

WEEK 5

NAME : SUNDEEP A
ROLL NO : 48

SRN : PES1UG20CS445
SECTION: H

Step 1: UDP and DNS

Q1) Open Wireshark and set up our privacy filter so that you display only DNS traffic to or from your computer (Filter: dns && ip.addr==<your IP address>).

dns && ip.addr == 192.168.100.137					
No.	Time	Source	Destination	Protocol	Length Info
4	0.000479589	192.168.100.137	192.168.100.2	DNS	81 Standard query 0xe696 A www.pluralsight.com
9	0.246002138	192.168.100.2	192.168.100.137	DNS	165 Standard query response 0xe696 A www.pluralsight.com CNAME ww...

▶ Frame 4: 81 bytes on wire (648 bits), 81 bytes captured (648 bits) on interface any, id 0
▶ Linux cooked capture
▶ Internet Protocol Version 4, Src: 192.168.100.137, Dst: 192.168.100.2
▼ User Datagram Protocol, Src Port: 46488, Dst Port: 53
Source Port: 46488
Destination Port: 53
Length: 45
▶ Checksum: 0x4a1b incorrect, should be 0xffff (maybe caused by "UDP checksum offload"?)
[Checksum Status: Bad]
[Stream index: 3]
[Timestamps]
▶ Domain Name System (query)

Q2) Use dig to generate a DNS query to lookup the domain name

```
sundeep@sundeep:~$ dig www.pluralsight.com

; <<>> DiG 9.16.1-Ubuntu <<>> www.pluralsight.com
;; global options: +cmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 9404
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;www.pluralsight.com.      IN      A

;; ANSWER SECTION:
www.pluralsight.com.      5      IN      CNAME   www.pluralsight.com.cdn.cloudflare.net.
www.pluralsight.com.cdn.cloudflare.net. 4 IN A    104.19.161.127
www.pluralsight.com.cdn.cloudflare.net. 4 IN A    104.19.162.127

;; Query time: 252 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: Tue Apr 19 22:41:32 IST 2022
;; MSG SIZE rcvd: 132
```

Q3) Before you look at the packets in Wireshark, think for a minute about what you expect to see as the UDP segment headers. What can you reasonably predict, and what could you figure out if you had some time and a calculator handy? Use your knowledge of UDP to inform your predictions.

I expect to see Source-port and Destination-port numbers, Length, Check-sum.

Q4) Take a look at the query packet on Wireshark. You'll see a bunch of bytes (70-75 bytes) listed as the actual packet contents in the bStom Wireshark window. The bytes at offsets up to number 33-34 are generated by the lower-level prSocols.You will also see Wireshark interpret the header contents. Match up the bytes in the packet contents window with each field of the UDP header. Were your predictions correct?

dns && ip.addr == 192.168.100.137						
No.	Time	Source	Destination	Protocol	Length	Info
4	0.000479589	192.168.100.137	192.168.100.2	DNS	81	Standard query 0xe696 A www.pluralsight.com
9	0.246002138	192.168.100.2	192.168.100.137	DNS	165	Standard query response 0xe696 A www.pluralsight.com CNAME ww...

▶ Frame 4: 81 bytes on wire (648 bits), 81 bytes captured (648 bits) on interface any, id 0 ▶ Linux cooked capture ▶ Internet Protocol Version 4, Src: 192.168.100.137, Dst: 192.168.100.2 ▶ User Datagram Protocol, Src Port: 46488, Dst Port: 53						
Source Port: 46488 Destination Port: 53 Length: 45						
Checksum: 0x4a1b incorrect, should be 0xffff (maybe caused by "UDP checksum offload"?) ▶ [Expert Info (Error/Checksum): Bad checksum [should be 0xffff]] [Calculated Checksum: 0xffff] [Checksum Status: Bad] [Stream index: 3] ▶ [Timestamps]						
▶ Domain Name System (query)						

0000	00 04 00 01 00 06 00 0c 29 60 c2 69 00 00 08 00).i....
0010	45 00 00 41 5d 47 40 00 40 11 93 88 c0 a8 64 89	E..A]G@. @.....d.
0020	c0 a8 64 02 b5 98 00 35 00 2d 4a 1b e6 96 01 00	.d...5..J....
0030	00 01 00 00 00 00 00 00 03 77 77 77 0b 70 6c 75www.plu
0040	72 61 6c 73 69 67 68 74 03 63 6f 6d 00 00 01 00	alsight .com...
0050	01	.

In bits:

dns && ip.addr == 192.168.100.137						
No.	Time	Source	Destination	Protocol	Length	Info
4	0.000479589	192.168.100.137	192.168.100.2	DNS	81	Standard query 0xe696 A www.pluralsight.com
9	0.246002138	192.168.100.2	192.168.100.137	DNS	165	Standard query response 0xe696 A www.pluralsight.com CNAME ww...

▶ Frame 4: 81 bytes on wire (648 bits), 81 bytes captured (648 bits) on interface any, id 0 ▶ Linux cooked capture ▶ Internet Protocol Version 4, Src: 192.168.100.137, Dst: 192.168.100.2 ▶ User Datagram Protocol, Src Port: 46488, Dst Port: 53						
Source Port: 46488 Destination Port: 53 Length: 45						
Checksum: 0x4a1b incorrect, should be 0xffff (maybe caused by "UDP checksum offload"?) ▶ [Expert Info (Error/Checksum): Bad checksum [should be 0xffff]] [Calculated Checksum: 0xffff] [Checksum Status: Bad] [Stream index: 3] ▶ [Timestamps]						
▶ Domain Name System (query)						

0000	00000000 00000100 00000000 00000001 00000000 00000110 00000000 00001100)-B....
0008	00101001 01100000 11000010 01101001 00000000 00000000 00001000 00000000	...). .
0016	01000101 00000000 00000000 01000001 01011101 01000111 01000000 00000000	..h{y.i
0024	01000000 00010001 10010011 10001000 11000000 10101000 01100100 10001001	{y..q5
0032	11000000 10101000 01100100 00000010 00110101 10011000 00000000 00110101	...wo..
0040	00000000 00101101 01001010 00011011 11100110 10010110 00000001 00000000%
0048	00000011 01110111 01110111 01110111 00001011 01110000 01101100 01110101	..?_....
0056	01110010 01100001 01101100 01110011 01101001 01100111 01101000 01110100	
0064	00000011 01100011 01101111 01101101 00000000 00000000 00000001 00000000	
0072	00000001	

Q5) Continue to examine the DNS request packet. Which fields does the UDP checksum cover? Wireshark probably shows the UDP checksum as “Validation Disabled”. Why is that?

Due to the prevalence of offloading in modern hardware and operating systems

Step 2: TCP

Q11) Take a screenshot showing the three-way handshake.

Here the first 3 packets show the 3 way handshake..

Destination: 128.2.131.88

Transmission Control Protocol, Src Port: 60030, Dst Port: 80, Seq: 0, Len: 0

Source Port: 60030
Destination Port: 80
[Stream index: 1]
[TCP Segment Len: 0]
Sequence number: 0 (relative sequence number)
Sequence number (raw): 4014126031
[Next sequence number: 1 (relative sequence number)]
Acknowledgment number: 0
Acknowledgment number (raw): 0
1010 = Header Length: 40 bytes (10)

Flags: 0x002 (SYN)

000. = Reserved: Not set
...0 = Nonce: Not set
....0. = Congestion Window Reduced (CWR): Not set
....0. = ECN-Echo: Not set
....0. = Urgent: Not set
....0. = Acknowledgment: Not set
....0. = Push: Not set
....0. = Reset: Not set

....1. = Syn: Set
....0. = Fin: Not set
[TCP Flags:S.]
Window size value: 64240
[Calculated window size: 64240]

0010 01000101 00000000 00000000 00111100 10111000 10111100 01000000 00000000 ...<W..
0018 01000000 00000110 00101011 01110011 11000000 10101000 01100100 10001001 ...+.y.i
0020 10000000 00000010 10000011 01011000 11010101 01111110 00000000 01010000 ...c..=d
0028 11101111 01000010 10110011 11001111 00000000 00000000 00000000 00000000
0030 10100000 00000010 11111010 11110000 00101000 10111011 00000000 00000000 ...0(...
0038 00000010 00000100 00000101 10110100 00000100 00000010 00001000 00001010
0040 00111110 01101101 00110000 11000001 00000000 00000000 00000000 00000000 ...A....
0048 00000001 00000011 00000011 00000111 00000000 00000000 00000000 00000000

Q12) What is the IP address and TCP port number used by your computer (client) to transfer the file? What is the IP address of the server? On what port number is it sending and receiving TCP segments for this transfer of the file?

