Lab 5 – Understanding Transport and Network Layer using Wireshark
GAURAV MAHAJAN PES1UG20CS150 SEC C

Objective

In this lab, you will continue to use Wireshark, you will explore the transport and network layers. You will examine various UDP, TCP and ICMP transmissions. Write a report, to show you have executed the lab procedures. In this report, also answer any questions that are interleaved among the procedures. Feel free to also include questions, thoughts, and any interesting stuff you observed.

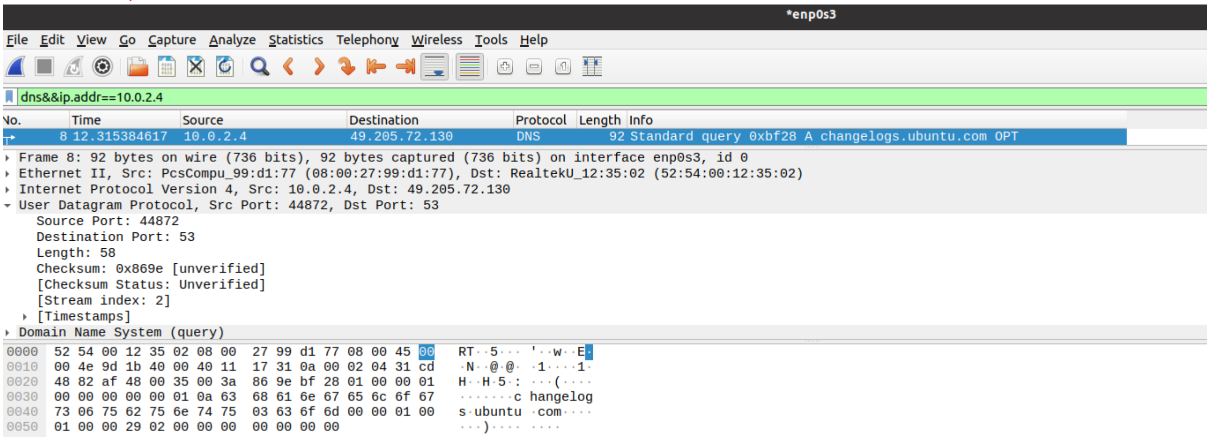
Note: Take screenshots wherever necessary.

Step 1: UDP and DNS

Let's start by examining a few UDP segments. UDP is a streamlined, no-frills transport protocol. All state information is conveyed in each individual UDP segment. In Lab 4, we used dig to generate DNS traffic with the intent of examining the DNS protocol. In this lab, we will use dig to generate DNS traffic, but with the intent of examining the UDP protocol.

