Week #5

Week 5 - Understanding Transport layer and Network Layer

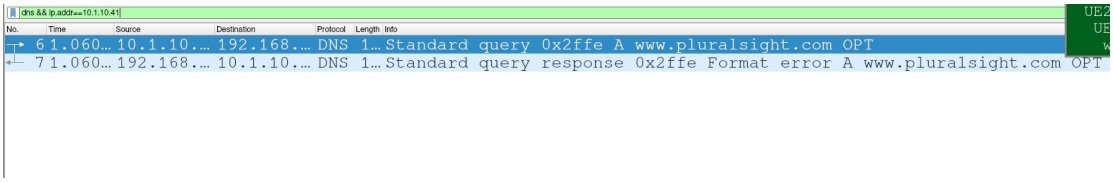
SRN : PES2UG20CS237

Name : P K Navin Shrinivas

Section : D

Task 1 :UDP and DNS

- Opening wireshark and filtering for DNS and UDP signals specific to my ip address.
- Pinging www.pluralsight.com to generate DNS packets.



Filtered packets

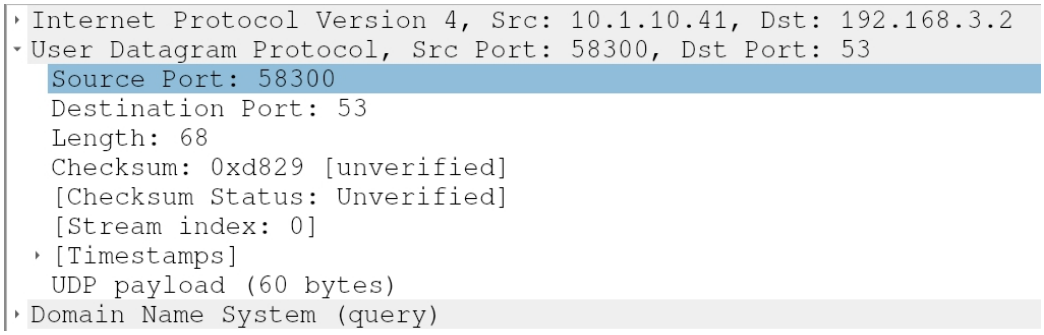
```
navin@usermachine:~ » dig www.pluralsight.com

; <<>> DiG 9.18.1 <<>> www.pluralsight.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: FORMERR, id: 12286
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: ba9a2fe3b83195f2 (echoed)
;; QUESTION SECTION:
;www.pluralsight.com.           IN      A

;; Query time: 0 msec
;; SERVER: 192.168.3.2#53(192.168.3.2) (UDP)
;; WHEN: Thu Mar 31 05:46:32 UTC 2022
;; MSG SIZE  rcvd: 60
```

dig command to generate UDP packets

- (Q) My predictions for UDP structure :

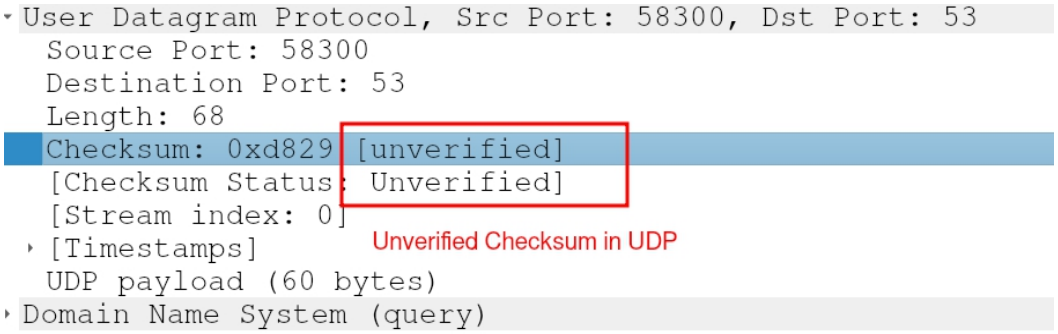
2 bytes for source port	2 bytes for dest port
2 bytes for length	2 bytes for UDP checksum
X bytes for payload	X bytes of payload

- (Q) And in wireshark : Hence concluding predictions were right!



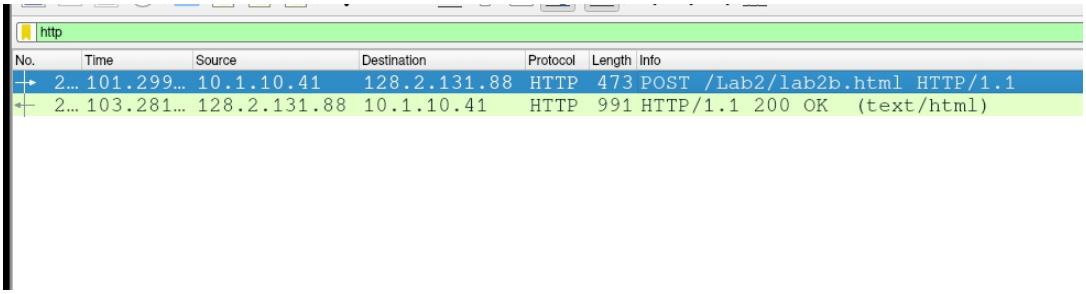
UDP in Wireshark

- Why is the checksum “unverified” in wireshark UDP packets? It is because these UDP packets were genrated by dig, iptrace or tcpdump will calculate these checksums and elad to “verified” status.

