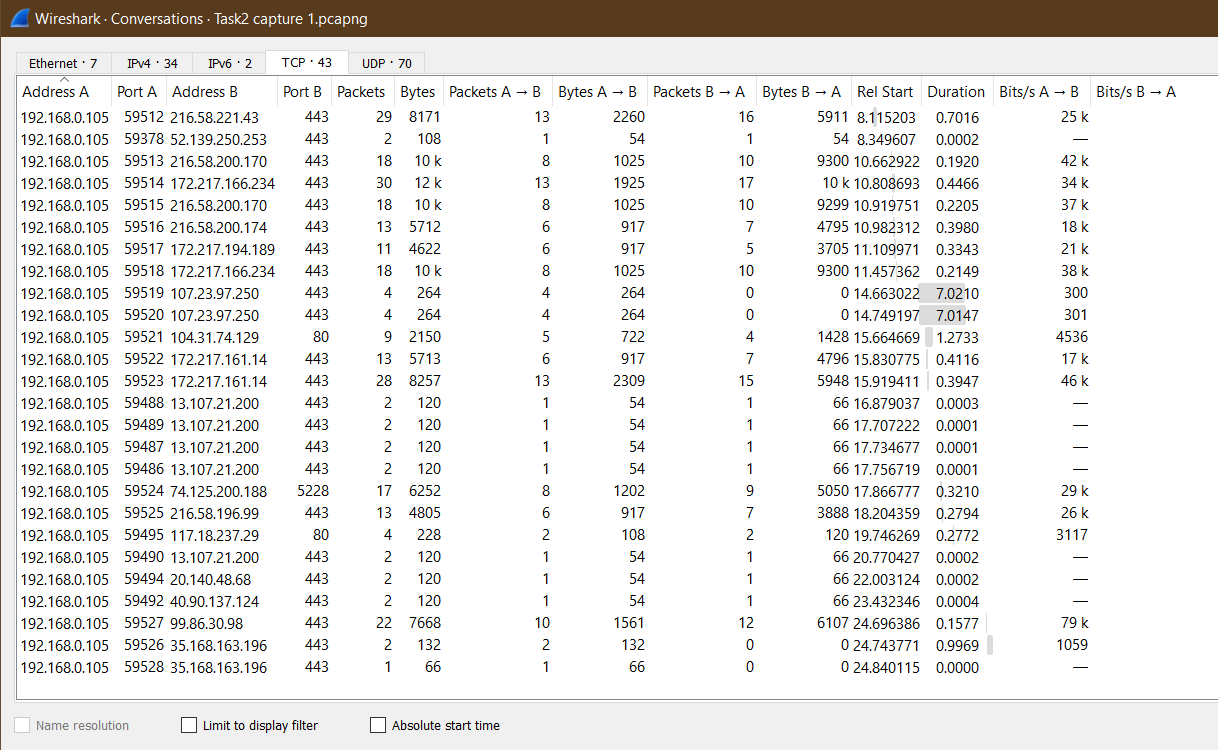
**Task 2**

**Capture 1:**

How many TCP packets are exchanged in this communication client and remote server?

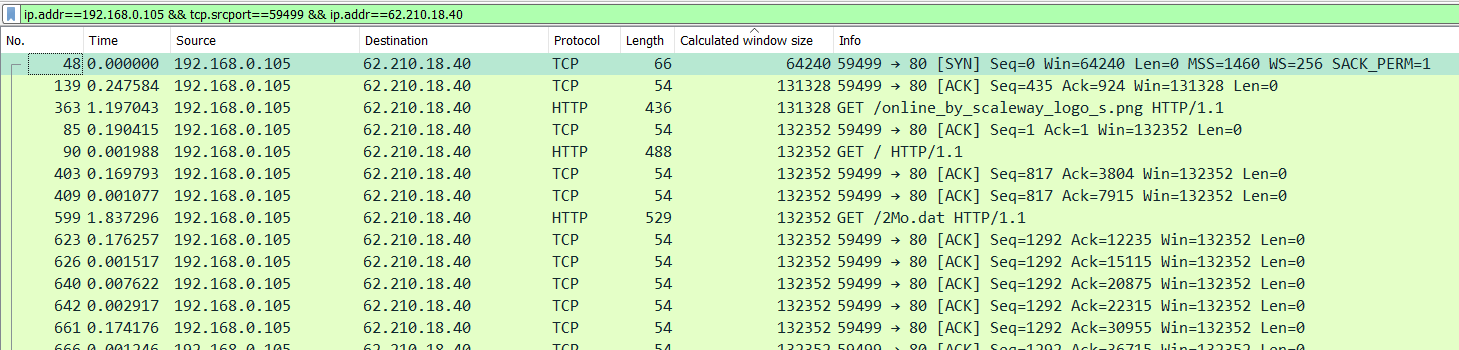
Ans 2065 Packets



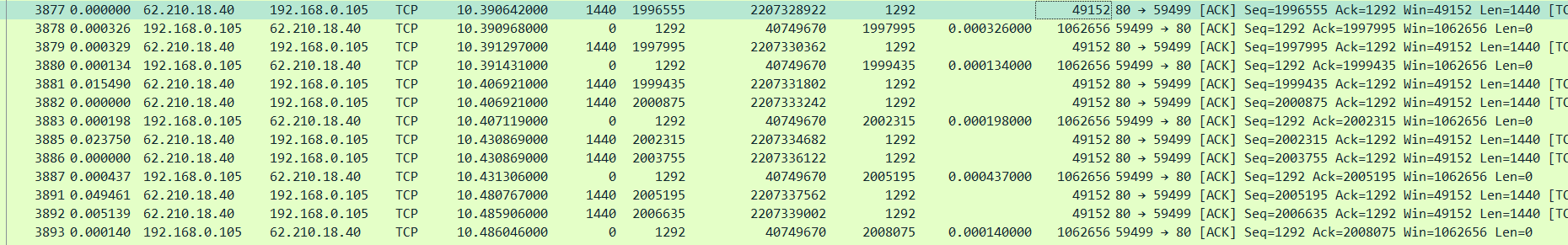


What is the minimum amount of available buffer space advertised at the client/receiver for the entire trace?

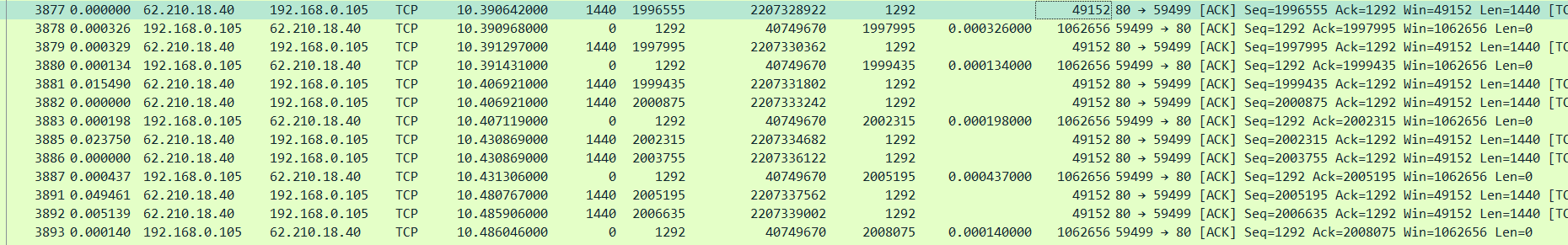
64240 is the minimum window size at receiver side.



Pick any 5 TCP segments from **server to client** which are **not** part of initial TCP connection establishment and final connection termination.



Make a table listing for each of these segments, the length of each of these TCP segments, the sequence number, time when the segment was sent, time when the respective ACK for each segment was received, length of the respective ACK segment. Place the screenshot of Wireshark of at least one such segment with respective ACK as a proof of observation and calculation. What is the maximum length out of all?



