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

```
PS C:\Users\SHIVANGI BITHEL\Downloads\iperf-3.1.3-win64\iperf-3.1.3-win64> .\iperf3 -u -t 10 -c
-R
Connecting to host ping.online.net, port 5208
Reverse mode, remote host ping.online.net is sending
[  4] local 192.168.0.105 port 57513 connected to 62.210.18.40 port 5208
iperf3: OUT OF ORDER - incoming packet = 3 and received packet = 4 AND SP = 4
iperf3: OUT OF ORDER - incoming packet = 6 and received packet = 7 AND SP = 4
iperf3: OUT OF ORDER - incoming packet = 11 and received packet = 12 AND SP = 4
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[  4]   0.00-1.00   sec   120 KBytes   982 Kbits/sec  45.099 ms  4/16 (25%)
iperf3: OUT OF ORDER - incoming packet = 22 and received packet = 23 AND SP = 4
[  4]   1.00-2.00   sec  72.0 KBytes   590 Kbits/sec  26.390 ms  6/14 (43%)
iperf3: OUT OF ORDER - incoming packet = 38 and received packet = 39 AND SP = 4
[  4]   2.00-3.00   sec   104 KBytes   851 Kbits/sec  20.043 ms  6/18 (33%)
[  4]   3.00-4.00   sec  88.0 KBytes   721 Kbits/sec  11.399 ms  6/17 (35%)
iperf3: OUT OF ORDER - incoming packet = 70 and received packet = 71 AND SP = 4
[  4]   4.00-5.00   sec   112 KBytes   918 Kbits/sec  11.378 ms  2/15 (13%)
[  4]   5.00-6.00   sec  72.0 KBytes   591 Kbits/sec  7.653 ms  8/17 (47%)
[  4]   6.00-7.00   sec  48.0 KBytes   393 Kbits/sec  6.447 ms  3/9 (33%)
[  4]   7.00-8.00   sec   112 KBytes   918 Kbits/sec  21.536 ms  9/23 (39%)
iperf3: OUT OF ORDER - incoming packet = 139 and received packet = 140 AND SP = 4
iperf3: OUT OF ORDER - incoming packet = 144 and received packet = 145 AND SP = 4
[  4]   8.00-9.00   sec   128 KBytes  1.05 Mbits/sec  11.114 ms  2/16 (12%)
[  4]   9.00-10.00  sec   120 KBytes   983 Kbits/sec  5.868 ms  0/15 (0%)
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[  4]   0.00-10.00  sec  1.28 MBytes  1.07 Mbits/sec  5.221 ms  47/164 (29%)
```

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

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

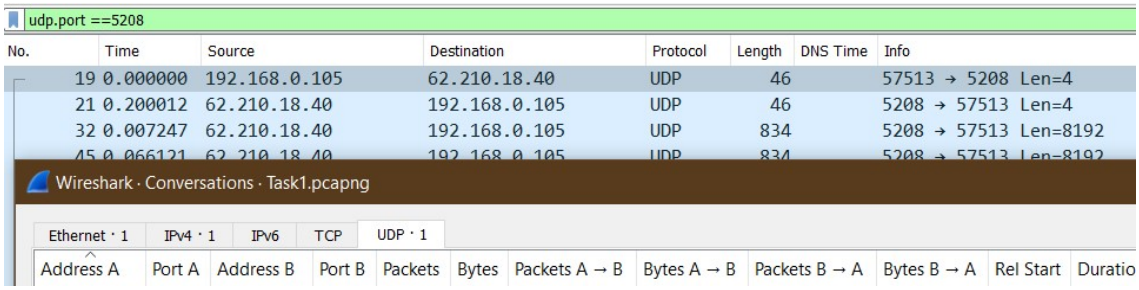
   Ans. Source/Sender of bulk data: 62.210.18.40 (http://ping.online.net/)
   Receiver of bulk data: 192.168.0.105
   Average packet size = 822 B

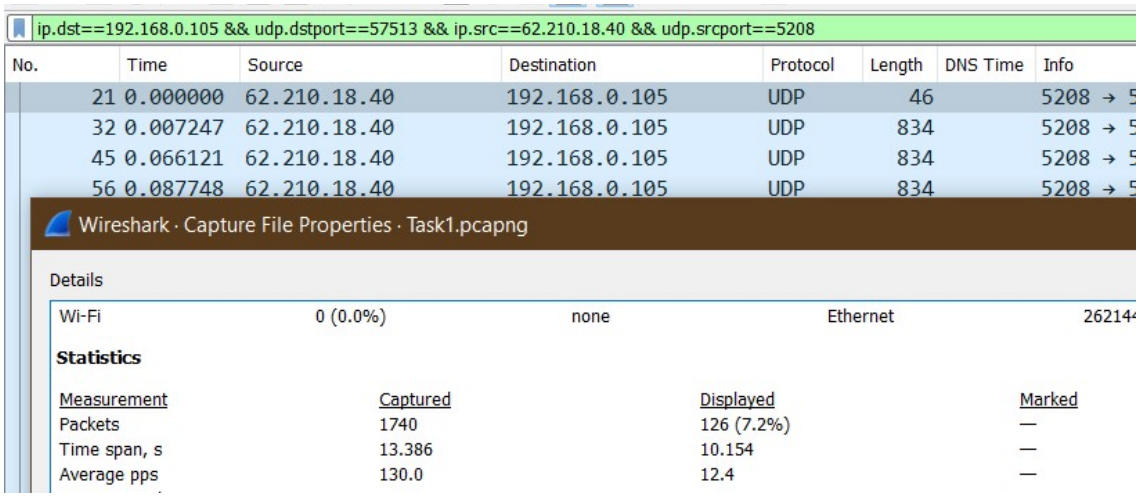
This average size can be seen in statistics displayed by wireshark. As we are doing fragmentation of 8200 bytes UDP packet, therefore this value is displayed is less.

Statistics

Measurement	Captured	Displayed
Packets	1740	127 (7.3%)
Time span, s	13.386	10.354
Average pps	130.0	12.3
Average packet size, B	762	822
Bytes	1325754	104342 (7.9%)
Average bytes/s	99 k	10 k

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"

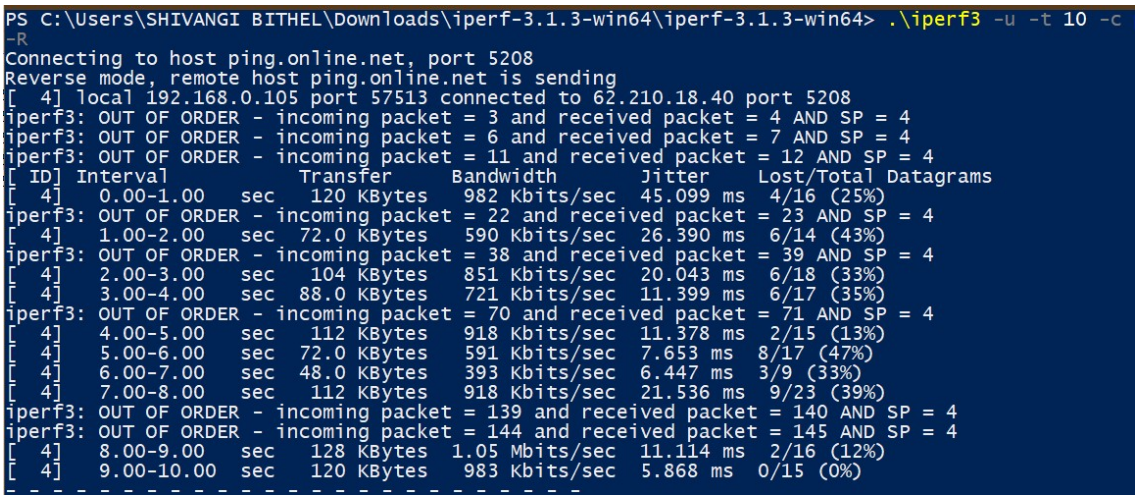
Average bytes/sec = 104296 bytes / 10.15 second = 10275.4 bytes/second

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:



```
PS C:\Users\SHIVANGI BITHEL\Downloads\iperf-3.1.3-win64\iperf-3.1.3-win64> .\iperf3 -u -t 10 -c
-R
Connecting to host ping.online.net, port 5208
Reverse mode, remote host ping.online.net is sending
[  4] local 192.168.0.105 port 57513 connected to 62.210.18.40 port 5208
iperf3: OUT OF ORDER - incoming packet = 3 and received packet = 4 AND SP = 4
iperf3: OUT OF ORDER - incoming packet = 6 and received packet = 7 AND SP = 4
iperf3: OUT OF ORDER - incoming packet = 11 and received packet = 12 AND SP = 4
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[  4]   0.00-1.00   sec   120 KBytes   982 Kbits/sec  45.099 ms  4/16 (25%)
iperf3: OUT OF ORDER - incoming packet = 22 and received packet = 23 AND SP = 4
[  4]   1.00-2.00   sec  72.0 KBytes   590 Kbits/sec  26.390 ms  6/14 (43%)
iperf3: OUT OF ORDER - incoming packet = 38 and received packet = 39 AND SP = 4
[  4]   2.00-3.00   sec   104 KBytes   851 Kbits/sec  20.043 ms  6/18 (33%)
[  4]   3.00-4.00   sec  88.0 KBytes   721 Kbits/sec  11.399 ms  6/17 (35%)
iperf3: OUT OF ORDER - incoming packet = 70 and received packet = 71 AND SP = 4
[  4]   4.00-5.00   sec   112 KBytes   918 Kbits/sec  11.378 ms  2/15 (13%)
[  4]   5.00-6.00   sec  72.0 KBytes   591 Kbits/sec  7.653 ms  8/17 (47%)
[  4]   6.00-7.00   sec  48.0 KBytes   393 Kbits/sec  6.447 ms  3/9 (33%)
[  4]   7.00-8.00   sec   112 KBytes   918 Kbits/sec  21.536 ms  9/23 (39%)
iperf3: OUT OF ORDER - incoming packet = 139 and received packet = 140 AND SP = 4
iperf3: OUT OF ORDER - incoming packet = 144 and received packet = 145 AND SP = 4
[  4]   8.00-9.00   sec   128 KBytes  1.05 Mbits/sec  11.114 ms  2/16 (12%)
[  4]   9.00-10.00  sec   120 KBytes   983 Kbits/sec  5.868 ms  0/15 (0%)
- - - - - - - - - - - - - - - - - - - - - - - - -
```