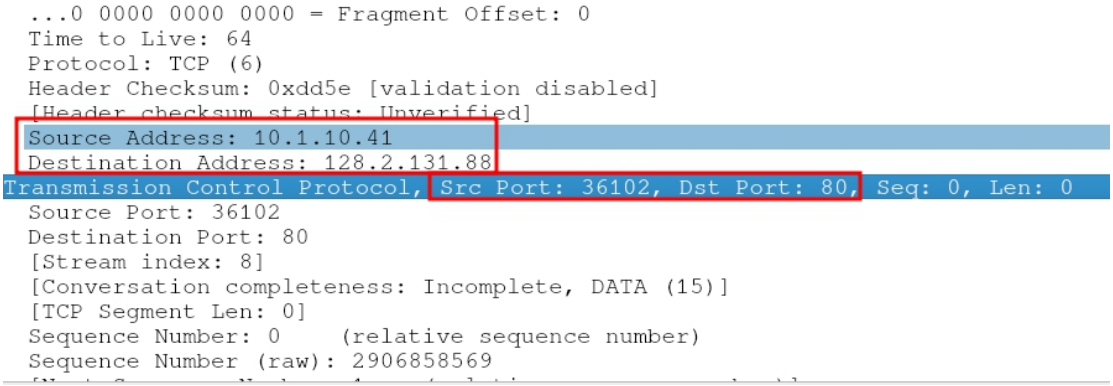


Task 2 :TCP

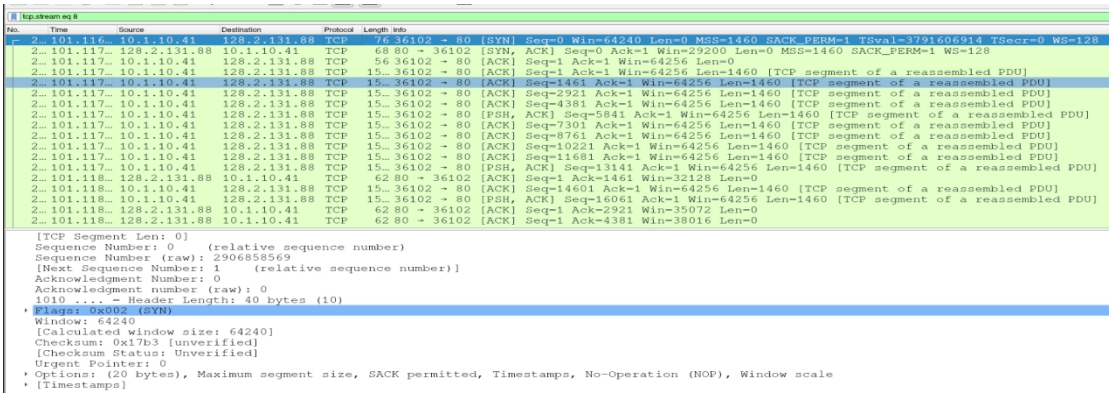
- Download the file from http://www.gutenberg.org/ebooks/2383.txt.utf-8
- Open up wireshark and keep it prime to filter TCP packets from and to http://www.ini740.com/Lab2/lab2a.html
- Upload the files and stop wireshark, wireshark output :



- (Q) What is the source/dest ip address and source port? IP :

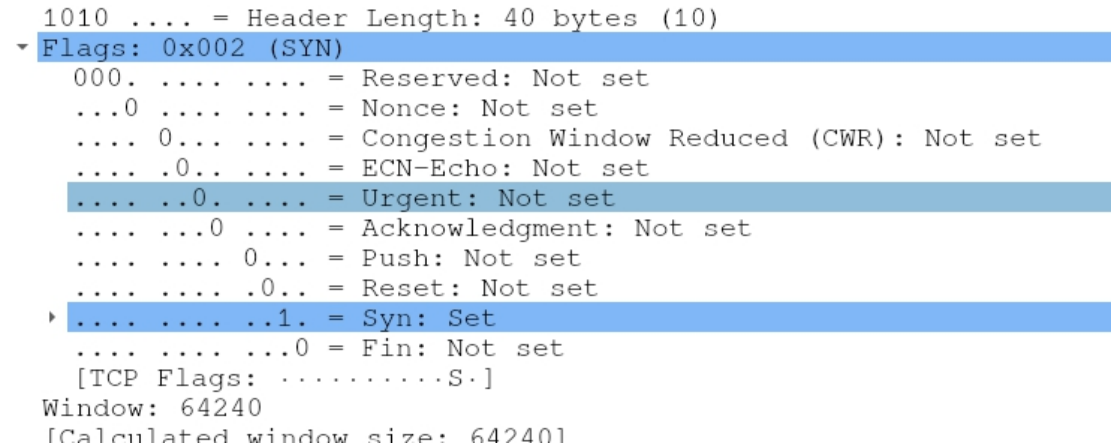


- Point of this task is to observe TCP packets, hence filter for them :

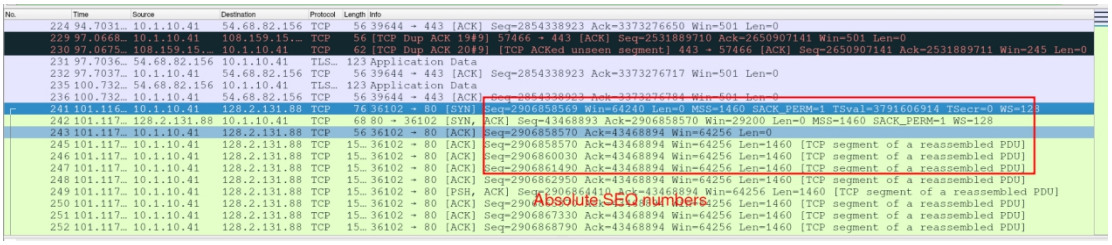


Task 2b : TCP Basics

- (Q14) Sequence number of SYN packets are 0, SYN segments can be identified using the TCP flag segments. Yes wireshark can display absolute seq numbers by : edit->prefrence->protocols->tcp->untick relative numbering.