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Segment Number | Length | Sequence Number | Time of segment | ACK segment no | Time of ACK | Length of ACK |
| 3877 | 1440 B | 1996555 | 10.39064 s | 3878 | 10.39096 s | 0 B |
| 3879 | 1440 B | 1997995 | 10.39129 s | 3880 | 10.39143 s | 0 B |
| 3882 | 1440 B | 2000875 | 10.40692 s | 3883 | 10.40711 s | 0 B |
| 3886 | 1440 B | 2003755 | 10.43086 s | 3887 | 10.43130 s | 0 B |
| 3892 | 1440 B | 2006635 | 10.48590 s | 3893 | 10.48604 s | 0 B |

All packets have same length= 1440 bytes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Segment Number | Time of segment rcvd | Time of ACK sent | RTT | Estimated RTT |
| 3877 | 10.39064 s | 10.39096 s | 0.000326 s | 0.000326 |
| 3879 | 10.39129 s | 10.39143 s | 0.000134 s | 0.000302 |
| 3882 | 10.40692 s | 10.40711 s | 0.000198 s | 0.000289 |
| 3886 | 10.43086 s | 10.43130 s | 0.000437 s | 0.000307 |
| 3892 | 10.48590 s | 10.48604 s | 0.000140 s | 0.000286 |

**Calculating RTT between what client has sent as ACK and when the next packet was received by client from server as data**

**Estimated RTT calculation:**

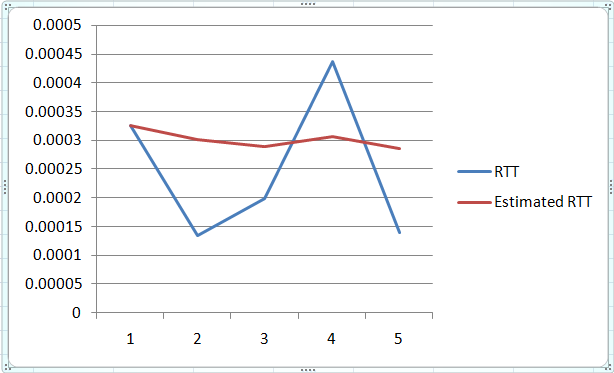
Segment 1: estimated rtt = actual rtt = 0.000326 s

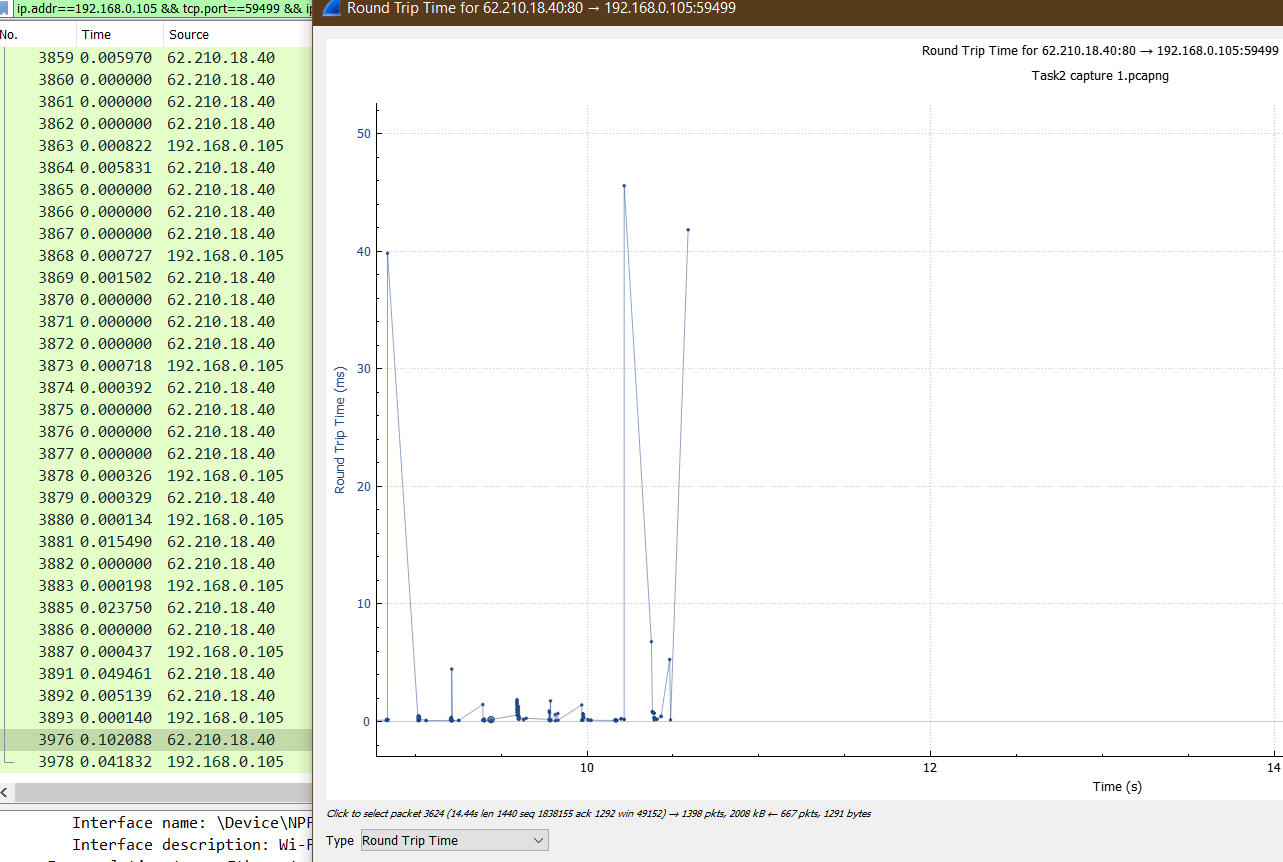
Segment 2 : estimated rtt = (1-0.125) \* 0.000326 + 0.125 \* 0.000134 =0.000302 s

Segment 3 : estimated rtt = (1-0.125) \* 0.000302 + 0.125 \* 0.000198 =0.000289 s

Segment 4 : estimated rtt = (1-0.125) \* 0.000289 + 0.125 \* 0.000437 =0.000307 s

Segment 5 : estimated rtt = (1-0.125) \* 0.000307 + 0.125 \* 0.000140 =0.000286 s

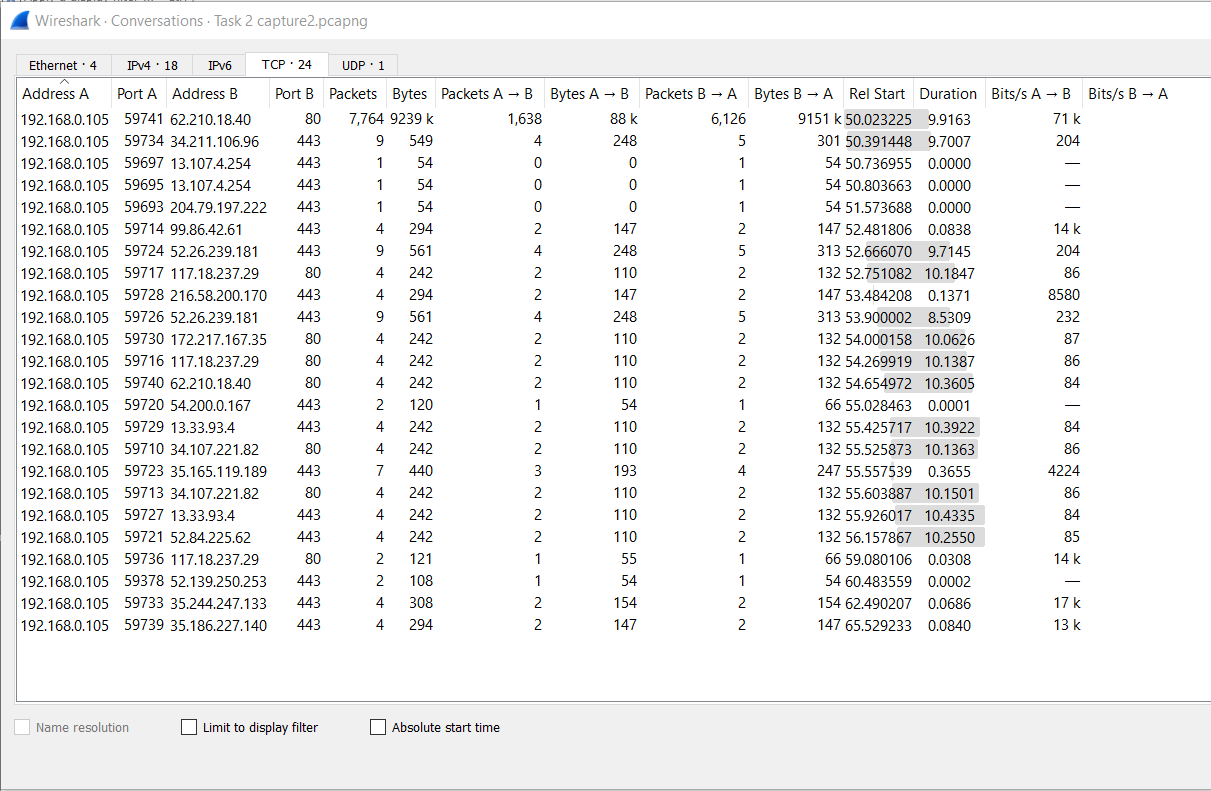


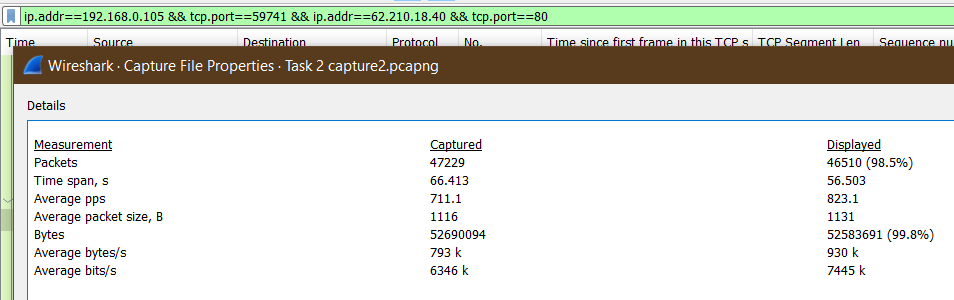


**Capture 2: for 50MB**

How many TCP packets are exchanged in this communication client and remote server?

