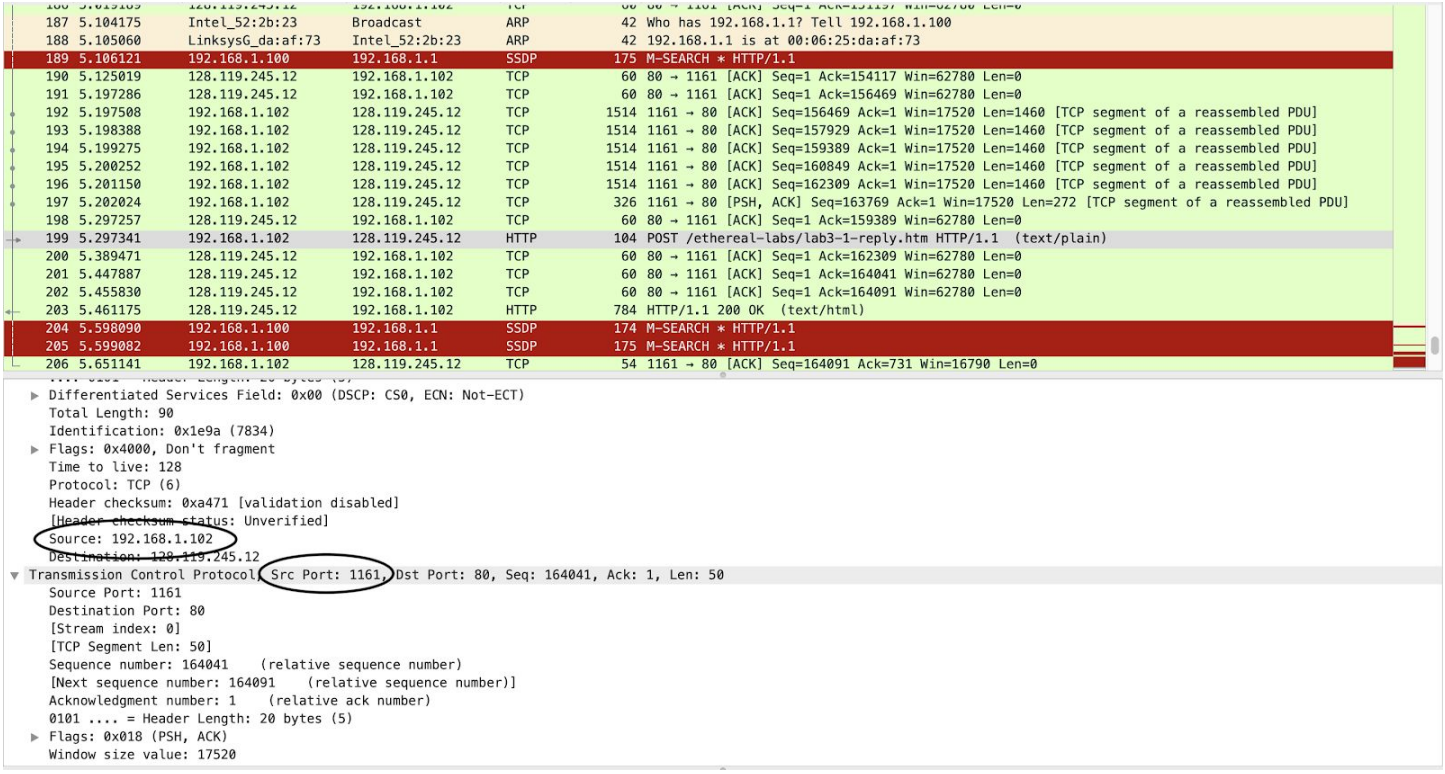
1.

What is the IP address and TCP port number used by the client computer (source) that is transferring the file to gaia.cs.umass.edu? To answer this question, it's probably easiest to select an HTTP message and explore the details of the TCP packet used to carry this HTTP message, using the "details of the selected packet header window" (refer to Figure 2 in the "Getting Started with Wireshark" Lab if you're uncertain about the Wireshark windows.



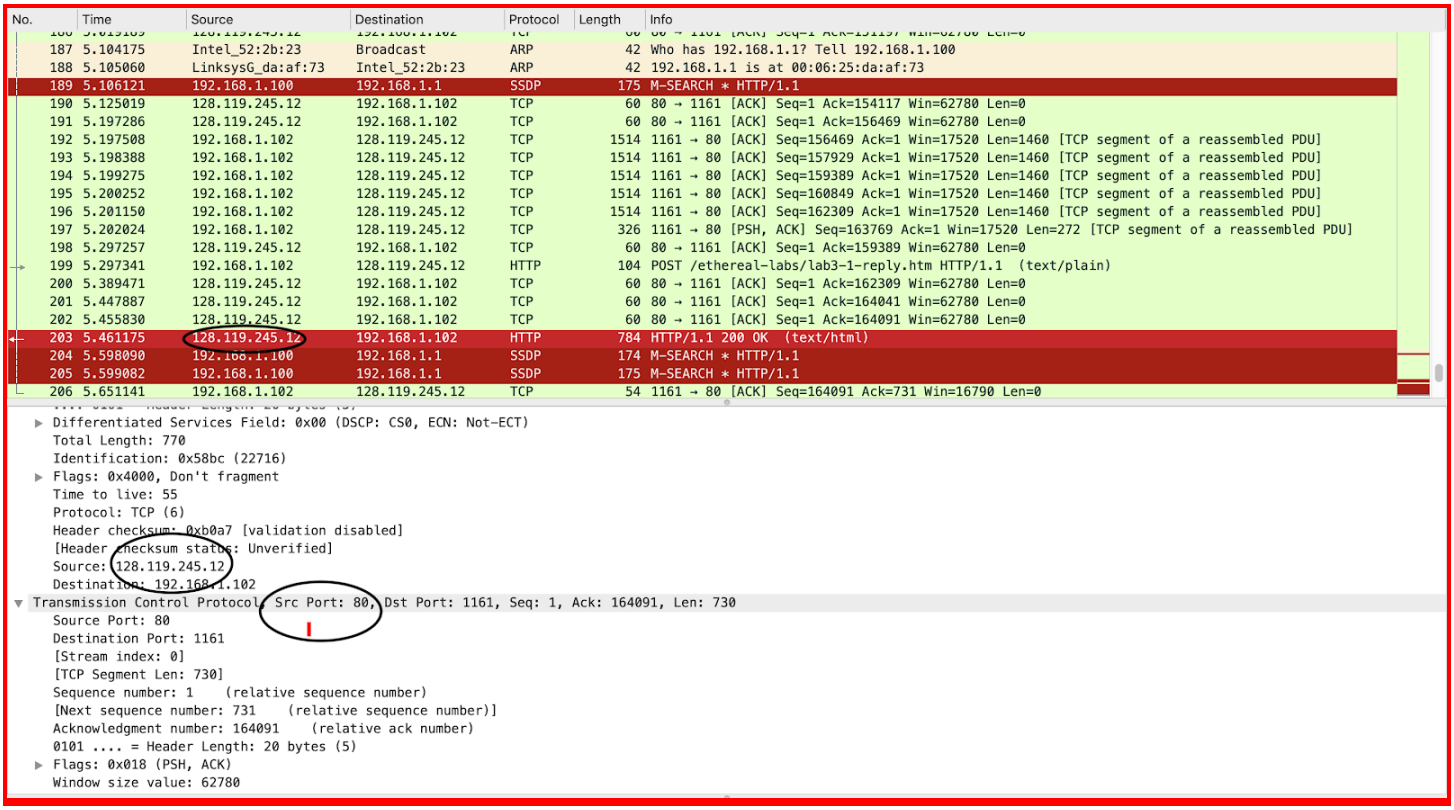
Printed Packet:

No. Time Source Destination Protocol Length Info
199 5.297341 192.168.1.102 128.119.245.12 HTTP 104 POST /

ethereal-labs/lab3-1-reply.htm HTTP/1.1 (text/plain)
Frame 199: 104 bytes on wire (832 bits), 104 bytes captured (832 bits)
Ethernet II, Src: Actionte_8a:70:1a (00:20:e0:8a:70:1a), Dst: LinksysG_da:af:73 (00:06:25:da:af:73)
Internet Protocol Version 4, Src: 192.168.1.102, Dst: 128.119.245.12
Transmission Control Protocol, Src Port: 1161, Dst Port: 80, Seq: 164041, Ack: 1, Len: 50
[122 Reassembled TCP Segments (164090 bytes): #4(565), #5(1460), #7(1460), #8(1460), #10(1460),

#11(1460), #13(1147), #18(1460), #19(1460), #20(1460), #21(1460), #22(1460), #23(892), #30(1460), #31(1460), #32(1460), #33(1460), #34(1460), #3]
Hypertext Transfer Protocol
MIME Multipart Media Encapsulation, Type: multipart/form-data, Boundary: "--------------------------265001916915724"

2.
What is the IP address of gaia.cs.umass.edu? On what port number is it sending and receiving TCP segments for this connection?