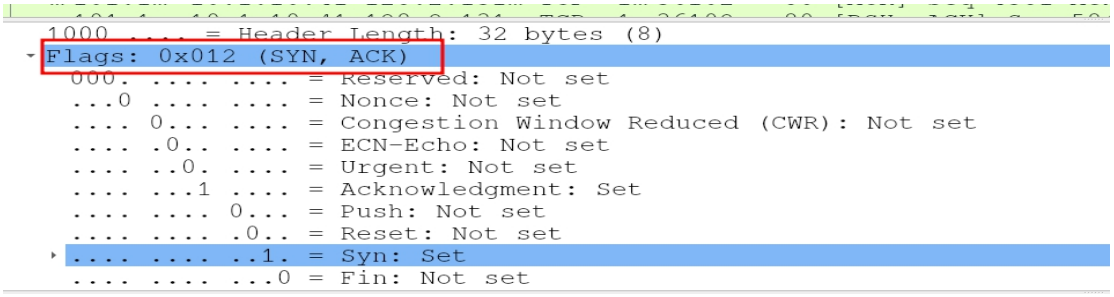


SYN Identification

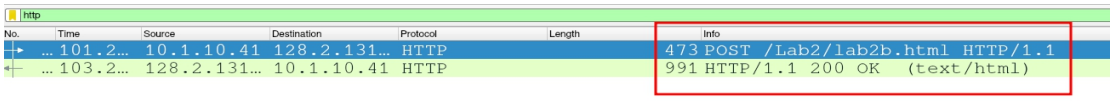


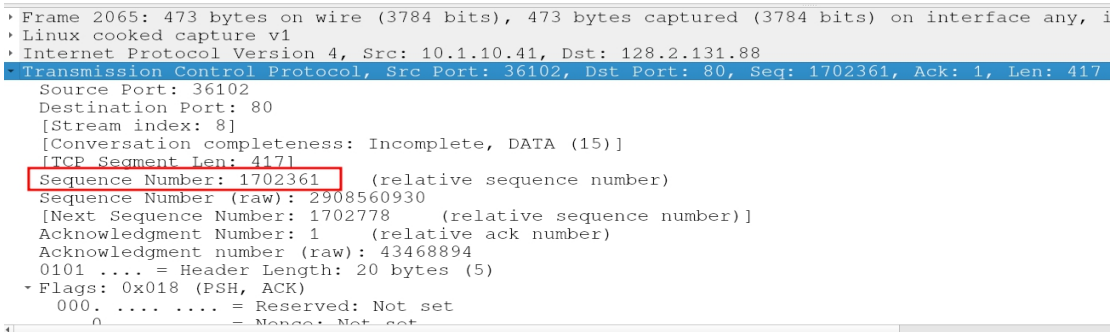
Relative TCP seq numbering

- (Q15) SYNACK has a relative sequence number of 1, ACK number of SYNACK packets is 1, this is because the server is asking for the best bit waiting for the three way handshake.The Flag field in TCP packets shows it is a SYNACK packet:



- (Q16) TCP SEQ number (relative) of HTTP post is : 1702361, this I feel is the entirty of uploaded content and remaining payload (1702278) are for headers.

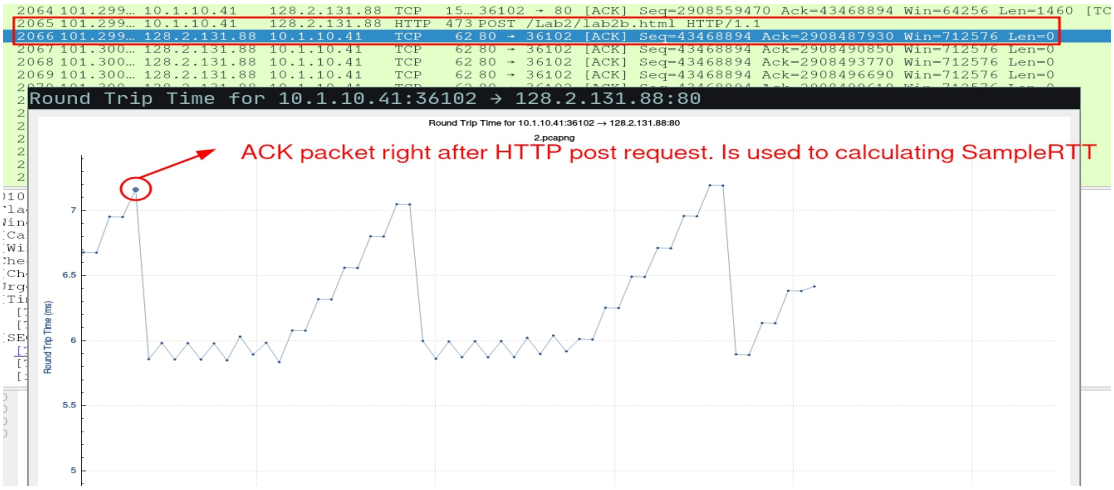




- (Q17)Sample RTT is simply the time taken to ACK a given segment, but to get smoother EstimatedRTT we take many samples from recent packets :

