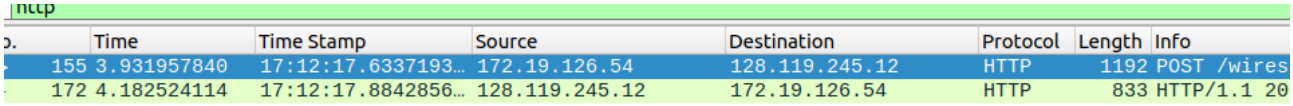
Assignment 3: Wireshark With TCP

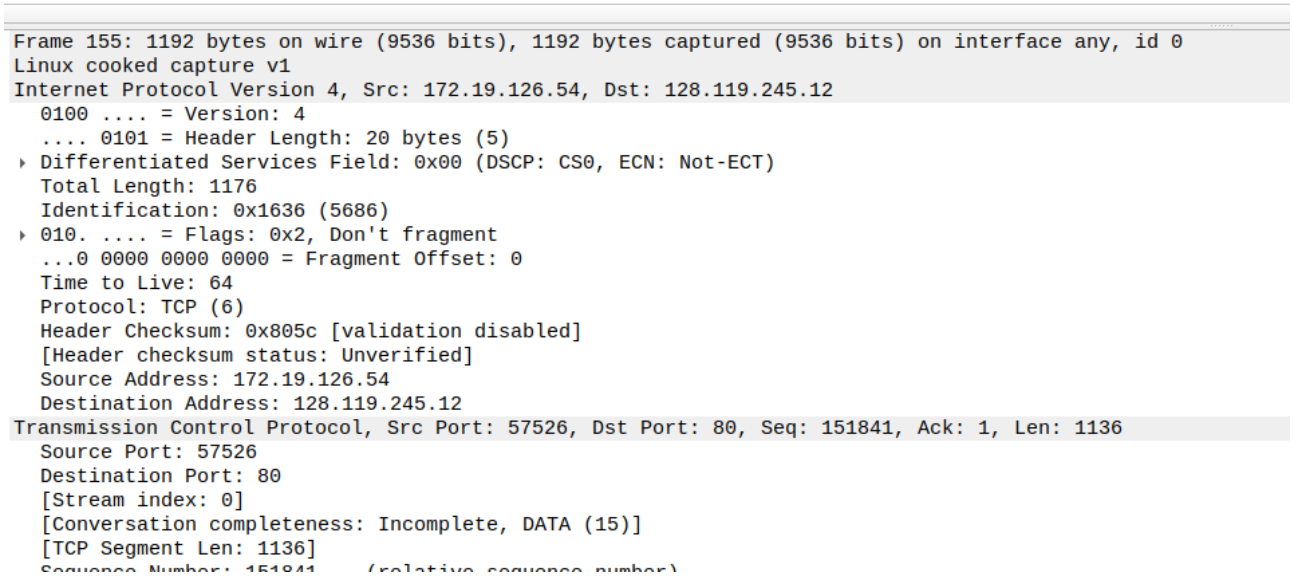
PART-A

Question-1

Source IP Address: 172.19.126.54
Source Port Number: 57526



No.	Time	Time Stamp	Source	Destination	Protocol	Length	Info
155	3.931957840	17:12:17.6337193…	172.19.126.54	128.119.245.12	HTTP	1192	POST /wires
172	4.182524114	17:12:17.8842856…	128.119.245.12	172.19.126.54	HTTP	833	HTTP/1.1 20



```
Frame 155: 1192 bytes on wire (9536 bits), 1192 bytes captured (9536 bits) on interface any, id 0
Linux cooked capture v1
Internet Protocol Version 4, Src: 172.19.126.54, Dst: 128.119.245.12
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 1176
  Identification: 0x1636 (5686)
▸ 010. .... = Flags: 0x2, Don't fragment
  ...0 0000 0000 0000 = Fragment Offset: 0
  Time to Live: 64
  Protocol: TCP (6)
  Header Checksum: 0x805c [validation disabled]
  [Header checksum status: Unverified]
  Source Address: 172.19.126.54
  Destination Address: 128.119.245.12
Transmission Control Protocol, Src Port: 57526, Dst Port: 80, Seq: 151841, Ack: 1, Len: 1136
  Source Port: 57526
  Destination Port: 80
  [Stream index: 0]
  [Conversation completeness: Incomplete, DATA (15)]
  [TCP Segment Len: 1136]
  Sequence Number: 151841    (relative sequence number)
```