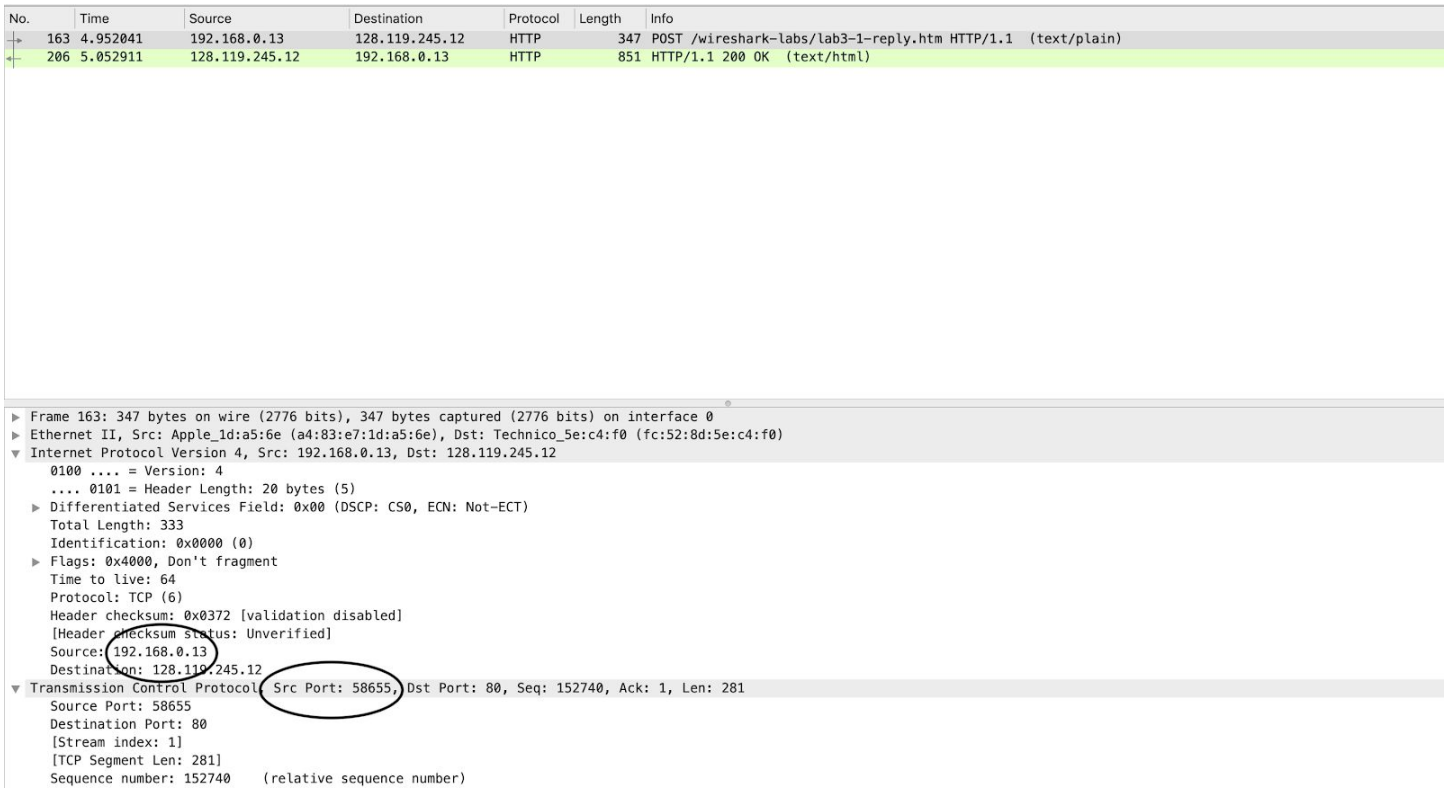
Packet Print

No.	Time	Source	Destination	Protocol	Length	Info
203	5.461175	128.119.245.12	192.168.1.102	HTTP	784	HTTP/1.1 200

OK (text/html)
Frame 203: 784 bytes on wire (6272 bits), 784 bytes captured (6272 bits)

Ethernet II, Src: LinksysG_da:af:73 (00:06:25:da:af:73), Dst: Actionte_8a:70:1a (00:20:e0:8a:70:1a)
Internet Protocol Version 4, Src: 128.119.245.12, Dst: 192.168.1.102
Transmission Control Protocol, Src Port: 80, Dst Port: 1161, Seq: 1, Ack: 164091, Len: 730
Hypertext Transfer Protocol
Line-based text data: text/html (11 lines)

3.
What is the IP address and TCP port number used by your client computer (source) to transfer the file to gaia.cs.umass.ed



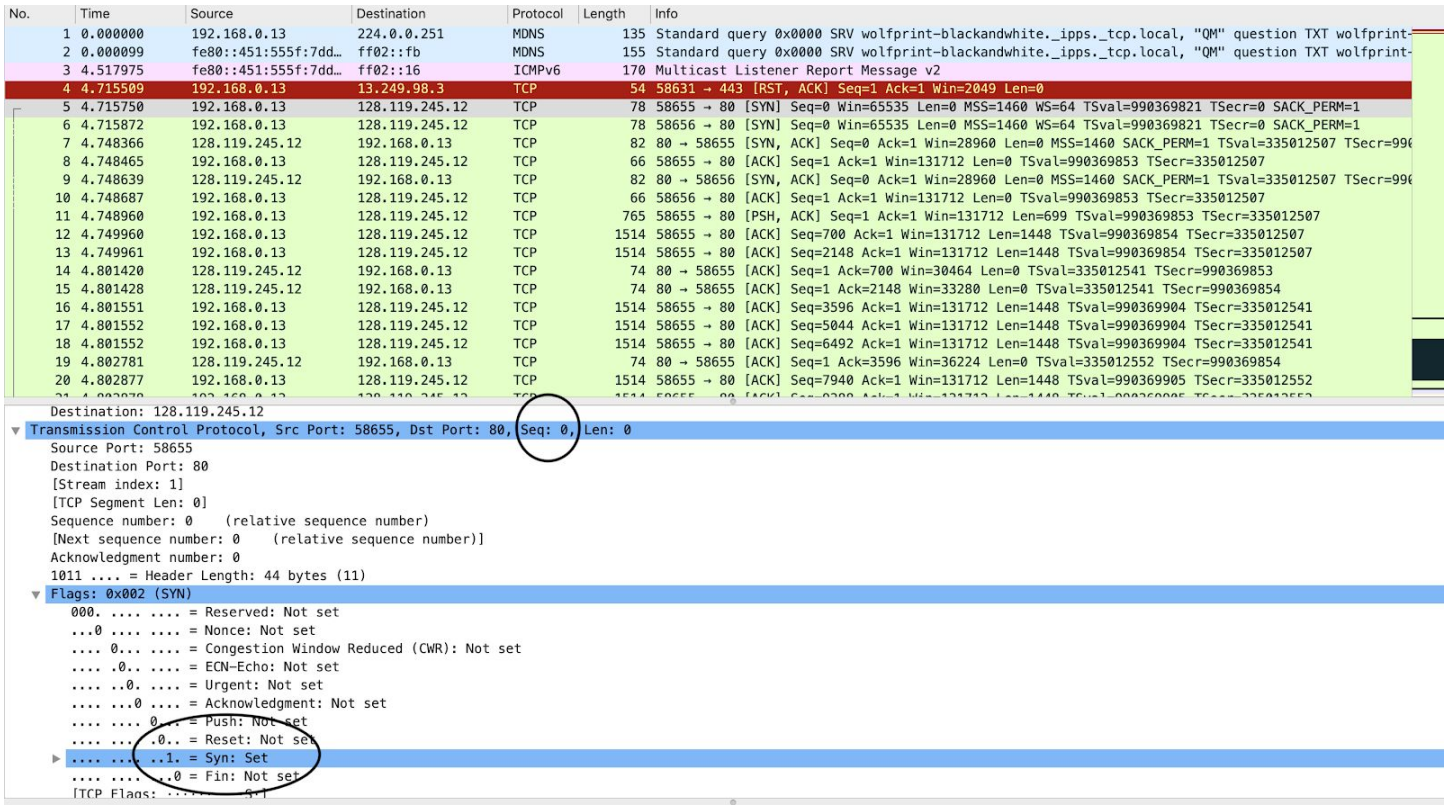
The source IP address is 192.168.0.13
The source port is 58655

Print Packet

No. Time Source Destination Protocol Length Info
163 4.952041 192.168.0.13 128.119.245.12 HTTP 347 POST /

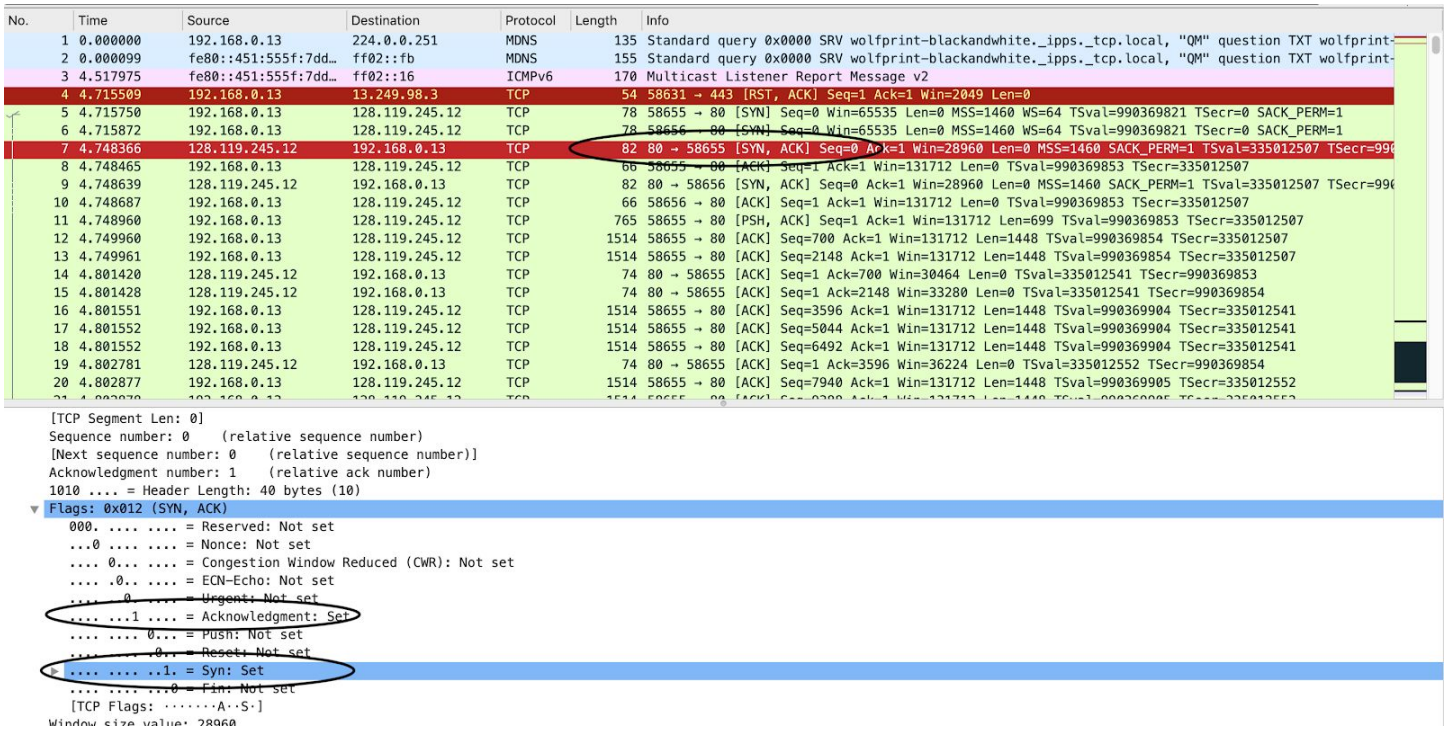
wireshark-labs/lab3-1-reply.htm HTTP/1.1 (text/plain)
Frame 163: 347 bytes on wire (2776 bits), 347 bytes captured (2776 bits) on interface 0
Ethernet II, Src: Apple_1d:a5:6e (a4:83:e7:1d:a5:6e), Dst: Technico_5e:c4:f0 (fc:52:8d:5e:c4:f0)
Internet Protocol Version 4, Src: 192.168.0.13, Dst: 128.119.245.12
Transmission Control Protocol, Src Port: 58655, Dst Port: 80, Seq: 152740, Ack: 1, Len: 281
[107 Reassembled TCP Segments (153020 bytes): #11(699), #12(1448), #13(1448), #16(1448), #17(1448), #18(1448), #20(1448), #21(1448), #23(1448), #24(1448), #27(1448), #28(1448), #29(1448), #30(1448), #31(1448), #32(1448), #34(1448), #35(1448)]
Hypertext Transfer Protocol
MIME Multipart Media Encapsulation, Type: multipart/form-data, Boundary: "----WebKitFormBoundarykFfvTdSCjsoHX2Ub"

4.
What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client computer and gaia.cs.umass.edu? What is it in the segment that identifies the segment as a SYN segment?