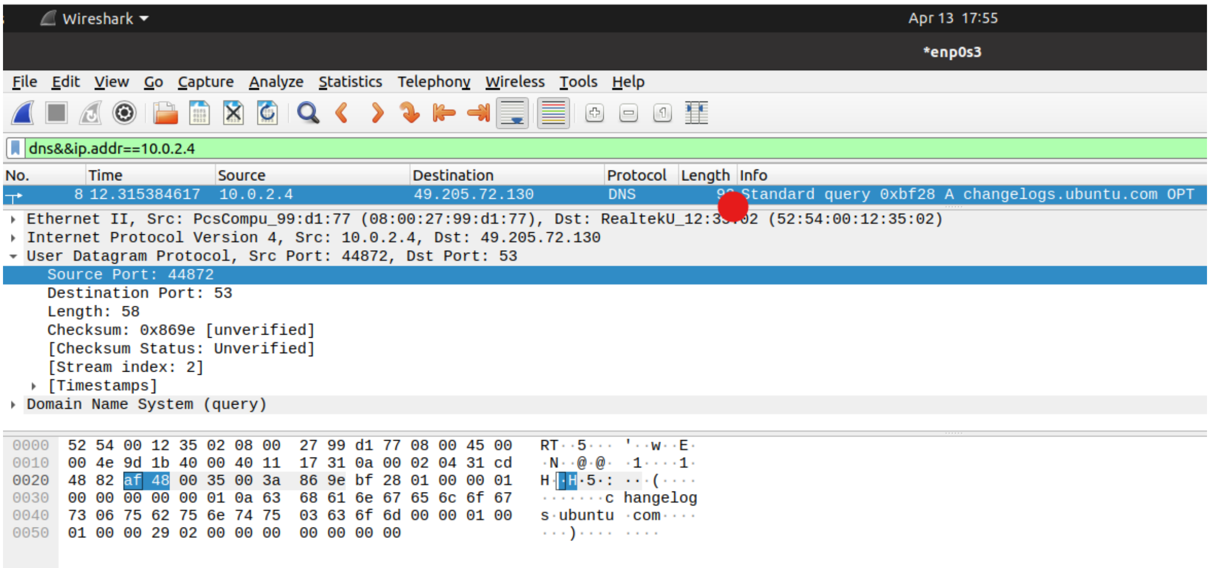
Procedures

1) 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>**).
2) Use dig to generate a DNS query to lookup the domain name "**www.pluralsight.com**". Then, stop the capture.
3) 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.
   A UDP HEADER HAS A SOURCE PORT NUMBER,DESTINATION PORT NUMBER,TOTAL LENGTH AND CHECKSUM FIELDS.EACH FIELD IS OF 2 BYTES.



4) 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 bottom Wireshark window. The bytes at offsets up to number 33-34 are generated by the lower-level protocols. If you click on the "User Datagram Protocol" line in the packet details window, you'll see the UDP contents get highlighted in the packet contents window. 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?
YES

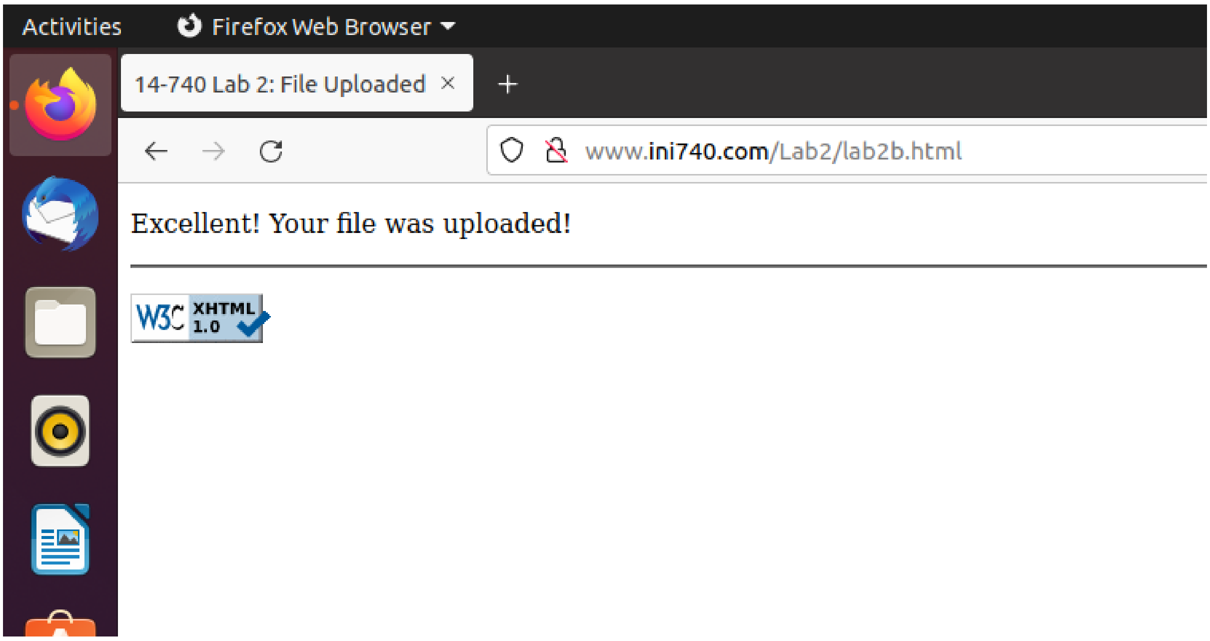


5) Continue to examine the DNS request packet. Which fields does the UDP checksum cover? Wireshark probably shows the UDP checksum as “Validation Disabled”. Why is that?
IF CHECKSUM DETECTS A BIT ERROR ,THEN THE MODERN SYSTEMS PREVALENTLY OFFLOAD THE PACKETS ,SINCE WE ARE TRACING EVERY PACKET IN THE PACKET TRACER ,THE VALIDATION IS DISABLED
6) Save your capture file. Restrict the range of saved packets to only those in the DNS query.