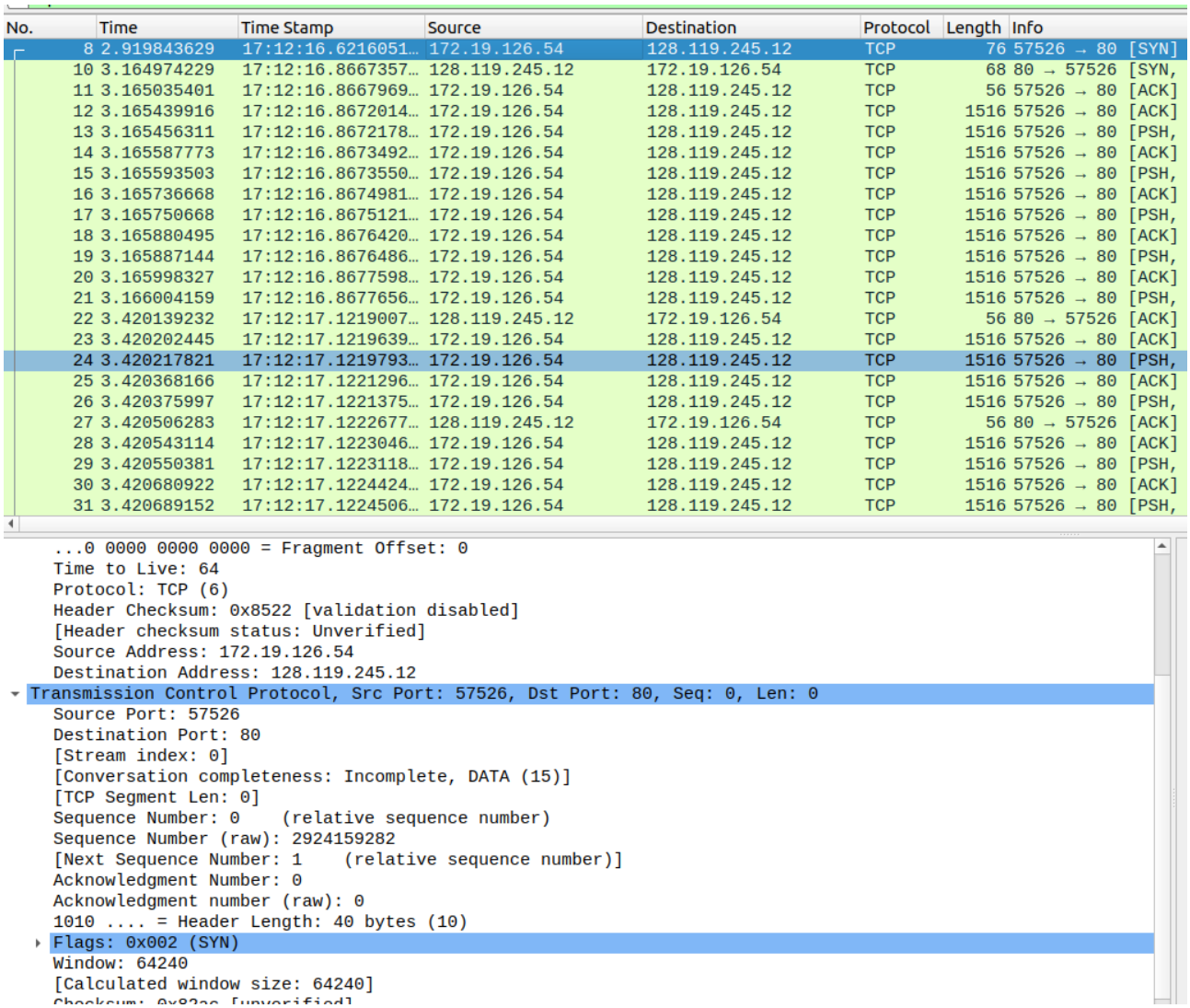
figure 1

Question-2

Destination IP Address: 128.119.245.12
Destination Port Number: 80
Reference: *figure 1*

Question-3

a) Sequence Number: 2924159282
b) It has set its Flag Feild value as 0x0002 [SYN] that sets the SYN Flag bit which will ultimately make it identified as SYN Segment
c)Yes, it will allow reciever to selectively acknowledge the out of order pakages and specify which segments are missing as it allows SACK, it can be inspected by observing the TCP Options of TCP headers of Packets Exhnaged during HandShake i.e. Kind: SACK Permitted (4) as of the current example. [*figure 2.2*]



No.	Time	Time Stamp	Source	Destination	Protocol	Length	Info
8	2.919843629	17:12:16.6216051…	172.19.126.54	128.119.245.12	TCP	76	57526 → 80 [SYN]
10	3.164974229	17:12:16.8667357…	128.119.245.12	172.19.126.54	TCP	68	80 → 57526 [SYN,
11	3.165035401	17:12:16.8667969…	172.19.126.54	128.119.245.12	TCP	56	57526 → 80 [ACK]
12	3.165439916	17:12:16.8672014…	172.19.126.54	128.119.245.12	TCP	1516	57526 → 80 [ACK]
13	3.165456311	17:12:16.8672178…	172.19.126.54	128.119.245.12	TCP	1516	57526 → 80 [PSH,
14	3.165587773	17:12:16.8673492…	172.19.126.54	128.119.245.12	TCP	1516	57526 → 80 [ACK]
15	3.165593503	17:12:16.8673550…	172.19.126.54	128.119.245.12	TCP	1516	57526 → 80 [PSH,
16	3.165736668	17:12:16.8674981…	172.19.126.54	128.119.245.12	TCP	1516	57526 → 80 [ACK]
17	3.165750668	17:12:16.8675121…	172.19.126.54	128.119.245.12	TCP	1516	57526 → 80 [PSH,
18	3.165880495	17:12:16.8676420…	172.19.126.54	128.119.245.12	TCP	1516	57526 → 80 [ACK]
19	3.165887144	17:12:16.8676486…	172.19.126.54	128.119.245.12	TCP	1516	57526 → 80 [PSH,
20	3.165998327	17:12:16.8677598…	172.19.126.54	128.119.245.12	TCP	1516	57526 → 80 [ACK]
21	3.166004159	17:12:16.8677656…	172.19.126.54	128.119.245.12	TCP	1516	57526 → 80 [PSH,
22	3.420139232	17:12:17.1219007…	128.119.245.12	172.19.126.54	TCP	56	80 → 57526 [ACK]
23	3.420202445	17:12:17.1219639…	172.19.126.54	128.119.245.12	TCP	1516	57526 → 80 [ACK]
24	3.420217821	17:12:17.1219793…	172.19.126.54	128.119.245.12	TCP	1516	57526 → 80 [PSH,
25	3.420368166	17:12:17.1221296…	172.19.126.54	128.119.245.12	TCP	1516	57526 → 80 [ACK]
26	3.420375997	17:12:17.1221375…	172.19.126.54	128.119.245.12	TCP	1516	57526 → 80 [PSH,
27	3.420506283	17:12:17.1222677…	128.119.245.12	172.19.126.54	TCP	56	80 → 57526 [ACK]
28	3.420543114	17:12:17.1223046…	172.19.126.54	128.119.245.12	TCP	1516	57526 → 80 [ACK]
29	3.420550381	17:12:17.1223118…	172.19.126.54	128.119.245.12	TCP	1516	57526 → 80 [PSH,
30	3.420680922	17:12:17.1224424…	172.19.126.54	128.119.245.12	TCP	1516	57526 → 80 [ACK]
31	3.420689152	17:12:17.1224506…	172.19.126.54	128.119.245.12	TCP	1516	57526 → 80 [PSH,

```
...0 0000 0000 0000 = Fragment Offset: 0
Time to Live: 64
Protocol: TCP (6)
Header Checksum: 0x8522 [validation disabled]
[Header checksum status: Unverified]
Source Address: 172.19.126.54
Destination Address: 128.119.245.12
Transmission Control Protocol, Src Port: 57526, Dst Port: 80, Seq: 0, Len: 0
Source Port: 57526
Destination Port: 80
[Stream index: 0]
[Conversation completeness: Incomplete, DATA (15)]
[TCP Segment Len: 0]
Sequence Number: 0    (relative sequence number)
Sequence Number (raw): 2924159282
[Next Sequence Number: 1    (relative sequence number)]
Acknowledgment Number: 0
Acknowledgment number (raw): 0
1010 .... = Header Length: 40 bytes (10)
Flags: 0x002 (SYN)
Window: 64240
[Calculated window size: 64240]
```

Figure 2.1

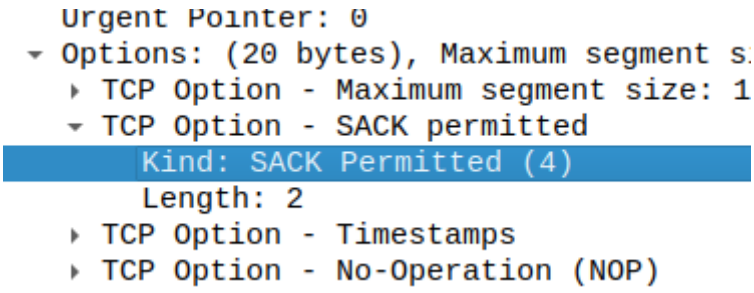


Figure 2.2

Question-4

a) Sequence Number: 2495720350
b) Flag: 0x0012 (SYN, ACK)
c) Acknowledgement Number: 2924159283
d) As gaia.cs.umass.edu recieved a TCP Connection Request with a Squence Number 2924159282 and Length 0, from that it can understand that the next packet it must expect would be next Byte in Sequence Number. So it sends that as ACK Number.