The sequence number of the TCP SYN segment is 0 as it is used to initiate the TCP connection between the client computer and server. In the Flags section, the Syn flag is set to 1 that indicates, this segment is a SYN segment.

5.
What is the sequence number of the SYN ACK segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN? What is the value of the Acknowledgement field in the SYNACK segment? How did gaia.cs.umass.edu determine that value? What is it in the segment that identifies the segment as a SYNACK segment?



Printed Packet

/var/folders/gz/wqm9l2y918g3kj8tl3zd0cf80000gn/T//wireshark_Wi-Fi_20191002235443_g9rBzs.pcapng 211 total packets, 211 shown

No. Time Source Destination Protocol Length Info
7 4.748366 128.119.245.12 192.168.0.13 TCP 82 80 → 58655

[SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=335012507 TSecr=990369821
WS=128
Frame 7: 82 bytes on wire (656 bits), 82 bytes captured (656 bits) on interface 0
Ethernet II, Src: Technico_5e:c4:f0 (fc:52:8d:5e:c4:f0), Dst: Apple_1d:a5:6e (a4:83:e7:1d:a5:6e)
Internet Protocol Version 4, Src: 128.119.245.12, Dst: 192.168.0.13

0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 60
Identification: 0x0000 (0)
Flags: 0x4000, Don't fragment
Time to live: 50
Protocol: TCP (6)
Header checksum: 0x1283 [validation disabled]
[Header checksum status: Unverified]
Source: 128.119.245.12
Destination: 192.168.0.13

Transmission Control Protocol, Src Port: 80, Dst Port: 58655, Seq: 0, Ack: 1, Len: 0
Source Port: 80

Destination Port: 58655
[Stream index: 1]
[TCP Segment Len: 0]
Sequence number: 0 (relative sequence number)
[Next sequence number: 0 (relative sequence number)]
Acknowledgment number: 1 (relative ack number)

1010 .... = Header Length: 40 bytes (10)
Flags: 0x012 (SYN, ACK)

000. .... = Reserved: Not set
...0 .... = Nonce: Not set
.... 0... .... = Congestion Window Reduced (CWR): Not set
.... .0.. .... = ECN-Echo: Not set
.... ..0. .... = Urgent: Not set
....1 .... = Acknowledgment: Set
.... 0... = Push: Not set
.... .0.. = Reset: Not set
....1. = Syn: Set
....0 = Fin: Not set
[TCP Flags: ·····A··S·]

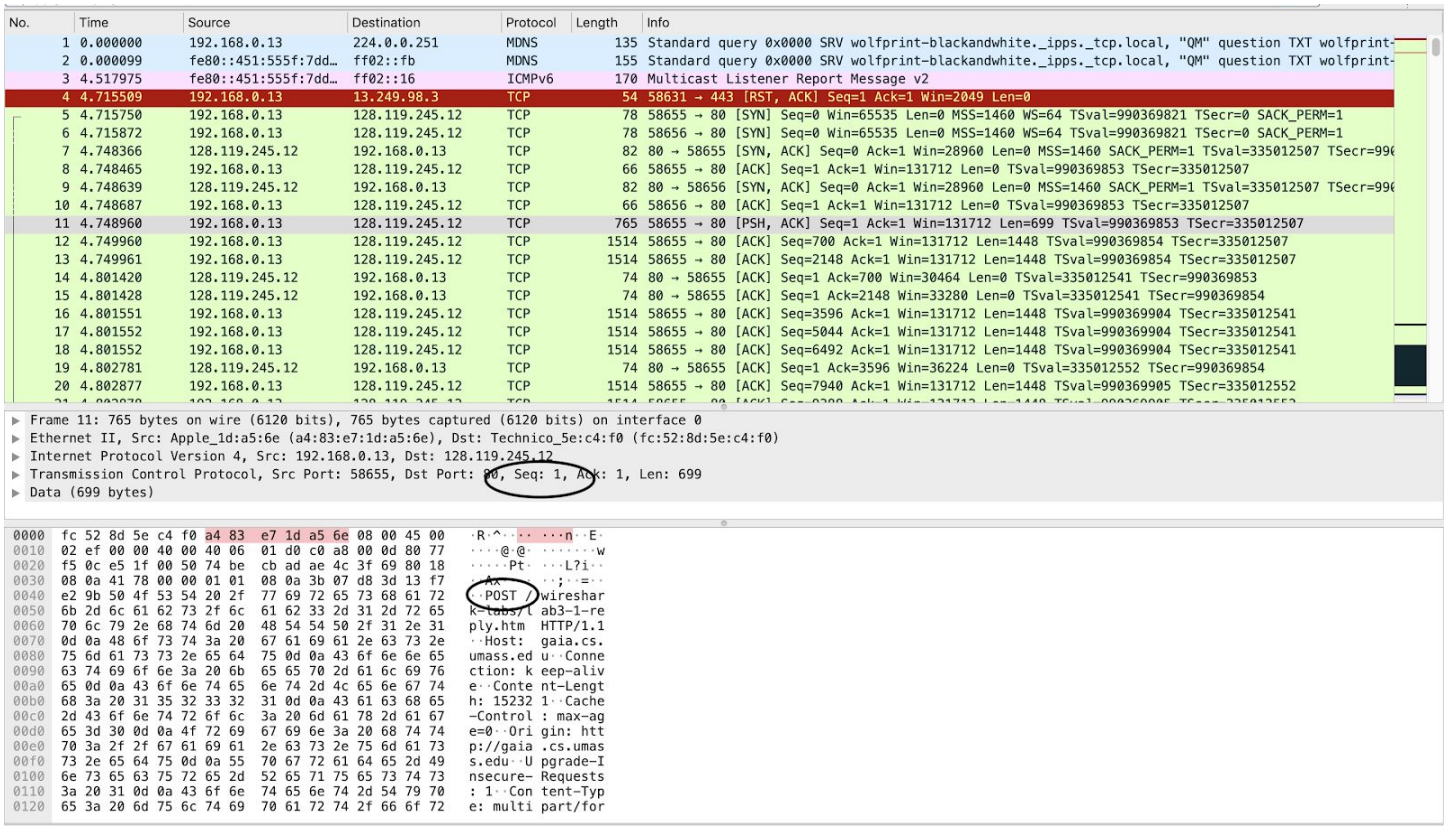
Window size value: 28960
[Calculated window size: 28960]
Checksum: 0x834d [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0

Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP),

Window scale
[SEQ/ACK analysis]
[Timestamps]

> the sequence number of the SYNACK segment sent by server to the client in reply to the SYN is 0.
>The value of the acknowledgment field in the SYNACK segment is 1
>The server adds 1 to the sequence number of SYN segment form the client as this will be the next sequence number that it is expecting.
>A segment will be identified as a SYNACK segment if both SYN flag and Acknowledgement in the segment are set to 1.

6.
What is the sequence number of the TCP segment containing the HTTP POST command? Note that in order to find the POST command, you'll need to dig into the packet content field at the bottom of the Wireshark window, looking for a segment with a "POST" within its DATA field.



The sequence number in the Post command will is 1.