	A	B	C	D
1	RTT From 11 packets after HTTP POST request			
2	Relative SEQ #	Response to SEQ #	RTT as per Wireshark	EstimatedRTT
3	2066	1992	0.005857519	0.005857519
4	2067	1995	0.005854865	0.00585718725
5	2068	1998	0.005853887	0.005856774719
6	2069	2001	0.005848369	0.005855724004
7	2070	2004	0.005892719	0.005860348378
8	2071	2007	0.005834262	0.005857087581
9	2072	2009	0.006075354	0.005884370883
10	2073	2011	0.006315608	0.005938275523
11	2074	2013	0.006557328	0.006015657083
12	2075	2015	0.006799189	0.006113598572
13	2076	2017	0.007047824	0.006230376751
14		SUM of 11 RTT's:	0.067936924	0.06532691974
15		Average/SampleRTT:	0.005643582273	0.005938810886

Source

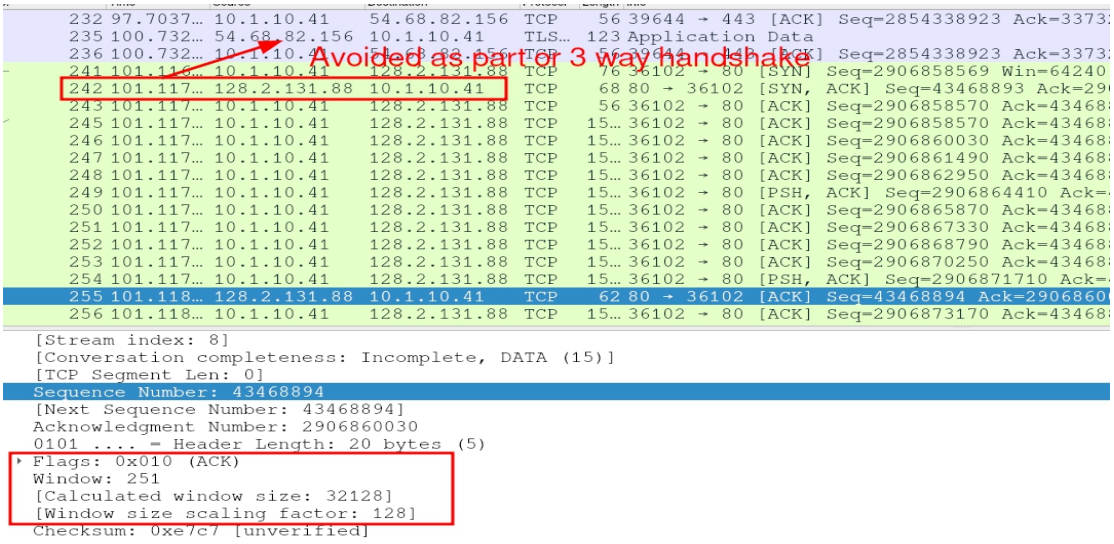


Now calculating the EstimatedRTT (In Table):

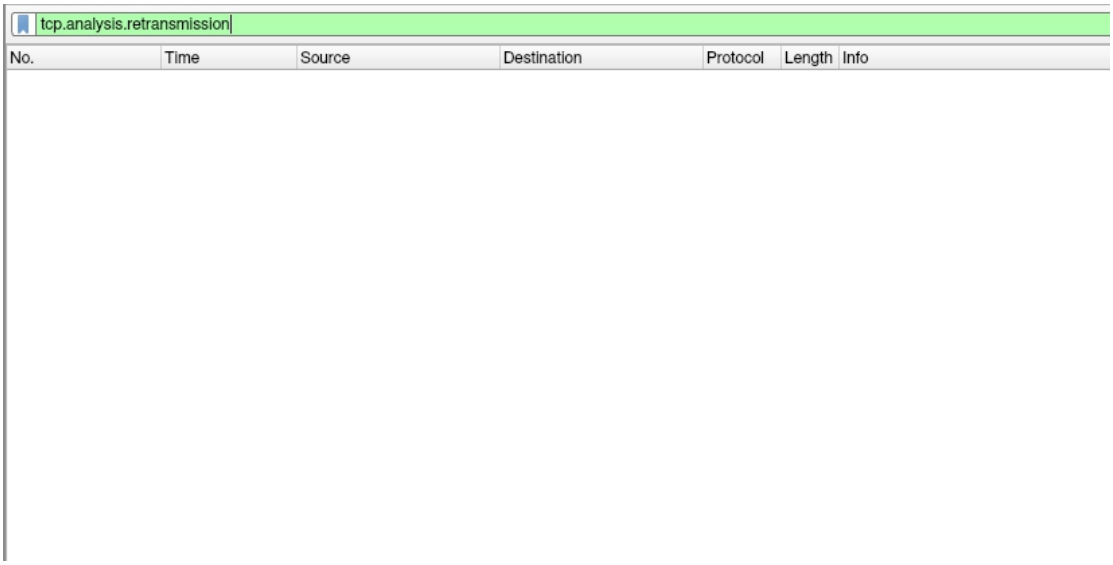
$$\text{EstimatedRTT} = (1-\alpha)*\text{EstimatedRTT} + \alpha*\text{SampleRTT}$$