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

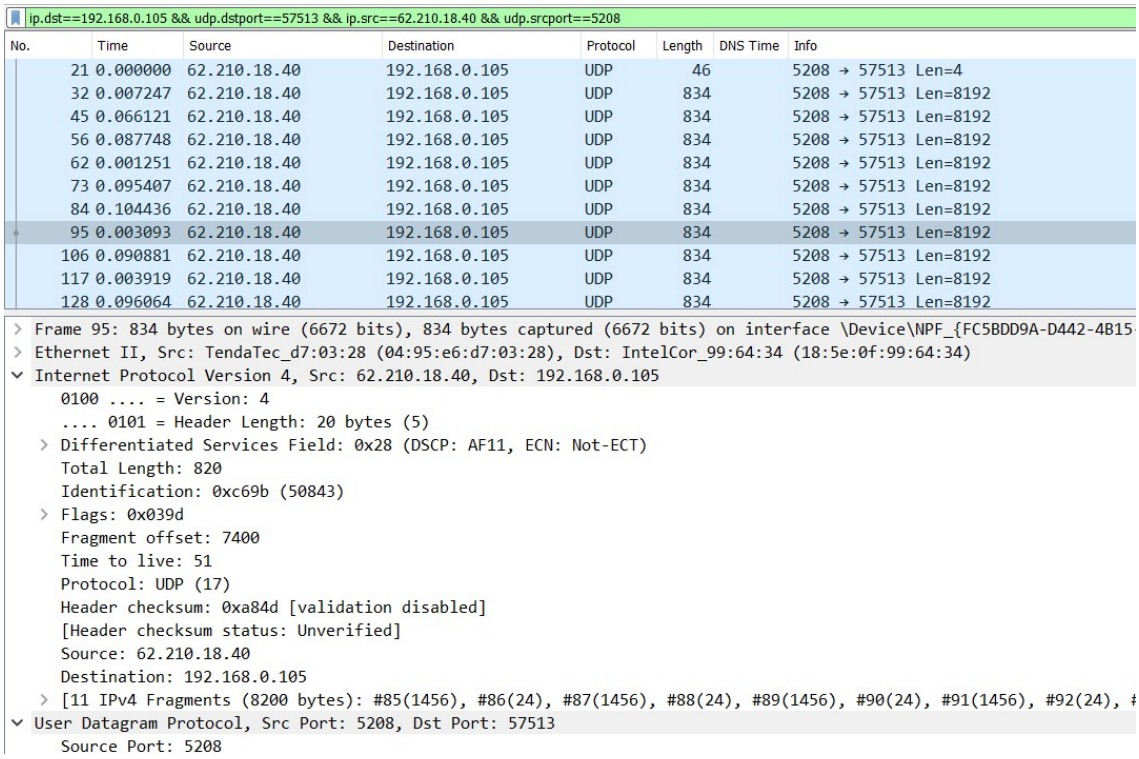
Adding UDP packet payload: 0.81 Mbits/sec

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.



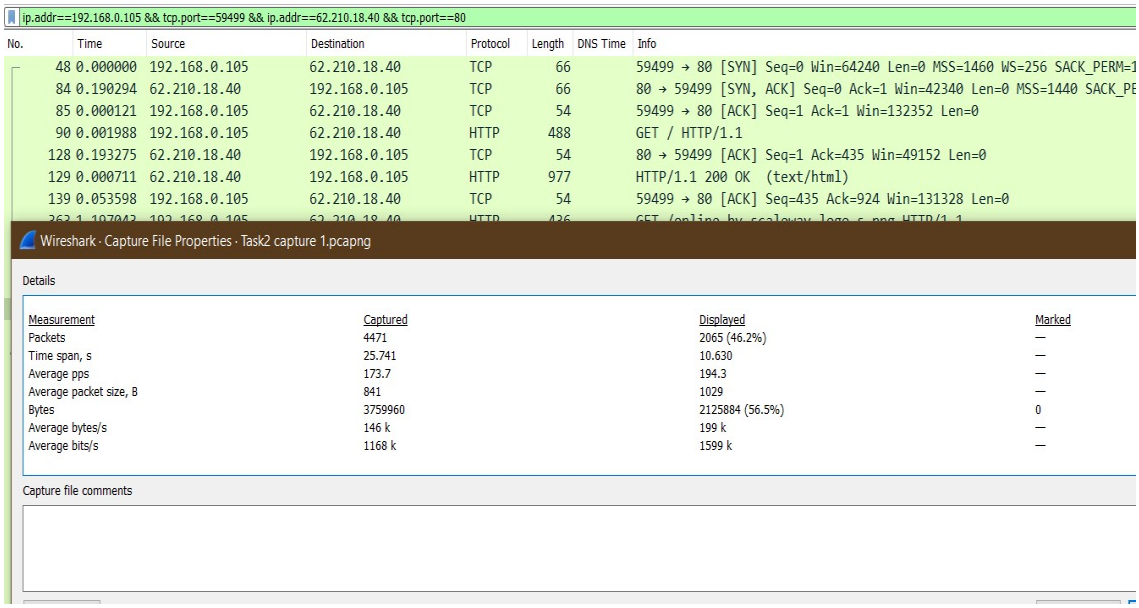
As can been seen in the above screenshot, the packet is being fragmented and thus the value of 834 bytes on wire gives an average packet size value of 822 B.

Task 2

Capture 1:

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

Ans 2065 Packets



ip.addr==192.168.0.105 && tcp.port==59499 && ip.addr==62.210.18.40 && tcp.port==80

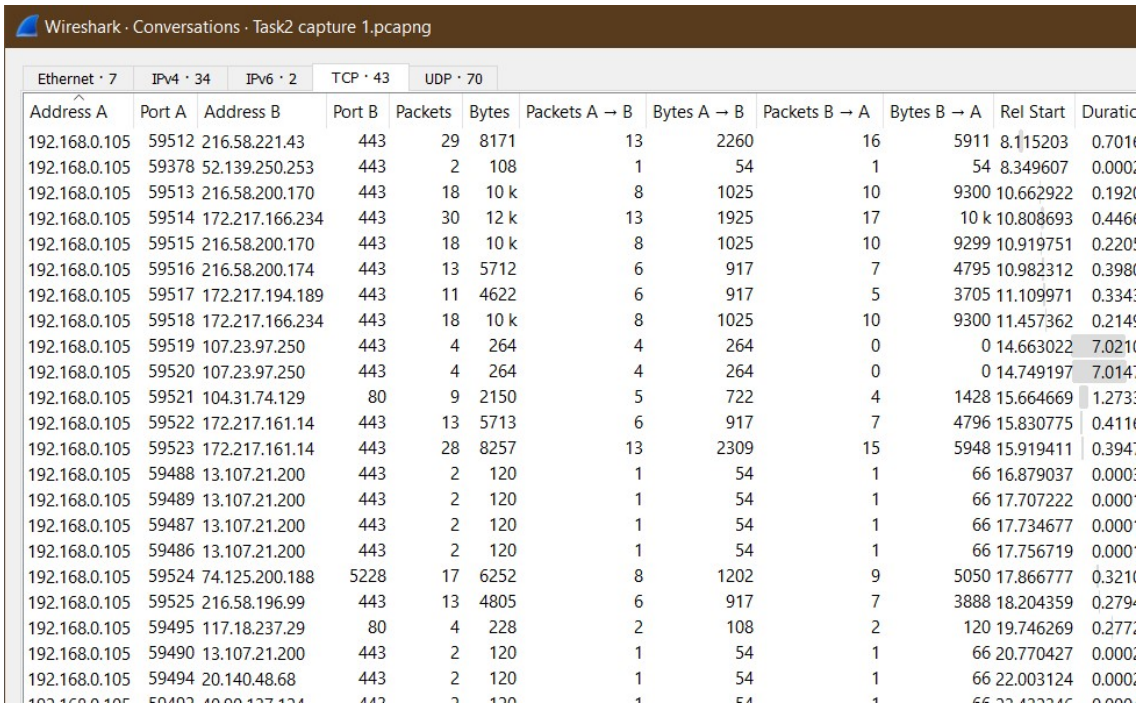
No.	Time	Source	Destination	Protocol	Length	DNS Time	Info
48	0.000000	192.168.0.105	62.210.18.40	TCP	66		59499 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
84	0.190294	62.210.18.40	192.168.0.105	TCP	66		80 → 59499 [SYN, ACK] Seq=0 Ack=1 Win=42340 Len=0 MSS=1440 SACK_PE
85	0.000121	192.168.0.105	62.210.18.40	TCP	54		59499 → 80 [ACK] Seq=1 Ack=1 Win=132352 Len=0
90	0.001988	192.168.0.105	62.210.18.40	HTTP	488		GET / HTTP/1.1
128	0.193275	62.210.18.40	192.168.0.105	TCP	54		80 → 59499 [ACK] Seq=1 Ack=435 Win=49152 Len=0
129	0.000711	62.210.18.40	192.168.0.105	HTTP	977		HTTP/1.1 200 OK (text/html)
139	0.053598	192.168.0.105	62.210.18.40	TCP	54		59499 → 80 [ACK] Seq=435 Ack=924 Win=131328 Len=0

Wireshark · Capture File Properties · Task2 capture 1.pcapng

Details

Measurement	Captured	Displayed	Marked
Packets	4471	2065 (46.2%)	—
Time span, s	25.741	10.630	—
Average pps	173.7	194.3	—
Average packet size, B	841	1029	—
Bytes	3759960	2125884 (56.5%)	0
Average bytes/s	146 k	199 k	—
Average bits/s	1168 k	1599 k	—

Capture file comments



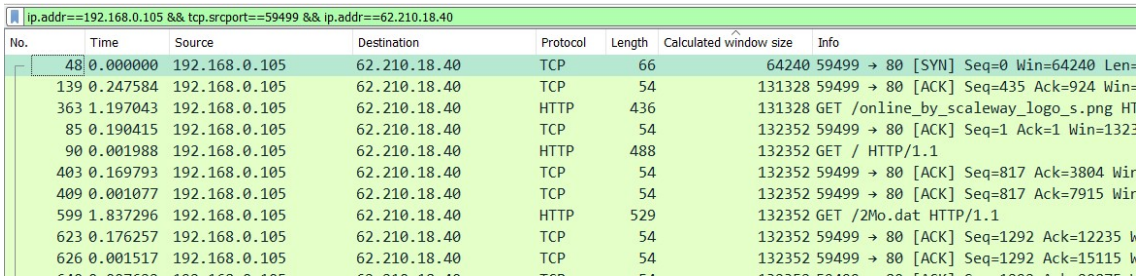
Wireshark · Conversations · Task2 capture 1.pcapng