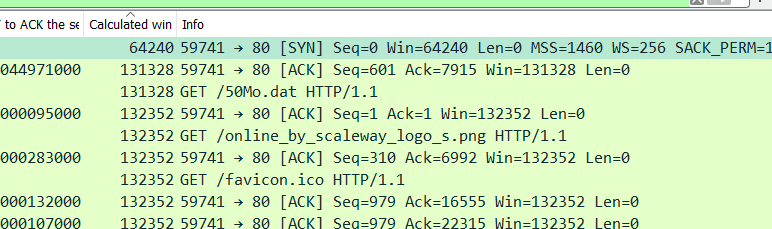
46510 packets



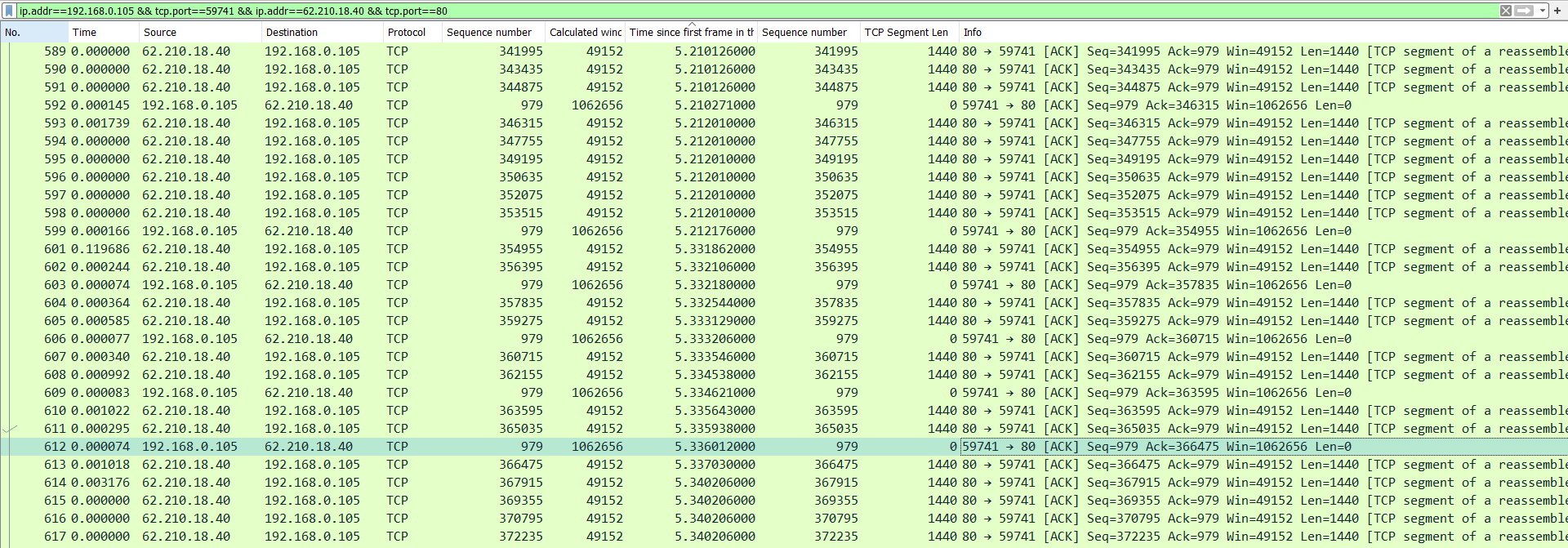


What is the minimum amount of available buffer space advertised at the client/receiver for the entire trace?

64240 is the minimum window size at receiver side.



**Calculating RTT between what client has sent as ACK and when the next packet was received by client from server as data**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Segment Number | Length | Sequence Number | Time of segment(s) | Time of ACK | Length of ACK |
| 601 | 1440 B | 354955 | 5.33186 | 5.33218 | 0 B |
| 604 | 1440 B | 357835 | 5.33254 | 5.33320 | 0 B |
| 607 | 1440 B | 360715 | 5.33354 | 5.33462 | 0 B |
| 610 | 1440 B | 363595 | 5.33564 | 5.33601 | 0 B |
| 613 | 1440 B | 366475 | 5.33703 | 5.34085 | 0 B |

All packets have same length= 1440 bytes

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Segment Number | Sequence Number | Time of segment(s) | Time of Ack sent to server | RTT (sec) | Estimated RTT(sec) |
| 601 | 354955 | 5.33186 | 5.21217 | 0.11967 | 0.11967 |
| 604 | 357835 | 5.33254 | 5.33218 | 0.00036 | 0.10475 |
| 607 | 360715 | 5.33354 | 5.33320 | 0.00034 | 0.09169 |
| 610 | 363595 | 5.33564 | 5.33462 | 0.00102 | 0.08035 |
| 613 | 366475 | 5.33703 | 5.33601 | 0.00102 | 0.07043 |

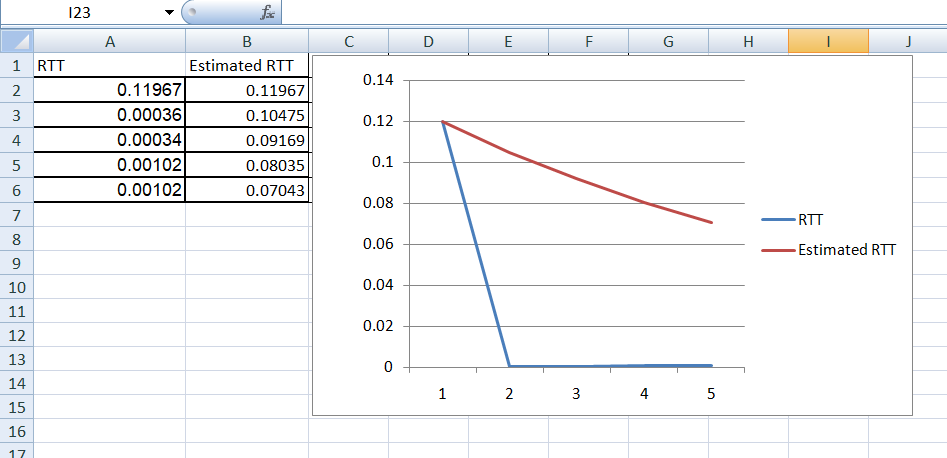
Segment 1: estimated rtt = actual rtt = 0.11967 s

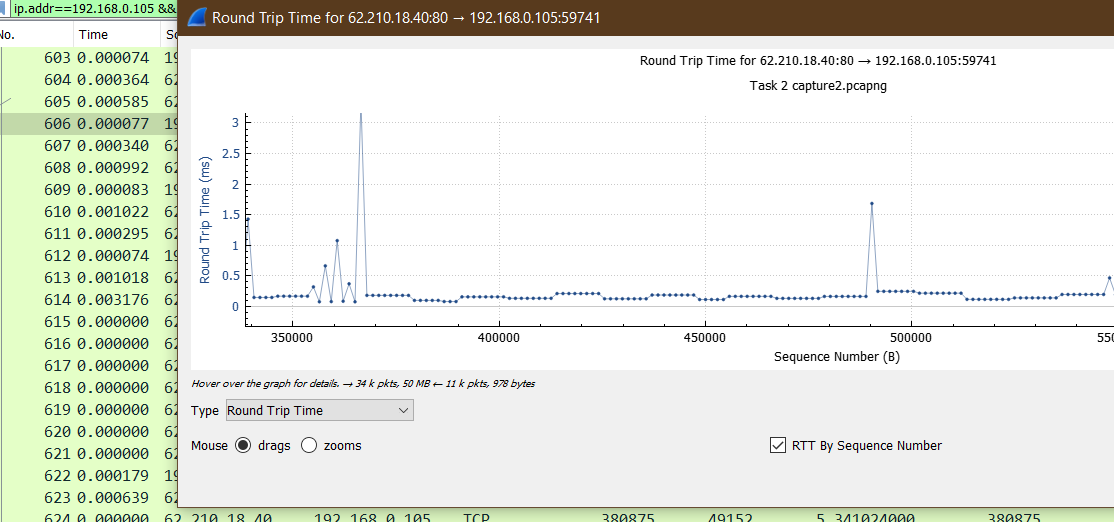
Segment 2 : estimated rtt = (1-0.125) \* 0.11967 + 0.125 \* 0.00036 =0.10475 s