Printed Packet:

```
No.     Time           Source                Destination           Protocol Length Info
     11 4.748960       192.168.0.13          128.119.245.12        TCP      765    58655 → 80
[PSH, ACK] Seq=1 Ack=1 Win=131712 Len=699 TSval=990369853 TSecr=335012507
Frame 11: 765 bytes on wire (6120 bits), 765 bytes captured (6120 bits) on interface 0
Ethernet II, Src: Apple_1d:a5:6e (a4:83:e7:1d:a5:6e), Dst: Technico_5e:c4:f0 (fc:52:8d:5e:c4:f0)
Internet Protocol Version 4, Src: 192.168.0.13, Dst: 128.119.245.12
Transmission Control Protocol, Src Port: 58655, Dst Port: 80, Seq: 1, Ack: 1, Len: 699
Data (699 bytes)
0000  50 4f 53 54 20 2f 77 69 72 65 73 68 61 72 6b 2d   POST /wireshark-
0010  6c 61 62 73 2f 6c 61 62 33 2d 31 2d 72 65 70 6c   labs/lab3-1-repl
0020  79 2e 68 74 6d 20 48 54 54 50 2f 31 2e 31 0d 0a   y.htm HTTP/1.1..
0030  48 6f 73 74 3a 20 67 61 69 61 2e 63 73 2e 75 6d   Host: gaia.cs.um
0040  61 73 73 2e 65 64 75 0d 0a 43 6f 6e 6e 65 63 74   ass.edu..Connect
0050  69 6f 6e 3a 20 6b 65 65 70 2d 61 6c 69 76 65 0d   ion: keep-alive.
0060  0a 43 6f 6e 74 65 6e 74 2d 4c 65 6e 67 74 68 3a   .Content-Length:
0070  20 31 35 32 33 32 31 0d 0a 43 61 63 68 65 2d 43    152321..Cache-C
0080  6f 6e 74 72 6f 6c 3a 20 6d 61 78 2d 61 67 65 3d   ontrol: max-age=
0090  30 0d 0a 4f 72 69 67 69 6e 3a 20 68 74 74 70 3a   0..Origin: http:
00a0  2f 2f 67 61 69 61 2e 63 73 2e 75 6d 61 73 73 2e   //gaia.cs.umass.
00b0  65 64 75 0d 0a 55 70 67 72 61 64 65 2d 49 6e 73   edu..Upgrade-Ins
00c0  65 63 75 72 65 2d 52 65 71 75 65 73 74 73 3a 20   ecure-Requests:
00d0  31 0d 0a 43 6f 6e 74 65 6e 74 2d 54 79 70 65 3a   1..Content-Type:
00e0  20 6d 75 6c 74 69 70 61 72 74 2f 66 6f 72 6d 2d    multipart/form-
00f0  64 61 74 61 3b 20 62 6f 75 6e 64 61 72 79 3d 2d   data; boundary=-
0100  2d 2d 2d 57 65 62 4b 69 74 46 6f 72 6d 42 6f 75   ---WebKitFormBou
0110  6e 64 61 72 79 6b 46 66 76 54 64 53 43 6a 73 6f   ndarykFfvTdSCjso
0120  48 58 32 55 62 0d 0a 55 73 65 72 2d 41 67 65 6e   HX2Ub..User-Agen
0130  74 3a 20 4d 6f 7a 69 6c 6c 61 2f 35 2e 30 20 28   t: Mozilla/5.0 (
0140  4d 61 63 69 6e 74 6f 73 68 3b 20 49 6e 74 65 6c   Macintosh; Intel
0150  20 4d 61 63 20 4f 53 20 58 20 31 30 5f 31 34 5f    Mac OS X 10_14_
0160  33 29 20 41 70 70 6c 65 57 65 62 4b 69 74 2f 35   3) AppleWebKit/5
0170  33 37 2e 33 36 20 28 4b 48 54 4d 4c 2c 20 6c 69   37.36 (KHTML, li
0180  6b 65 20 47 65 63 6b 6f 29 20 43 68 72 6f 6d 65   ke Gecko) Chrome
0190  2f 37 37 2e 30 2e 33 38 36 35 2e 39 30 20 53 61   /77.0.3865.90 Sa
01a0  66 61 72 69 2f 35 33 37 2e 33 36 0d 0a 41 63 63   fari/537.36..Acc
01b0  65 70 74 3a 20 74 65 78 74 2f 68 74 6d 6c 2c 61   ept: text/html,a
01c0  70 70 6c 69 63 61 74 69 6f 6e 2f 78 68 74 6d 6c   pplication/xhtml
01d0  2b 78 6d 6c 2c 61 70 70 6c 69 63 61 74 69 6f 6e   +xml,application
01e0  2f 78 6d 6c 3b 71 3d 30 2e 39 2c 69 6d 61 67 65   /xml;q=0.9,image
01f0  2f 77 65 62 70 2c 69 6d 61 67 65 2f 61 70 6e 67   /webp,image/apng
0200  2c 2a 2f 2a 3b 71 3d 30 2e 38 2c 61 70 70 6c 69   ,*/*;q=0.8,appli
0210  63 61 74 69 6f 6e 2f 73 69 67 6e 65 64 2d 65 78   cation/signed-ex
0220  63 68 61 6e 67 65 3b 76 3d 62 33 0d 0a 52 65 66   change;v=b3..Ref
0230  65 72 65 72 3a 20 68 74 74 70 3a 2f 2f 67 61 69   erer: http://gai
0240  61 2e 63 73 2e 75 6d 61 73 73 2e 65 64 75 2f 77   a.cs.umass.edu/w
0250  69 72 65 73 68 61 72 6b 2d 6c 61 62 73 2f 54 43   ireshark-labs/TC
0260  50 2d 77 69 72 65 73 68 61 72 6b 2d 66 69 6c 65   P-wireshark-file
0270  31 2e 68 74 6d 6c 0d 0a 41 63 63 65 70 74 2d 45   1.html..Accept-E
0280  6e 63 6f 64 69 6e 67 3a 20 67 7a 69 70 2c 20 64   ncoding: gzip, d
0290  65 66 6c 61 74 65 0d 0a 41 63 63 65 70 74 2d 4c   eflate..Accept-L
02a0  61 6e 67 75 61 67 65 3a 20 65 6e 2d 55 53 2c 65   anguage: en-US,e
02b0  6e 3b 71 3d 30 2e 39 0d 0a 0d 0a                  n;q=0.9....
```

7.

Consider the TCP segment containing the HTTP POST as the first segment in the TCP connection. What are the sequence numbers of the first six segments in the TCP connection (including the segment containing the HTTP POST)? At what time was each segment sent? When was the ACK for each segment received? Given the difference between when each TCP segment was sent, and when its acknowledgement was received, what is the RTT value for each of the six segments? What is the EstimatedRTT value (see Section 3.5.3, page 239 in text) after the receipt of each ACK? Assume that the value of the EstimatedRTT is equal to the

measured RTT for the first segment, and then is computed using the EstimatedRTT equation on page 239 for all subsequent segments.

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

The segments that are sending the data are :
11,12,13,16,17,18

The acknowledgments to these segments are:

14,15,19,22,25,26

The acknowledgment on line 25 acts as a cumulative ack for 17 and 18

The sequence number of the segment:

11 - 1

12 - 700

13 - 2184

16 - 3596

17 - 5044

18 - 6492

Recording the sending time for the segment and the receiving time for the acknowledgment:

	Sent Time	*Ack receive time*	*RTT*
Seg1	*4.748960*	*4.801420*	*0.05246*
Seg2	*4.749960*	*4.801428*	*0.051468*
Seg3	*4.749961*	*4.801551*	*0.05159*
Seg4	*4.801551*	*4.834084*	*0.032533*
Seg5	*4.801552*	*4.834511*	*0.032959*

Seg6	*4.801552*	*4.834511*	*0.032959*

EstimatedRTT = (1 – alpha) • EstimatedRTT + alpha • SampleRTT

Alpha = 0.125

EstimatedRTT = 0.875 * EstimatedRTT + 0.125 * SampleRTT

Estimated RTT after receipt of ACK1 that is for seg 1

Estimated RTT = RTT after segment 1 = *0.05246 seconds*

Estimated RTT after receipt of ACK 2 that is for seg 2

Estimated RTT=0.875 * *0.05246* + 0.125 **0.051468 = 0.052336 seconds*

Estimated RTT after receipt of ACK 3 that is for seg 3

Estimated RTT=0.875 * *0.052336* + 0.125 **0.05159= 0.05224275 seconds*

Estimated RTT after receipt of ACK 4 that is for seg 4

Estimated RTT=0.875 **0.05224275* + 0.125 * *0.032533* = 0.04977903125 seconds

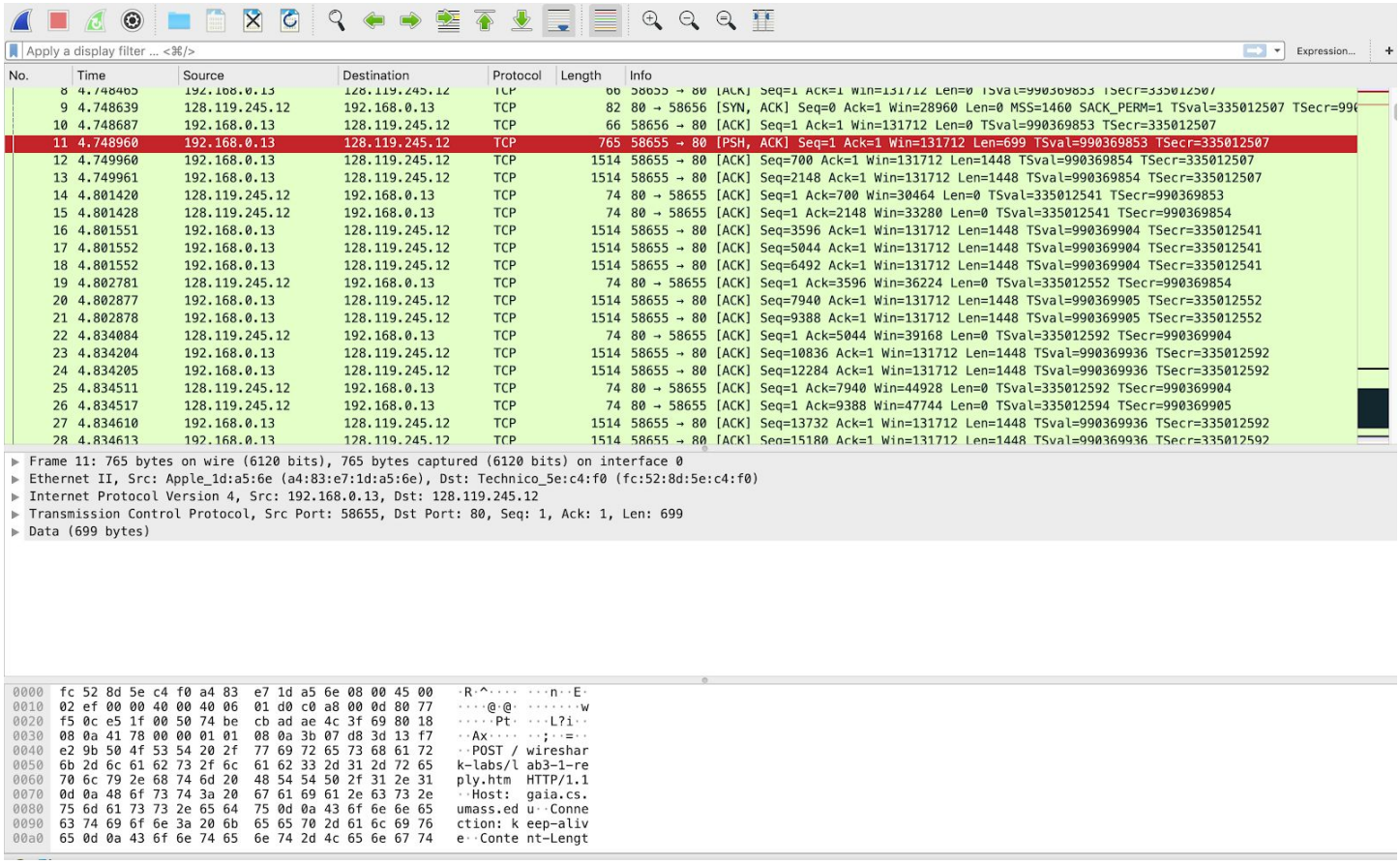
Estimated RTT after receipt of ACK 5 that is for seg 5