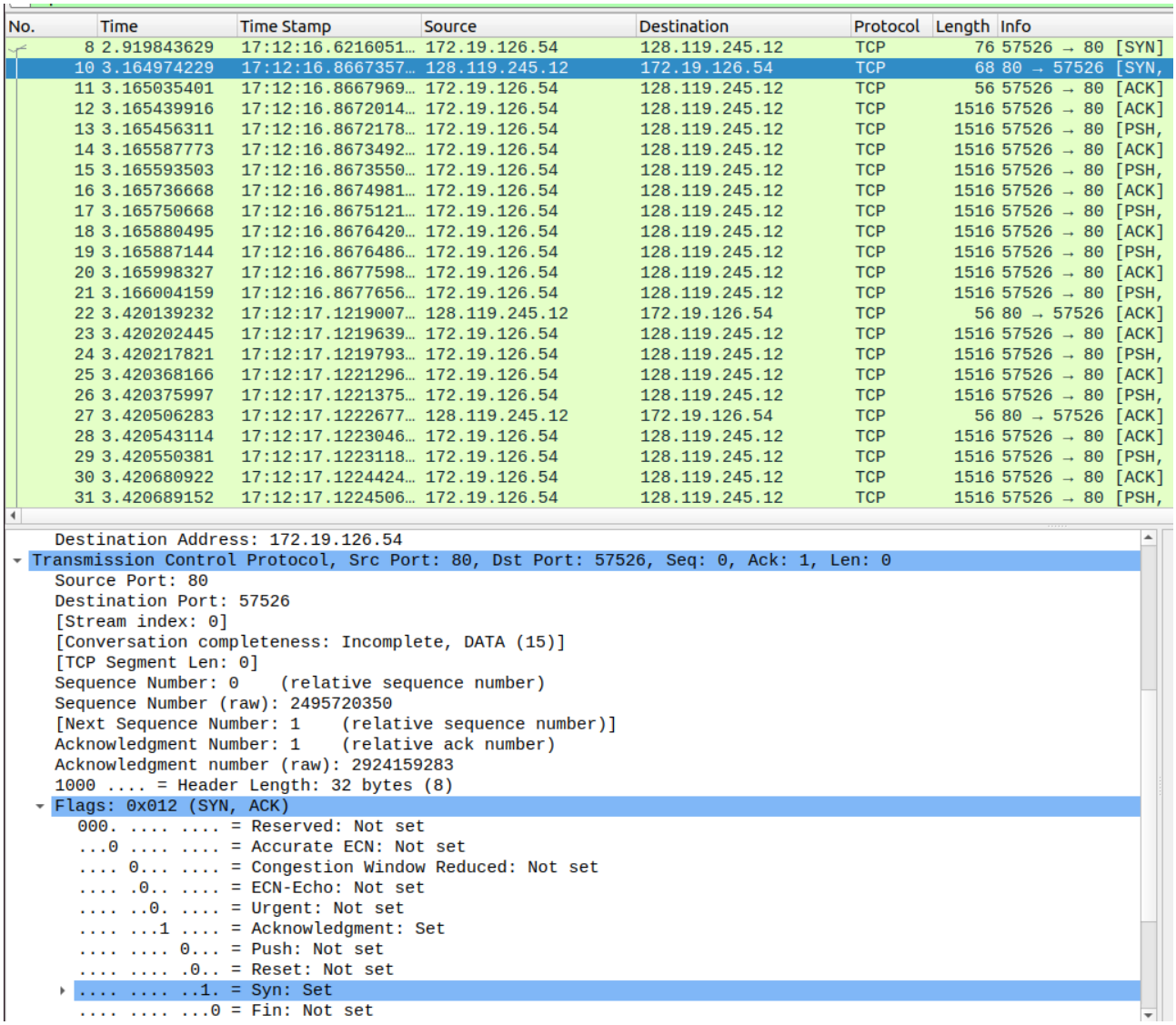


Figure 3

Question-5

a) Sequence Number: 2924311123
b) Payload: 1136
c) The size of file is 152.1 kB (1,52,138 bytes) which is much less then tha TCP Payload so all the data of the transferred file can’t sit and fit into a single TCP packet.

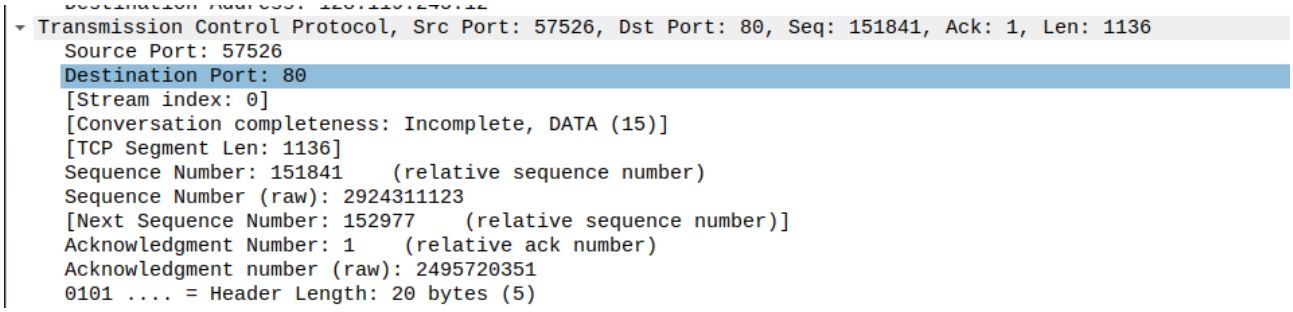


Figure 4

Question-6

a) 17:12:16.867201424 (or) 3.165439916 (s) after the TCP SYN Segment was sent [*figure 6.1*]
b) 17:12:17.121900740 [*figure 6.2*]
c) It Actually didn't Recieved ACK for the First Packet sent [*figure 6.2*]
d) RTT for this Second data-containing segment: 0.254682921 (s)
e) As I didn't recieved ACK for the First Packet, considering results of my friend's wireshark capture. He got the following result:

RTT for this first data-containing segment is **0.000705242** seconds.

RTT value of the second data-carrying TCP segment and its ACK is **0.003076674** seconds.