Client:

IP : 192.168.100.137

port:60030

Server:

IP: 128.2.131.88

port: 80

Q13) After disabling HTTP protocol in the Wireshark:

tcp && ip.addr == 192.168.100.137

No.	Time	Source	Destination	Protocol	Length	Info
18	5.736749970	192.168.100.137	128.2.131.88	TCP	76	60028 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 T...
19	5.988490732	192.168.100.137	128.2.131.88	TCP	76	60028 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 T...
24	6.352233177	128.2.131.88	192.168.100.137	TCP	62	80 → 60028 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
25	6.352312931	192.168.100.137	128.2.131.88	TCP	56	60028 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
26	6.352948036	192.168.100.137	128.2.131.88	TCP	2976	60028 → 80 [PSH, ACK] Seq=1 Ack=1 Win=64240 Len=2920
27	6.353066401	192.168.100.137	128.2.131.88	TCP	2976	60028 → 80 [PSH, ACK] Seq=2921 Ack=1 Win=64240 Len=2920
28	6.353158270	192.168.100.137	128.2.131.88	TCP	2976	60028 → 80 [PSH, ACK] Seq=5841 Ack=1 Win=64240 Len=2920
29	6.353252233	192.168.100.137	128.2.131.88	TCP	2976	60028 → 80 [PSH, ACK] Seq=8761 Ack=1 Win=64240 Len=2920
30	6.353320980	128.2.131.88	192.168.100.137	TCP	62	80 → 60028 [ACK] Seq=1 Ack=1461 Win=64240 Len=0
31	6.353388881	192.168.100.137	128.2.131.88	TCP	2976	60028 → 80 [PSH, ACK] Seq=11681 Ack=1 Win=64240 Len=2920
32	6.353402682	128.2.131.88	192.168.100.137	TCP	62	80 → 60028 [ACK] Seq=1 Ack=2921 Win=64240 Len=0
33	6.353418900	192.168.100.137	128.2.131.88	TCP	2976	60028 → 80 [PSH, ACK] Seq=14601 Ack=1 Win=64240 Len=2920
34	6.353476055	128.2.131.88	192.168.100.137	TCP	62	80 → 60028 [ACK] Seq=1 Ack=4381 Win=64240 Len=0
35	6.353548355	192.168.100.137	128.2.131.88	TCP	2976	60028 → 80 [PSH, ACK] Seq=17521 Ack=1 Win=64240 Len=2920
36	6.353549843	128.2.131.88	192.168.100.137	TCP	62	80 → 60028 [ACK] Seq=1 Ack=5841 Win=64240 Len=0
37	6.353619175	192.168.100.137	128.2.131.88	TCP	2976	60028 → 80 [PSH, ACK] Seq=20441 Ack=1 Win=64240 Len=2920
38	6.353699634	128.2.131.88	192.168.100.137	TCP	62	80 → 60028 [ACK] Seq=1 Ack=7301 Win=64240 Len=0
39	6.353761826	192.168.100.137	128.2.131.88	TCP	2976	60028 → 80 [PSH, ACK] Seq=23361 Ack=1 Win=64240 Len=2920
40	6.353820994	192.168.100.137	128.2.131.88	TCP	2976	60028 → 80 [PSH, ACK] Seq=26281 Ack=1 Win=64240 Len=2920
41	6.353861404	128.2.131.88	192.168.100.137	TCP	62	80 → 60028 [ACK] Seq=1 Ack=8761 Win=64240 Len=0
42	6.353961627	128.2.131.88	192.168.100.137	TCP	62	80 → 60028 [ACK] Seq=1 Ack=10221 Win=64240 Len=0
43	6.353928804	192.168.100.137	128.2.131.88	TCP	2976	60028 → 80 [PSH, ACK] Seq=29201 Ack=1 Win=64240 Len=2920
44	6.354028674	192.168.100.137	128.2.131.88	TCP	2976	60028 → 80 [PSH, ACK] Seq=32121 Ack=1 Win=64240 Len=2920

Header checksum: 0x2b73 [validation disabled]
[Header checksum status: Unverified]
Source: 192.168.100.137
Destination: 128.2.131.88

Transmission Control Protocol, Src Port: 60030, Dst Port: 80, Seq: 0, Len: 0

Source Port: 60030
Destination Port: 80
[Stream index: 1]
[TCP Segment Len: 0]
Sequence number: 0 (relative sequence number)
Sequence number (raw): 4014126031
[Next sequence number: 1 (relative sequence number)]
Acknowledgment number: 0
Acknowledgment number (raw): 0
1010 = Header Length: 40 bytes (10)

Flags: 0x002 (SYN)

000. = Reserved: Not set
...0 = Nonce: Not set
....0. = Congestion Window Reduced (CWR): Not set
....0. = ECN-Echo: Not set
....0. = Urgent: Not set
....0. = Acknowledgment: Not set
....0. = Push: Not set
....0. = Reset: Not set

....1. = Syn: Set
....0. = Fin: Not set
[TCP Flags:S.]
Window size value: 64240
[Calculated window size: 64240]

0010 01000101 00000000 00000000 00111100 11100110 10111100 01000000 00000000 ...<W..
0018 01000000 00000110 00101011 01110011 11000000 10101000 01100100 10001001 ...+.y.i
0020 10000000 00000010 10000011 01011000 11010101 01111110 00000000 01010000 ...c..=d
0028 11101111 01000010 10110011 11001111 00000000 00000000 00000000 00000000
0030 10100000 00000010 11111010 11110000 00101000 10111011 00000000 00000000 ...0(...
0038 00000010 00000100 00000101 10110100 00000100 00000010 00001000 00001010
0040 00111110 01101101 00110000 11000001 00000000 00000000 00000000 00000000 ...A....
0048 00000001 00000011 00000011 00000111 00000000 00000000 00000000 00000000

Step 2b: TCP Basics

Q14) What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection?

Relative Sequence Number:0

Raw Sequence number:4014126031

What element of the segment identifies it as a SYN segment?

From the above figure, we can see that the SYN bit is set to 1 in the flags section .So, it can be identified as a SYN segment.

Q15) What is the sequence number of the SYNACK segment sent by the server in reply to the SYN? What is the value of the Acknowledgement field in the SYNACK segment? How did the server determine that value? What element in the segment identifies it as a SYNACK segment?

From the below figure we can see that,

sequence number of the SYNACK segment sent by the server in reply to the SYN = 0
value of the Acknowledgement field in the SYNACK segment = 1

How did the server determine that value?

o The server adds 1 to the initial sequence number of SYN segment from the client computer. For this case, the initial sequence number of SYN segment from the client computer is 0, thus the value of the ACKnowledgement field in the SYNACK segment is 1.