Estimated RTT=0.875 * 0.04977903125 + 0.125 * *0.032959 = 0.04767652734 seconds*

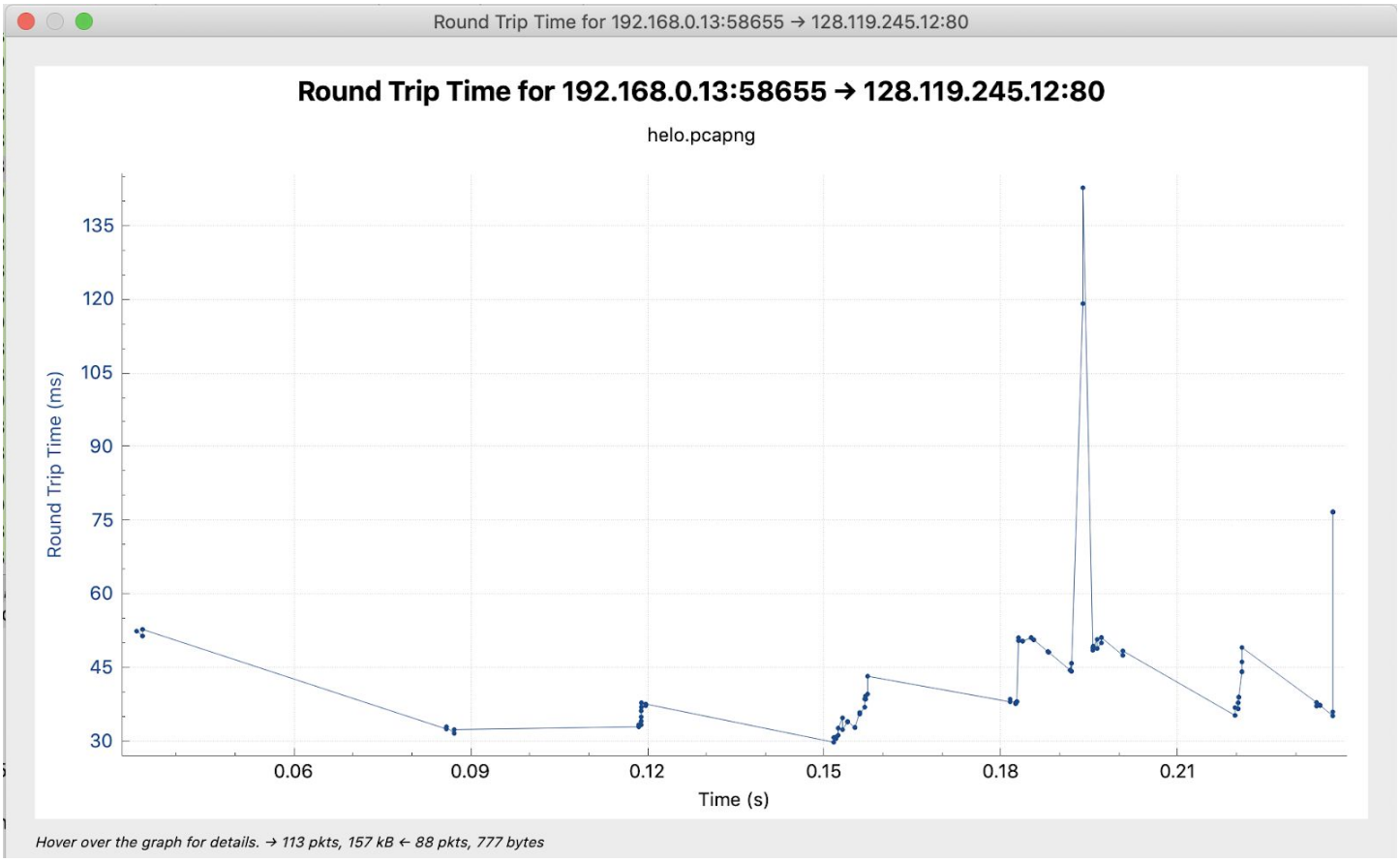
Estimated RTT after receipt of ACK 6 that is for seg 6

Estimated RTT=0.875 * *0.04767652734* + 0.125 * *0.032959 = 0.04583683642 seconds*

You can see all these packets in the below packet capture :



Graph for RTT



8.

What is the length of each of the first six TCP segments?