Segment 3 : estimated rtt = (1-0.125) \* 0.10475 + 0.125 \* 0.00034 =0.09169 s

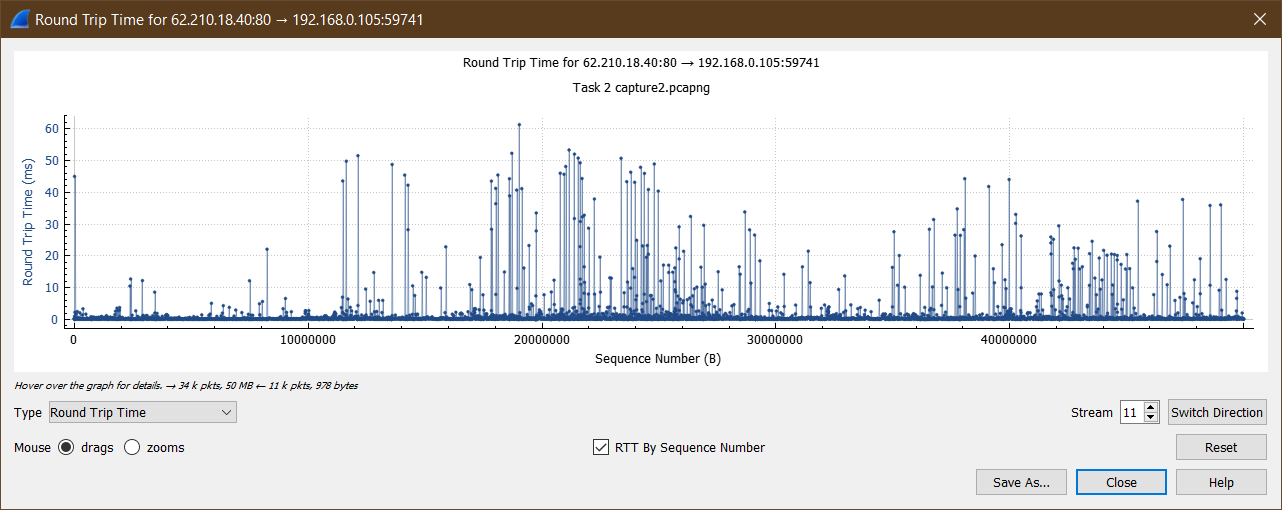
Segment 4 : estimated rtt = (1-0.125) \* 0.09169 + 0.125 \* 0.00102 =0.08035 s

Segment 5 : estimated rtt = (1-0.125) \* 0.08035 + 0.125 \* 0.00102 =0.07043 s





The RTT Graph from wireshark is given above.



Ques 3.4 Comment on your understanding of Estimated RTT calculation and plotted RTT

graphs.

EstimatedRTT = (1 − α) × EstimatedRTT + α × SampleRTT

where α = 0.125 (that is, 1/8) [RFC 6298]

Increase in the value of Estimated RTT represent that there is a congestion in the network. And similarly if the value of estimated RTT decreases, we can say that there is no congestion and the packets are arriving early.

We cannot always fix the value of RTT for every packet as value of actual RTT depend on congestion, so if fixed value is used, sometimes it will be too large to give excess tome out time and sometime too less to give time out and retransmission. So to give a factor that represent the network situation also to the this RTT ad TOT we are using a factor alpha with sample RTT and and 1- alpha with estimated RTT. This will smoothen the value and will also have an effect of network congestion on it as it considers the actual RTT value as sample RTT.

**From plotted graphs this can be seen that the RTT value is less than estimated RTT value. For 50 MB packet capture.**

Ques 4. Calculate the overall throughput (bytes transferred per unit time) for this TCP conversation using different fields of TCP from the captured file. Explain how you calculated this value using Wireshark capture in this experiment along with relevant screenshots. Verify your calculation with the one done by Wireshark using “Capture File properties”. If you observe the major difference in your calculation and one calculated by Wireshark, comment why and how?

**Capture 1**

The average throughput for this TCP connection is computed as the ratio between the total amount data and the total transmission time

Throughput = total bytes transferred / total time taken

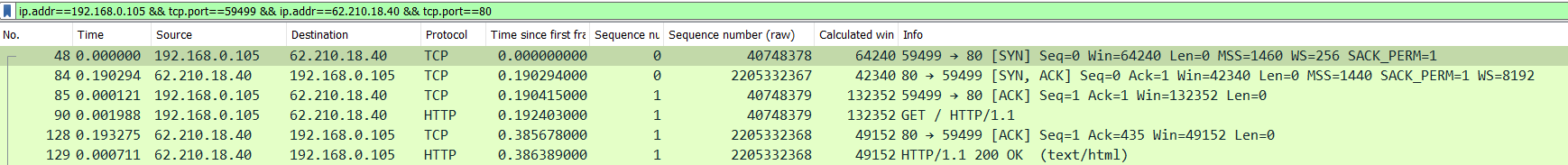
Total bytes transferred can be calculated using sequence number fields= initial seq no of first TCP segment – sequence no of last ack packet of this conversation

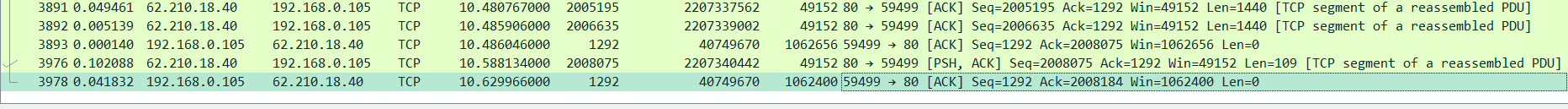
Total time taken= time of (first TCP Segment – last ack packet) of conversation

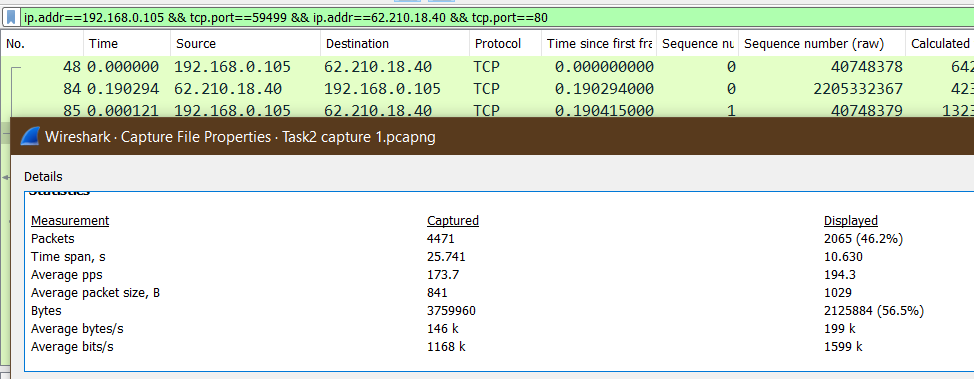
Throughput = (2008075 – 1) /( 10.629966000- 0.192403000) =192389.162106 bytes/sec

Throughput (by wireshark) = 199k bytes/sec

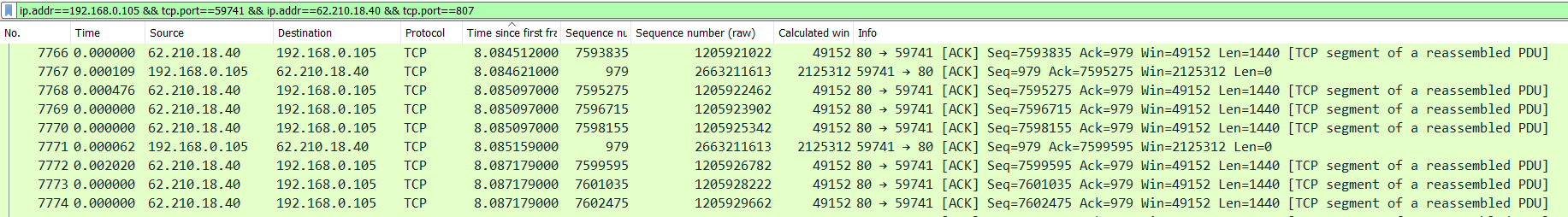
The difference in value is because wireshark takes complete data payload along with header of TCP, IP, Ethernet header. While what I calculated using sequence no. considered only data bytes not the header. Total time being same here but as there is a difference in the value of numerator, the ratio value changes. Thus wireshark is giving 199k bytes/s which is more than 192k bytes/s as I calculated using different fields of TCP.