Ethernet · 7 | IPv4 · 34 | IPv6 · 2 | TCP · 43 | UDP · 70

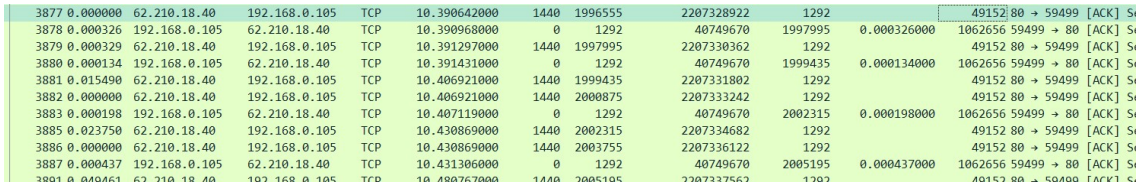
Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duratic
192.168.0.105	59512	216.58.221.43	443	29	8171	13	2260	16	5911	8.115203	0.701
192.168.0.105	59378	52.139.250.253	443	2	108	1	54	1	54	8.349607	0.000
192.168.0.105	59513	216.58.200.170	443	18	10 k	8	1025	10	9300	10.662922	0.192
192.168.0.105	59514	172.217.166.234	443	30	12 k	13	1925	17	10 k	10.808693	0.446
192.168.0.105	59515	216.58.200.170	443	18	10 k	8	1025	10	9299	10.919751	0.220
192.168.0.105	59516	216.58.200.174	443	13	5712	6	917	7	4795	10.982312	0.398
192.168.0.105	59517	172.217.194.189	443	11	4622	6	917	5	3705	11.109971	0.334
192.168.0.105	59518	172.217.166.234	443	18	10 k	8	1025	10	9300	11.457362	0.214
192.168.0.105	59519	107.23.97.250	443	4	264	4	264	0	0	14.663022	7.021
192.168.0.105	59520	107.23.97.250	443	4	264	4	264	0	0	14.749197	7.014
192.168.0.105	59521	104.31.74.129	80	9	2150	5	722	4	1428	15.664669	1.273
192.168.0.105	59522	172.217.161.14	443	13	5713	6	917	7	4796	15.830775	0.411
192.168.0.105	59523	172.217.161.14	443	28	8257	13	2309	15	5948	15.919411	0.394
192.168.0.105	59488	13.107.21.200	443	2	120	1	54	1	66	16.879037	0.000
192.168.0.105	59489	13.107.21.200	443	2	120	1	54	1	66	17.707222	0.000
192.168.0.105	59487	13.107.21.200	443	2	120	1	54	1	66	17.734677	0.000
192.168.0.105	59486	13.107.21.200	443	2	120	1	54	1	66	17.756719	0.000
192.168.0.105	59524	74.125.200.188	5228	17	6252	8	1202	9	5050	17.866777	0.321
192.168.0.105	59525	216.58.196.99	443	13	4805	6	917	7	3888	18.204359	0.279
192.168.0.105	59495	117.18.237.29	80	4	228	2	108	2	120	19.746269	0.277
192.168.0.105	59490	13.107.21.200	443	2	120	1	54	1	66	20.770427	0.000
192.168.0.105	59494	20.140.48.68	443	2	120	1	54	1	66	22.003124	0.000

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

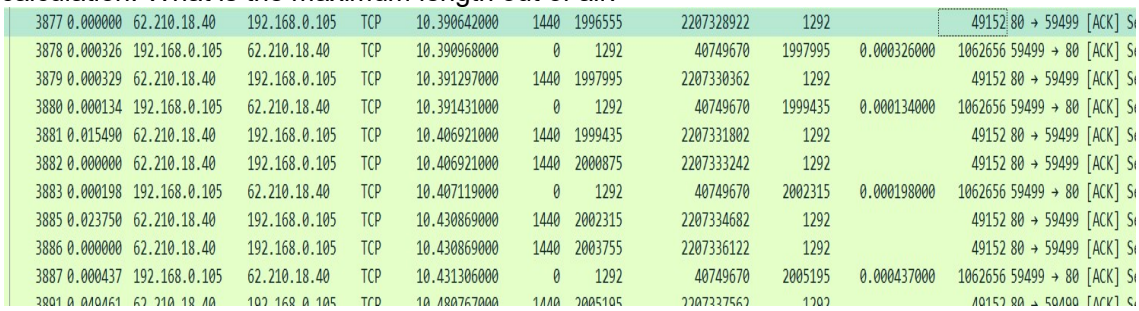
64240 is the minimum window size at receiver side.



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

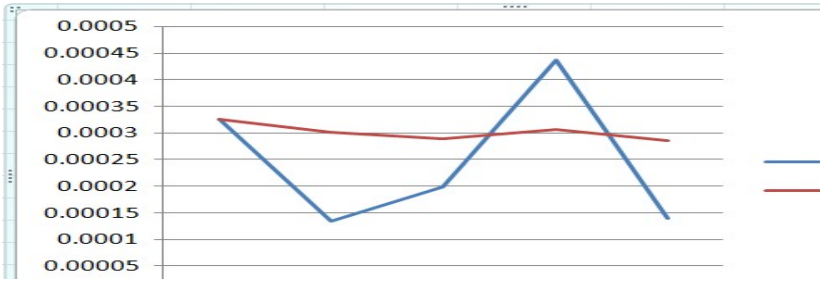
All packets have same length= 1440 bytes

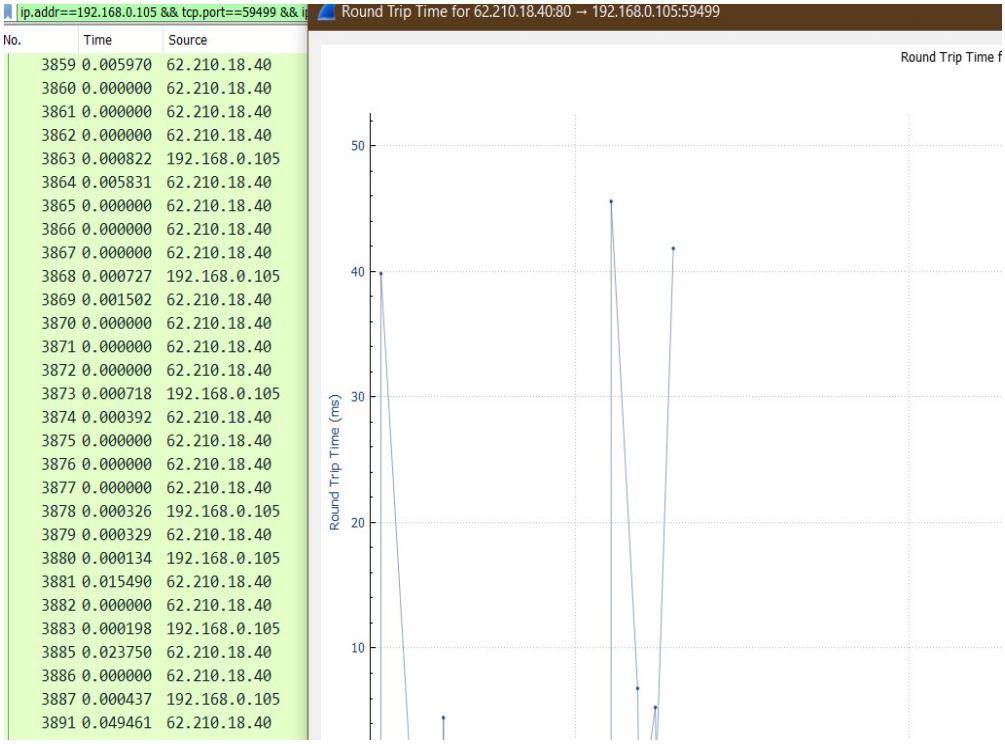
Segment Number	Time of segment rcvd	Time of ACK sent	RTT	Estimated RTT
3877	10.39064 s	10.39096 s	0.000326 s	0.000326
3879	10.39129 s	10.39143 s	0.000134 s	0.000302
3882	10.40692 s	10.40711 s	0.000198 s	0.000289
3886	10.43086 s	10.43130 s	0.000437 s	0.000307
3892	10.48590 s	10.48604 s	0.000140 s	0.000286

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





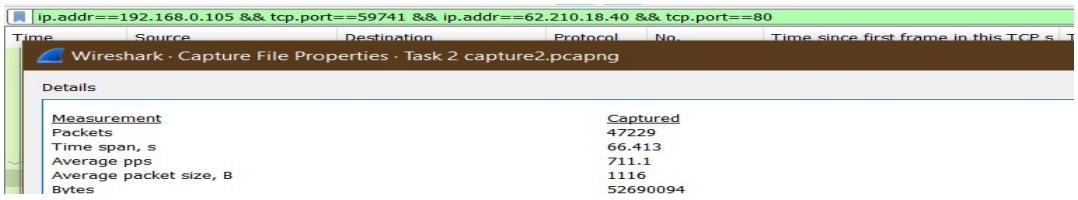
Capture 2: for 50MB

How many TCP packets are exchanged in this communication client and remote server?
46510 packets

Wireshark · Conversations · Task 2 capture2.pcapng

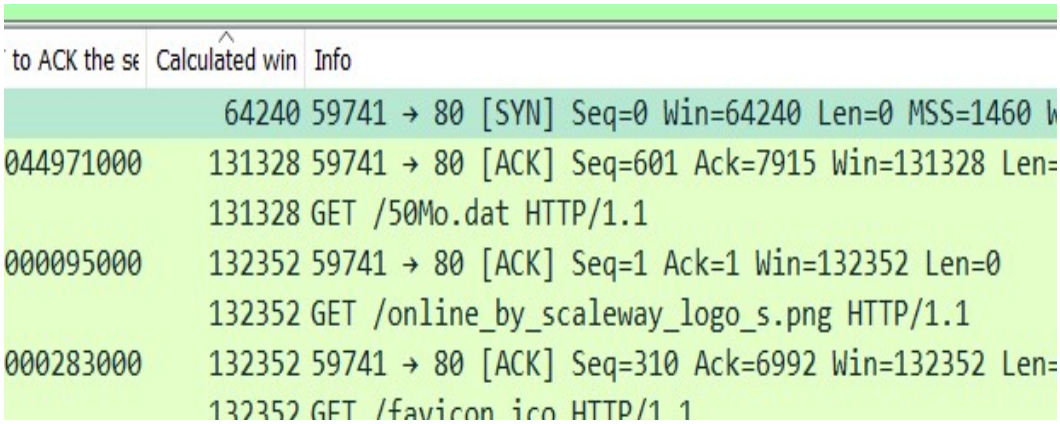
Ethernet · 4 | IPv4 · 18 | IPv6 | TCP · 24 | UDP · 1