What element in the segment identifies it as a SYNACK segment?

o A segment will be identified as a SYNACK segment if both SYN flag and Acknowledgement in the segment are set to 1.

tcp && ip.addr == 192.168.100.137						
No.	Time	Source	Destination	Protocol	Length	Info
18	5.736749979	192.168.100.137	128.2.131.88	TCP	76	60028 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 T...
19	5.988490732	192.168.100.137	128.2.131.88	TCP	76	60030 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 T...
24	6.352233177	128.2.131.88	192.168.100.137	TCP	62	80 → 60028 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
25	6.352312931	192.168.100.137	128.2.131.88	TCP	56	60028 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
26	6.352948036	192.168.100.137	128.2.131.88	TCP	2976	60028 → 80 [PSH, ACK] Seq=1 Ack=1 Win=64240 Len=2920
27	6.353066401	192.168.100.137	128.2.131.88	TCP	2976	60028 → 80 [PSH, ACK] Seq=2921 Ack=1 Win=64240 Len=2920
28	6.353158270	192.168.100.137	128.2.131.88	TCP	2976	60028 → 80 [PSH, ACK] Seq=5841 Ack=1 Win=64240 Len=2920
29	6.353252233	192.168.100.137	128.2.131.88	TCP	2976	60028 → 80 [PSH, ACK] Seq=8761 Ack=1 Win=64240 Len=2920
30	6.353320980	128.2.131.88	192.168.100.137	TCP	62	80 → 60028 [ACK] Seq=1 Ack=1461 Win=64240 Len=0
31	6.353338881	192.168.100.137	128.2.131.88	TCP	2976	60028 → 80 [PSH, ACK] Seq=11681 Ack=1 Win=64240 Len=2920
32	6.353402682	128.2.131.88	192.168.100.137	TCP	62	80 → 60028 [ACK] Seq=1 Ack=2921 Win=64240 Len=0
33	6.353418900	192.168.100.137	128.2.131.88	TCP	2976	60028 → 80 [PSH, ACK] Seq=14601 Ack=1 Win=64240 Len=2920
34	6.353476055	128.2.131.88	192.168.100.137	TCP	62	80 → 60028 [ACK] Seq=1 Ack=4381 Win=64240 Len=0
35	6.353548355	192.168.100.137	128.2.131.88	TCP	2976	60028 → 80 [PSH, ACK] Seq=17521 Ack=1 Win=64240 Len=2920
36	6.353549843	128.2.131.88	192.168.100.137	TCP	62	80 → 60028 [ACK] Seq=1 Ack=5841 Win=64240 Len=0
37	6.353619175	192.168.100.137	128.2.131.88	TCP	2976	60028 → 80 [PSH, ACK] Seq=20441 Ack=1 Win=64240 Len=2920
38	6.353699634	128.2.131.88	192.168.100.137	TCP	62	80 → 60028 [ACK] Seq=1 Ack=7301 Win=64240 Len=0
39	6.353701826	192.168.100.137	128.2.131.88	TCP	2976	60028 → 80 [PSH, ACK] Seq=23361 Ack=1 Win=64240 Len=2920
40	6.353820994	192.168.100.137	128.2.131.88	TCP	2976	60028 → 80 [PSH, ACK] Seq=26281 Ack=1 Win=64240 Len=2920
41	6.353901404	128.2.131.88	192.168.100.137	TCP	62	80 → 60028 [ACK] Seq=1 Ack=8761 Win=64240 Len=0
42	6.353901627	128.2.131.88	192.168.100.137	TCP	62	80 → 60028 [ACK] Seq=1 Ack=10221 Win=64240 Len=0
43	6.353928804	192.168.100.137	128.2.131.88	TCP	2976	60028 → 80 [PSH, ACK] Seq=29201 Ack=1 Win=64240 Len=2920
44	6.354028674	192.168.100.137	128.2.131.88	TCP	2976	60028 → 80 [PSH, ACK] Seq=32121 Ack=1 Win=64240 Len=2920
Frame 24: 62 bytes on wire (496 bits), 62 bytes captured (496 bits) on interface any, id 0						
Linux cooked capture						
Internet Protocol Version 4, Src: 128.2.131.88, Dst: 192.168.100.137						
Transmission Control Protocol, Src Port: 80, Dst Port: 60028, Seq: 0, Ack: 1, Len: 0						
Source Port: 80						
Destination Port: 60028						
[Stream index: 0]						
[TCP Segment Len: 0]						
Sequence number: 0 (relative sequence number)						
Sequence number (raw): 514225060						
[Next sequence number: 1 (relative sequence number)]						
Acknowledgment number: 1 (relative ack number)						
Acknowledgment number (raw): 3766173931						
0110 = Header Length: 24 bytes (6)						
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: 64240						
[Calculated window size: 64240]						
Checksum: 0xd61a [unverified]						
[Checksum Status: Unverified]						
0020	11000000	10101000	01100100	10001001	00000000	01010000 11101010 01111100 {y.i.&.0
0028	00011110	10100110	01110011	10100100	11100000	01111011 01000000 11101011 .w.u/# .
0030	01100000	00010010	11111010	11110000	11010110	00011010 00000000 00000000 .00...
0038	00000010	00000100	00000101	10110100	00000000	00000000

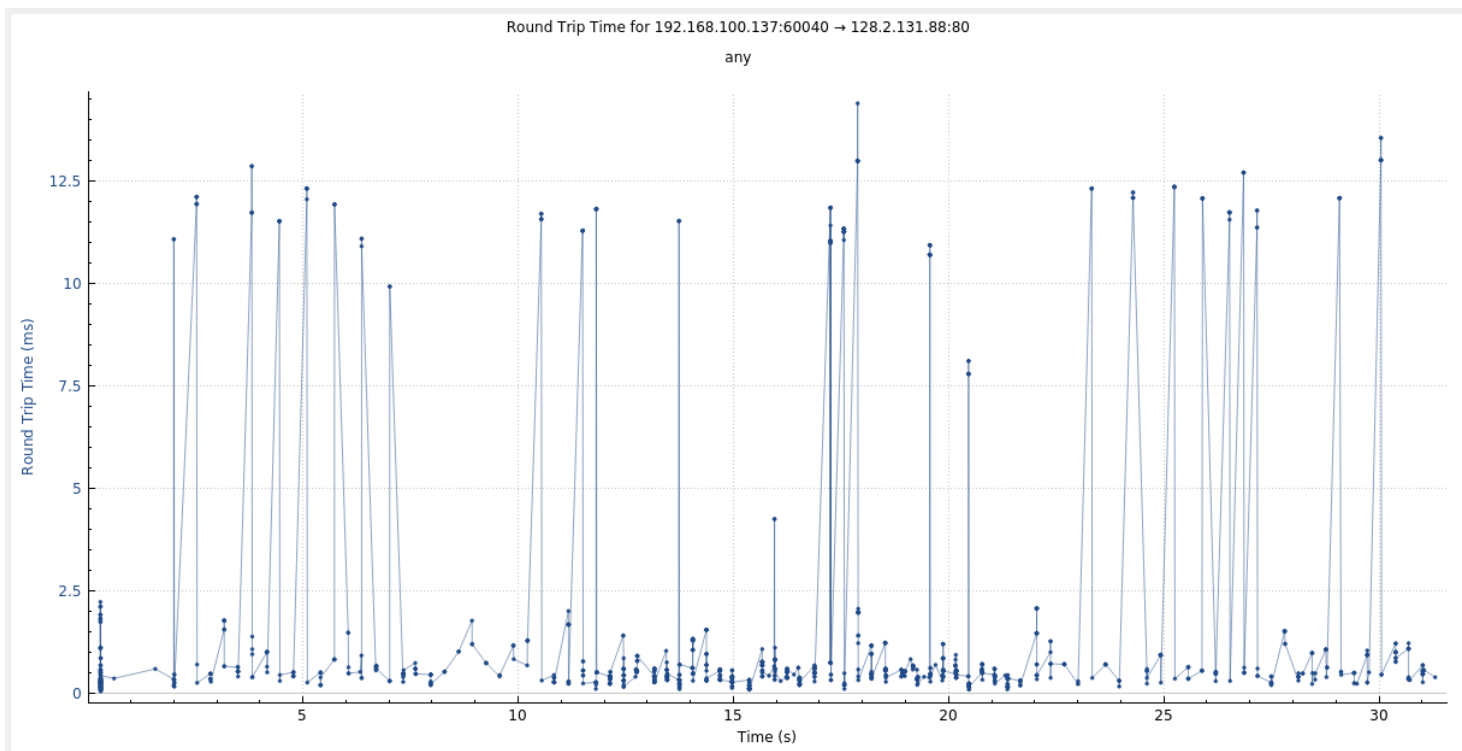
Q16) What is the sequence number of the TCP segment containing the HTTP POST command? Note that in order to find the POST command, you'll need to dig into the packet content field at the bottom of the Wireshark window, looking for a segment with a "POST" within its DATA field.