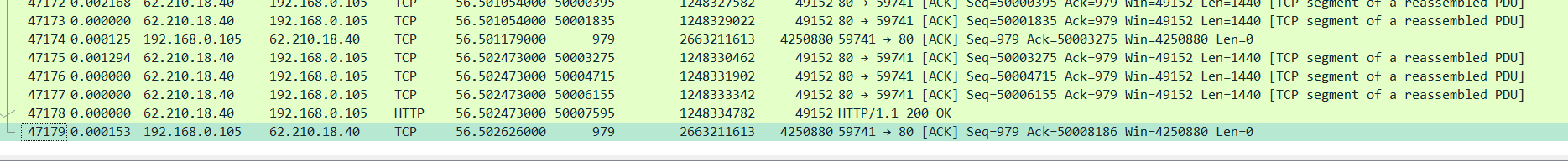






**Capture 2**





The average throughput for this TCP connection is computed as the ratio between the total amount data and the total transmission time

Throughput = total bytes transferred / total time taken

Total bytes transferred can be calculated using sequence number fields= initial seq no of first TCP segment – sequence no of last ack packet of this conversation

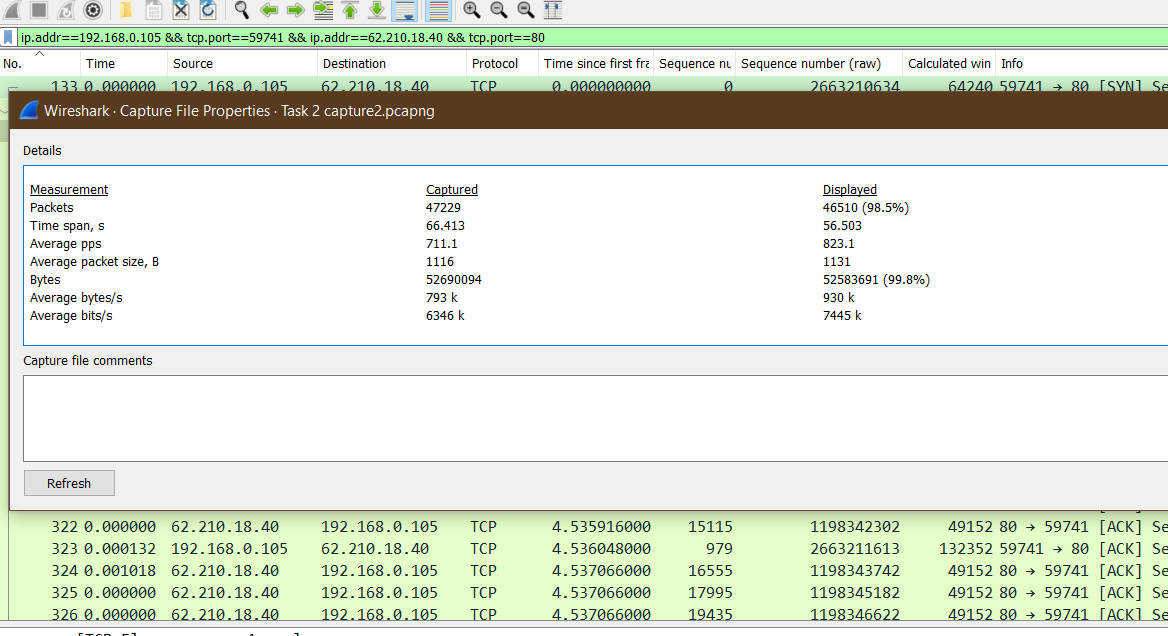
Total time taken= time of (first TCP Segment – last ack packet) of conversation

Throughput = (50008186 – 1) /( 56.502626000- 0.182719000) =887930.887386 bytes/sec

Throughput (by wireshark) = 930k bytes/sec

The difference in value is because wireshark takes complete data payload along with header of TCP, IP, Ethernet header. While what I calculated using sequence no. considered only data bytes not the header.

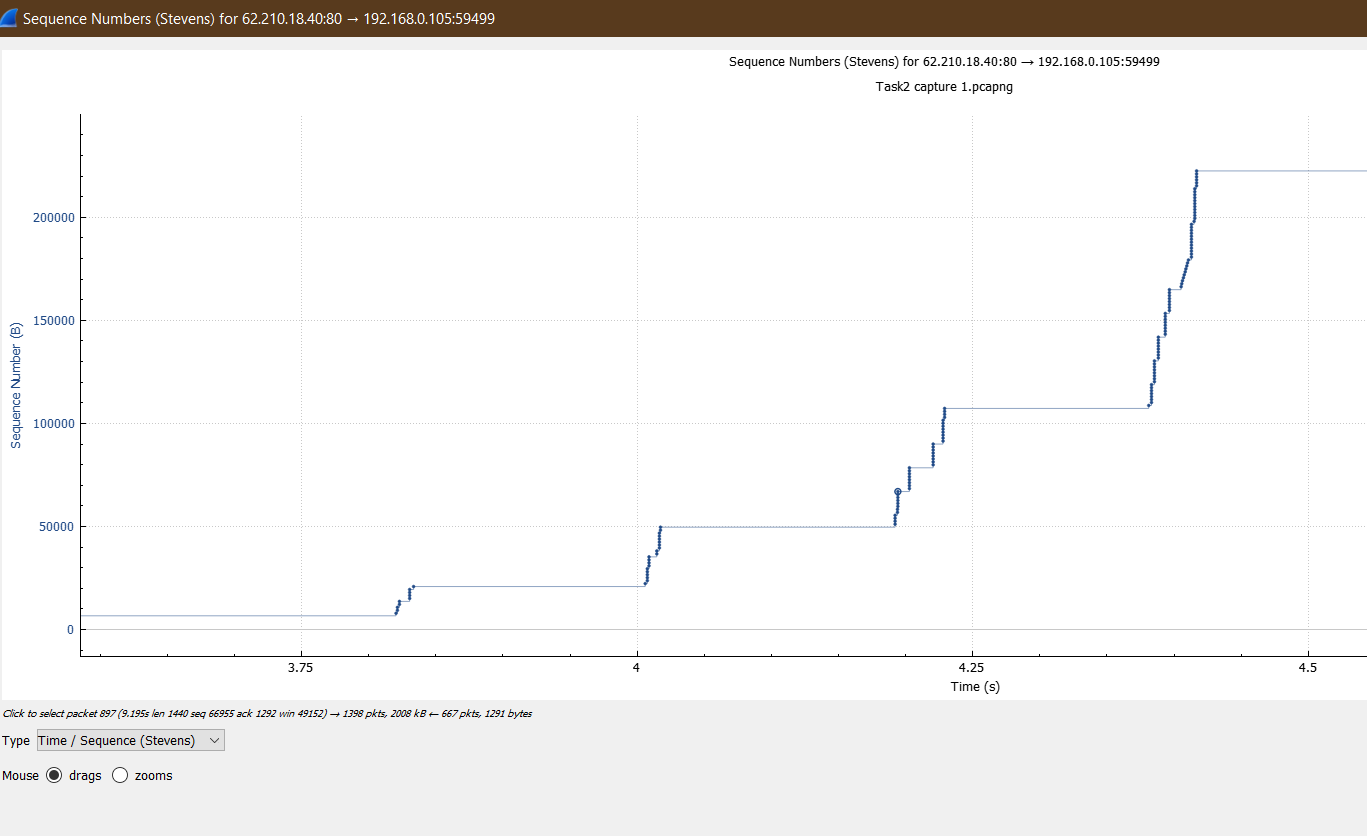
Total time being same here but as there is a difference in the value of numerator, the ratio value changes. Thus wireshark is giving 199k bytes/s which is more than 192k bytes/s as I calculated using different fields of TCP.

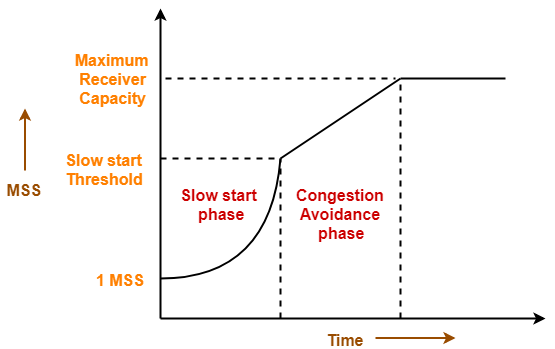


Ques 5. Using any active TCP segment (pick the packet of bulk data length, e.g: 5668) involved in the download process from server to client, capture the TCP’s functioning using the Time-Sequence-Graph (Stevens) plotting tool to view the sequence number versus time plot of segments being sent from the server to the client. Can you identify where TCP’s slow start phase begins and ends, and where congestion avoidance takes over? If not possible, why ?