Estimated RTT after the ACK for the second data-carrying segment is received is **0.001001671** seconds

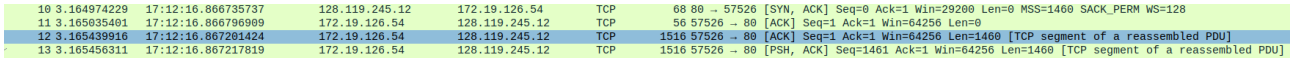
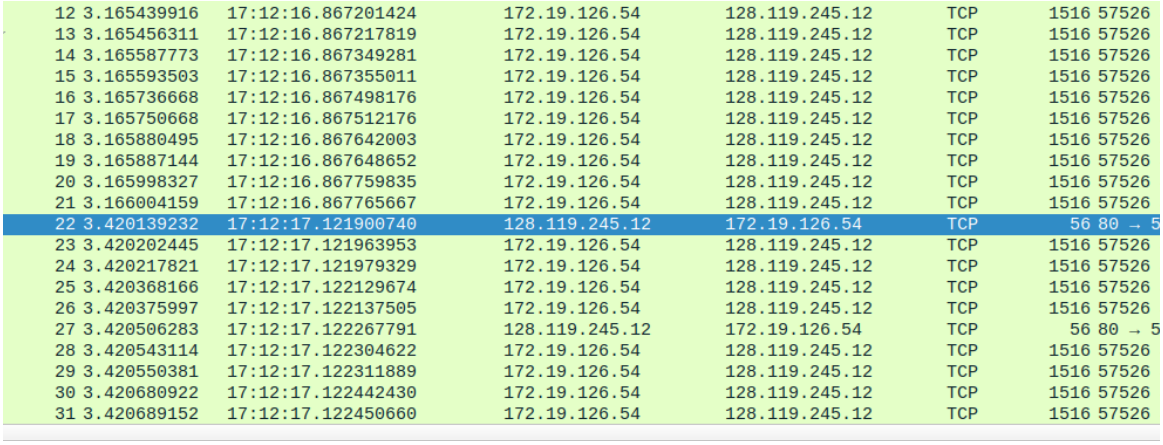


figure 6.1



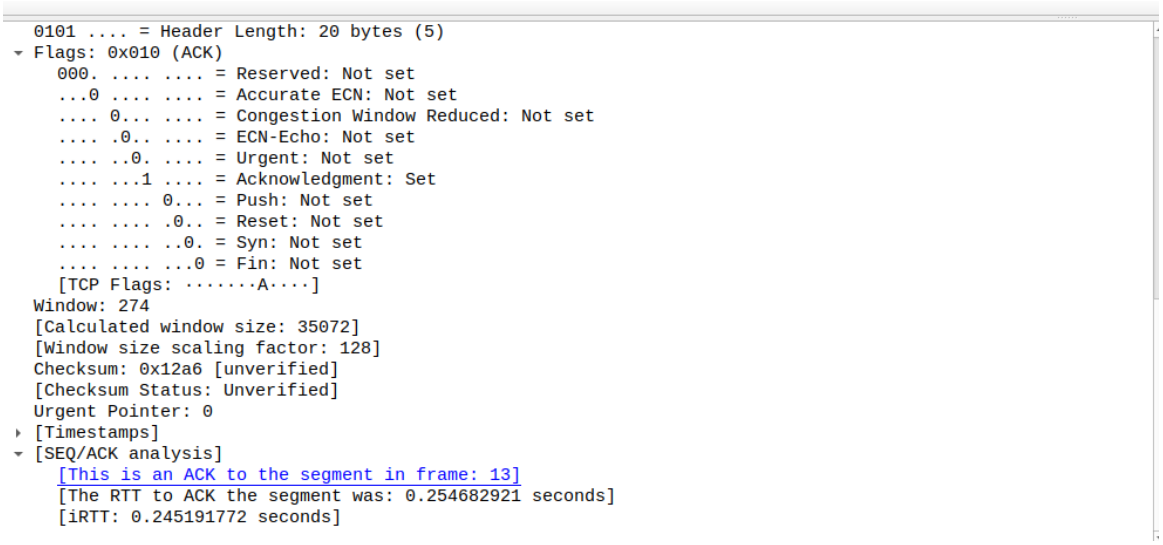


figure 6.2

Question-7

Each one of four has Total: 1480 (Talking just at TCP Level)

Source	Destination	Protocol	Length	Info
172.19.126.54	128.119.245.12	TCP	76	57526 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM
128.119.245.12	172.19.126.54	TCP	68	80 → 57526 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
172.19.126.54	128.119.245.12	TCP	56	57526 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0
172.19.126.54	128.119.245.12	TCP	1516	57526 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=1460 [TCP segme
172.19.126.54	128.119.245.12	TCP	1516	57526 → 80 [PSH, ACK] Seq=1461 Ack=1 Win=64256 Len=1460 [T
172.19.126.54	128.119.245.12	TCP	1516	57526 → 80 [ACK] Seq=2921 Ack=1 Win=64256 Len=1460 [TCP se
172.19.126.54	128.119.245.12	TCP	1516	57526 → 80 [PSH, ACK] Seq=4381 Ack=1 Win=64256 Len=1460 [T
172.19.126.54	128.119.245.12	TCP	1516	57526 → 80 [ACK] Seq=5841 Ack=1 Win=64256 Len=1460 [TCP se
172.19.126.54	128.119.245.12	TCP	1516	57526 → 80 [PSH, ACK] Seq=7301 Ack=1 Win=64256 Len=1460 [T
172.19.126.54	128.119.245.12	TCP	1516	57526 → 80 [ACK] Seq=8761 Ack=1 Win=64256 Len=1460 [TCP se
172.19.126.54	128.119.245.12	TCP	1516	57526 → 80 [PSH, ACK] Seq=10221 Ack=1 Win=64256 Len=1460 [
172.19.126.54	128.119.245.12	TCP	1516	57526 → 80 [ACK] Seq=11681 Ack=1 Win=64256 Len=1460 [TCP s
172.19.126.54	128.119.245.12	TCP	1516	57526 → 80 [PSH, ACK] Seq=13141 Ack=1 Win=64256 Len=1460 [
128.119.245.12	172.19.126.54	TCP	56	80 → 57526 [ACK] Seq=1 Ack=2921 Win=35072 Len=0
172.19.126.54	128.119.245.12	TCP	1516	57526 → 80 [ACK] Seq=14601 Ack=1 Win=64256 Len=1460 [TCP s
172.19.126.54	128.119.245.12	TCP	1516	57526 → 80 [PSH, ACK] Seq=16061 Ack=1 Win=64256 Len=1460 [
172.19.126.54	128.119.245.12	TCP	1516	57526 → 80 [ACK] Seq=17521 Ack=1 Win=64256 Len=1460 [TCP s
172.19.126.54	128.119.245.12	TCP	1516	57526 → 80 [PSH, ACK] Seq=18981 Ack=1 Win=64256 Len=1460 [
128.119.245.12	172.19.126.54	TCP	56	80 → 57526 [ACK] Seq=1 Ack=5841 Win=40960 Len=0
172.19.126.54	128.119.245.12	TCP	1516	57526 → 80 [ACK] Seq=20441 Ack=1 Win=64256 Len=1460 [TCP s
172.19.126.54	128.119.245.12	TCP	1516	57526 → 80 [PSH, ACK] Seq=21901 Ack=1 Win=64256 Len=1460 [
172.19.126.54	128.119.245.12	TCP	1516	57526 → 80 [ACK] Seq=23361 Ack=1 Win=64256 Len=1460 [TCP s
172.19.126.54	128.119.245.12	TCP	1516	57526 → 80 [PSH, ACK] Seq=24821 Ack=1 Win=64256 Len=1460 [