- Exponential weighted moving average
- influence of past sample decreases exponentially fast
- typical value: $\alpha = 0.125$

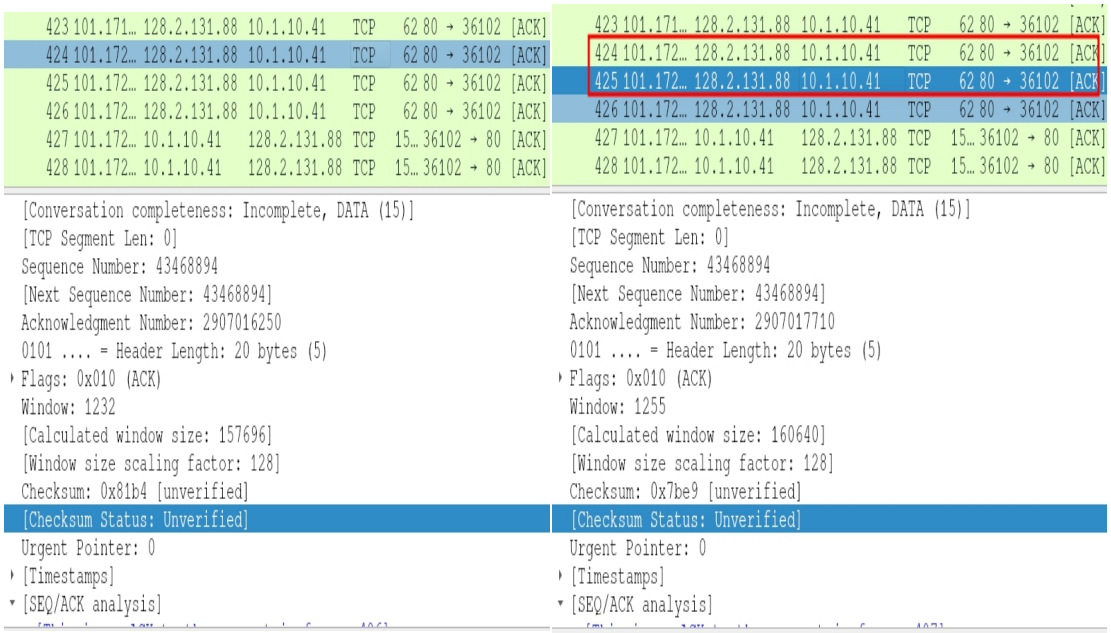
- (Q18)The window size on packets from A to B indicate how much buffer space is available on A for receiving packets. So when B receives a packet with window size 1, it would tell B how many bytes it is allowed to send to A. Hence looking at the window sizes here on reponses (ACK). We also see the windows are scaled with a factor of 128. Lowest observed window size : 32128



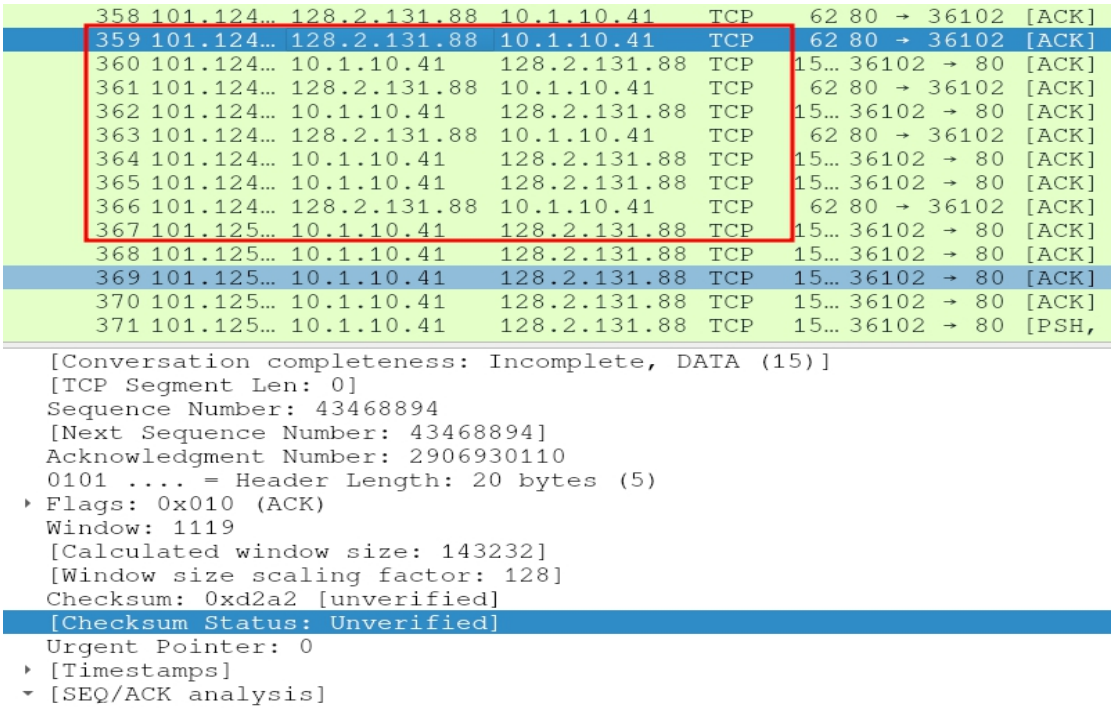
- (Q19)No, there are no retransmitted packets, I checked for these by filtering them using : tcp.analysis.retrasmission :