1-The length of the first TCP segment is 699 Bytes

2-1448 Bytes

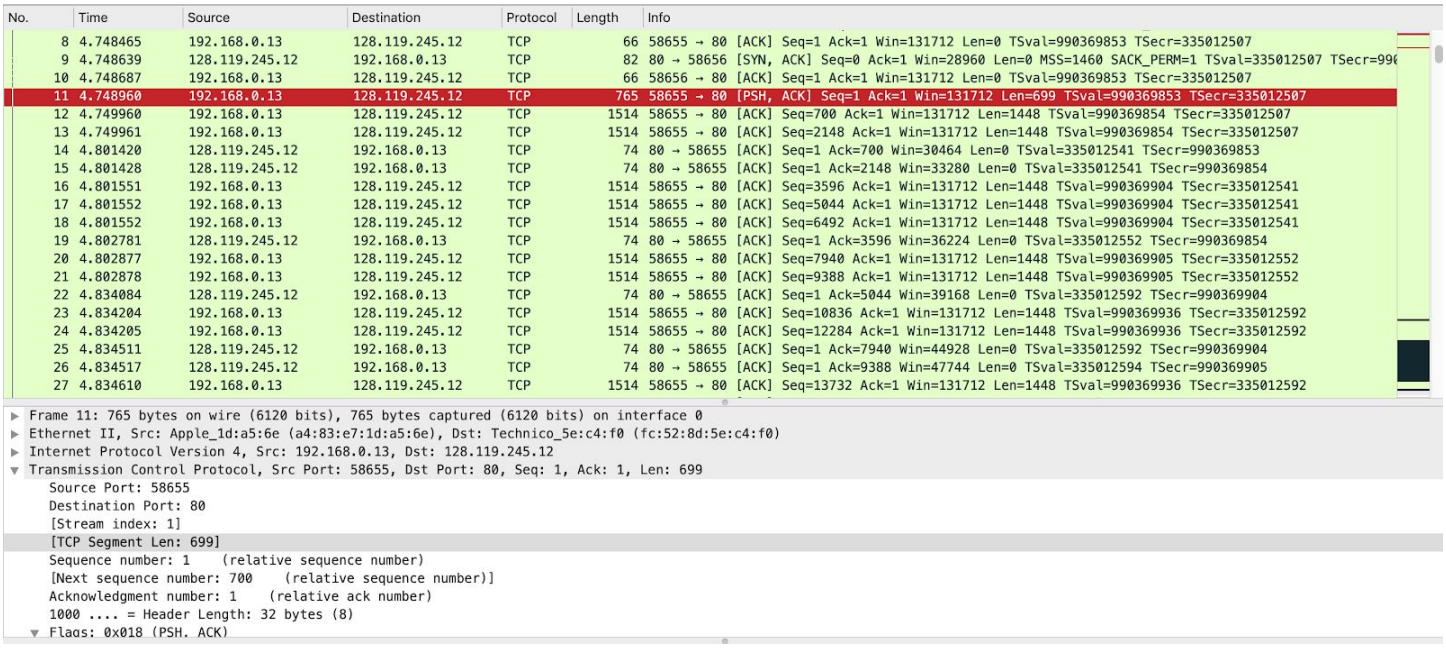
3-1448 Bytes

4-1448 Bytes

5-1448 Bytes

6-1448 Bytes

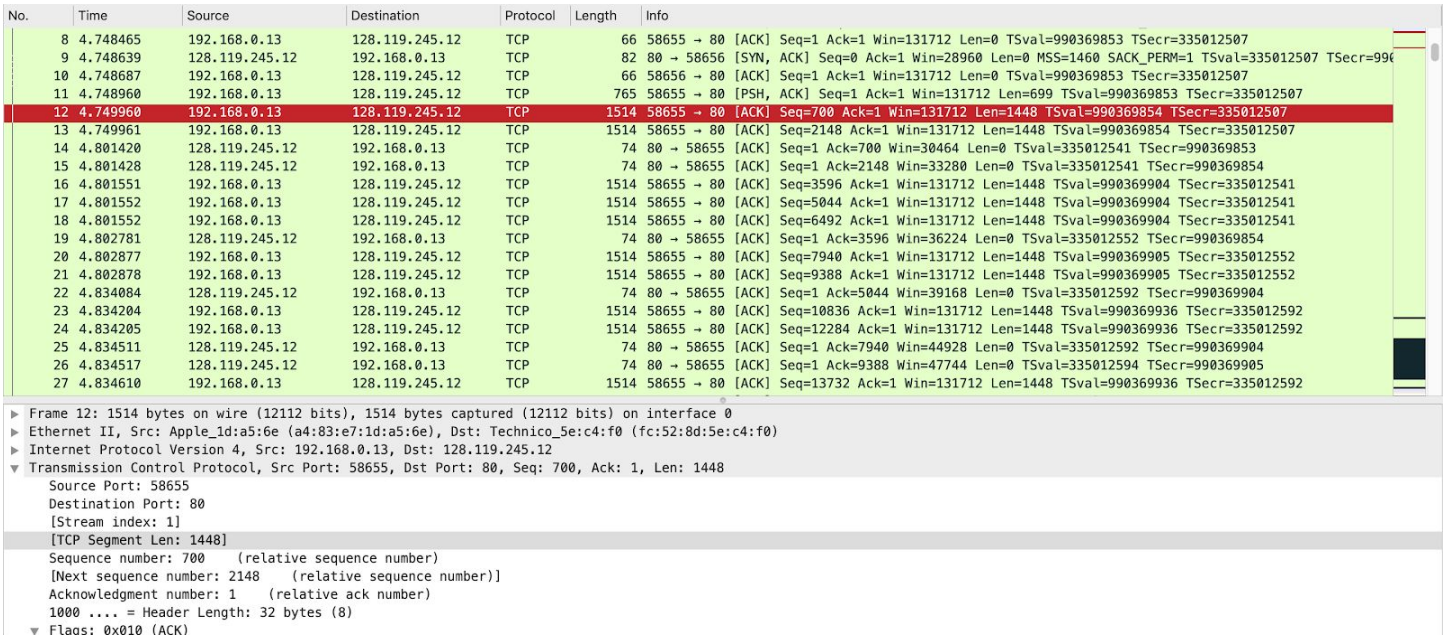
Packet 1



No.	Time	Source	Destination	Protocol	Length	Info
8	4.748465	192.168.0.13	128.119.245.12	TCP	66	58655 → 80 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=990369853 TSecr=335012507
9	4.748639	128.119.245.12	192.168.0.13	TCP	82	80 → 58656 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=335012507 TSecr=990
10	4.748687	192.168.0.13	128.119.245.12	TCP	66	58656 → 80 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=990369853 TSecr=335012507
11	4.748960	192.168.0.13	128.119.245.12	TCP	765	58655 → 80 [PSH, ACK] Seq=1 Ack=1 Win=131712 Len=699 TSval=990369853 TSecr=335012507
12	4.749960	192.168.0.13	128.119.245.12	TCP	1514	58655 → 80 [ACK] Seq=700 Ack=1 Win=131712 Len=1448 TSval=990369854 TSecr=335012507
13	4.749961	192.168.0.13	128.119.245.12	TCP	1514	58655 → 80 [ACK] Seq=2148 Ack=1 Win=131712 Len=1448 TSval=990369854 TSecr=335012507
14	4.801420	128.119.245.12	192.168.0.13	TCP	74	80 → 58655 [ACK] Seq=1 Ack=700 Win=30464 Len=0 TSval=335012541 TSecr=990369853
15	4.801428	128.119.245.12	192.168.0.13	TCP	74	80 → 58655 [ACK] Seq=1 Ack=2148 Win=33280 Len=0 TSval=335012541 TSecr=990369854
16	4.801551	192.168.0.13	128.119.245.12	TCP	1514	58655 → 80 [ACK] Seq=3596 Ack=1 Win=131712 Len=1448 TSval=990369904 TSecr=335012541
17	4.801552	192.168.0.13	128.119.245.12	TCP	1514	58655 → 80 [ACK] Seq=5044 Ack=1 Win=131712 Len=1448 TSval=990369904 TSecr=335012541
18	4.801552	192.168.0.13	128.119.245.12	TCP	1514	58655 → 80 [ACK] Seq=6492 Ack=1 Win=131712 Len=1448 TSval=990369904 TSecr=335012541
19	4.802781	128.119.245.12	192.168.0.13	TCP	74	80 → 58655 [ACK] Seq=1 Ack=3596 Win=36224 Len=0 TSval=335012552 TSecr=990369854
20	4.802877	192.168.0.13	128.119.245.12	TCP	1514	58655 → 80 [ACK] Seq=7940 Ack=1 Win=131712 Len=1448 TSval=990369905 TSecr=335012552
21	4.802878	192.168.0.13	128.119.245.12	TCP	1514	58655 → 80 [ACK] Seq=9388 Ack=1 Win=131712 Len=1448 TSval=990369905 TSecr=335012552
22	4.834084	128.119.245.12	192.168.0.13	TCP	74	80 → 58655 [ACK] Seq=1 Ack=5044 Win=39168 Len=0 TSval=335012592 TSecr=990369904
23	4.834204	192.168.0.13	128.119.245.12	TCP	1514	58655 → 80 [ACK] Seq=10836 Ack=1 Win=131712 Len=1448 TSval=990369936 TSecr=335012592
24	4.834205	192.168.0.13	128.119.245.12	TCP	1514	58655 → 80 [ACK] Seq=12284 Ack=1 Win=131712 Len=1448 TSval=990369936 TSecr=335012592
25	4.834511	128.119.245.12	192.168.0.13	TCP	74	80 → 58655 [ACK] Seq=1 Ack=7940 Win=44928 Len=0 TSval=335012592 TSecr=990369904
26	4.834517	128.119.245.12	192.168.0.13	TCP	74	80 → 58655 [ACK] Seq=1 Ack=9388 Win=47744 Len=0 TSval=335012594 TSecr=990369905
27	4.834610	192.168.0.13	128.119.245.12	TCP	1514	58655 → 80 [ACK] Seq=13732 Ack=1 Win=131712 Len=1448 TSval=990369936 TSecr=335012592

```
▶ Frame 11: 765 bytes on wire (6120 bits), 765 bytes captured (6120 bits) on interface 0
▶ Ethernet II, Src: Apple_1d:a5:6e (a4:83:e7:1d:a5:6e), Dst: Technico_5e:c4:f0 (fc:52:8d:5e:c4:f0)
▶ Internet Protocol Version 4, Src: 192.168.0.13, Dst: 128.119.245.12
▼ Transmission Control Protocol, Src Port: 58655, Dst Port: 80, Seq: 1, Ack: 1, Len: 699
    Source Port: 58655
    Destination Port: 80
    [Stream index: 1]
    [TCP Segment Len: 699]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 700    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    1000 .... = Header Length: 32 bytes (8)
  ▼ Flags: 0x018 (PSH, ACK)
```

Packet 2 You can see TCP segment length as 1448 Bytes



No.	Time	Source	Destination	Protocol	Length	Info
8	4.748465	192.168.0.13	128.119.245.12	TCP	66	58655 → 80 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=990369853 TSecr=335012507
9	4.748639	128.119.245.12	192.168.0.13	TCP	82	80 → 58656 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=335012507 TSecr=990
10	4.748687	192.168.0.13	128.119.245.12	TCP	66	58656 → 80 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=990369853 TSecr=335012507
11	4.748960	192.168.0.13	128.119.245.12	TCP	765	58655 → 80 [PSH, ACK] Seq=1 Ack=1 Win=131712 Len=699 TSval=990369853 TSecr=335012507
12	4.749960	192.168.0.13	128.119.245.12	TCP	1514	58655 → 80 [ACK] Seq=700 Ack=1 Win=131712 Len=1448 TSval=990369854 TSecr=335012507
13	4.749961	192.168.0.13	128.119.245.12	TCP	1514	58655 → 80 [ACK] Seq=2148 Ack=1 Win=131712 Len=1448 TSval=990369854 TSecr=335012507
14	4.801420	128.119.245.12	192.168.0.13	TCP	74	80 → 58655 [ACK] Seq=1 Ack=700 Win=30464 Len=0 TSval=335012541 TSecr=990369853
15	4.801428	128.119.245.12	192.168.0.13	TCP	74	80 → 58655 [ACK] Seq=1 Ack=2148 Win=33280 Len=0 TSval=335012541 TSecr=990369854
16	4.801551	192.168.0.13	128.119.245.12	TCP	1514	58655 → 80 [ACK] Seq=3596 Ack=1 Win=131712 Len=1448 TSval=990369904 TSecr=335012541
17	4.801552	192.168.0.13	128.119.245.12	TCP	1514	58655 → 80 [ACK] Seq=5044 Ack=1 Win=131712 Len=1448 TSval=990369904 TSecr=335012541
18	4.801552	192.168.0.13	128.119.245.12	TCP	1514	58655 → 80 [ACK] Seq=6492 Ack=1 Win=131712 Len=1448 TSval=990369904 TSecr=335012541
19	4.802781	128.119.245.12	192.168.0.13	TCP	74	80 → 58655 [ACK] Seq=1 Ack=3596 Win=36224 Len=0 TSval=335012552 TSecr=990369854
20	4.802877	192.168.0.13	128.119.245.12	TCP	1514	58655 → 80 [ACK] Seq=7940 Ack=1 Win=131712 Len=1448 TSval=990369905 TSecr=335012552
21	4.802878	192.168.0.13	128.119.245.12	TCP	1514	58655 → 80 [ACK] Seq=9388 Ack=1 Win=131712 Len=1448 TSval=990369905 TSecr=335012552
22	4.834084	128.119.245.12	192.168.0.13	TCP	74	80 → 58655 [ACK] Seq=1 Ack=5044 Win=39168 Len=0 TSval=335012592 TSecr=990369904
23	4.834204	192.168.0.13	128.119.245.12	TCP	1514	58655 → 80 [ACK] Seq=10836 Ack=1 Win=131712 Len=1448 TSval=990369936 TSecr=335012592
24	4.834205	192.168.0.13	128.119.245.12	TCP	1514	58655 → 80 [ACK] Seq=12284 Ack=1 Win=131712 Len=1448 TSval=990369936 TSecr=335012592
25	4.834511	128.119.245.12	192.168.0.13	TCP	74	80 → 58655 [ACK] Seq=1 Ack=7940 Win=44928 Len=0 TSval=335012592 TSecr=990369904
26	4.834517	128.119.245.12	192.168.0.13	TCP	74	80 → 58655 [ACK] Seq=1 Ack=9388 Win=47744 Len=0 TSval=335012594 TSecr=990369905
27	4.834610	192.168.0.13	128.119.245.12	TCP	1514	58655 → 80 [ACK] Seq=13732 Ack=1 Win=131712 Len=1448 TSval=990369936 TSecr=335012592

```
▶ Frame 12: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
▶ Ethernet II, Src: Apple_1d:a5:6e (a4:83:e7:1d:a5:6e), Dst: Technico_5e:c4:f0 (fc:52:8d:5e:c4:f0)
▶ Internet Protocol Version 4, Src: 192.168.0.13, Dst: 128.119.245.12
▼ Transmission Control Protocol, Src Port: 58655, Dst Port: 80, Seq: 700, Ack: 1, Len: 1448
    Source Port: 58655
    Destination Port: 80
    [Stream index: 1]
    [TCP Segment Len: 1448]
    Sequence number: 700    (relative sequence number)
    [Next sequence number: 2148    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    1000 .... = Header Length: 32 bytes (8)
  ▼ Flags: 0x010 (ACK)
```