figure 7

Question-8

a) Window: 275 with Scaling Factor: 128
b) No it won't throttle the sender for the first five data carrying segment as the buffer space is not sufficiently small (e.g., is not close to zero or significantly smaller than the sender's congestion window)

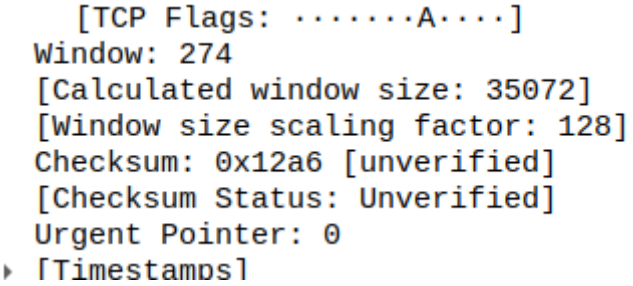


figure 8

Question-9

Its same accross all the Data Carrying Packets, Window: 502, with Window Scaling Factor: 128. It is the case because the sender is not receiving and payload packets from the reciever, its only being acknowledged for the packets its has send. So there won't be any case of throttling or congestion at sender by reciever as recievers ACK packets are of length 0.

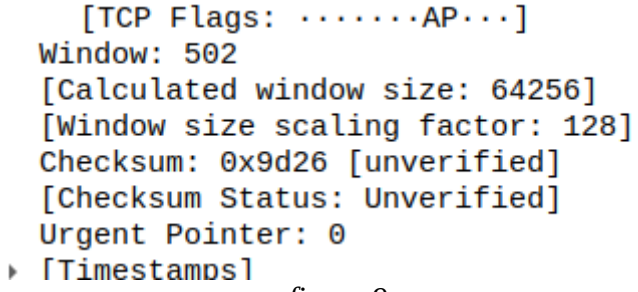


figure 9

Question-10

No there aren't any retransmitted segments in the trace file. For this conclusion I observed for non-duplicacy of acknowledge number in the TCP-ACK Packets from the reciever.

Question-11

a) For the First 10 data-carrying Segemnts it Acknowledged 5 packets where the bytes acknowledged are 2920,1460,1460,1460 giving an average of 1752 bytes
b) No, as it showed comulativce acknowldegements

11	3.165035401	17:12:16.866796909	172.19.126.54	128.119.245.12
12	3.165439916	17:12:16.867201424	172.19.126.54	128.119.245.12
13	3.165456311	17:12:16.867217819	172.19.126.54	128.119.245.12
14	3.165587773	17:12:16.867349281	172.19.126.54	128.119.245.12
15	3.165593503	17:12:16.867355011	172.19.126.54	128.119.245.12
16	3.165736668	17:12:16.867498176	172.19.126.54	128.119.245.12
17	3.165750668	17:12:16.867512176	172.19.126.54	128.119.245.12
18	3.165880495	17:12:16.867642003	172.19.126.54	128.119.245.12
19	3.165887144	17:12:16.867648652	172.19.126.54	128.119.245.12
20	3.165998327	17:12:16.867759835	172.19.126.54	128.119.245.12
21	3.166004159	17:12:16.867765667	172.19.126.54	128.119.245.12
22	3.420139232	17:12:17.121900740	128.119.245.12	172.19.126.54
23	3.420202445	17:12:17.121963953	172.19.126.54	128.119.245.12
24	3.420217821	17:12:17.121979329	172.19.126.54	128.119.245.12
25	3.420368166	17:12:17.122129674	172.19.126.54	128.119.245.12
26	3.420375997	17:12:17.122137505	172.19.126.54	128.119.245.12
27	3.420506283	17:12:17.122267791	128.119.245.12	172.19.126.54
28	3.420543114	17:12:17.122304622	172.19.126.54	128.119.245.12
29	3.420550381	17:12:17.122311889	172.19.126.54	128.119.245.12
30	3.420680922	17:12:17.122442430	172.19.126.54	128.119.245.12
31	3.420689152	17:12:17.122450660	172.19.126.54	128.119.245.12

```
  0101 .... = Header Length: 20 bytes (5)
▾ Flags: 0x010 (ACK)
    000. .... .... = Reserved: Not set
    ...0 .... .... = Accurate ECN: Not set
    .... 0... .... = Congestion Window Reduced: Not set
    .... .0.. .... = ECN-Echo: Not set
    .... ..0. .... = Urgent: Not set
    .... ...1 .... = Acknowledgment: Set
    .... .... 0... = Push: Not set
    .... .... .0.. = Reset: Not set
    .... .... ..0. = Syn: Not set
    .... .... ...0 = Fin: Not set
    [TCP Flags: ·······A····]
  Window: 274
  [Calculated window size: 35072]
  [Window size scaling factor: 128]
  Checksum: 0x12a6 [unverified]
  [Checksum Status: Unverified]
  Urgent Pointer: 0
▸ [Timestamps]
▾ [SEQ/ACK analysis]
    [This is an ACK to the segment in frame: 13]
    [The RTT to ACK the segment was: 0.254682921 seconds]
    [iRTT: 0.245191772 seconds]
```

figure 11: Its the first ACK from the reciever and its showing comutativeness which can be observed from its ACK feild which shows that the reciever is expecting packet 3

Question-12

Total Message Size: 1,52,138 Bytes
First Packet Time-Stamp: 2.919843629...(a)
Last Packet Time-Stamp: 4.182574266...(b)

Throughput = Transfer Size/ Transfer Time
= 1,52,138/((b)-(a))
= 1,52,138/1.262730637
= **120483.336304764 bytes/sec**