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duratio
192.168.0.105	59741	62.210.18.40	80	7,764	9239 k	1,638	88 k	6,126	9151 k	50.023225	9.9163
192.168.0.105	59734	34.211.106.96	443	9	549	4	248	5	301	50.391448	9.7007
192.168.0.105	59697	13.107.4.254	443	1	54	0	0	1	54	50.736955	0.0000
192.168.0.105	59695	13.107.4.254	443	1	54	0	0	1	54	50.803663	0.0000
192.168.0.105	59693	204.79.197.222	443	1	54	0	0	1	54	51.573688	0.0000
192.168.0.105	59714	99.86.42.61	443	4	294	2	147	2	147	52.481806	0.0838
192.168.0.105	59724	52.26.239.181	443	9	561	4	248	5	313	52.666070	9.7145
192.168.0.105	59717	117.18.237.29	80	4	242	2	110	2	132	52.751082	10.184
192.168.0.105	59728	216.58.200.170	443	4	294	2	147	2	147	53.484208	0.1371
192.168.0.105	59726	52.26.239.181	443	9	561	4	248	5	313	53.900002	8.5309
192.168.0.105	59730	172.217.167.35	80	4	242	2	110	2	132	54.000158	10.062
192.168.0.105	59716	117.18.237.29	80	4	242	2	110	2	132	54.269919	10.138
192.168.0.105	59740	62.210.18.40	80	4	242	2	110	2	132	54.654972	10.360
192.168.0.105	59720	54.200.0.167	443	2	120	1	54	1	66	55.028463	0.0001
192.168.0.105	59729	13.33.93.4	443	4	242	2	110	2	132	55.425717	10.392
192.168.0.105	59710	34.107.221.82	80	4	242	2	110	2	132	55.525873	10.136
192.168.0.105	59723	35.165.119.189	443	7	440	3	193	4	247	55.557539	0.3655
192.168.0.105	59713	34.107.221.82	80	4	242	2	110	2	132	55.603887	10.150
192.168.0.105	59727	13.33.93.4	443	4	242	2	110	2	132	55.926017	10.433
192.168.0.105	59721	52.84.225.62	443	4	242	2	110	2	132	56.157867	10.255
192.168.0.105	59736	117.18.237.29	80	2	121	1	55	1	66	59.080106	0.0308
192.168.0.105	59378	52.139.250.253	443	2	108	1	54	1	54	60.483559	0.0002
192.168.0.105	59733	35.244.247.133	443	4	308	2	154	2	154	62.490207	0.0686
192.168.0.105	59739	35.186.227.140	443	4	294	2	147	2	147	65.529233	0.0840

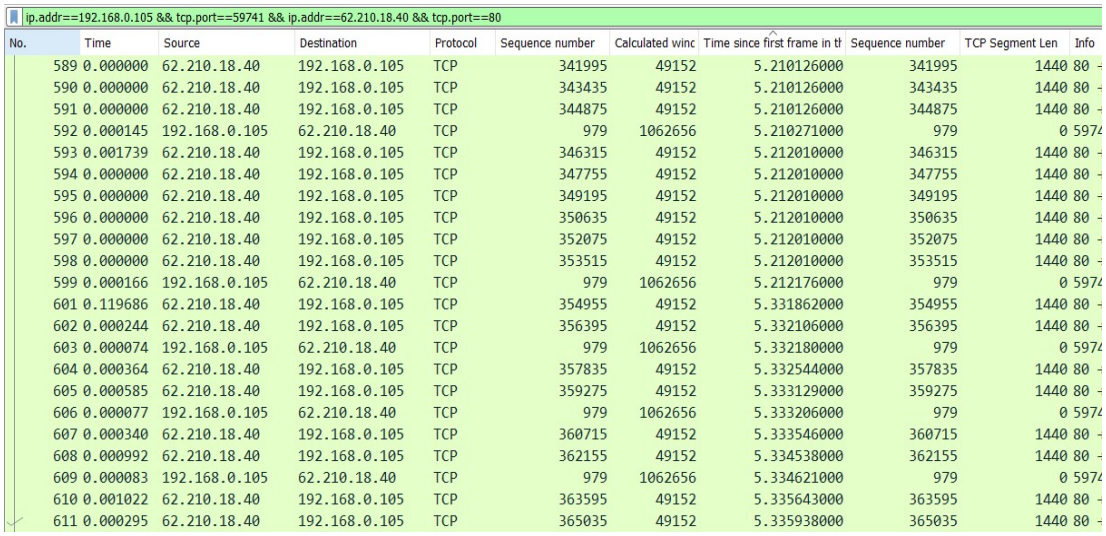


What is the minimum amount of available buffer space advertised at the client/receiver for the entire trace?

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



ip.addr==192.168.0.105 && tcp.port==59741 && ip.addr==62.210.18.40 && tcp.port==80

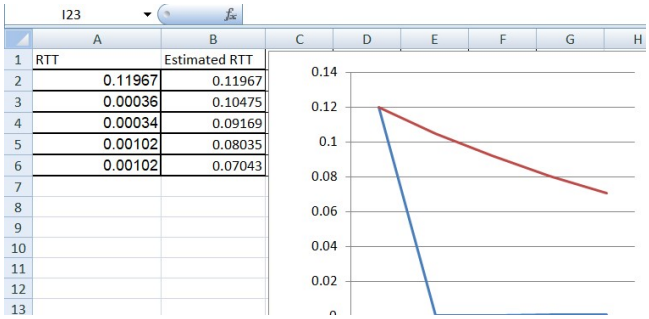
No.	Time	Source	Destination	Protocol	Sequence number	Calculated winc	Time since first frame in th	Sequence number	TCP Segment Len	Info
589	0.000000	62.210.18.40	192.168.0.105	TCP	341995	49152	5.210126000	341995	1440	80 -
590	0.000000	62.210.18.40	192.168.0.105	TCP	343435	49152	5.210126000	343435	1440	80 -
591	0.000000	62.210.18.40	192.168.0.105	TCP	344875	49152	5.210126000	344875	1440	80 -
592	0.000145	192.168.0.105	62.210.18.40	TCP	979	1062656	5.210271000	979	0	5974
593	0.001739	62.210.18.40	192.168.0.105	TCP	346315	49152	5.212010000	346315	1440	80 -
594	0.000000	62.210.18.40	192.168.0.105	TCP	347755	49152	5.212010000	347755	1440	80 -
595	0.000000	62.210.18.40	192.168.0.105	TCP	349195	49152	5.212010000	349195	1440	80 -
596	0.000000	62.210.18.40	192.168.0.105	TCP	350635	49152	5.212010000	350635	1440	80 -
597	0.000000	62.210.18.40	192.168.0.105	TCP	352075	49152	5.212010000	352075	1440	80 -
598	0.000000	62.210.18.40	192.168.0.105	TCP	353515	49152	5.212010000	353515	1440	80 -
599	0.000166	192.168.0.105	62.210.18.40	TCP	979	1062656	5.212176000	979	0	5974
601	0.119686	62.210.18.40	192.168.0.105	TCP	354955	49152	5.331862000	354955	1440	80 -
602	0.000244	62.210.18.40	192.168.0.105	TCP	356395	49152	5.332106000	356395	1440	80 -
603	0.000074	192.168.0.105	62.210.18.40	TCP	979	1062656	5.332180000	979	0	5974
604	0.000364	62.210.18.40	192.168.0.105	TCP	357835	49152	5.332544000	357835	1440	80 -
605	0.000585	62.210.18.40	192.168.0.105	TCP	359275	49152	5.333129000	359275	1440	80 -
606	0.000077	192.168.0.105	62.210.18.40	TCP	979	1062656	5.333206000	979	0	5974
607	0.000340	62.210.18.40	192.168.0.105	TCP	360715	49152	5.333546000	360715	1440	80 -
608	0.000992	62.210.18.40	192.168.0.105	TCP	362155	49152	5.334538000	362155	1440	80 -
609	0.000083	192.168.0.105	62.210.18.40	TCP	979	1062656	5.334621000	979	0	5974
610	0.001022	62.210.18.40	192.168.0.105	TCP	363595	49152	5.335643000	363595	1440	80 -
611	0.000295	62.210.18.40	192.168.0.105	TCP	365035	49152	5.335938000	365035	1440	80 -

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

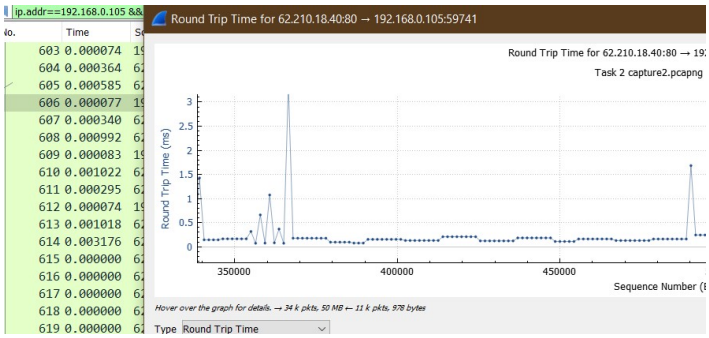
All packets have same length= 1440 bytes

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

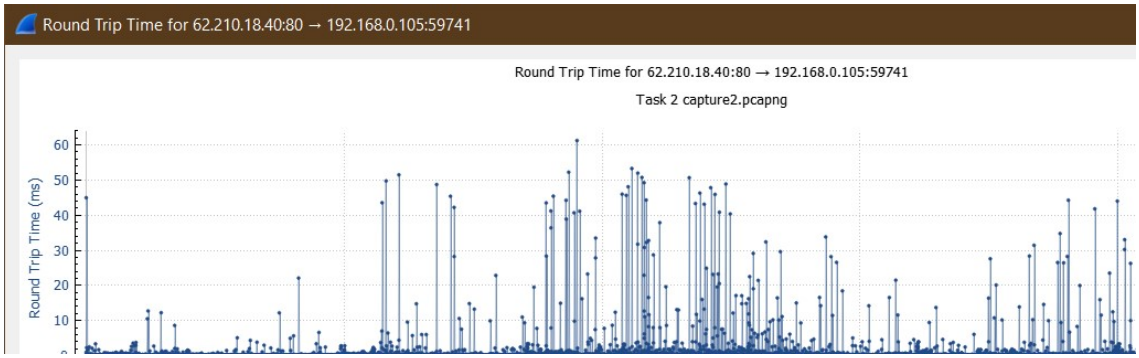
Segment 1: estimated rtt = actual rtt = 0.11967 s
Segment 2 : estimated rtt = (1-0.125) * 0.11967 + 0.125 * 0.00036 =0.10475 s
Segment 3 : estimated rtt = (1-0.125) * 0.10475 + 0.125 * 0.00034 =0.09169 s
Segment 4 : estimated rtt = (1-0.125) * 0.09169 + 0.125 * 0.00102 =0.08035 s
Segment 5 : estimated rtt = (1-0.125) * 0.08035 + 0.125 * 0.00102 =0.07043 s



	A	B
1	RTT	Estimated RTT
2	0.11967	0.11967
3	0.00036	0.10475
4	0.00034	0.09169
5	0.00102	0.08035
6	0.00102	0.07043