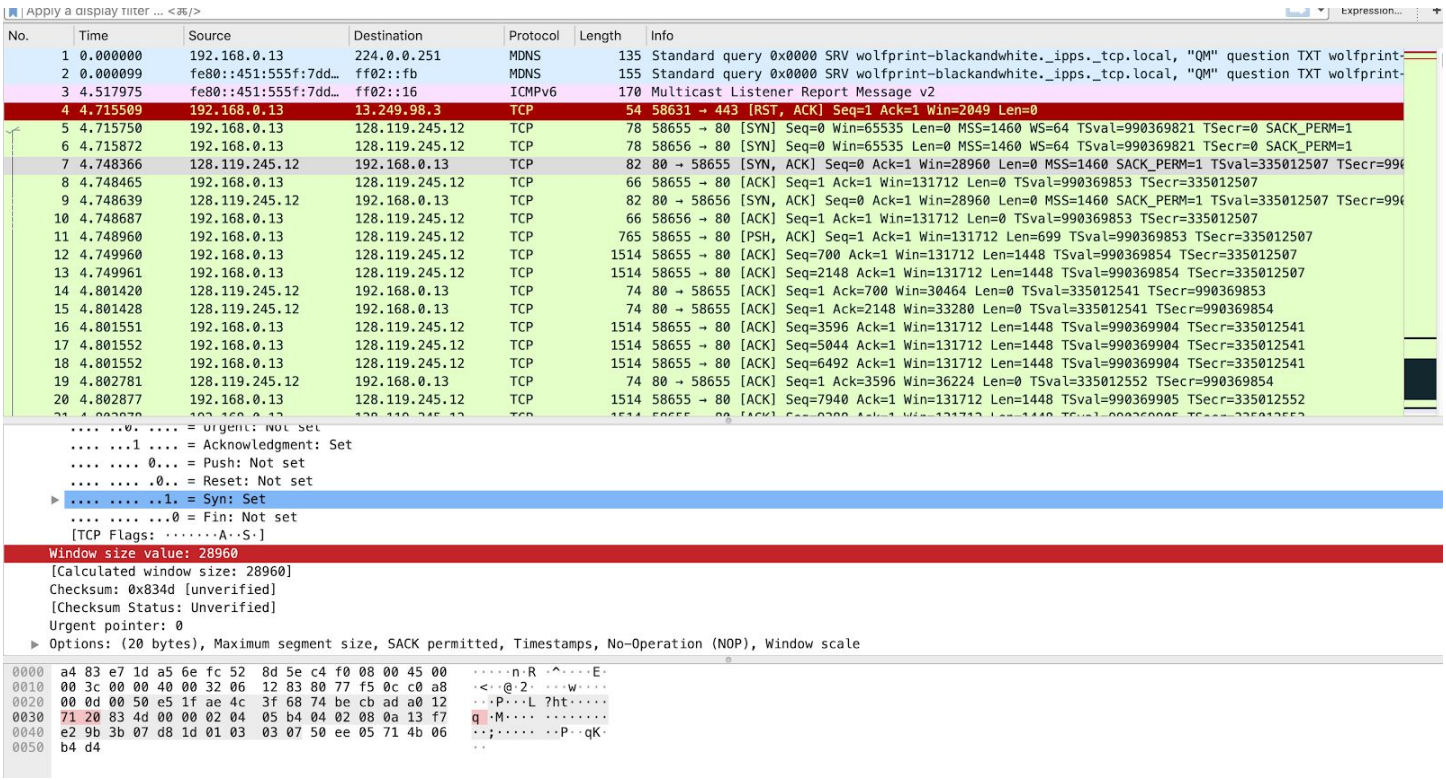
Similarly, it is for all the other 4 packets also.

9.

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?

The minimum amount of available buffer space advertised at the received for the entire trace is indicated the first ACK from the server, its value is 28960 bytes

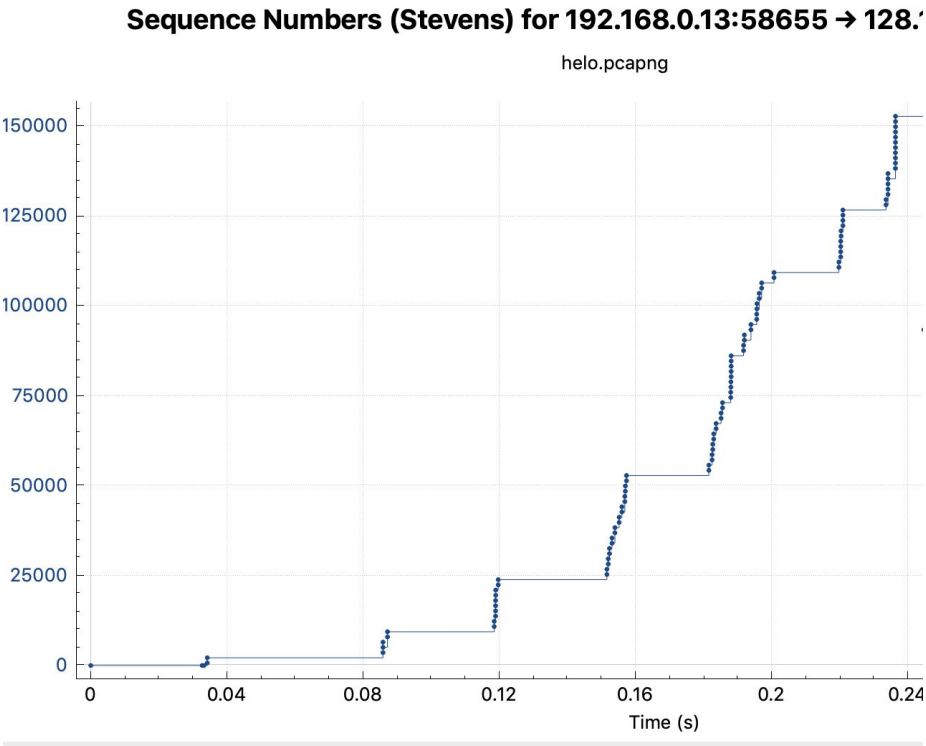
We can see from the trace that the sender is never throttled due to lack of receiver buffer space.r



The value of the receiver window increases to a value of 226688.

10.

There are no retransmitted segments in the trace file and the same can be seen in the time sequence graph (stevens), as the sequence numbers are increasing monotonically



.

11.

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

We can see that

Packet on line 11 is acknowledged by a packet on line 14

12 by 15

13 by 19

16 by 22

But 17 and 18 both by 25

12.

The throughput of the network:

152,138 bytes = File size

Total time = last ack - first tcp = 5.052911 - 4.748960 = 0.303951

500,534.6 bytes/ sec

13.

In the Stevens graphs that was given to us (*tcp-etherealtrace-1)*

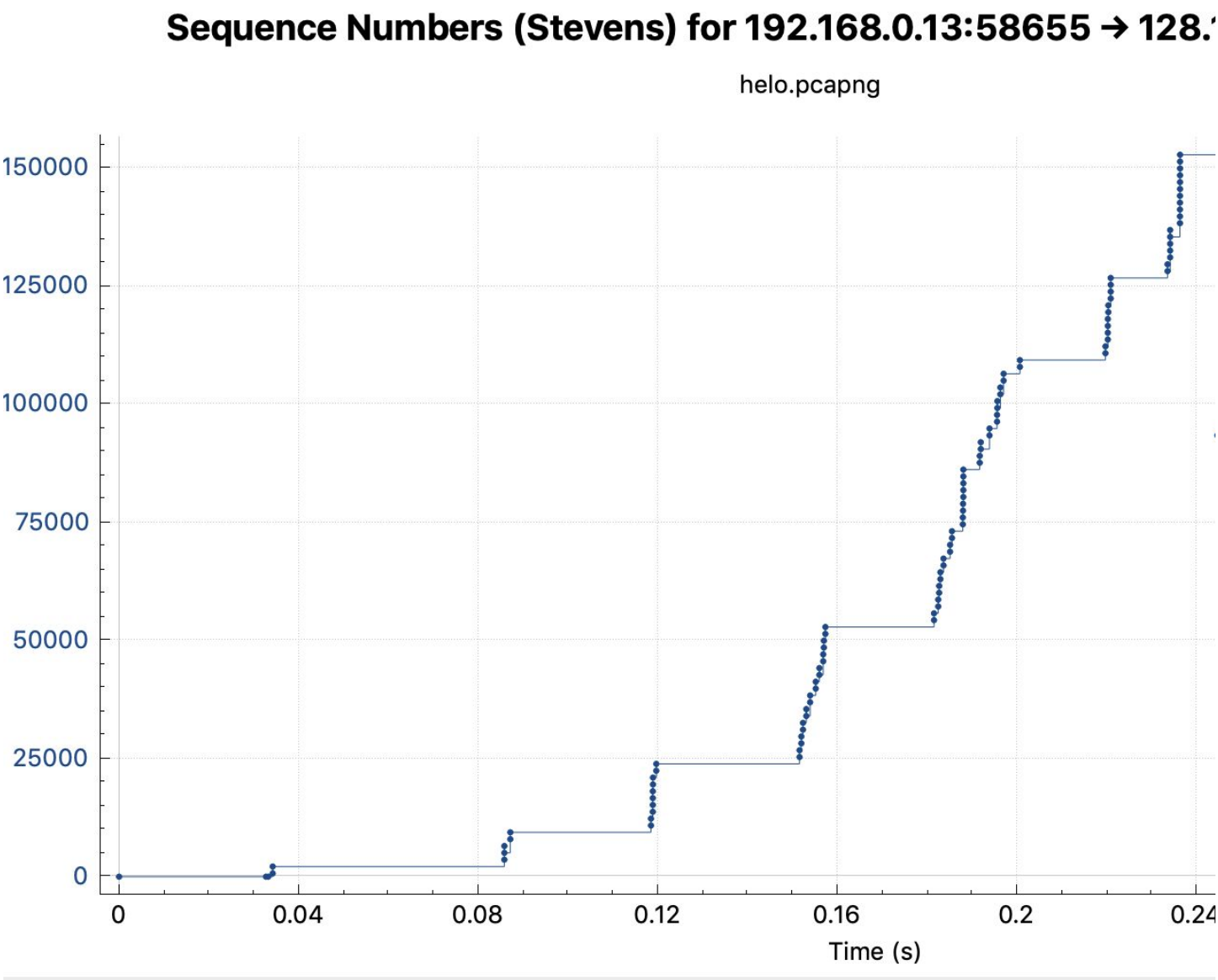
I could observe that :

Here also I could not observe the linear increase behavior as there is in case of congestion avoidance. There is a possibility that flow control is not being practised as the receiver window is more than the number of packets being sent in a batch.

The slow start phase is there only for a few seconds that is 1 to 1.5 second.
After that it is always in the congestion avoidance state.

14.