	Time	Time Stamp	Source	Destination
8	2.919843629	17:12:16.621605137	172.19.126.54	128.119.245.12
10	3.164974229	17:12:16.866735737	128.119.245.12	172.19.126.54
11	3.165035401	17:12:16.866796909	172.19.126.54	128.119.245.12
170	4.179798151	17:12:17.881559659	128.119.245.12	172.19.126.54
171	4.179798232	17:12:17.881559740	128.119.245.12	172.19.126.54
172	4.182524114	17:12:17.884285622	128.119.245.12	172.19.126.54
173	4.182574266	17:12:17.884335774	172.19.126.54	128.119.245.12

figure 12

PART-B

Question-1

Source IP Address: 172.19.126.253
Source Port Number: 54570

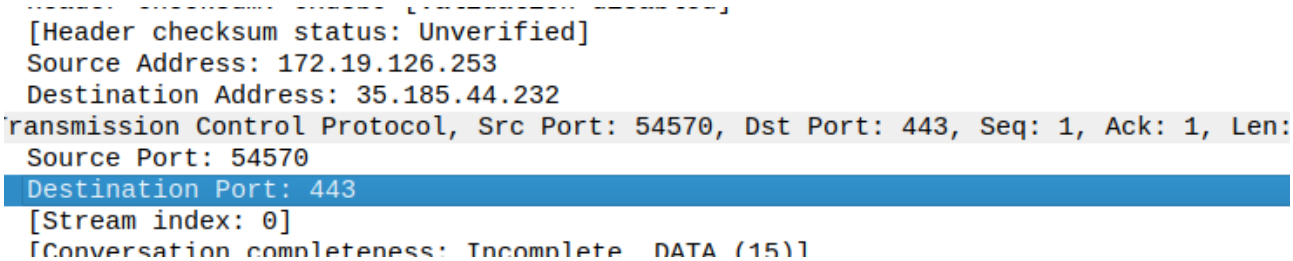


figure 1

Question-2

Destination IP Address: 35.185.44.232
Destination Port Number: 443
Reference: *figure 1*

Question-3

a) Sequence Number: 2833411430 [*figure 2.2*]
b) It has set its Flag Feild value as 0x0002 [SYN] that sets the SYN Flag bit which will ultimately make it identified as SYN Segment [*figure 2.2*]
c)Yes, it will allow reciever to selectively acknowledge the out of order pakages and specify which segments are missing as it allows SACK, it can be inspected by observing the TCP Options of TCP headers of Packets Exhnaged during HandShake i.e. Kind: SACK Permitted (4) as of the current example. [*figure 2.1*]