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

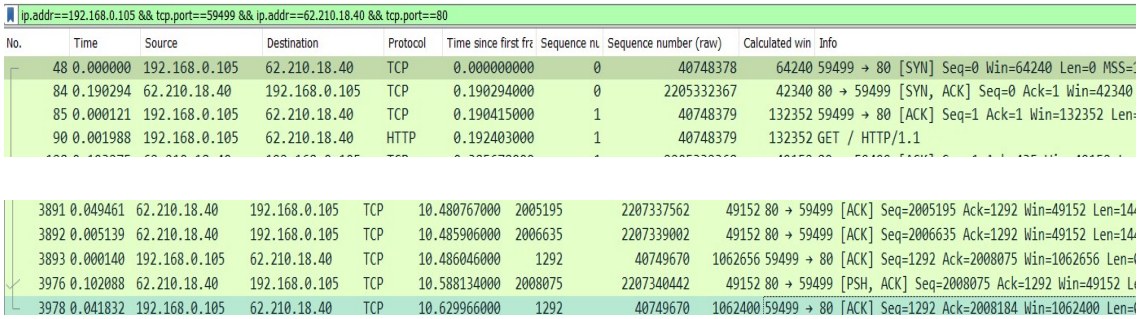
Total bytes transferred can be calculated using sequence number fields= initial seq no of first TCP segment – sequence no of last ack packet of this conversation

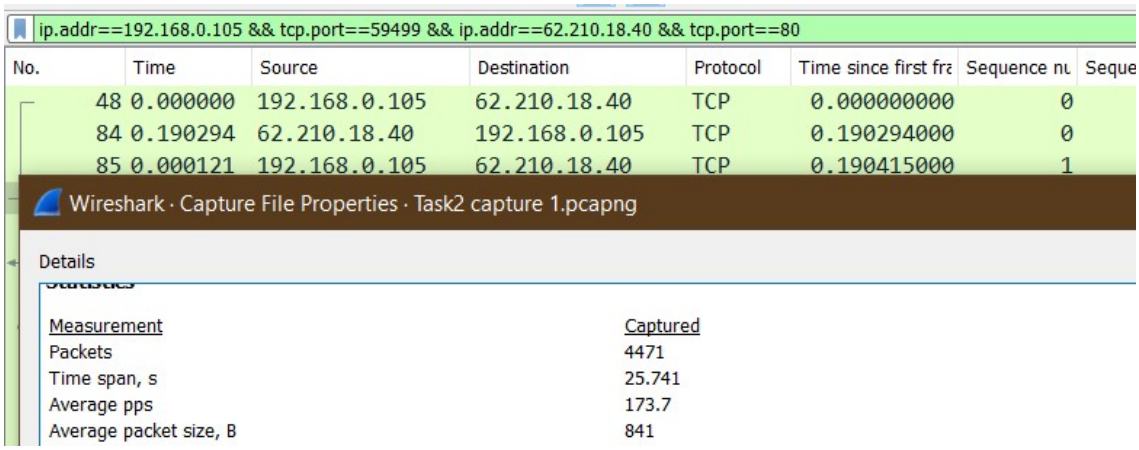
Total time taken= time of (first TCP Segment – last ack packet) of conversation

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

ip.addr==192.168.0.105 && tcp.port==59741 && ip.addr==62.210.18.40 && tcp.port==807

No.	Time	Source	Destination	Protocol	Time since first fra	Sequence nu	Sequence number (raw)	Calculated win	Info
7766	0.000000	62.210.18.40	192.168.0.105	TCP	8.084512000	7593835	1205921022	49152	80 → 59741 [ACK] Seq=7593835 Ack=979 Win=49152 Len=1
7767	0.000109	192.168.0.105	62.210.18.40	TCP	8.084621000	979	2663211613	2125312	59741 → 80 [ACK] Seq=979 Ack=7595275 Win=2125312 Ler
7768	0.000476	62.210.18.40	192.168.0.105	TCP	8.085097000	7595275	1205922462	49152	80 → 59741 [ACK] Seq=7595275 Ack=979 Win=49152 Len=1
7769	0.000000	62.210.18.40	192.168.0.105	TCP	8.085097000	7596715	1205923902	49152	80 → 59741 [ACK] Seq=7596715 Ack=979 Win=49152 Len=1
7770	0.000000	62.210.18.40	192.168.0.105	TCP	8.085097000	7598155	1205925342	49152	80 → 59741 [ACK] Seq=7598155 Ack=979 Win=49152 Len=1
7771	0.000062	192.168.0.105	62.210.18.40	TCP	8.085159000	979	2663211613	2125312	59741 → 80 [ACK] Seq=979 Ack=7599595 Win=2125312 Ler
7772	0.002020	62.210.18.40	192.168.0.105	TCP	8.087179000	7599595	1205926782	49152	80 → 59741 [ACK] Seq=7599595 Ack=979 Win=49152 Len=1
47173	0.000000	62.210.18.40	192.168.0.105	TCP	56.501054000	50001835	1248329022	49152	80 → 59741 [ACK] Seq=50001835 Ack=979 Win=49152 Len=144
47174	0.000125	192.168.0.105	62.210.18.40	TCP	56.501179000	979	2663211613	4250880	59741 → 80 [ACK] Seq=979 Ack=50003275 Win=4250880 Len=0
47175	0.001294	62.210.18.40	192.168.0.105	TCP	56.502473000	50003275	1248330462	49152	80 → 59741 [ACK] Seq=50003275 Ack=979 Win=49152 Len=144
47176	0.000000	62.210.18.40	192.168.0.105	TCP	56.502473000	50004715	1248331902	49152	80 → 59741 [ACK] Seq=50004715 Ack=979 Win=49152 Len=144
47177	0.000000	62.210.18.40	192.168.0.105	TCP	56.502473000	50006155	1248333342	49152	80 → 59741 [ACK] Seq=50006155 Ack=979 Win=49152 Len=144
47178	0.000000	62.210.18.40	192.168.0.105	HTTP	56.502473000	50007595	1248334782	49152	HTTP/1.1 200 OK

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

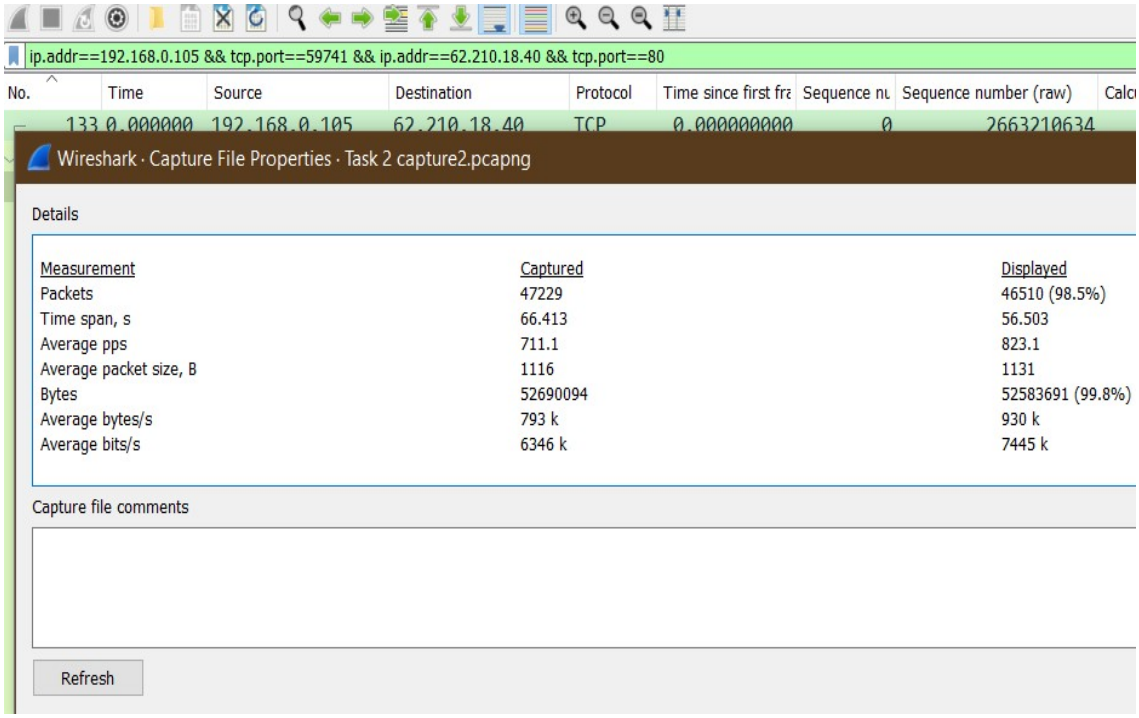
Total time taken= time of (first TCP Segment – last ack packet) of conversation

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.



ip.addr==192.168.0.105 && tcp.port==59741 && ip.addr==62.210.18.40 && tcp.port==80

No.	Time	Source	Destination	Protocol	Time since first fra	Sequence nu	Sequence number (raw)	Calcu
133	0.000000	192.168.0.105	62.210.18.40	TCP	0.000000000	0	2663210634	

Wireshark · Capture File Properties · Task 2 capture2.pcapng

Details

Measurement	Captured	Displayed
Packets	47229	46510 (98.5%)
Time span, s	66.413	56.503
Average pps	711.1	823.1
Average packet size, B	1116	1131
Bytes	52690094	52583691 (99.8%)
Average bytes/s	793 k	930 k
Average bits/s	6346 k	7445 k