- (Q20)Comapring Sequence number of 2 consecutive ACK's will give us the answer we need, I am comparing these two :



Subracting the two sequence number gives us : 1460, this the amount of bytes ACK'ed in each ACK response.Also, here we can see where the server is ACK'ing every other packet :



- (Q21)The HTTP reponse has a “time since request” value of 1.98 seconds, the HTTP POST has a content size of 1702219 bytes. This gives us a throughput of : ~840KB/second

Task 2c : Statistics

- (Q22) The most common packet length is 1280-2559 :

Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
Packet Lengths	2111	867.26	44	1516	0.0202	100%	12.9800	101.171
0-19	0	-	-	-	0.0000	0.00%	-	-
20-39	0	-	-	-	0.0000	0.00%	-	-
40-79	880	61.68	44	78	0.0084	41.69%	4.8300	101.171
80-159	60	111.75	80	159	0.0006	2.84%	0.0400	46.801
160-319	3	224.33	183	245	0.0000	0.14%	0.0100	89.279
320-639	1	473.00	473	473	0.0000	0.05%	0.0100	101.300
640-1279	1	991.00	991	991	0.0000	0.05%	0.0100	103.282
1280-2559	1166	1516.00	1516	1516	0.0112	55.23%	8.1700	101.186
2560-5119	0	-	-	-	0.0000	0.00%	-	-
5120 and greater	0	-	-	-	0.0000	0.00%	-	-

- (Q23) Throughput and other statistics :

Measurement	Captured	Displayed	Marked
Packets	2111	2111 (100.0%)	–
Time span, s	104.257	104.257	–
Average pps	20.2	20.2	–
Average packet size, B	867	867	–
Bytes	1830780	1830780 (100.0%)	0
Average bytes/s	17 k	17 k	–
Average bits/s	140 k	140 k	–

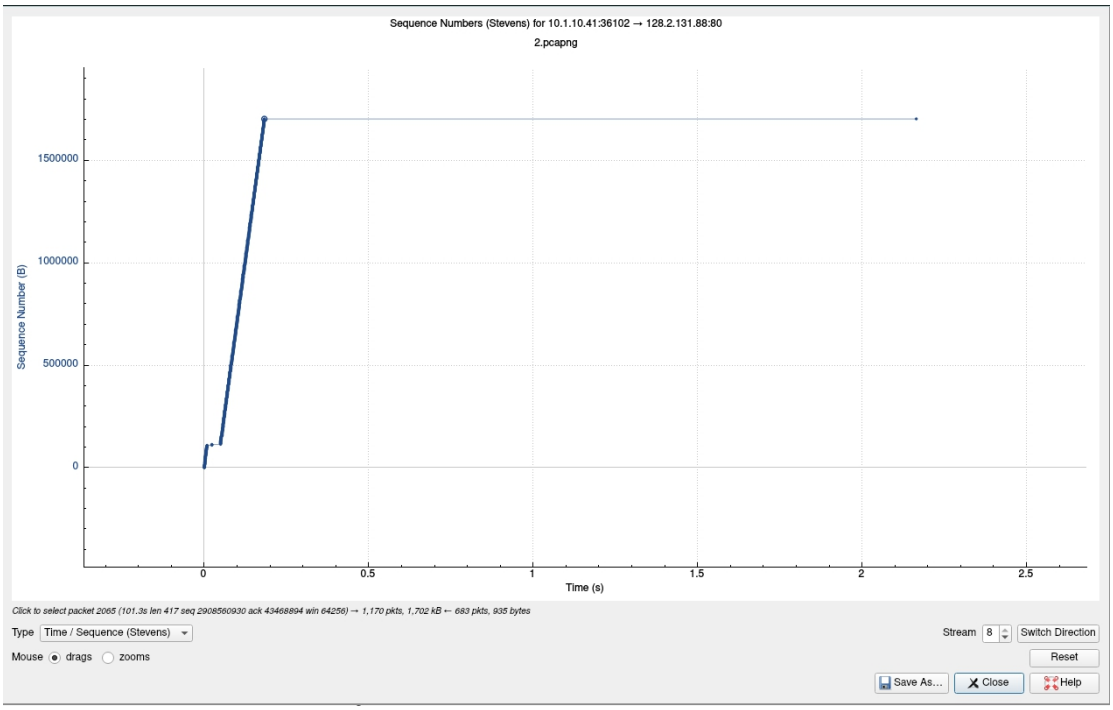
- (Q24)Details about converstation between hosts is as follows :

Ethernet | IPv4 · 12 | IPv6 · 1 | TCP · 9 | UDP · 6

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
10.1.10.21	10.1.10.255	4	527	4	527	0	0	73.851272	29.8901	141	
10.1.10.29	10.1.10.255	1	245	1	245	0	0	89.279014	0.0000	—	
10.1.10.41	192.168.254.1	5	298	2	112	3	186	0.000000	0.5316	1,685	
10.1.10.41	108.159.15.119	24	1,434	13	752	11	682	6.613564	95.1916	63	
10.1.10.41	142.250.195.65	4	308	2	151	2	157	46.801058	0.0365	33 k	
10.1.10.41	142.250.196.34	4	308	2	151	2	157	46.801304	0.0309	39 k	
10.1.10.41	142.250.193.164	4	308	2	151	2	157	47.801861	0.0981	12 k	
10.1.10.41	34.120.115.102	4	308	2	151	2	157	49.803385	0.1036	11 k	
10.1.10.41	34.120.208.123	4	322	2	158	2	164	50.803972	0.0561	22 k	
10.1.10.41	192.168.3.2	10	997	5	384	5	613	101.116150	2.2203	1,383	
10.1.10.41	128.2.131.88	1,853	1,811 k	1,170	1,768 k	683	43 k	101.116860	2.1651	6,533 k	
54.68.82.156	10.1.10.41	70	6,265	35	4,305	35	1,960	1.701126	101.9966	337	