In the below figure we can see that the data field starts with the word "POST".

The sequence number of the TCP segment containing the HTTP POST is seq = 1

tcp && ip.addr == 192.168.100.137						
No.	Time	Source	Destination	Protocol	Length	Info
18	5.736749979	192.168.100.137	128.2.131.88	TCP	76	60028 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 T...
19	5.988490732	192.168.100.137	128.2.131.88	TCP	76	60030 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 T...
24	6.352233177	128.2.131.88	192.168.100.137	TCP	62	80 → 60028 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
25	6.352312931	192.168.100.137	128.2.131.88	TCP	56	60028 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
26	6.352948036	192.168.100.137	128.2.131.88	TCP	2976	60028 → 80 [PSH, ACK] Seq=1 Ack=1 Win=64240 Len=2920 [TCP seq...
27	6.353066401	192.168.100.137	128.2.131.88	TCP	2976	60028 → 80 [PSH, ACK] Seq=2921 Ack=1 Win=64240 Len=2920 [TCP ...
28	6.353158270	192.168.100.137	128.2.131.88	TCP	2976	60028 → 80 [PSH, ACK] Seq=5841 Ack=1 Win=64240 Len=2920 [TCP ...
29	6.353252233	192.168.100.137	128.2.131.88	TCP	2976	60028 → 80 [PSH, ACK] Seq=8761 Ack=1 Win=64240 Len=2920 [TCP ...
30	6.353320980	128.2.131.88	192.168.100.137	TCP	62	80 → 60028 [ACK] Seq=1 Ack=1461 Win=64240 Len=0
31	6.353338881	192.168.100.137	128.2.131.88	TCP	2976	60028 → 80 [PSH, ACK] Seq=11681 Ack=1 Win=64240 Len=2920 [TCP...
32	6.353402682	128.2.131.88	192.168.100.137	TCP	62	80 → 60028 [ACK] Seq=1 Ack=2921 Win=64240 Len=0
33	6.353418900	192.168.100.137	128.2.131.88	TCP	2976	60028 → 80 [PSH, ACK] Seq=14601 Ack=1 Win=64240 Len=2920 [TCP...
34	6.353476055	128.2.131.88	192.168.100.137	TCP	62	80 → 60028 [ACK] Seq=1 Ack=4381 Win=64240 Len=0
35	6.353548355	192.168.100.137	128.2.131.88	TCP	2976	60028 → 80 [PSH, ACK] Seq=17521 Ack=1 Win=64240 Len=2920 [TCP...
36	6.353549843	128.2.131.88	192.168.100.137	TCP	62	80 → 60028 [ACK] Seq=1 Ack=5841 Win=64240 Len=0
37	6.353619175	192.168.100.137	128.2.131.88	TCP	2976	60028 → 80 [PSH, ACK] Seq=20441 Ack=1 Win=64240 Len=2920 [TCP...
38	6.353699634	128.2.131.88	192.168.100.137	TCP	62	80 → 60028 [ACK] Seq=1 Ack=7301 Win=64240 Len=0
39	6.353701826	192.168.100.137	128.2.131.88	TCP	2976	60028 → 80 [PSH, ACK] Seq=23361 Ack=1 Win=64240 Len=2920 [TCP...
40	6.353820994	192.168.100.137	128.2.131.88	TCP	2976	60028 → 80 [PSH, ACK] Seq=26281 Ack=1 Win=64240 Len=2920 [TCP...
41	6.353901404	128.2.131.88	192.168.100.137	TCP	62	80 → 60028 [ACK] Seq=1 Ack=8761 Win=64240 Len=0
42	6.353901627	128.2.131.88	192.168.100.137	TCP	62	80 → 60028 [ACK] Seq=1 Ack=10221 Win=64240 Len=0
43	6.353928804	192.168.100.137	128.2.131.88	TCP	2976	60028 → 80 [PSH, ACK] Seq=29201 Ack=1 Win=64240 Len=2920 [TCP...
44	6.354028674	192.168.100.137	128.2.131.88	TCP	2976	60028 → 80 [PSH, ACK] Seq=32121 Ack=1 Win=64240 Len=2920 [TCP...
.....0.. = ECN-Echo: Not set0.. = Urgent: Not set1.. = Acknowledgment: Set1.. = Push: Set0.. = Reset: Not set0.. = Syn: Not set0.. = Fin: Not set [TCP Flags:AP...] Window size value: 64240 [Calculated window size: 64240] [Window size scaling factor: -2 (no window scaling used)] Checksum: 0x340f [unverified] [Checksum Status: Unverified] Urgent pointer: 0 [SEQ/ACK analysis] [RTT: 0.615562952 seconds] [Bytes in flight: 2920] [Bytes sent since last PSH flag: 2920] [Timestamps] TCP payload (2920 bytes) [Reassembled PDU in frame: 2148] TCP segment data (2920 bytes)						
0030	50 18 fa f0 34 0f 00 00	50 4f 53 54 20 2f 4c 61	P...4... POST /La			
0040	62 32 2f 6c 61 62 32 62	2e 68 74 6d 6c 20 48 54	b2/lab2b...html HT			
0050	54 50 2f 31 2e 31 0d 0a	48 6f 73 74 3a 20 77 77	TP/1.1...Host: ww			
0060	77 2e 69 6e 69 37 34 30	2e 63 6f 6d 0d 0a 55 73	w.ini740...com:Us			
0070	65 72 2d 41 67 65 6e 74	3a 20 4d 6f 7a 69 6c 6c	er-Agent : Mozill			
0080	61 2f 35 2e 30 20 28 58	31 31 3b 20 55 62 75 6e	a/5.0 (X 11; Ubun			
0090	74 75 3b 20 4c 69 6e 75	78 20 78 38 36 5f 36 34	tu; Linu x x86_64			
00a0	3b 20 72 76 3a 39 39 2e	30 29 20 47 65 63 6b 67	; rv:99.0) Gecko			
00b0	2f 32 30 31 30 30 31 30	31 20 46 69 72 65 66 6f	/2010010 1 Firefo			
00c0	70 2f 39 39 2e 30 0d 0a	41 63 63 65 70 74 3a 20	x/99.0...Accept:			
00d0	74 65 78 74 2f 68 74 6d	6c 2c 61 70 70 6c 69 63	text/html ,applic			

Q17) Consider the TCP segment containing the HTTP POST as the first segment in the non-overhead part of the TCP connection. For the segments which follow, put together a table with one row per segment (and columns for whatever data you think is useful) until you have enough segments to calculate four SampleRTT values according to the RTT estimation techniques discussed in class. Calculate what those SampleRTT values are, as well as the EstimatedRTT after each Sample is collected. Discuss this calculation, including what your initial EstimatedRTT was, your choice of parameters, and any segments that weren't used in the calculation. 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 server. Then select: Statistics → TCP Stream Graph → Round Trip Time Graph.



Q18) What is the minimum amount of available buffer space advertised at the receiver for the entire trace? Does the lack of receiver buffer space ever throttle the sender?