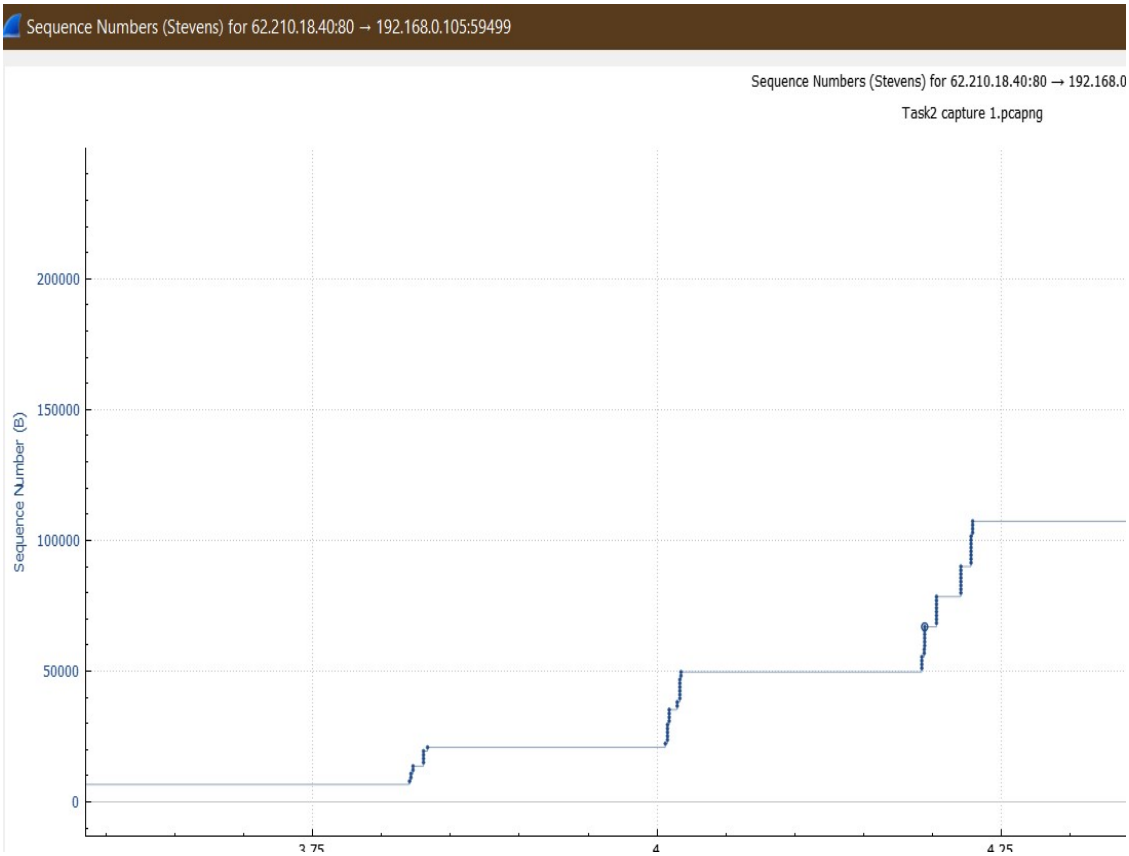
Capture file comments

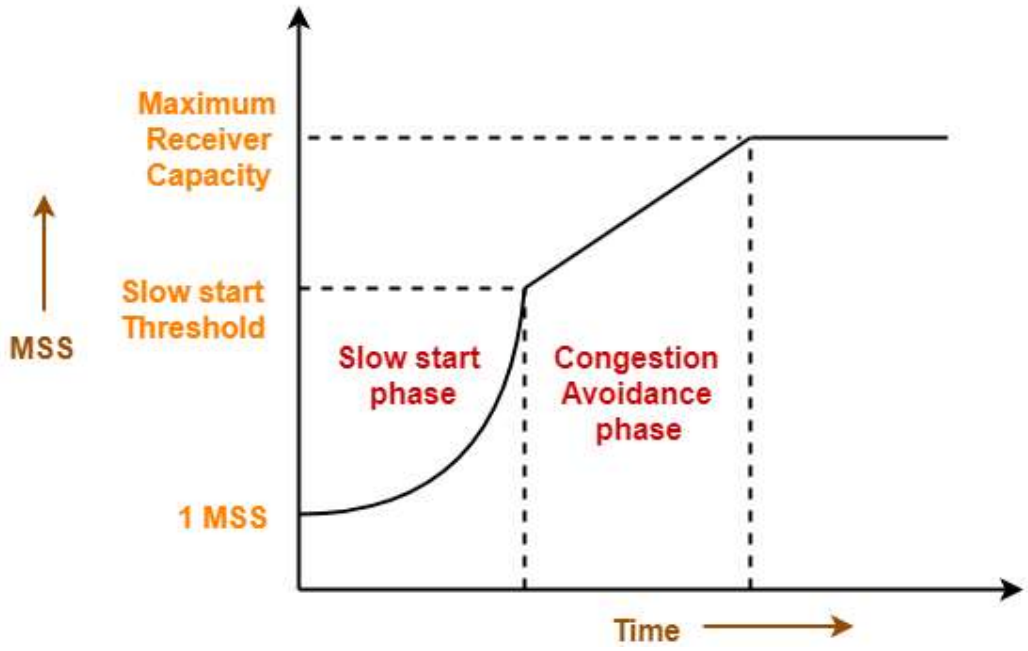
Refresh

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

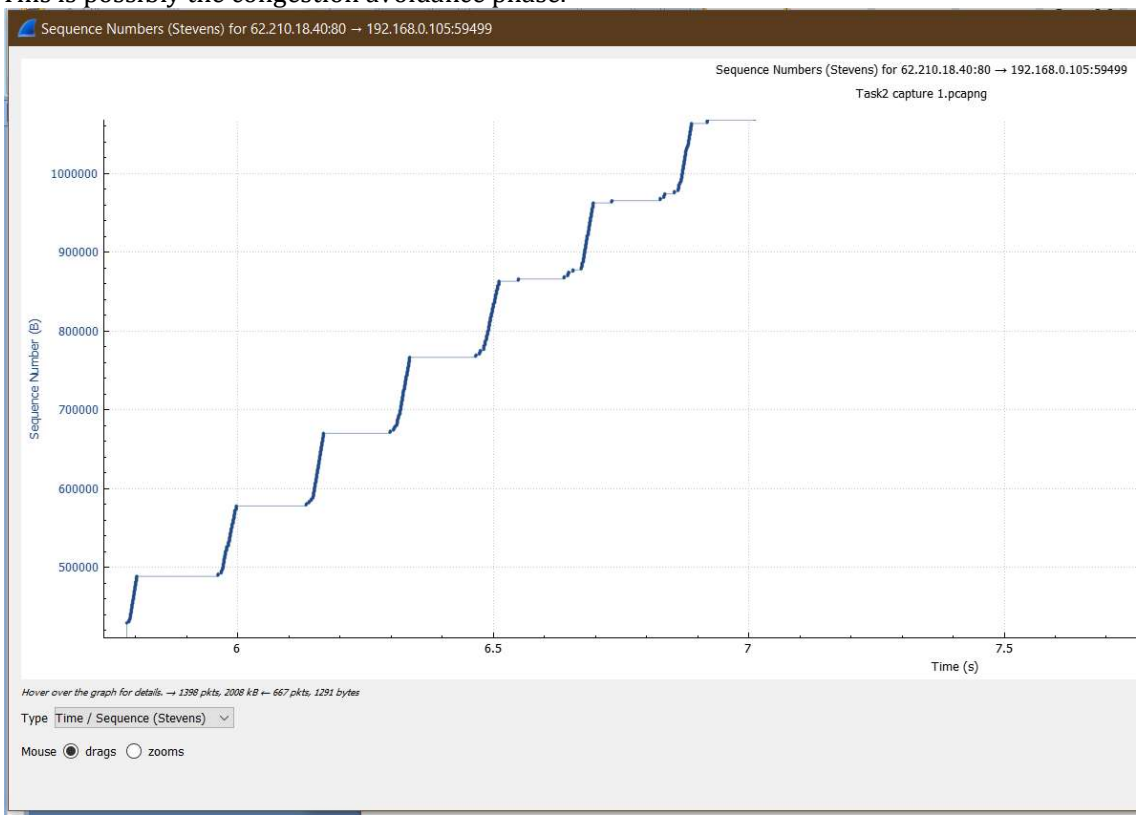
The figure below shows the TCP slow start phase...we can count the number of packets from here its increasing like 10---20----40---80 and so on

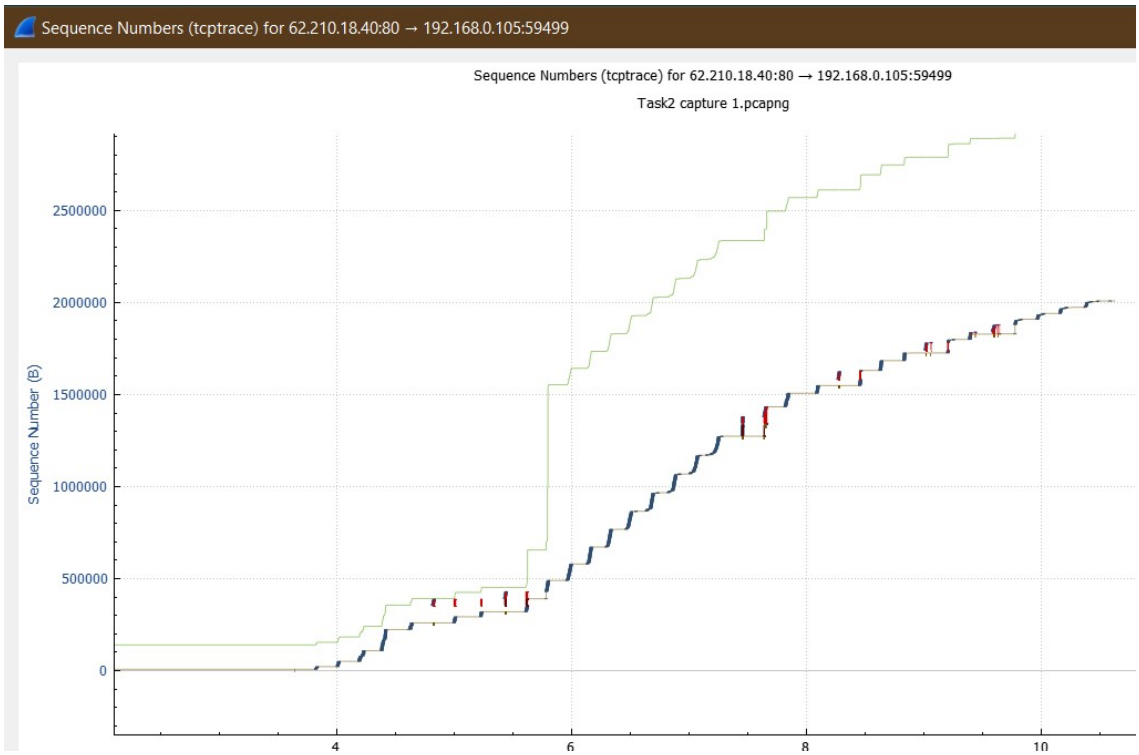




In congestion avoidance TCP linearly increases the window size by 1 after every ACK.

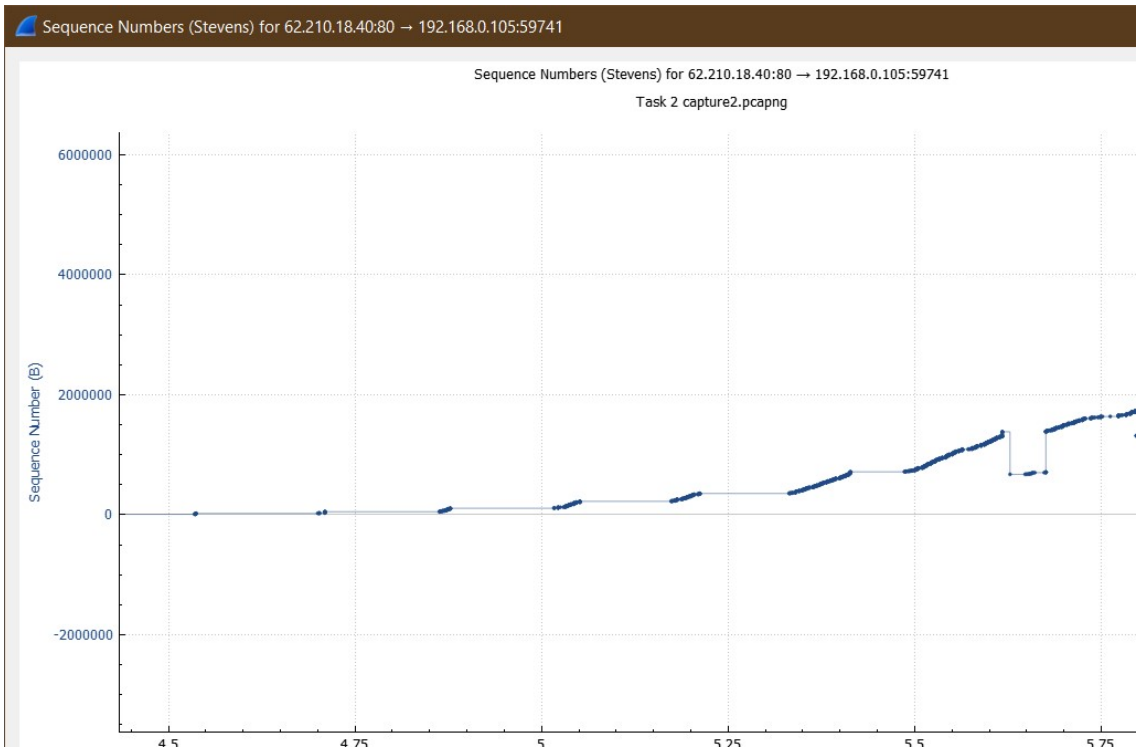
The below picture shows the linear increase in the packet numbers when counted manually. This is possibly the congestion avoidance phase.