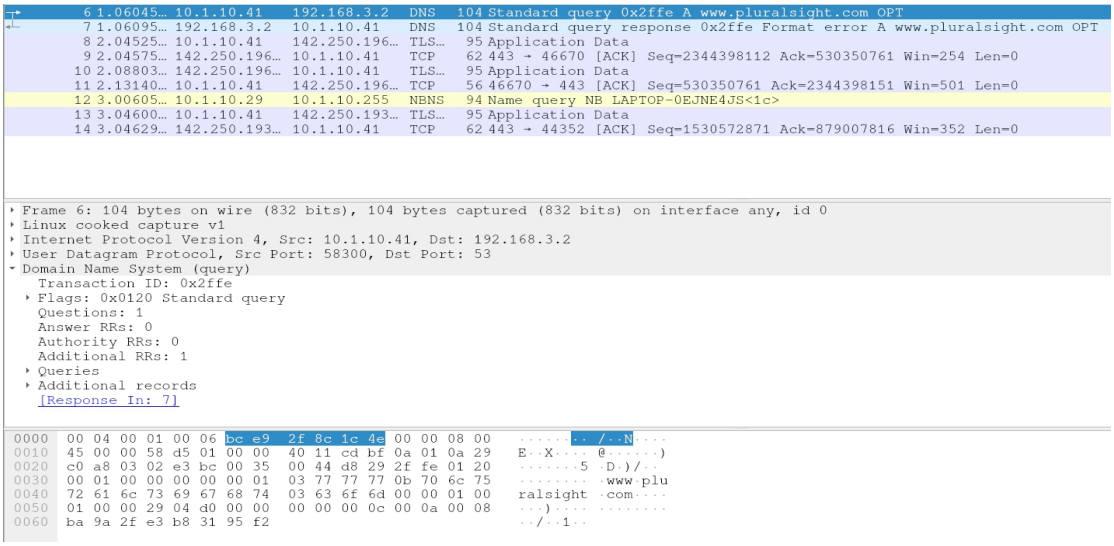
Task 3 : Congestion Control

- (Q25 and Q26)The observed graph using wiresharks TCP graphs is :



Task 4 : Network layer

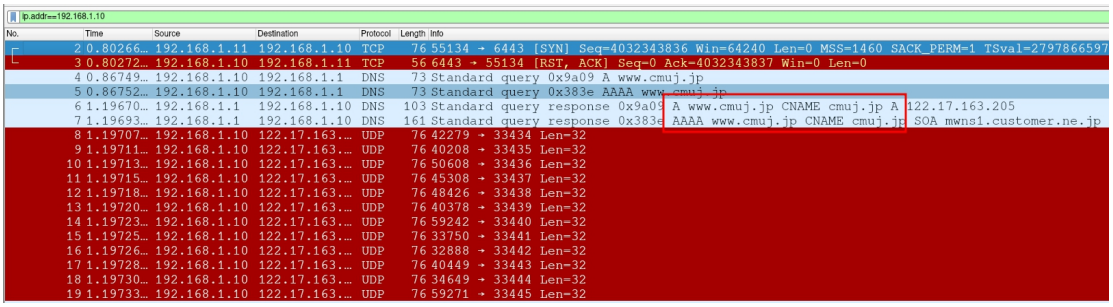
- The DNS query was made with www.pluralsight.com as the Domain name. These were the observed query packets in wireshark



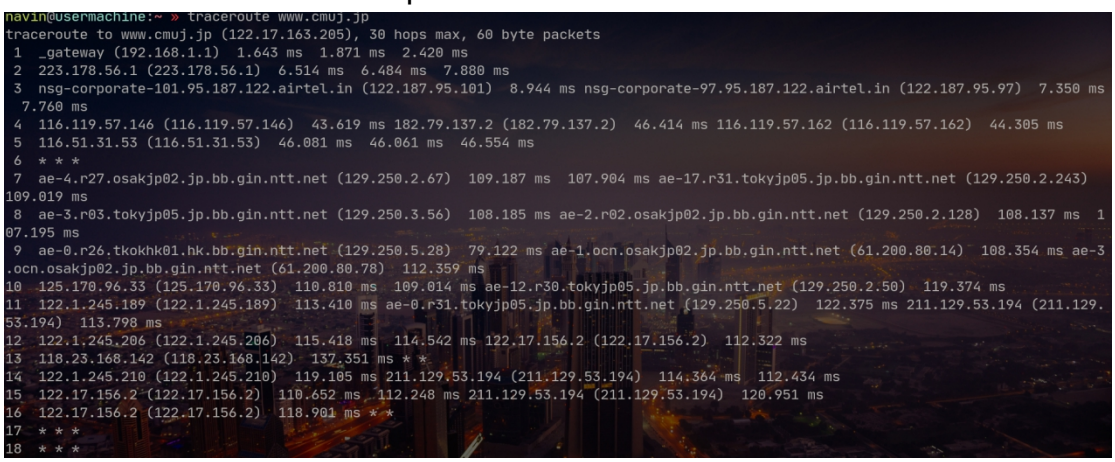
- (Q28 and Q29) All the needed fields do match up
- (Q31) TTL is set as 64, my OS being arch derivative of linux running on linux 5.16.