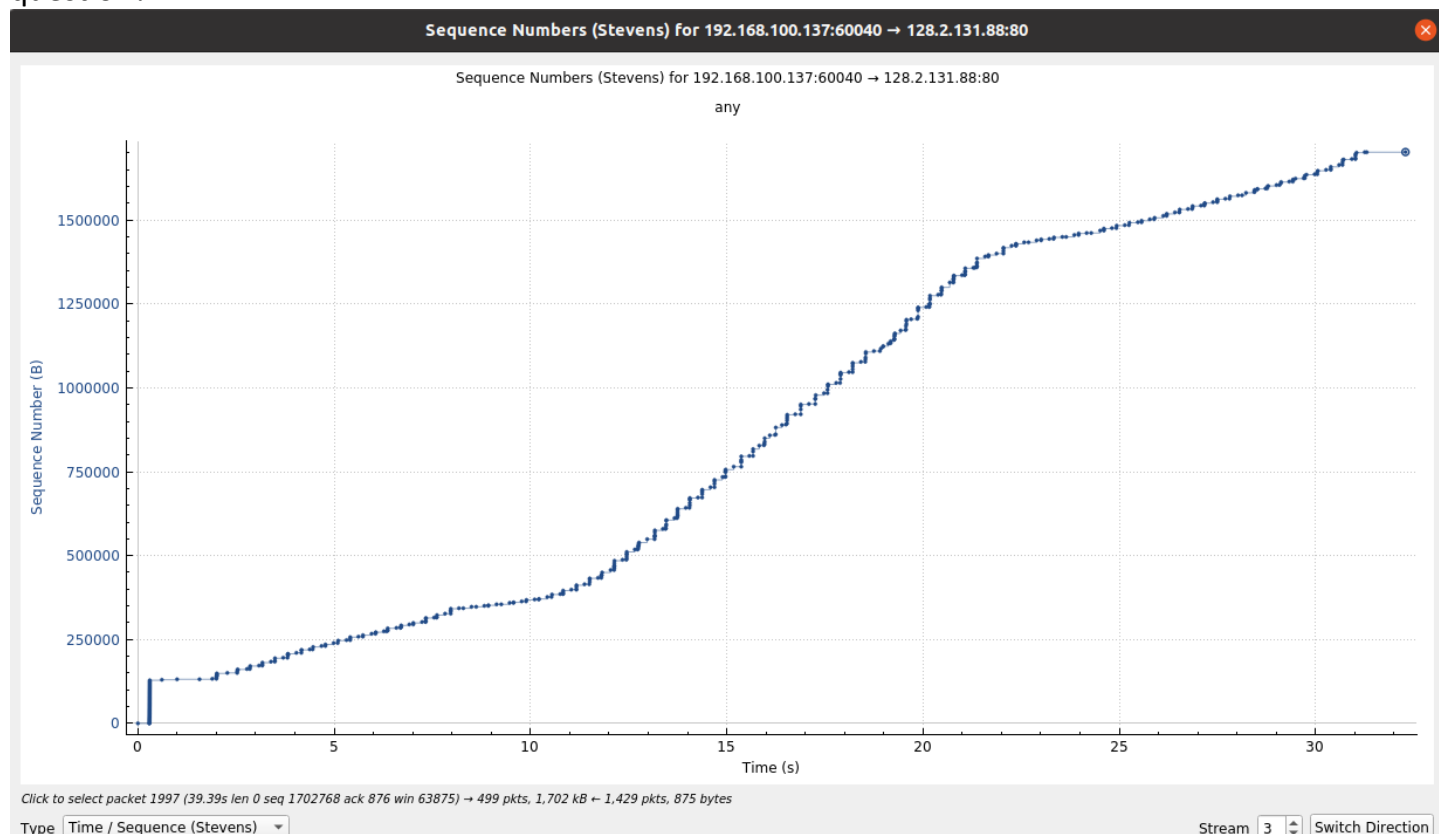
```

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
.... .... = Push: Not set
.... ..0. = Reset: Not set
.... ...1 = Syn: Set
.... .... = Fin: Not set
[TCP Flags: .....A..S.]
Window size value: 64240
[Calculated window size: 64240]

```

The minimum amount of available buffer space advertised at the received for the entire trace is indicated in the first ACK from the server, its value is 64240 bytes (shown in above figure).

Q19) Are there any retransmitted segments? What did you check for (in the trace) to answer this question?



As the sequence number-time graph is monotonically increasing from the above graph, we can conclude that no packet is retransmitted.

OR

tcp.analysis.retransmission						
No.	Time	Source	Destination	Protocol	Length	Info

Q20) How much data does the receiver typically acknowledge in an ACK? Can you identify cases where the receiver is delayed ACKing segments? Explain how or why not

50	6.354191561	128.2.131.88	192.168.100.137	TCP	62	80 → 60028 [ACK] Seq=1 Ack=16061 Win=64240 Len=0
51	6.354191657	128.2.131.88	192.168.100.137	TCP	62	80 → 60028 [ACK] Seq=1 Ack=17521 Win=64240 Len=0
52	6.354302366	128.2.131.88	192.168.100.137	TCP	62	80 → 60028 [ACK] Seq=1 Ack=18981 Win=64240 Len=0
53	6.354302477	128.2.131.88	192.168.100.137	TCP	62	80 → 60028 [ACK] Seq=1 Ack=20441 Win=64240 Len=0

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

The receiver is ACKing every other segment. For example, segment of Number. 51 acknowledged data with 18981-17521=1460 bytes.

Q21) What is the throughput (bytes transferred per unit time) for the TCP connection? Explain how you calculated this value

The file on the hard drive is 17,01,981 bytes, and the download time is 51.057952 (last TCP segment) - 48.441203 (last ACK) = 2.616749 second. Therefore, the throughput for the TCP connection is computed as 17,01,981/2.616749=650418.133340 bytes/second.

Step 2c: Statistics

Q22) What is the most common TCP packet length range? What is the second most common TCP packet length range? Why is the ratio of TCP packets of length < 40 bytes equal to zero? Describe what actions you took to get answers to these questions from Wireshark

Wireshark · Packet Lengths · any								
Topic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate	Burst start
▼ Packet Lengths	2255	827.28	56	14656	0.0501	100%	1.3700	6.352
0-19	0	-	-	-	0.0000	0.00%	-	-
20-39	0	-	-	-	0.0000	0.00%	-	-
40-79	1675	62.17	56	78	0.0372	74.28%	0.9400	6.352
80-159	76	93.89	83	159	0.0017	3.37%	0.0800	42.000
160-319	13	175.92	162	194	0.0003	0.58%	0.0400	42.815
320-639	2	440.00	414	466	0.0000	0.09%	0.0100	41.691
640-1279	1	931.00	931	931	0.0000	0.04%	0.0100	40.256
1280-2559	199	1514.08	1292	1516	0.0044	8.82%	0.1100	8.445
2560-5119	176	3442.86	2976	4843	0.0039	7.80%	0.4200	6.353
5120 and greater	113	7459.36	5896	14656	0.0025	5.01%	0.0700	10.773

the most common TCP packet length range: 40-79

is the second most common TCP packet length range : 1280-2559

The header length is 40 bytes in handshaking stage as it consists of 10 headers so as the minimum packet length is 40 bytes without any data in it, the ratio of tcp packets of length <40 is 0.

This information can be obtained by navigating to ‘statistics<packet lenth’s’ in wireshark menu.

Q23) What average throughput did you use in Mbps? How many packets were captured in the packet capture session? How many bytes in total? Explain your methods.