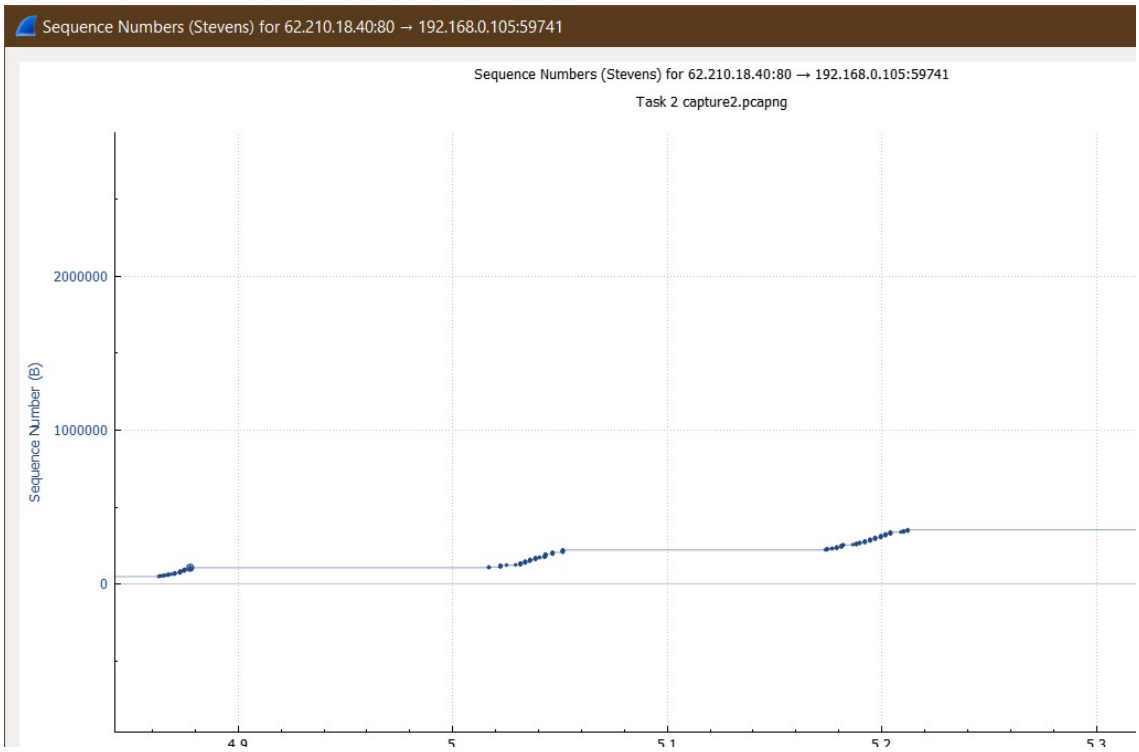




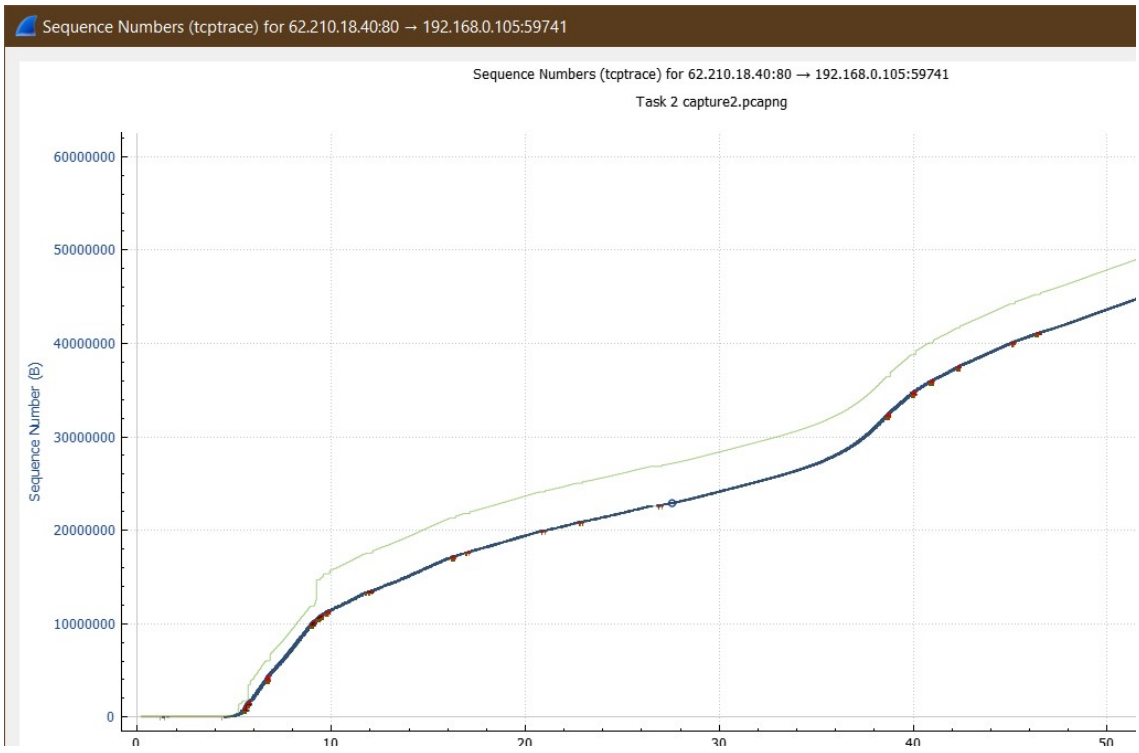
More can be seen from TCP trace graph for congestion.

Capture 2

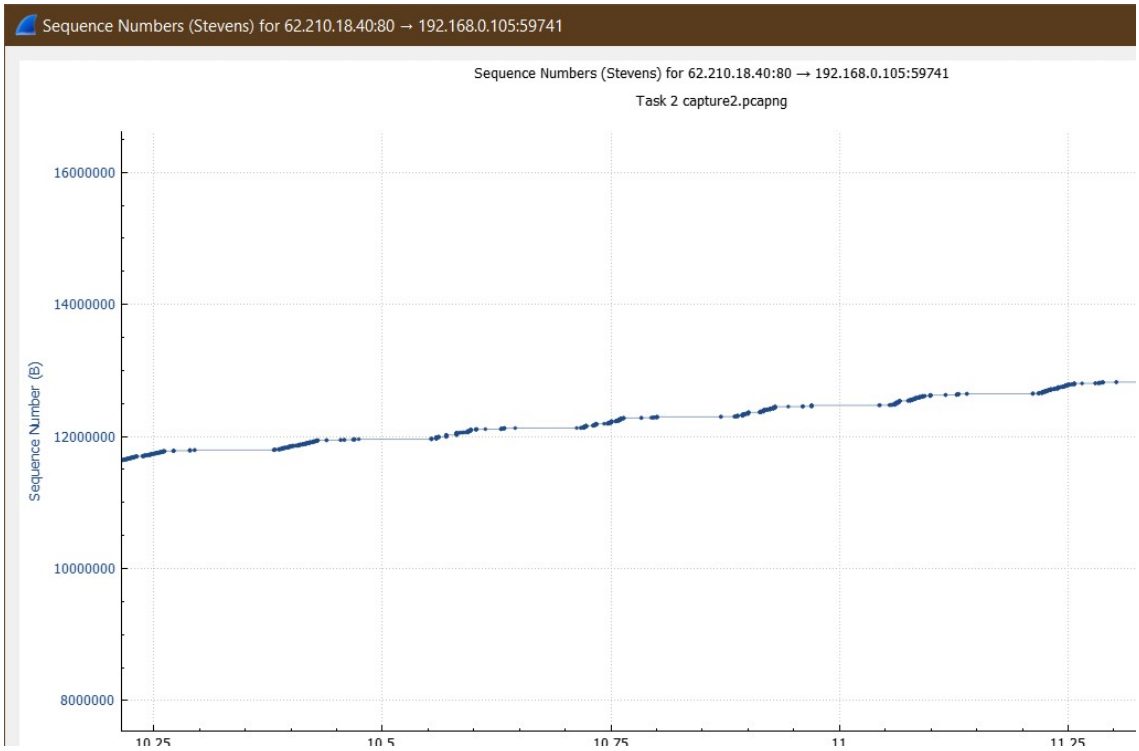




TCP slow start phase where there is a exponential increase in number of packets.

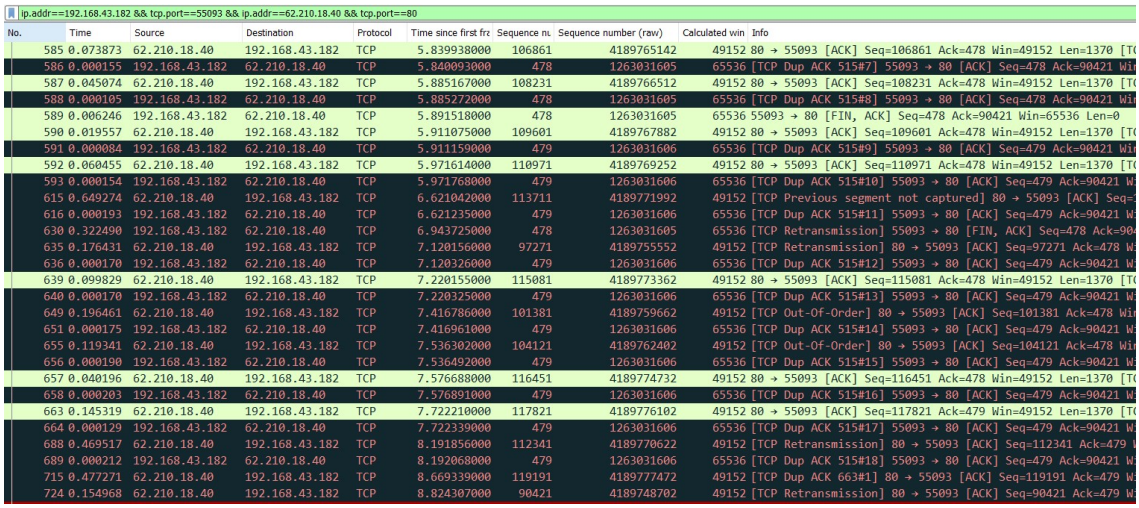


More can be seen from TCP trace graph for congestion.