Stevens Graph for my wireshark capture:



Similar observation for my graph plot as well.

Answers to the Research Papers

Paper 1

1.

What are the two main approaches for improving TCP performance over networks with lossy links (i.e., wireless networks with significant losses due to bit errors and handoffs)? Are hybrid schemes possible?

Ans.

The first approach hides any non-congestion-related losses from the TCP sender and therefore it does not require to change the existing implementations in the server. The idea behind solving this problem is that since the problem is local, it should be solved locally and that the transport layer need not be aware of the characteristics of the individual links.

Some protocols that adopt this approach attempt to make the lossy links appear a better quality link with reduced effective bandwidth. As a result, most of the losses seen by the TCP sender are caused by congestion.
Examples of this approach include wireless links with reliable link layer protocols such as AIRMAIL, split connection approaches such as IndirectTCP, and TCP-aware link-layer schemes such as the snoop protocol.

The second technique attempted is to make the sender aware of the existence of wireless hops and realize that some packet losses are not due to congestion.

Yes, a hybrid approach is possible as the sender can then avoid invoking congestion control algorithms when non-congestion-related losses occur Hence, it is possible for a wireless-aware transport protocol to coexist with link-layer schemes to achieve good performance.

A few important protocols that have been proposed to improve the performance of TCP over wireless links are as follows:
Link-layer protocols- The 2 main techniques involved are error correction (using techniques such as forward error correction (FEC)), and retransmission of lost packets in response to automatic repeat request (ARQ) messages
-Indirect-TCP (I-TCP) protocol
-The Snoop Protocol
-Selective Acknowledgments

2.
The paper classifies the main schemes into three groups. Explain the underlying philosophy of each group.
Ans.
The main schemes have been classified into 3 groups in the paper. Below is the philosophy underlying each group:

- End-To-End Schemes
The E2E protocol improves the performance of TCP-Reno after multiple packet losses in a window by remaining in a fast recovery mode if the first new acknowledgment received after fast retransmission is “partial”, i.e, is less than the value of the last byte transmitted when the fast retransmission was done.

This method enables the connection to make progress at the rate of one segment per round trip time, rather than stall until a coarse timeout.

- Link-Layer Schemes
Existing link-layer protocols choose from techniques such as Stop-and-Wait, Go-Back-N, Selective Repeat, and Forward Error Correction to provide reliability.

The base link-layer algorithm, called LL, uses cumulative acknowledgments to discover the lost packets that are retransmitted locally from the base station to the host.

In order to minimize the overhead, the implementation of LL uses TCP acknowledgments instead of generating its own acknowledgments.

Further, the timeout based retransmissions are managed by a smoothed round-trip time estimate.

- Split-Connection Schemes
Uses an intermediate host to divide a TCP connection into two separate TCP connections.
The implementation avoids the copying of data copying in the intermediate hosts by passing the pointers to the same buffer between the two TCP connections

3.
Discuss the main conclusions that the authors draw from the experiments.

Several experiments were performed to determine the performance and efficiency of each of the protocols

The protocols were implemented as a set of modifications to the BSD/OS TCP/IP (Reno) network stack

Link-Layer Protocols
Traditional link-layer protocols operate independently of the higher-layer protocol, and consequently, do not necessarily shield the sender from the lossy link.

Hence the performance is impacted due to 2 reasons:
(i) competing retransmissions caused by an incompatible setting of timers at the two layers, and
(ii) the effect of the link-layer protocol on the TCP fast retransmission mechanism

The effects of the first situation were simulated and analyzed for a TCP-like transport protocol and a reliable link layer protocol.

It was concluded that unless the packet loss rate is high, competing retransmissions by the link and transport layers often lead to significant performance degradation.

However, this is not the dominating effect when link-layer schemes, such as LL, are used with TCP Reno and its variants.

The real problem is that when packets are lost, link-layer protocols that do not attempt in-order delivery across the link (e.g., LL) cause packets to reach the TCP receiver out-of-order. This leads to the generation of duplicate acknowledgments by the TCP receiver, which causes the sender to invoke fast retransmission and recovery, and can potentially cause degraded throughput and goodput, especially when the delay-bandwidth product is large.

A simple link-layer retransmission scheme could adversely impact TCP performance. An enhanced link-layer scheme, that uses knowledge of TCP semantics to prevent duplicate acknowledgments caused by wireless losses from reaching the sender, achieves significantly better performance.

End-To-End Protocols
>E2E-NEWRENO is better than E2E, especially for large socket buffer sizes.

>Adding ELN to TCP improves throughput significantly by successfully preventing unnecessary fluctuations in the transmission window.
>SACKs provide a significant improvement over TCP Reno but perform about 10-15% worse than the best local schemes in the LAN tests

Split-Connection Protocols
>The split-connection approach results in a better throughput if the wireless connection uses some special mechanisms, the performance does not exceed that of a well-tuned, TCP-aware link-layer protocol (LL-OPT).
>Moreover, the link-layer protocol maintains the end-to-end semantics of TCP acknowledgments, unlike the split-connection approach.
>This demonstrates that the end-to-end connection need not be split at the base station in order to achieve good performance.

Paper 2

1.
Describe the TCP ECN protocol and what actions various devices may take to subvert congestion control.

>Explicit Congestion Notification (ECN) [21], with active queue management in the form of RED gateways , has been proposed as a standard mechanism to improve congestion control in the Internet. With ECN, routers are able to mark packets to signal incipient congestion, as well as simply drop them during congestion. This avoids loss and improves performance

>The design of ECN is on TCP, because it is the only mainstream transport protocol for which ECN is currently defined.

>Explicit Congestion Notification (ECN) changes the character of congestion signaling to improve performance.

> It allows routers to signal congestion to end hosts explicitly, rather than implicitly via packet drops.

> Routers mark packets along congested links, and the receiver returns these congestion marks to the sender in a transport-specific manner.

Actions various devices may take to subvert congestion control:

>A receiver may receive marked packets but neglect to inform the sender

>A router on the reverse path may clear the congestion echo signals being returned to the sender.

> To signal congestion, routers set the Congestion Experienced (CE) state in the IP header of ECN-capable packets.

>The receiver returns this signal to the sender by setting the ECN-Echo (ECE) flag in the TCP header of subsequent acknowledgements. To ensure reliable delivery of this signal, the receiver continues to set the ECE flag in acknowledgements until a Congestion Window Reduced (CWR) flag is received, implying the sender has reacted to the congestion

>the design of ECN requires routers and receivers to explicitly and correctly participate in the congestion control loop, but has no means to check or enforce this cooperation.

2.
Explain why receiver misbehavior is worse than sender misbehavior, and describe the effects of the former.

Ans.

The receiver misbehavior is more dangerous than the sender's misbehavior because by hiding congestion signals, a misbehaving TCP receiver can persuade the sender into increasing the congestion window. Because the data packets are ECN-capable, they will not be dropped by the router until the link becomes congested. This behavior is dangerous because

>they subvert congestion control

>A receiver may receive marked packets but neglect to inform the sender

>A router may clear congestion signals received from upstream

>A router on the reverse path may clear the congestion echo signals being returned to the sender.

The effects of the receivers misbehavior are as follows:

>Less Bandwidth obtained-We show the bandwidth obtained by all flows relative to their “fair share” of the bottleneck as the number of competing flows varies. The “fair share” is easy to interpret, but underestimates the impact of misbehavior because not only does the misbehavior receive up to six times its fair share, the behavers receive as little as one tenth their fair share.

>Misbehavior gains a significant performance advantage over compliant flows, does not harm its own performance in the absence of contention, and greatly reduces the bandwidth available to compliant flows.

>When the misbehavior competes with ECN-enabled TCP connections, it forces the sender’s congestion window to increase until the router drops packets. When the number of flows is sufficient to saturate the router, it no longer marks packets to alleviate congestion and instead drops packets from all flows in proportion to their queue consumption, decreasing the misbehaver’s effectiveness

3.
Describe the main elements of the proposed robust ECN protocol.

When a router drops a packet to signal congestion, this signal is permanent: a downstream router cannot “undrop” a packet. If we enable the ECN sender to detect when marked packets are unmarked, we make ECN as robust a congestion signal as packet drops.

One-bit Nonces:

>Large nonces of 16 or 32 bits would be effective at identifying concealed ECN congestion signals.

> An insight enabling a less expensive implementation is that congestion control applies to a sequence of packets.

>Even a one-bit random nonce per packet is enough to detect misbehavior in the congestion control loop since each mark of congestion is a separate trial.

Cumulative Nonce Protocol

>Cumulative nonces allow the receiver to prove receipt of unmarked packets without returning every original nonce.

>The sender places a random nonce in each packet, which is cleared by a router to signal congestion.

>The receiver maintains a cumulative nonce, which is the sum of the nonces received for all in-order packets and includes it in every acknowledgment to be verified by the sender.

>Because every nonce is needed to calculate the correct cumulative nonce, it depends on the receipt of only unmarked packets

>In the case of TCP, we suspend the checking of the cumulative nonce while the Congestion Window Reduced (CWR) signal is delivered to the receiver.

>We reset the sender's cumulative nonce to the receiver's when the packet containing CWR is acknowledged.

Detected Misbehaviors

>ECN with nonces protects ECN from various abuses and incompatibilities. ECN-nonce senders are able to detect the dangerous misbehaviors and the potential misbehavior of network devices removing ECN capability from packets

> The ECN-nonce can also be used to protect other congestion-related protocols from misbehavior.

>The ECN-nonce also prevents the optimistic acknowledgment vulnerability

>The ECN-nonce provides a mechanism to detect misbehavior but leaves unspecified the sender-specific policy to address it.

Extensions for Other Transports

>The Stream Control Transmission Protocol is a new, reliable transport protocol being developed by the IETF that includes modern features such as multi-homing, framing, and multiple concurrent streams per connection.

>SCTP uses selective acknowledgments and supports ECN.

> The cumulative nonce can be used in SCTP because like TCP, it uses cumulative acknowledgments. The cumulative nonce can be carried in a new SCTP option, known as a chunk

>The ECN-nonce can also be applied to unreliable transports by taking advantage of the transport-specific acknowledgment mechanism.

>TCP-Friendly Rate Control (TFRC) is an unreliable transport protocol that uses a model of TCP performance to calculate a smooth sending rate based on the loss event rate and round trip time.

>TFRC receivers calculate the loss event rate from a weighted average of the length of recent loss intervals.