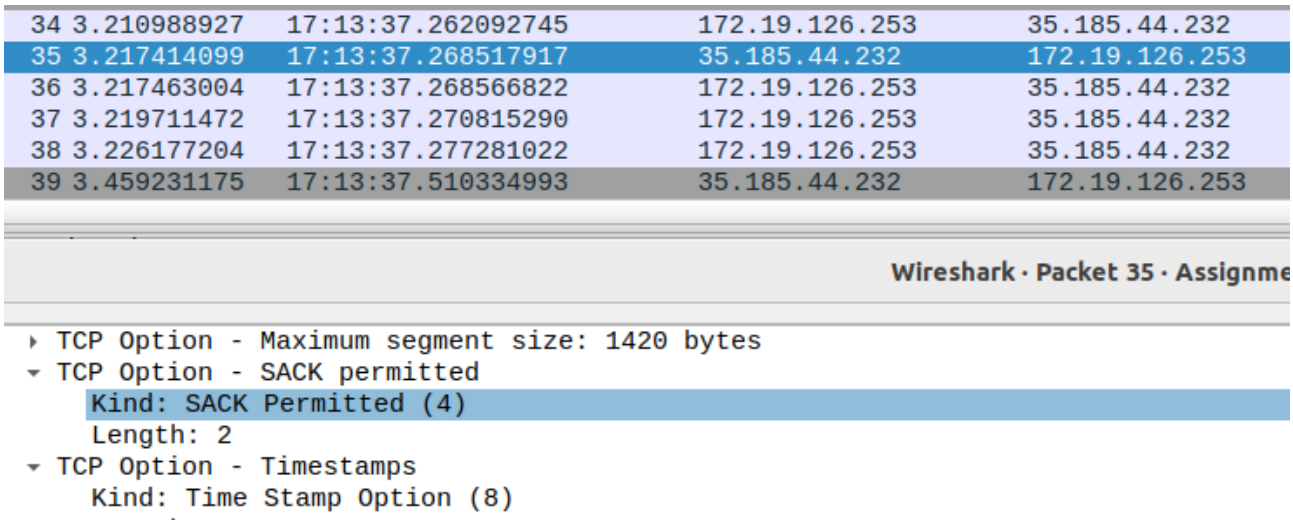


figure 2.2 IITH@CSE to Host

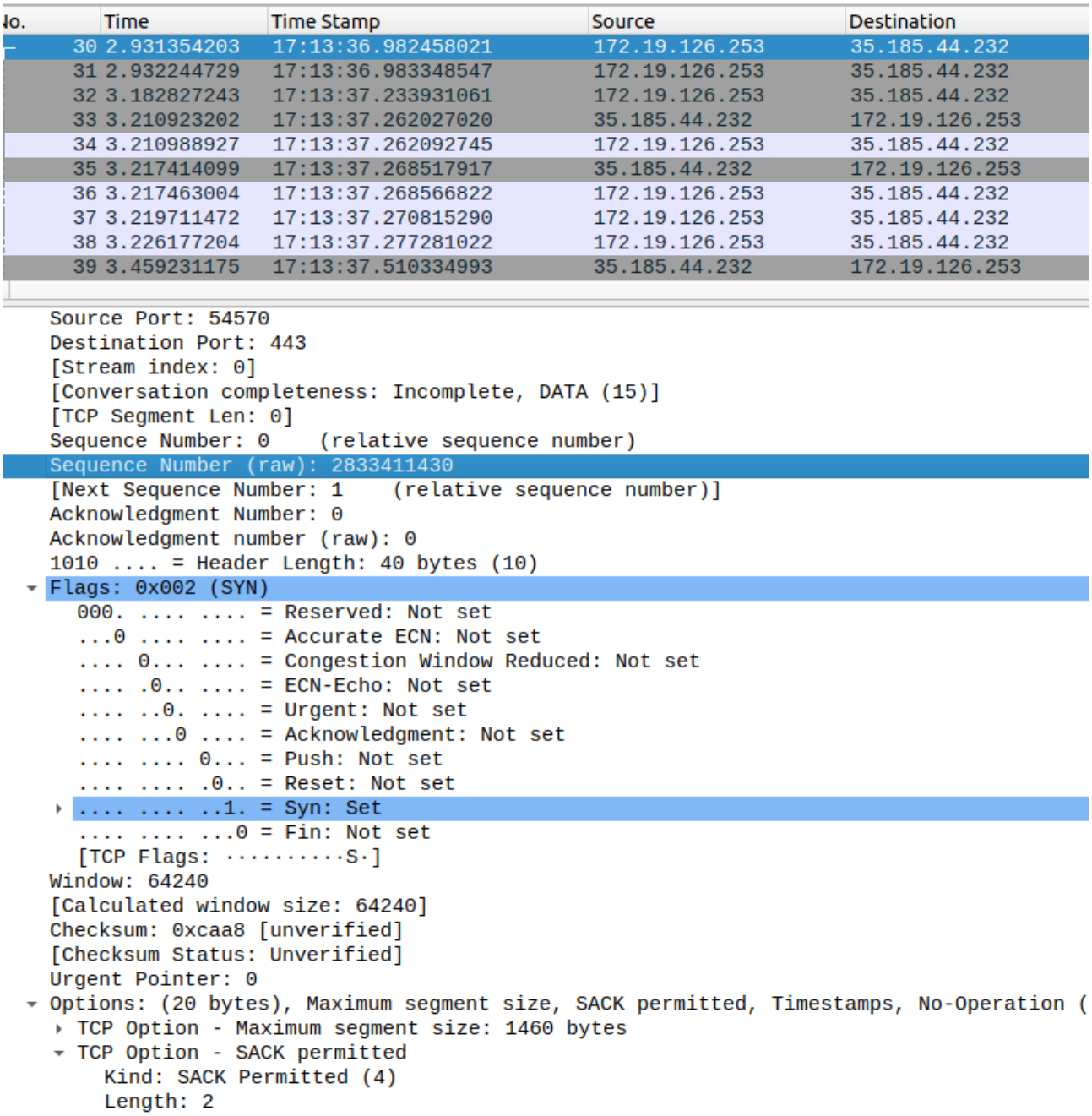


figure 2.2 Host to IIT@CSE

Question-4

a) Sequence Number: 1186615147
b) Flag: 0x0012 (SYN, ACK)
c) Acknowledgement Number: 2833411431
d) As gaia.cs.umass.edu recieved a TCP Connection Request with a Squence Number 2833411430 and Length 0, from that it can understand that the next packet it must expect would be next Byte in Sequence Number. So it sends that as ACK Number.

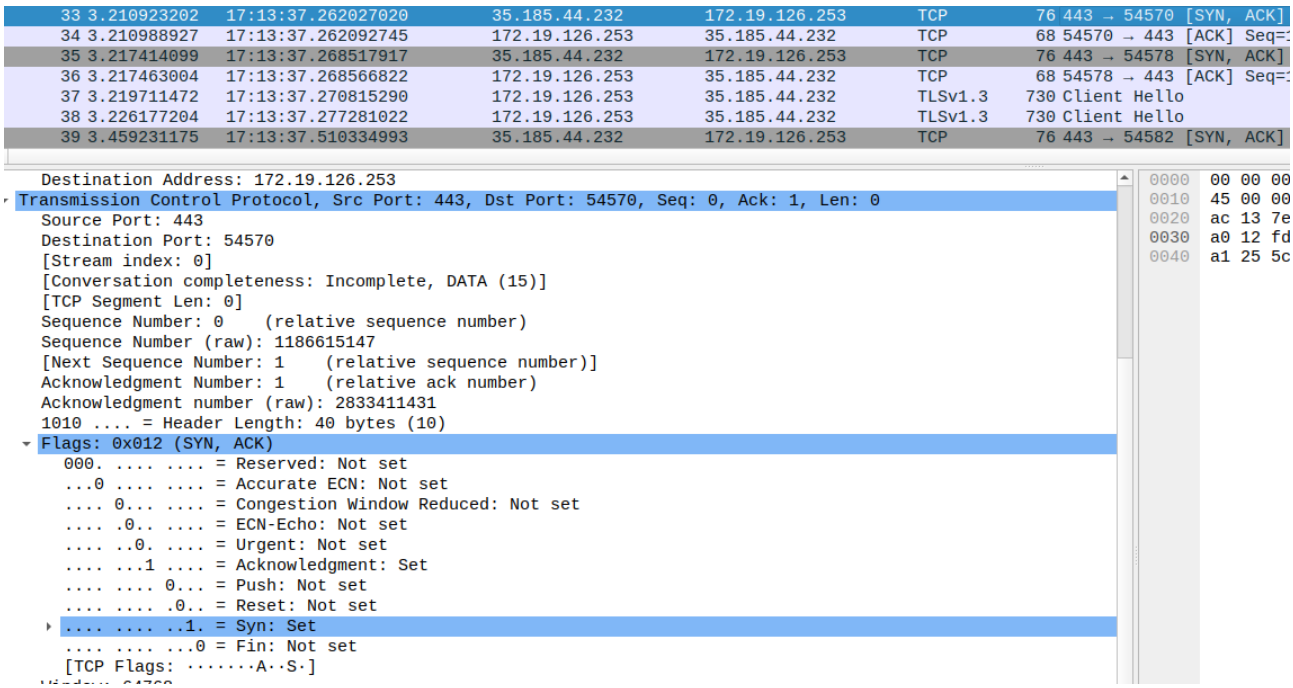


Figure 4

Question-5

a) Sequence Number: 1186615148
b) Payload: 5168
c) No, in this case if considering Web Objects from IITH@CSE