Task 5 : ICMP

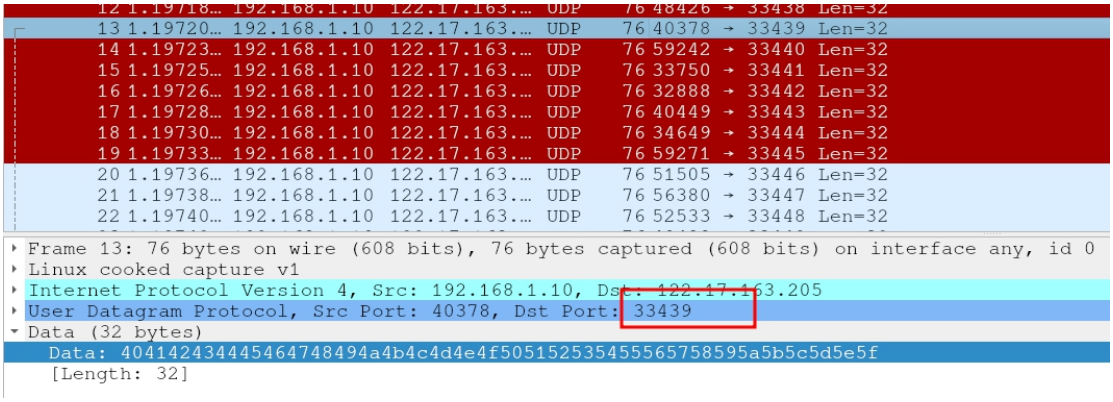
- Starting a new capture when simultaneously pinging : http://www.cmuj.jp/ using the traceroute utility :

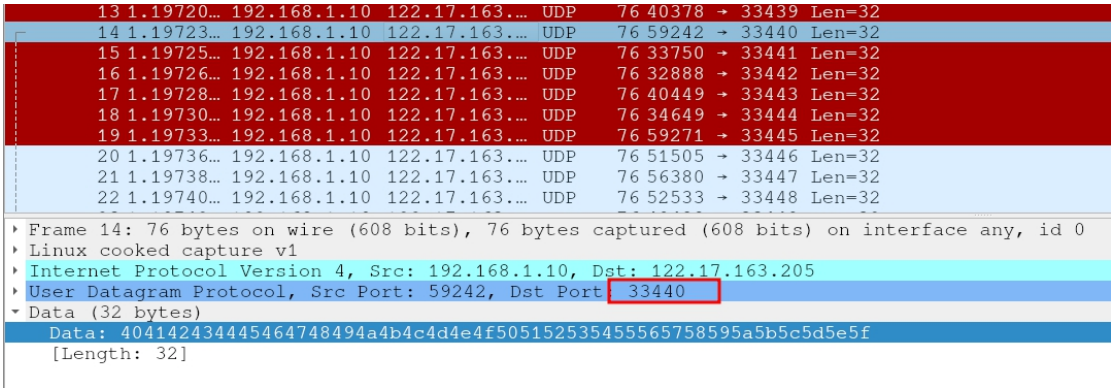


DNS queries of traceroute

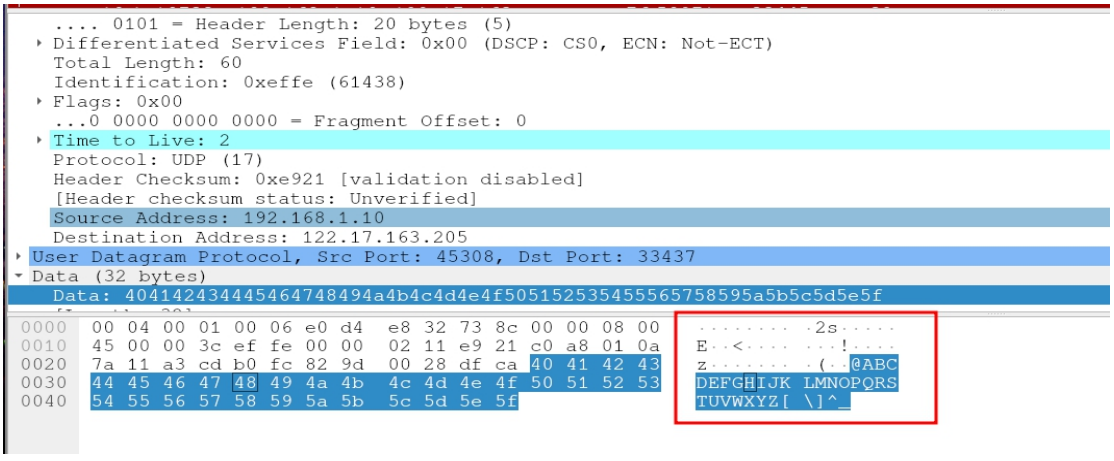


- (Q35) The dest port increases by one in every sent packet, indicating that traceroute tries to reach the server in multiple ports, as seen here:





- My chances at traceroute did not get a reponse back from the server, hence couldn't observe those packet.
- (Q37) ICMP packet do contain very intresting data values, The alphabets as seen here :



- (Q38) We can observe the ping request and reponse followed by a bunch of UDP packets to a fixed port of 443 with random data.