Average throughput = $1865523/45.046 = 0.041\text{Mbps}$

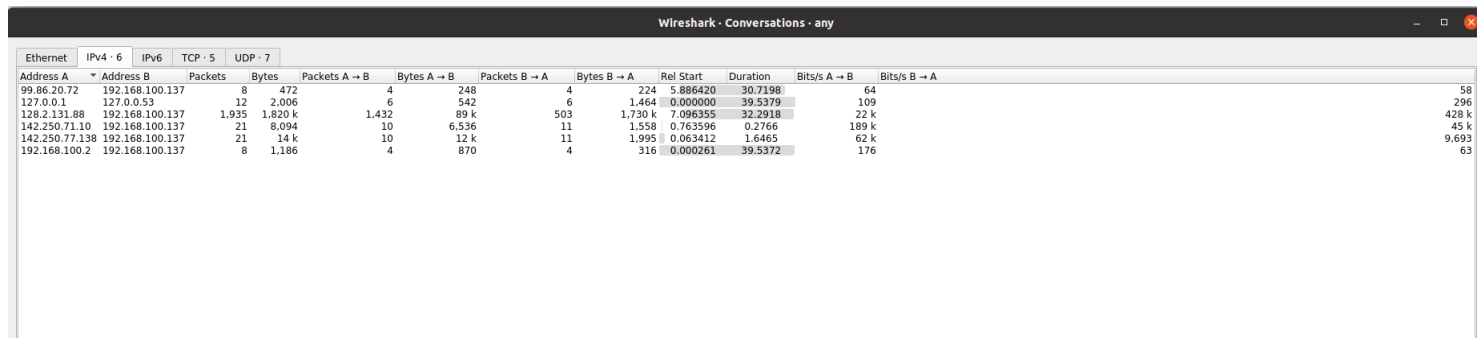
Number of packets captured = 2255

Total number of bytes = 1865523

For this information navigate to ‘statistics->capture file properties’

Statistics			
Measurement	Captured	Displayed	Marked
Packets	2255	2116 (93.8%)	—
Time span, s	45.046	39.309	—
Average pps	50.1	53.8	—
Average packet size, B	827	875	—
Bytes	1865523	1852540 (99.3%)	0
Average bytes/s	41 k	47 k	—
Average bits/s	331 k	377 k	—

Q24) A conversation represents a traffic between two hosts. With which remote host did your local host converse the most (in bytes)? How many packets were sent from your host? How many packets were sent from the remote host?



Wireshark - Conversations - any											
Ethernet		IPv4 - 6	IPv6	TCP - 5	UDP - 7						
Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
99.86.20.72	192.168.100.137	8	472	4	248	4	224	5.886420	30.7198	64	58
127.0.0.1	127.0.0.53	12	2,006	6	542	6	1,464	0.000000	39.5379	109	296
128.2.131.88	192.168.100.137	1,935	1,820 k	1,432	89 k	503	1,730 k	7.096355	32.2918	22 k	428 k
142.250.71.10	192.168.100.137	21	8,094	10	6,536	11	1,558	0.763596	0.2766	189 k	45 k
142.250.77.138	192.168.100.137	21	14 k	10	12 k	11	1,995	0.063412	1.6465	62 k	9,693
192.168.100.2	192.168.100.137	8	1,186	4	870	4	316	0.000261	39.5372	176	63

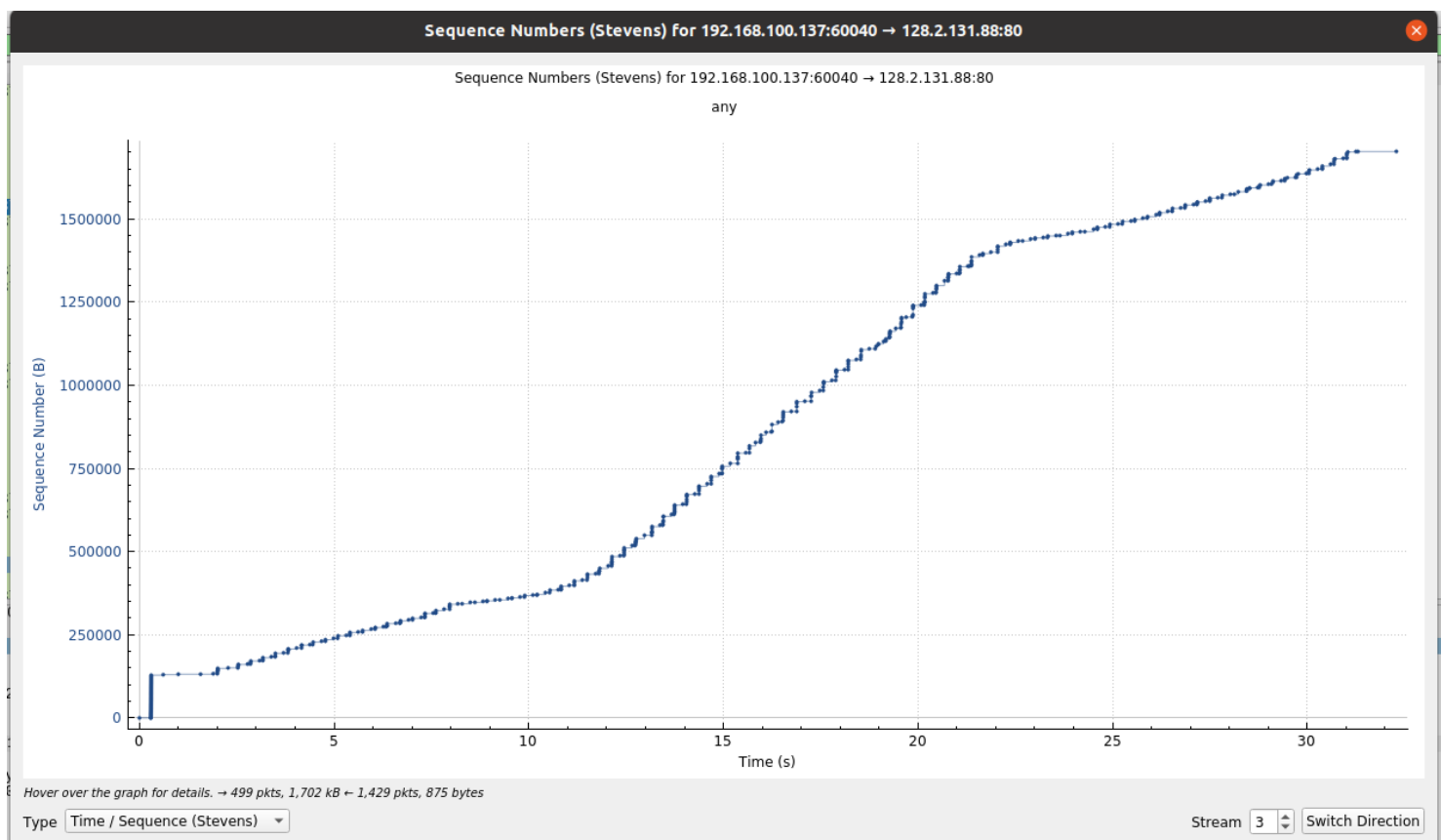
With 128.2.131.88 my localhost conversed the most.(1730Kbytes were transferred)

503 packets were sent from my localhost

1432 packets were sent from the remote host.

Step 3: Congestion Control

Q25). Select a TCP segment in the Wireshark’s “listing of captured-packets” window. Then select the menu: Statistics → TCP Stream Graph → Time-Sequence- Graph (Stevens). You should see a plot that looks like the following plot (though the individual plotted values may differ quite a bit).



Step 4: The Network Layer

Q28) Take a look at the IP section of the DNS query (the packet that was generated when you used dig to request the address of www.pluralsight.com). Match up the header fields with the format we discussed in class (don't just look through Wireshark's display -- instead, match the raw bytes with the pictures we saw in lecture, which I've copied on the right).

No.	Time	Source	Destination	Protocol	Length	Info
14	4.030224380	192.168.100.137	192.168.100.2	DNS	81	Standard query 0x44f1 A www.pluralsight.com
15	4.093980438	192.168.100.2	192.168.100.137	DNS	165	Standard query response 0x44f1 A www.pluralsight.com CNAME ww...