s

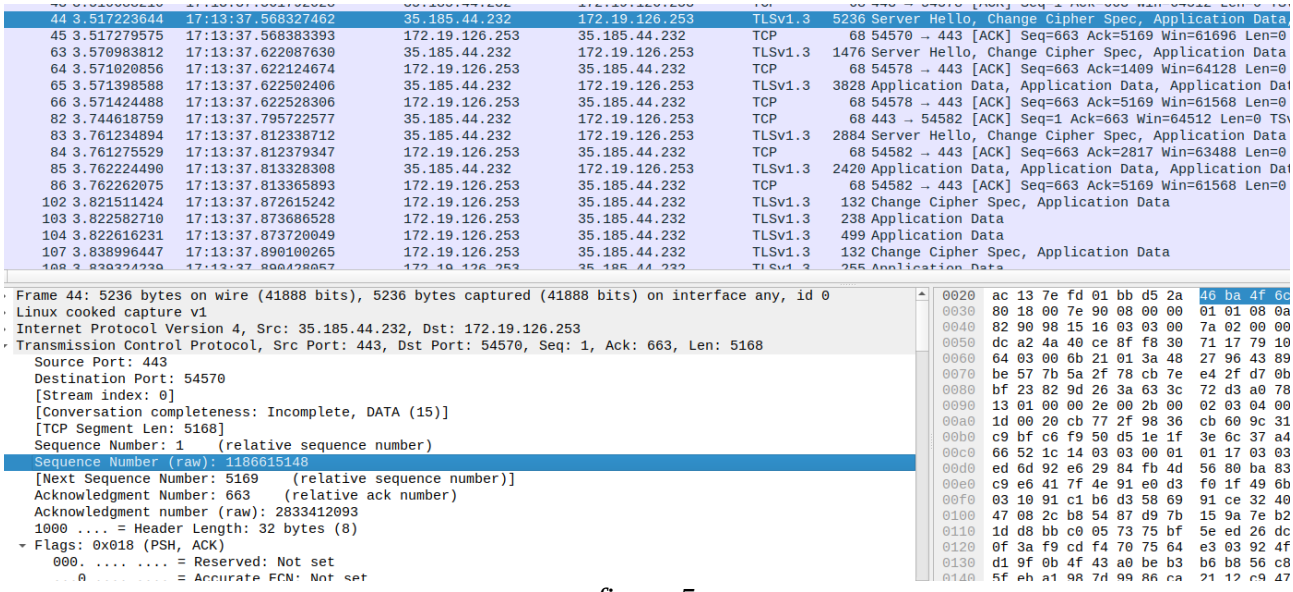


figure 5

Question-6

a) 17:13:38.208885094 (or) 4.157781276 (s) after the TCP SYN Segment was sent
b) 17:13:38.208897969
c) 0.000012875

d) Here for the second packet my host didn't sent the ACK to IITH@CSE rather it was a comutative in ACK of Packet 3. So due to this insufficient data I wont't be able to calculate EstimatedRRT.

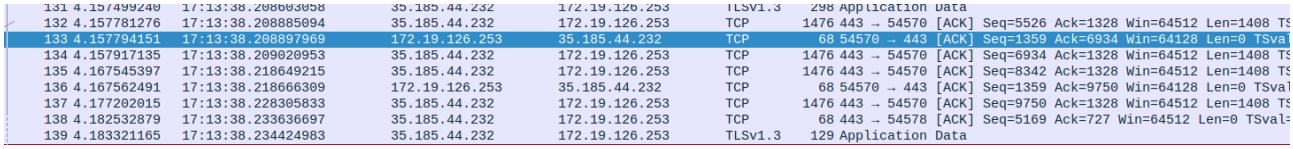


figure 6.1

Question-7

Each one of four has Total: 1408 (Talking just at TCP Level)

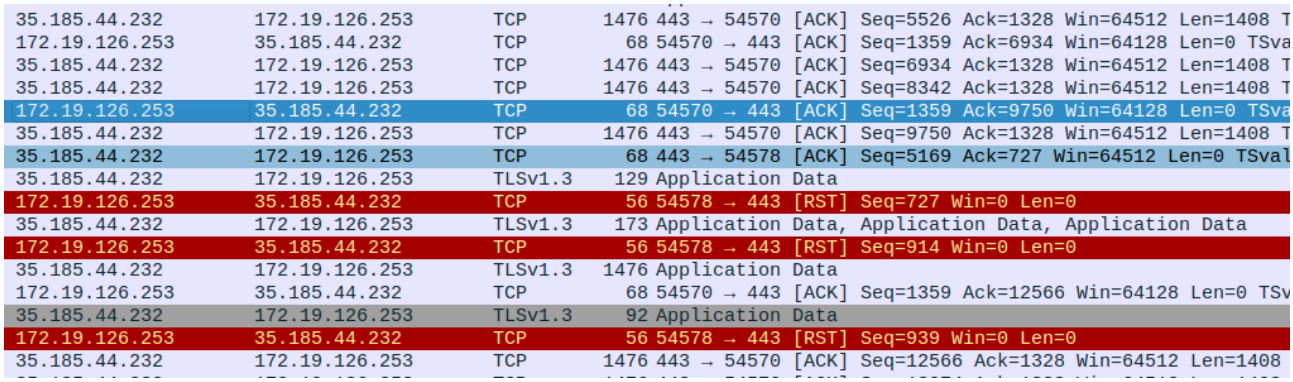


figure 7

Question-8

a) Window: 501 with Scaling Factor: 128
b) No it won't throttle the sender for the first five data carrying segment as the *buffer space is not sufficiently small (e.g., is not close to zero or significantly smaller than the sender's congestion window)*

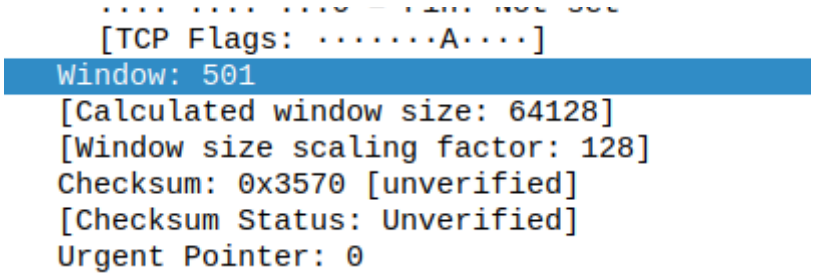


figure 8

Question-9

Its same accross all the Data Carrying Packets, Window: 502, with Window Scaling Factor: 128. It is the case because the server is not receiving and payload packets from the sender, its only being acknowledged for the packets its has send. So there won't be any case of throttling or congestion at sender by reciever as recievers ACK packets are of length 0.

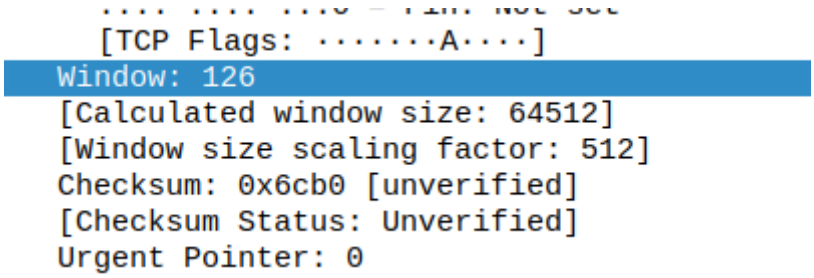


figure 9

Question-10

No there aren't any retransmitted segments in the trace file. For this conclusion I observed for non-duplicacy of acknowledge number in the TCP-ACK Packets from the reciever.

Question-11

a) 1408 Bytes, tho some were comulative ACK
b) No, as it showed comulativce acknowldegements

Question-12

Throughput = Tranfer size/ Transfer Time

Transfer size = Sequence Number of Last Data carrying segment + Its Length – Sequence Number of First Data Carrying segment
= 1217808 + 2186 – 5526
= 1214468 Bytes

First Packet Time-Stamp: 2.931354203...(a)
Last Packet Time-Stamp: 7.002261345...(b)
Transfer Time = (b) – (a)
= 4.070907142 (s)

So,

Throughput = 1214468/ 4.070907142
= **298328.593022965 bytes/sec**