**Capture 1**

The figure below shows the TCP slow start phase...we can count the number of packets from here its increasing like 10---20----40---80 and so on

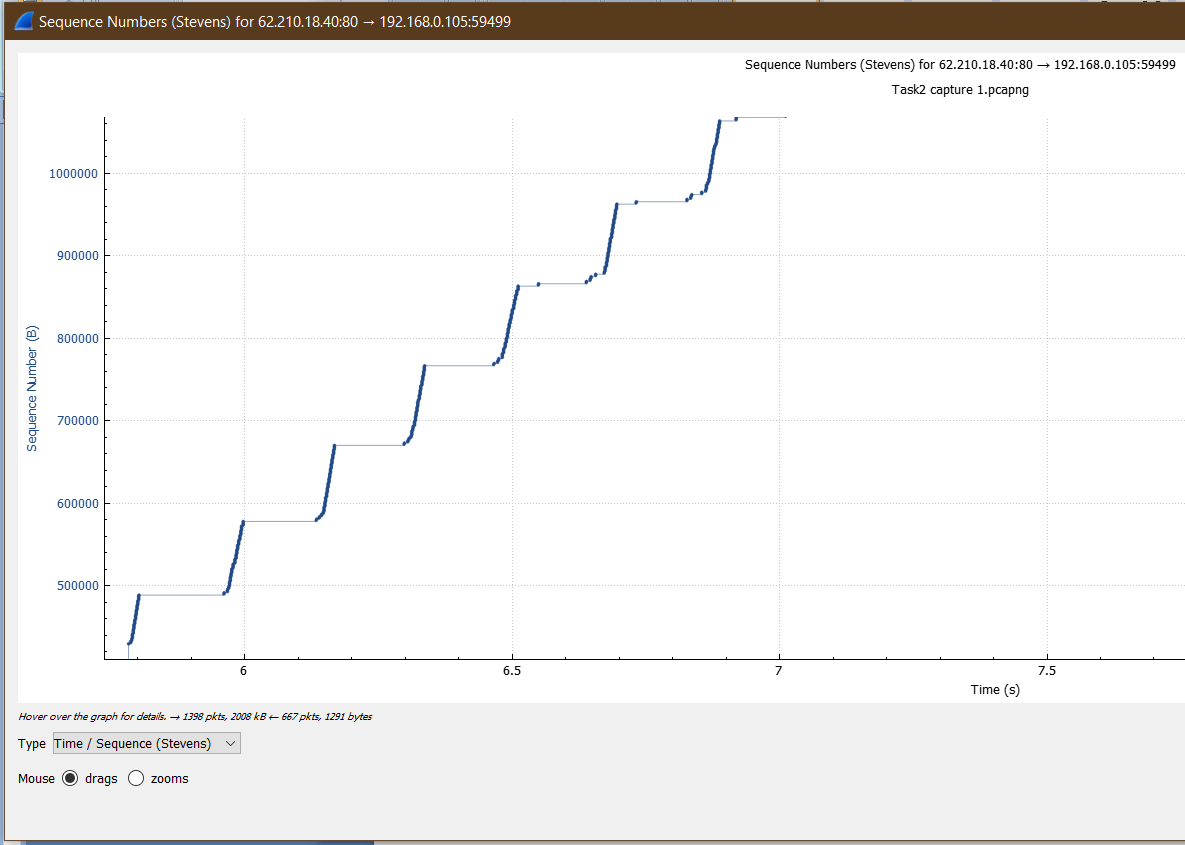


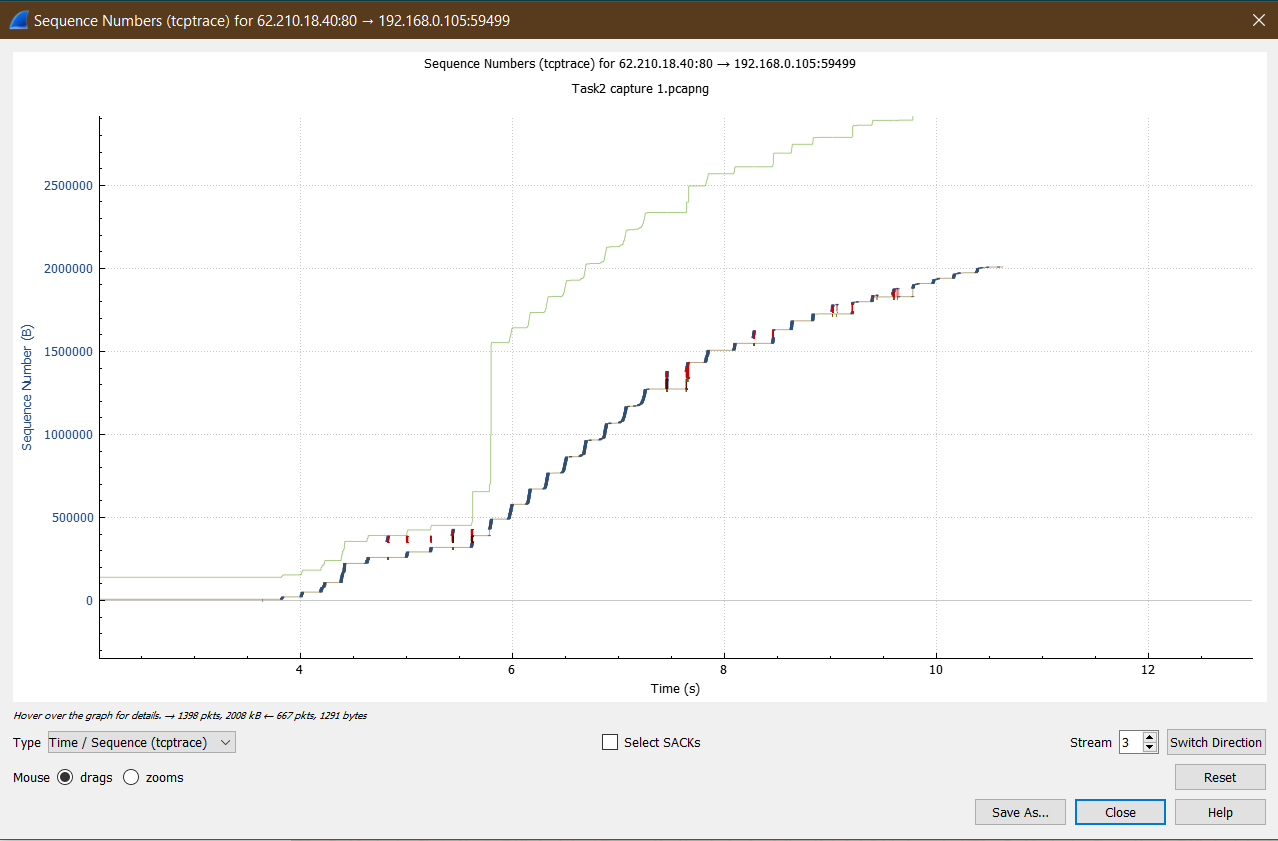


In congestion avoidance TCP linearly increases the window size by 1 after every ACK.

The below picture shows the linear increase in the packet numbers when counted manually.

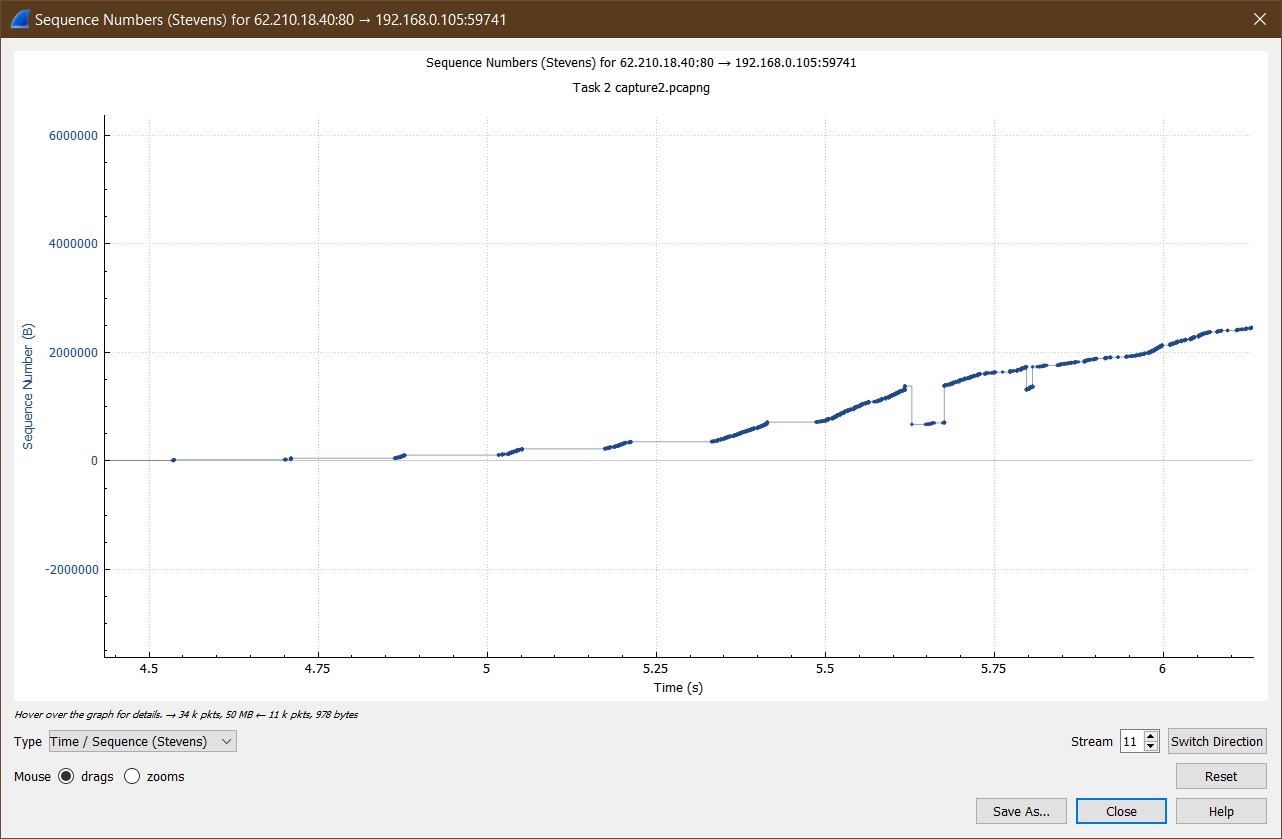
This is possibly the congestion avoidance phase.