Frame 14: 81 bytes on wire (648 bits), 81 bytes captured (648 bits) on interface any, id 0

Linux cooked capture

- Packet type: Sent by us (4)
- Link-layer address type: 1
- Link-layer address length: 6
- Source: VMware_60:c2:69 (00:0c:29:60:c2:69)
- Unused: 0000
- Protocol: IPv4 (0x0800)
- Internet Protocol Version 4, Src: 192.168.100.137, Dst: 192.168.100.2
 - 0100 = Version: 4
 - 0101 = Header Length: 20 bytes (5)
 - Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
 - Total Length: 65
 - Identification: 0xde43 (56899)
 - Flags: 0x4000, Don't fragment
 - Fragment offset: 0
 - Time to live: 64
 - Protocol: UDP (17)
 - Header checksum: 0x128c [validation disabled]
 - [Header checksum status: Unverified]
 - Source: 192.168.100.137
 - Destination: 192.168.100.2
- User Datagram Protocol, Src Port: 43021, Dst Port: 53
 - Source Port: 43021
 - Destination Port: 53
 - Length: 45
 - Checksum: 0x4a1b incorrect, should be 0xaf2d (maybe caused by "UDP checksum offload"?)
 - [Checksum Status: Bad]
 - [Stream index: 3]
 - [Timestamps]
- Domain Name System (query)
 - Transaction ID: 0x44f1
 - Flags: 0x0100 Standard query
 - Questions: 1
 - Answer RRs: 0
 - Authority RRs: 0
 - Additional RRs: 0
 - Queries
 - [Response In: 15]

Q29) Datagram length: 45

Upper-Layer protocol-IP

Source IP address : 192.168.100.137

Destination IP address : 192.168.100.2

Q31) we discussed the TTL field and determined that we didn't know a good way to set this. What does your OS set this field to?

TTL value = 64
OS = Ubuntu 20.04

Step 5: ICMP

Q33)

```
sundeep@sundeep:~$ traceroute www.cmuj.jp
traceroute to www.cmuj.jp (122.17.163.205), 30 hops max, 60 byte packets
 1  _gateway (192.168.100.2)  0.502 ms  0.576 ms  0.433 ms
 2  * * *
 3  * * *
 4  * * *
 5  * * *
 6  * * *
 7  * * *
 8  * * *
 9  * * *
10  * * *
11  * * *
12  * * *
13  * * *
14  * * *
15  * * *
16  * * *
17  * * *
18  * * *
19  * * *
20  * * *
21  * * *
22  * * *
23  * * *
24  * * *
25  * * *
26  * * *
27  * * *
28  * * *
29  * * *
30  * * *
```

Q32)

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.53	DNS	84	Standard query 0x35ab A www.cmuj.jp OPT
2	0.000054675	127.0.0.1	127.0.0.53	DNS	84	Standard query 0xcea0 AAAA www.cmuj.jp OPT
3	0.000280993	192.168.100.137	192.168.100.2	DNS	73	Standard query 0xb3c7 A www.cmuj.jp
4	0.000542686	192.168.100.137	192.168.100.2	DNS	73	Standard query 0xa38e AAAA www.cmuj.jp
5	0.725109521	192.168.100.2	192.168.100.137	DNS	155	Standard query response 0xb3c7 A www.cmuj.jp CNAME cmuj.jp A ...
6	0.726083712	127.0.0.53	127.0.0.1	DNS	114	Standard query response 0x35ab A www.cmuj.jp CNAME cmuj.jp A ...
7	1.298569536	192.168.100.2	192.168.100.137	DNS	161	Standard query response 0xa38e AAAA www.cmuj.jp CNAME cmuj.jp...
8	1.299283086	192.168.100.137	192.168.100.2	DNS	69	Standard query 0x4644 AAAA cmuj.jp
9	1.307883279	192.168.100.2	192.168.100.137	DNS	69	Standard query response 0x4644 AAAA cmuj.jp
10	1.308342982	127.0.0.53	127.0.0.1	DNS	98	Standard query response 0xcea0 AAAA www.cmuj.jp CNAME cmuj.jp...
11	1.308963909	192.168.100.137	122.17.163.205	UDP	76	44802 → 33434 Len=32 [UDP CHECKSUM INCORRECT]
12	1.309271081	192.168.100.137	122.17.163.205	UDP	76	49237 → 33435 Len=32 [UDP CHECKSUM INCORRECT]
13	1.309444820	192.168.100.2	192.168.100.137	ICMP	104	Time-to-live exceeded (Time to live exceeded in transit)
14	1.309671330	192.168.100.137	122.17.163.205	UDP	76	40120 → 33436 Len=32 [UDP CHECKSUM INCORRECT]
15	1.309828921	192.168.100.2	192.168.100.137	ICMP	104	Time-to-live exceeded (Time to live exceeded in transit)
16	1.309918129	192.168.100.137	122.17.163.205	UDP	76	57824 → 33437 Len=32 [UDP CHECKSUM INCORRECT]
17	1.310087505	192.168.100.2	192.168.100.137	ICMP	104	Time-to-live exceeded (Time to live exceeded in transit)
18	1.310177506	192.168.100.137	122.17.163.205	UDP	76	57927 → 33438 Len=32 [UDP CHECKSUM INCORRECT]
19	1.310358562	192.168.100.137	122.17.163.205	UDP	76	59798 → 33439 Len=32 [UDP CHECKSUM INCORRECT]
20	1.310641675	192.168.100.137	122.17.163.205	UDP	76	55119 → 33440 Len=32 [UDP CHECKSUM INCORRECT]
21	1.311871811	192.168.100.137	122.17.163.205	UDP	76	59103 → 33441 Len=32 [UDP CHECKSUM INCORRECT]
22	1.312161293	192.168.100.137	122.17.163.205	UDP	76	45384 → 33442 Len=32 [UDP CHECKSUM INCORRECT]
23	1.312442937	192.168.100.137	122.17.163.205	UDP	76	37430 → 33443 Len=32 [UDP CHECKSUM INCORRECT]
24	1.312727490	192.168.100.137	122.17.163.205	UDP	76	49077 → 33444 Len=32 [UDP CHECKSUM INCORRECT]
25	1.312992874	192.168.100.137	122.17.163.205	UDP	76	41266 → 33445 Len=32 [UDP CHECKSUM INCORRECT]
26	1.313214126	192.168.100.137	122.17.163.205	UDP	76	33419 → 33446 Len=32 [UDP CHECKSUM INCORRECT]
27	1.313532240	192.168.100.137	122.17.163.205	UDP	76	39264 → 33447 Len=32 [UDP CHECKSUM INCORRECT]
28	1.313762325	192.168.100.137	122.17.163.205	UDP	76	39824 → 33448 Len=32 [UDP CHECKSUM INCORRECT]
29	1.314004682	192.168.100.137	122.17.163.205	UDP	76	47151 → 33449 Len=32 [UDP CHECKSUM INCORRECT]
30	1.314842282	127.0.0.1	127.0.0.53	DNS	99	Standard query 0xe3ad PTR 2.100.168.192.in-addr.arpa OPT
31	1.315572030	192.168.100.137	192.168.100.2	DNS	88	Standard query 0xb65f PTR 2.100.168.192.in-addr.arpa
32	1.355722359	192.168.100.2	192.168.100.137	DNS	88	Standard query response 0xb65f No such name PTR 2.100.168.192...
33	1.356656805	127.0.0.53	127.0.0.1	DNS	121	Standard query response 0xe3ad PTR 2.100.168.192.in-addr.arpa...
34	1.359084540	192.168.100.137	122.17.163.205	UDP	76	47660 → 33450 Len=32 [UDP CHECKSUM INCORRECT]
35	1.359869020	192.168.100.137	122.17.163.205	UDP	76	38160 → 33451 Len=32 [UDP CHECKSUM INCORRECT]

▶ Frame 1: 84 bytes on wire (672 bits), 84 bytes captured (672 bits) on interface any, id 0

▶ Linux cooked capture

▶ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.53

▶ User Datagram Protocol, Src Port: 35679, Dst Port: 53

▶ Domain Name System (query)