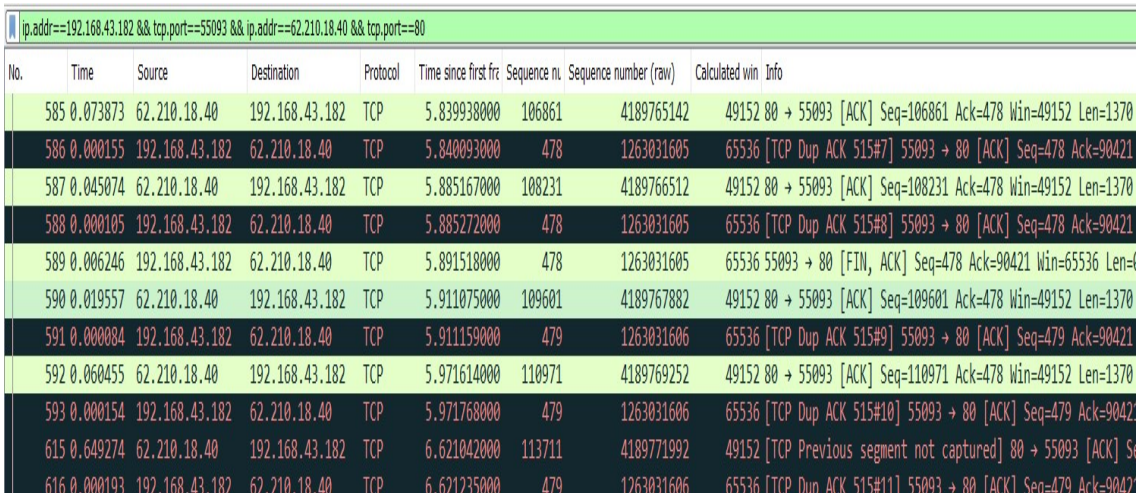
Possible congestion avoidance phase where we can see linear increase in number of packets.

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.



ip.addr==192.168.43.182 && tcp.port==55093 && ip.addr==62.210.18.40 && tcp.port==80

No.	Time	Source	Destination	Protocol	Time since first fra	Sequence nu	Sequence number (raw)	Calculated win	Info
585	0.073873	62.210.18.40	192.168.43.182	TCP	5.839938000	106861	4189765142	49152	80 → 55093 [ACK] Seq=106861 Ack=478 Win=49152 Len=1370 [T
586	0.000155	192.168.43.182	62.210.18.40	TCP	5.840093000	478	1263031605	65536	[TCP Dup ACK 515#7] 55093 → 80 [ACK] Seq=478 Ack=90421 Wi
587	0.045074	62.210.18.40	192.168.43.182	TCP	5.885167000	108231	4189766512	49152	80 → 55093 [ACK] Seq=108231 Ack=478 Win=49152 Len=1370 [T
588	0.000105	192.168.43.182	62.210.18.40	TCP	5.885272000	478	1263031605	65536	[TCP Dup ACK 515#8] 55093 → 80 [ACK] Seq=478 Ack=90421 Wi
589	0.006246	192.168.43.182	62.210.18.40	TCP	5.891518000	478	1263031605	65536	55093 → 80 [FIN, ACK] Seq=478 Ack=90421 Win=65536 Len=0
590	0.019557	62.210.18.40	192.168.43.182	TCP	5.911075000	109601	4189767882	49152	80 → 55093 [ACK] Seq=109601 Ack=478 Win=49152 Len=1370 [T
591	0.000084	192.168.43.182	62.210.18.40	TCP	5.911159000	479	1263031606	65536	[TCP Dup ACK 515#9] 55093 → 80 [ACK] Seq=479 Ack=90421 Wi
592	0.060455	62.210.18.40	192.168.43.182	TCP	5.971614000	110971	4189769252	49152	80 → 55093 [ACK] Seq=110971 Ack=478 Win=49152 Len=1370 [T
593	0.000154	192.168.43.182	62.210.18.40	TCP	5.971768000	479	1263031606	65536	[TCP Dup ACK 515#10] 55093 → 80 [ACK] Seq=479 Ack=90421 W
615	0.649274	62.210.18.40	192.168.43.182	TCP	6.621042000	113711	4189771992	49152	[TCP Previous segment not captured] 80 → 55093 [ACK] Seq=
616	0.000193	192.168.43.182	62.210.18.40	TCP	6.621235000	479	1263031606	65536	[TCP Dup ACK 515#11] 55093 → 80 [ACK] Seq=479 Ack=90421 W
630	0.322490	192.168.43.182	62.210.18.40	TCP	6.943725000	478	1263031605	65536	[TCP Retransmission] 55093 → 80 [FIN, ACK] Seq=478 Ack=904
635	0.176431	62.210.18.40	192.168.43.182	TCP	7.120156000	97271	4189755552	49152	[TCP Retransmission] 80 → 55093 [ACK] Seq=97271 Ack=478 W
636	0.000170	192.168.43.182	62.210.18.40	TCP	7.120326000	479	1263031606	65536	[TCP Dup ACK 515#12] 55093 → 80 [ACK] Seq=479 Ack=90421 W
639	0.099829	62.210.18.40	192.168.43.182	TCP	7.220155000	115081	4189773362	49152	80 → 55093 [ACK] Seq=115081 Ack=478 Win=49152 Len=1370 [T
640	0.000170	192.168.43.182	62.210.18.40	TCP	7.220325000	479	1263031606	65536	[TCP Dup ACK 515#13] 55093 → 80 [ACK] Seq=479 Ack=90421 W
649	0.196461	62.210.18.40	192.168.43.182	TCP	7.416786000	101381	4189759662	49152	[TCP Out-Of-Order] 80 → 55093 [ACK] Seq=101381 Ack=478 Wi
651	0.000175	192.168.43.182	62.210.18.40	TCP	7.416961000	479	1263031606	65536	[TCP Dup ACK 515#14] 55093 → 80 [ACK] Seq=479 Ack=90421 W
655	0.119341	62.210.18.40	192.168.43.182	TCP	7.536302000	104121	4189762402	49152	[TCP Out-Of-Order] 80 → 55093 [ACK] Seq=104121 Ack=478 Wi
656	0.000190	192.168.43.182	62.210.18.40	TCP	7.536492000	479	1263031606	65536	[TCP Dup ACK 515#15] 55093 → 80 [ACK] Seq=479 Ack=90421 W
657	0.040196	62.210.18.40	192.168.43.182	TCP	7.576688000	116451	4189774732	49152	80 → 55093 [ACK] Seq=116451 Ack=478 Win=49152 Len=1370 [T
658	0.000203	192.168.43.182	62.210.18.40	TCP	7.576891000	479	1263031606	65536	[TCP Dup ACK 515#16] 55093 → 80 [ACK] Seq=479 Ack=90421 W
663	0.145319	62.210.18.40	192.168.43.182	TCP	7.722210000	117821	4189776102	49152	80 → 55093 [ACK] Seq=117821 Ack=479 Win=49152 Len=1370 [T
664	0.000129	192.168.43.182	62.210.18.40	TCP	7.722339000	479	1263031606	65536	[TCP Dup ACK 515#17] 55093 → 80 [ACK] Seq=479 Ack=90421 W
688	0.469517	62.210.18.40	192.168.43.182	TCP	8.191856000	112341	4189770622	49152	[TCP Retransmission] 80 → 55093 [ACK] Seq=112341 Ack=479 W
689	0.000212	192.168.43.182	62.210.18.40	TCP	8.192068000	479	1263031606	65536	[TCP Dup ACK 515#18] 55093 → 80 [ACK] Seq=479 Ack=90421 W
715	0.477271	62.210.18.40	192.168.43.182	TCP	8.669339000	119191	4189777472	49152	[TCP Dup ACK 663#1] 80 → 55093 [ACK] Seq=119191 Ack=479 W
724	0.154968	62.210.18.40	192.168.43.182	TCP	8.824307000	90421	4189748702	49152	[TCP Retransmission] 80 → 55093 [ACK] Seq=90421 Ack=479 W



ip.addr==192.168.43.182 && tcp.port==55093 && ip.addr==62.210.18.40 && tcp.port==80

No.	Time	Source	Destination	Protocol	Time since first fra	Sequence nu	Sequence number (raw)	Calculated win	Info
585	0.073873	62.210.18.40	192.168.43.182	TCP	5.839938000	106861	4189765142	49152	80 → 55093 [ACK] Seq=106861 Ack=478 Win=49152 Len=1370
586	0.000155	192.168.43.182	62.210.18.40	TCP	5.840093000	478	1263031605	65536	[TCP Dup ACK 515#7] 55093 → 80 [ACK] Seq=478 Ack=90421
587	0.045074	62.210.18.40	192.168.43.182	TCP	5.885167000	108231	4189766512	49152	80 → 55093 [ACK] Seq=108231 Ack=478 Win=49152 Len=1370
588	0.000105	192.168.43.182	62.210.18.40	TCP	5.885272000	478	1263031605	65536	[TCP Dup ACK 515#8] 55093 → 80 [ACK] Seq=478 Ack=90421
589	0.006246	192.168.43.182	62.210.18.40	TCP	5.891518000	478	1263031605	65536	55093 → 80 [FIN, ACK] Seq=478 Ack=90421 Win=65536 Len=
590	0.019557	62.210.18.40	192.168.43.182	TCP	5.911075000	109601	4189767882	49152	80 → 55093 [ACK] Seq=109601 Ack=478 Win=49152 Len=1370
591	0.000084	192.168.43.182	62.210.18.40	TCP	5.911159000	479	1263031606	65536	[TCP Dup ACK 515#9] 55093 → 80 [ACK] Seq=479 Ack=90421
592	0.060455	62.210.18.40	192.168.43.182	TCP	5.971614000	110971	4189769252	49152	80 → 55093 [ACK] Seq=110971 Ack=478 Win=49152 Len=1370
593	0.000154	192.168.43.182	62.210.18.40	TCP	5.971768000	479	1263031606	65536	[TCP Dup ACK 515#10] 55093 → 80 [ACK] Seq=479 Ack=9042
615	0.649274	62.210.18.40	192.168.43.182	TCP	6.621042000	113711	4189771992	49152	[TCP Previous segment not captured] 80 → 55093 [ACK] S
616	0.000193	192.168.43.182	62.210.18.40	TCP	6.621235000	479	1263031606	65536	[TCP Dup ACK 515#11] 55093 → 80 [ACK] Seq=479 Ack=9042

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.