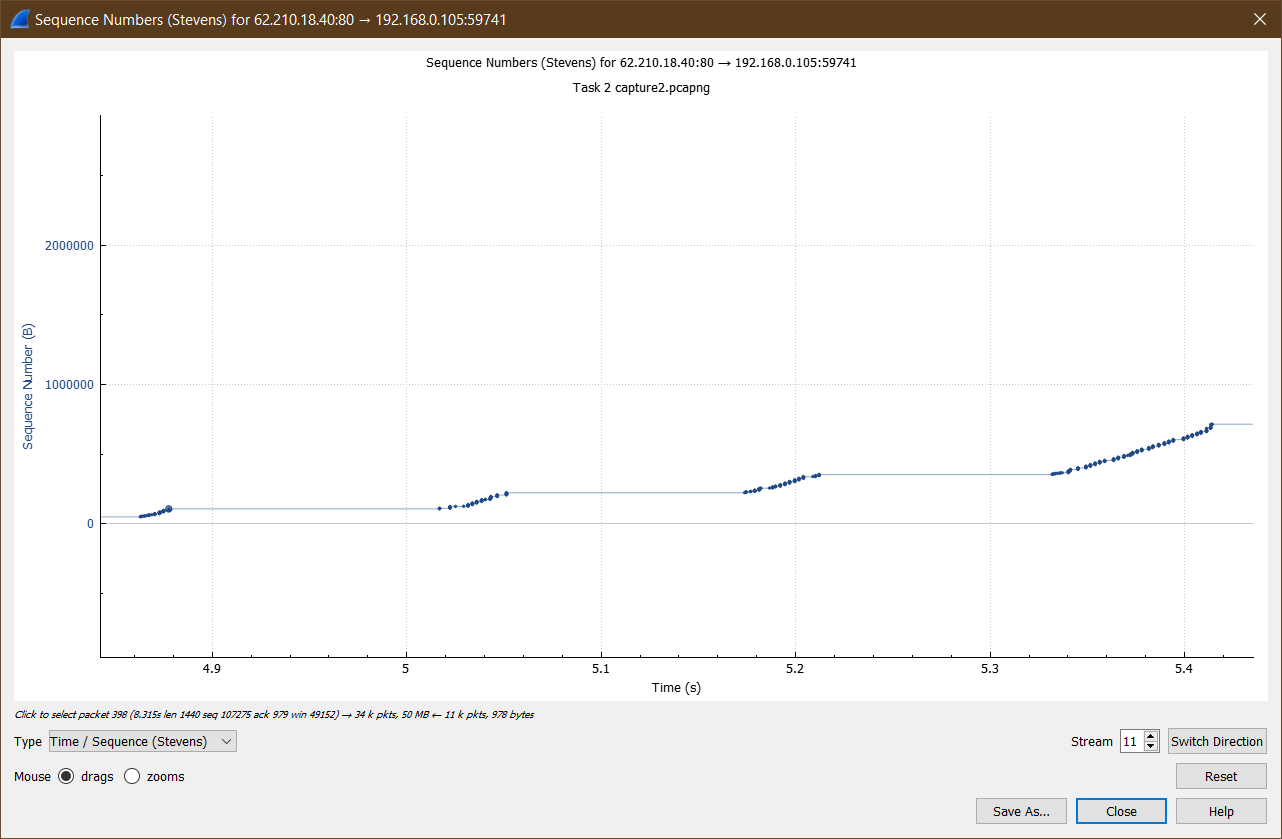




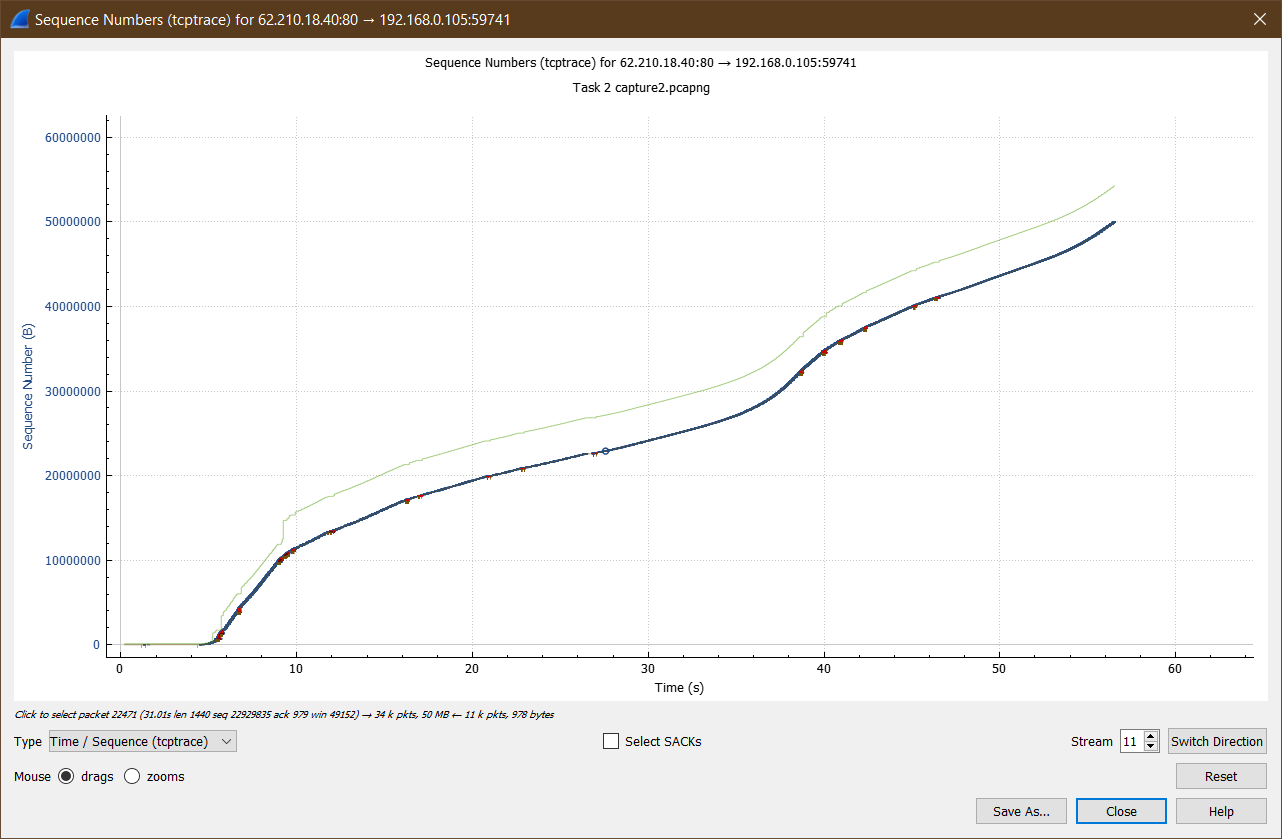
More can be seen from TCP trace graph for congestion.