0000	00 00 03 04 00 06 00 00	00 00 00 00 00 00 00 00
0010	45 00 00 44 f2 5b 40 00	40 11 4a 17 7f 00 00 01	E..D.[@.@.J.....
0020	7f 00 00 35 8b 5f 00 35	00 30 fe 77 35 ab 01 20	...5...5..0.w5...
0030	00 01 00 00 00 00 00 01	03 77 77 77 04 63 6d 75www.cmuj
0040	6a 02 6a 70 00 00 01 00	01 00 00 29 04 b0 00 00	j.jp.....)
0050	00 00 00 00	

No.	Time	Source	Destination	Protocol	Length	Info
13	1.309444820	192.168.100.2	192.168.100.137	ICMP	104	Time-to-live exceeded (Time to live exceeded in transit)
15	1.309828921	192.168.100.2	192.168.100.137	ICMP	104	Time-to-live exceeded (Time to live exceeded in transit)
17	1.310087595	192.168.100.2	192.168.100.137	ICMP	104	Time-to-live exceeded (Time to live exceeded in transit)

Total Length: 88 Identification: 0x4900 (18688) Flags: 0x0000 0... .. = Reserved bit: Not set .0... .. = Don't fragment: Not set ..0... .. = More fragments: Not set Fragment offset: 0 Time to live: 128 Protocol: ICMP (1) Header checksum: 0xa7c0 [validation disabled] [Header checksum status: Unverified] Source: 192.168.100.2 Destination: 192.168.100.137
Internet Control Message Protocol Type: 11 (Time-to-live exceeded) Code: 0 (Time to live exceeded in transit) Checksum: 0x384a [correct] [Checksum Status: Good] Unused: 00000000
Internet Protocol Version 4, Src: 192.168.100.137, Dst: 122.17.163.205 0100 = Version: 4 0101 = Header Length: 20 bytes (5) Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT) Total Length: 60 Identification: 0x8900 (35072) Flags: 0x0000 0... .. = Reserved bit: Not set .0... .. = Don't fragment: Not set ..0... .. = More fragments: Not set Fragment offset: 0 Time to live: 1 Protocol: UDP (17) Header checksum: 0xeda0 [validation disabled] [Header checksum status: Unverified] Source: 192.168.100.137 Destination: 122.17.163.205
User Datagram Protocol, Src Port: 44802, Dst Port: 33434 Source Port: 44802 Destination Port: 33434 Length: 40 Checksum: 0x95eb [correct] [Calculated Checksum: 0x95eb] [Checksum Status: Good] [Stream index: 1]

Q35) What are the transmitted segments like? Describe the important features of the segments you observe. In particular, examine the destination port field. What characteristics do you observe about this port number and why would it be chosen so?

Transmitted segments are of type ICMP which are mainly used to carry error messages Commonly used ICMP types are echo request and echo reply (used for ping) and time to live exceeded in transit (used for traceroute).

The destination port number is 33434. And The dest port increases by one in every sent packet, indicating that traceroute tries to reach the server in multiple ports, as seen here:

[Checksum Status: Good] Unused: 00000000 Internet Protocol Version 4, Src: 192.168.100.137, Dst: 122.17.163.205 User Datagram Protocol, Src Port: 44802, Dst Port: 33434 Data (32 bytes)
[Checksum Status: Good] Unused: 00000000 Internet Protocol Version 4, Src: 192.168.100.137, Dst: 122.17.163.205 User Datagram Protocol, Src Port: 49237, Dst Port: 33435 Data (32 bytes)
[Checksum Status: Good] Unused: 00000000 Internet Protocol Version 4, Src: 192.168.100.137, Dst: 122.17.163.205 User Datagram Protocol, Src Port: 40120, Dst Port: 33436 Data (32 bytes)

Q36) What about the return packets? What are the values of the various header fields?

Internet Control Message Protocol Type: 11 (Time-to-live exceeded) Code: 0 (Time to live exceeded in transit) Checksum: 0x384a [correct] [Checksum Status: Good] Unused: 00000000 Internet Protocol Version 4, Src: 192.168.100.137, Dst: 122.17.163.205
--

Q37) The ICMP packets carry some interesting data. What is it? Can you show the relationship to the sent packets?

ICMP packet do contain very intresting data values, The alphabets as seen here:

```

▶ Frame 17: 104 bytes on wire (832 bits), 104 bytes captured (832 bits) on interface any, id 0
▶ Linux cooked capture
▶ Internet Protocol Version 4, Src: 192.168.100.2, Dst: 192.168.100.137
▼ Internet Control Message Protocol
  Type: 11 (Time-to-live exceeded)
  Code: 0 (Time to live exceeded in transit)
  Checksum: 0x384a [correct]
  [Checksum Status: Good]
  Unused: 00000000
▶ Internet Protocol Version 4, Src: 192.168.100.137, Dst: 122.17.163.205
▶ User Datagram Protocol, Src Port: 40120, Dst Port: 33436
▼ Data (32 bytes)
  Data: 404142434445464748494a4b4c4d4e4f5051525354555657...
  [Length: 32]

```

```

0000 00 00 00 01 00 06 00 50 56 e6 5a 60 00 00 08 00 .....P V.Z`...
0010 45 00 00 58 49 02 00 00 80 01 a7 c6 c0 a8 64 02 E..XI... ..d.
0020 c0 a8 64 89 0b 00 38 4a 00 00 00 00 45 00 00 3c ..d..8J....E..<
0030 89 02 00 00 01 11 ed 9e c0 a8 64 89 7a 11 a3 cd .....d.z...
0040 9c b8 82 9c 00 28 a8 33 40 41 42 43 44 45 46 47 .....(.3 @ABCDEFG
0050 48 49 4a 4b 4c 4d 4e 4f 50 51 52 53 54 55 56 57 HIJKLMNO PQRSTUW
0060 58 59 5a 5b 5c 5d 5e 5f XYZ[\]^_

```

Q38) Lab1 asserted that ping operates in a similar fashion to traceroute. Use Wireshark to show the degree to which this is true. What differences and similarities are there between the network traffic of ping versus traceroute?

Traceroute can't tell you what happened in the past.

if you experience a slow connection, the Traceroute command that you subsequently issue might not reveal what happened because by that time. The problem that caused the delay may have been fixed and your Traceroute path may not be the same path that the slow connection used.

```

50.00025. 192.168.1.10 122.17.163. ICMP 100 Echo (ping) request id=0x0001, seq=1/256, ttl=64 (reply in 6)
60.11892. 122.17.163. 192.168.1.10 ICMP 100 Echo (ping) reply id=0x0001, seq=1/256, ttl=52 (request in 5)
70.11951. 192.168.1.10 192.168.1.1 DNS 89 Standard query PTR 205.163.17.122.in-addr.arpa
80.12137. 192.168.1.1 192.168.1.10 DNS 114 Standard query response 0x26cf PTR 205.163.17.122.in-addr.arpa PTR
90.89454. 192.168.1.10 142.250.196. UDP 12. 37248 -> 443 Len=1246
100.89463. 192.168.1.10 142.250.196. UDP 12. 37248 -> 443 Len=1250
110.89465. 192.168.1.10 142.250.196. UDP 12. 37248 -> 443 Len=1250
120.89468. 192.168.1.10 142.250.196. UDP 12. 37248 -> 443 Len=1250
130.89470. 192.168.1.10 142.250.196. UDP 11. 37248 -> 443 Len=1142
140.89594. 192.168.1.10 172.217.163. UDP 12. 35668 -> 443 Len=1243
150.89601. 192.168.1.10 172.217.163. UDP 12. 35668 -> 443 Len=1250
160.89603. 192.168.1.10 172.217.163. UDP 12. 35668 -> 443 Len=1250
170.89606. 192.168.1.10 172.217.163. UDP 12. 35668 -> 443 Len=1250
180.89608. 192.168.1.10 172.217.163. UDP 12. 35668 -> 443 Len=1250

```

We can observe the ping request and response followed by a bunch of UDP packets to a fixed port of 443 with random data.