**Capture 2**

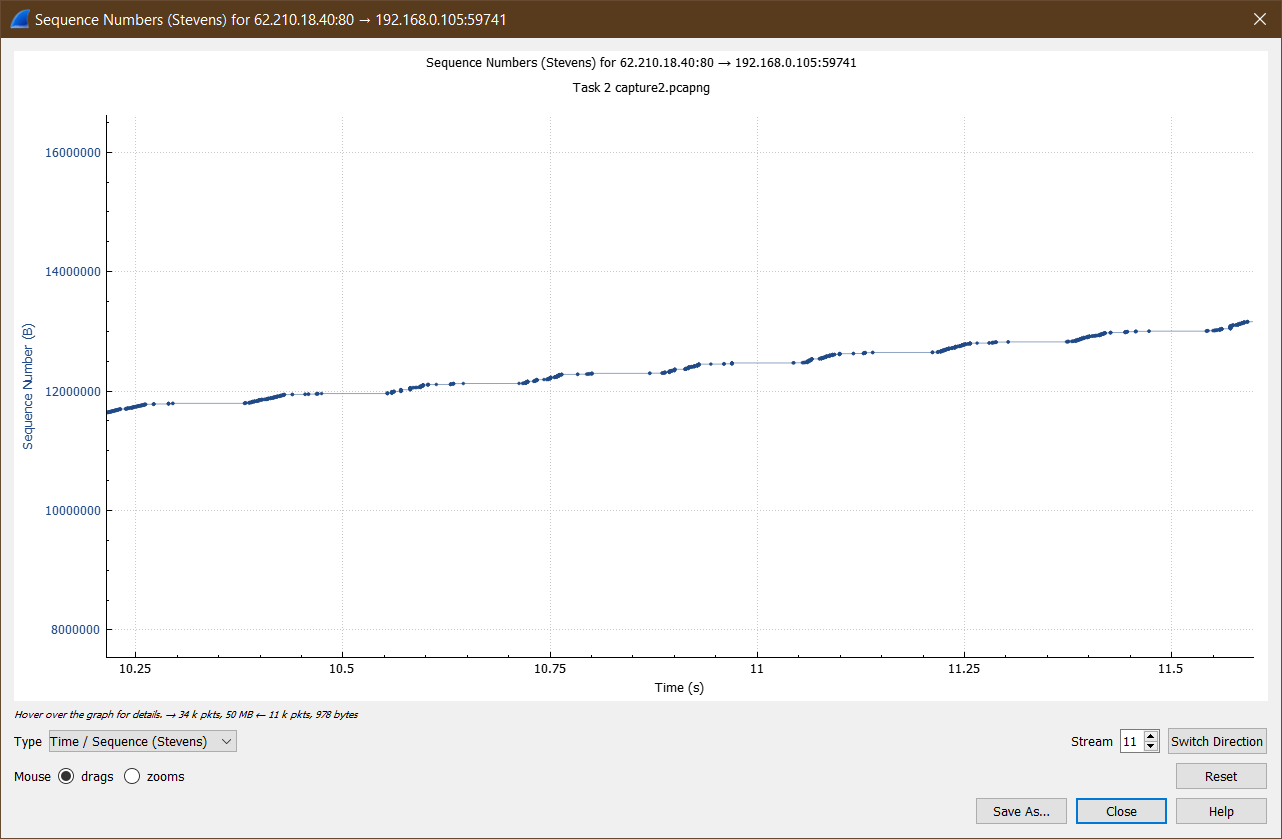




TCP slow start phase where there is a exponential increase in number of packets.

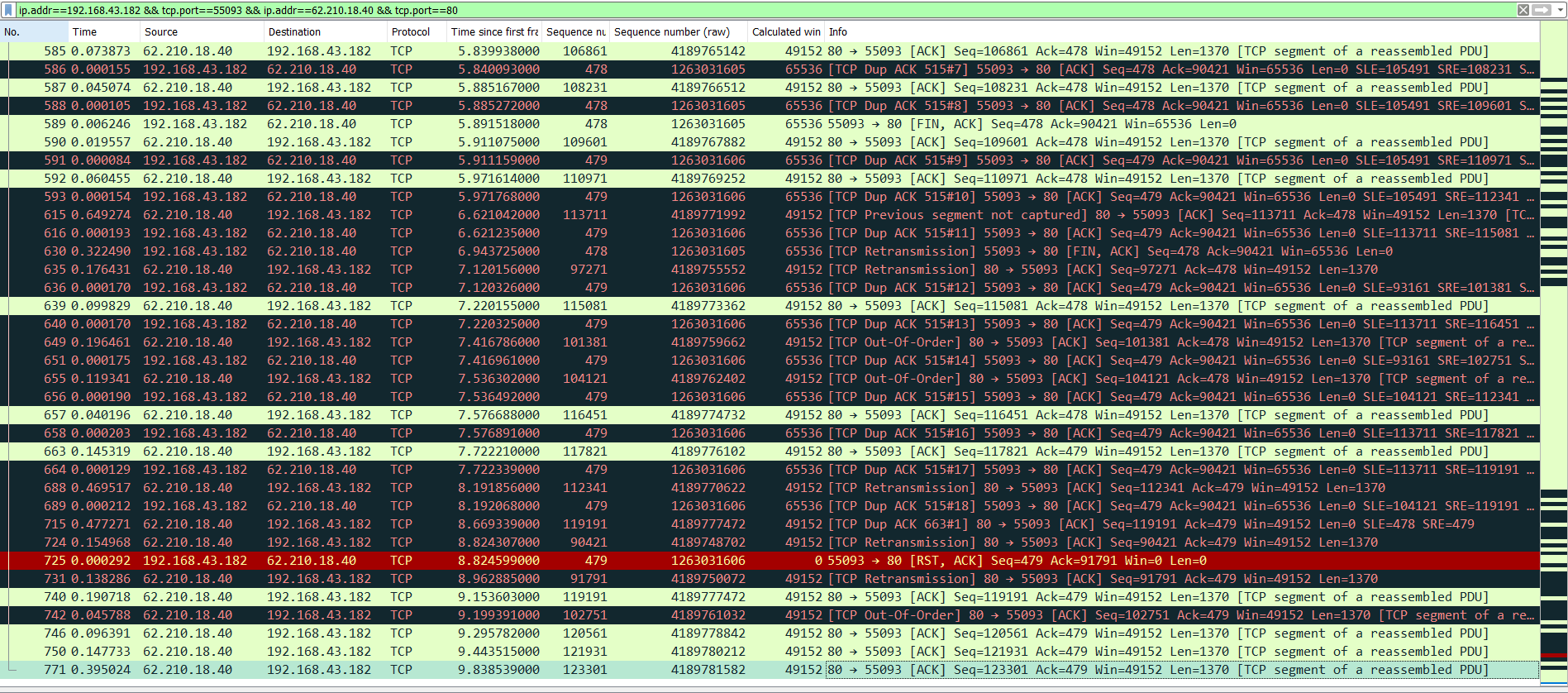


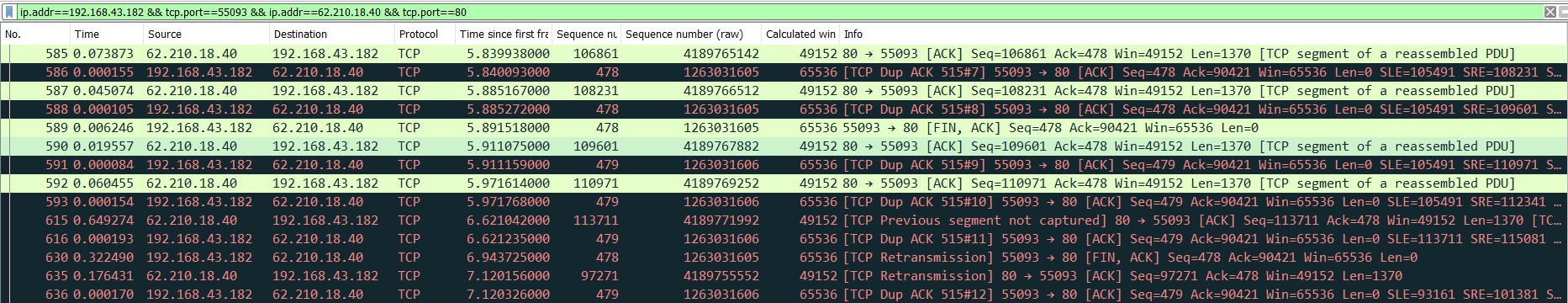
More can be seen from TCP trace graph for congestion.



Possible congestion avoidance phase where we can see linear increase in number of packets.

**Question. Only for #c above, observe and clearly explain with screenshots, how TCP connection gets terminated in this case, as well as which fields of TCP influence this, due to cancelling of the download in between.**





As we cancelled the download in between, we can see that my PC is sending a packet to server with FIN flag set and telling the server to close the connection.

Also the red coloured line has a TCP packet with reset flag being set.

An **RST** packet is sent either in the middle of the 3-way handshake when the server rejects the connection or is unavailable OR in the middle of data transfer when either the server or client rejects further communication bypassing the formal **4-way TCP** connection termination process.

Similarly we can see here that a TCP packet to reset the connect is being sent by my PC to server to immediately cancel/ terminate the connection as i don’t want to download more data.