Step 2: TCP

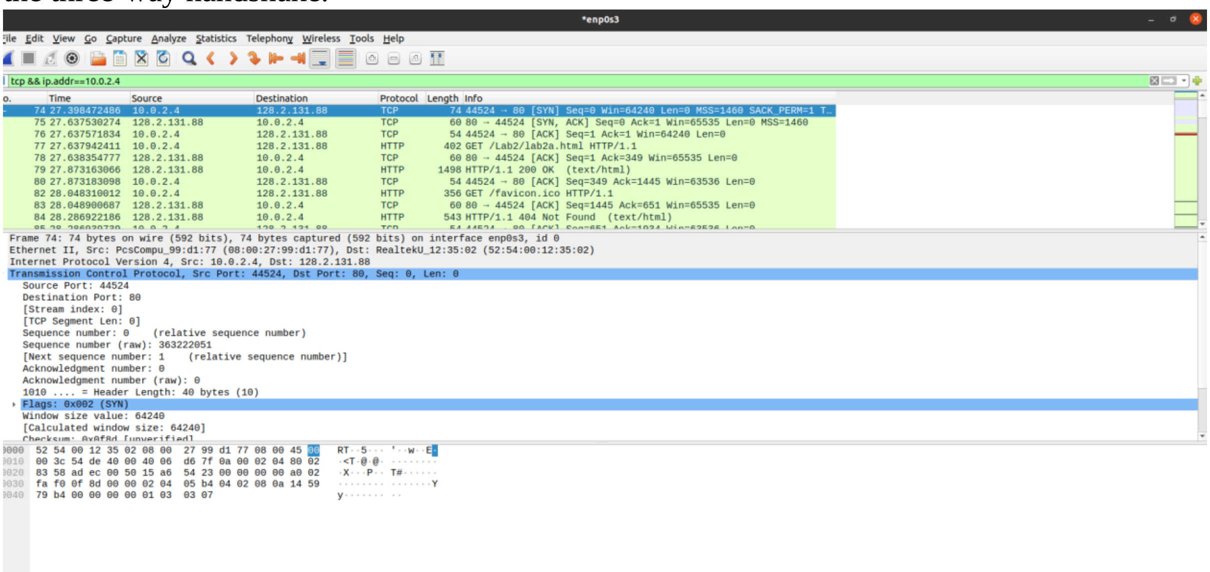
Now, let’s look at another transport protocol, TCP. We will use HTTP to invoke the sort of TCP behaviours we want to study -> I trust that you understand HTTP well enough by now.

7) Download and save a copy of Geoffrey Chaucer's Canterbury Tales and Other Poems from the Project Gutenberg website1. Grab the Plain Text UTF-8 version: **http://www.gutenberg.org/ebooks/2383.txt.utf-8**
8) Clear out Wireshark and start a new capture.
9) Go to the following website. When there, use the form to choose a file (the copy of the Canterbury Tales that you’ve stashed away somewhere on your hard drive) and upload the file. The point of this exercise is to capture a lengthy TCP stream which originates at your computer. [**http://www.ini740.com/Lab2/lab2a.html**](http://www.ini740.com/Lab2/lab2a.html)



10) Stop the Wireshark capture.

11) Let's look at what you captured. First, filter the results to look for TCP packets and to only look at those going to and from your computer with the filter **"tcp && ip.addr == <your IP address>"**. If you have other services running on your computer, you might want to further filter so you only display TCP packets between your computer and the ECE (Electrical and Computer Engineering department of CMU) webserver. What you should see is a series of TCP and HTTP messages between your computer and www.ece.cmu.edu. You should see the initial three-way handshake containing a SYN message. You should see an HTTP POST message and a series of "HTTP Continuation" messages being sent from your computer to the server. HTTP Continuation messages are Wireshark's way of indicating that there are multiple TCP segments being used to carry a single HTTP message. You should also see TCP ACK segments being returned from the server to your computer. Take a screenshot showing the three-way handshake.



12) What is the IP address and TCP port number used by your computer (client) to transfer the file?

THE IP ADDRESS BY MY COMPUTER IS 10.0.2.4 AND THE TCP PORT NUMBER IS 44524

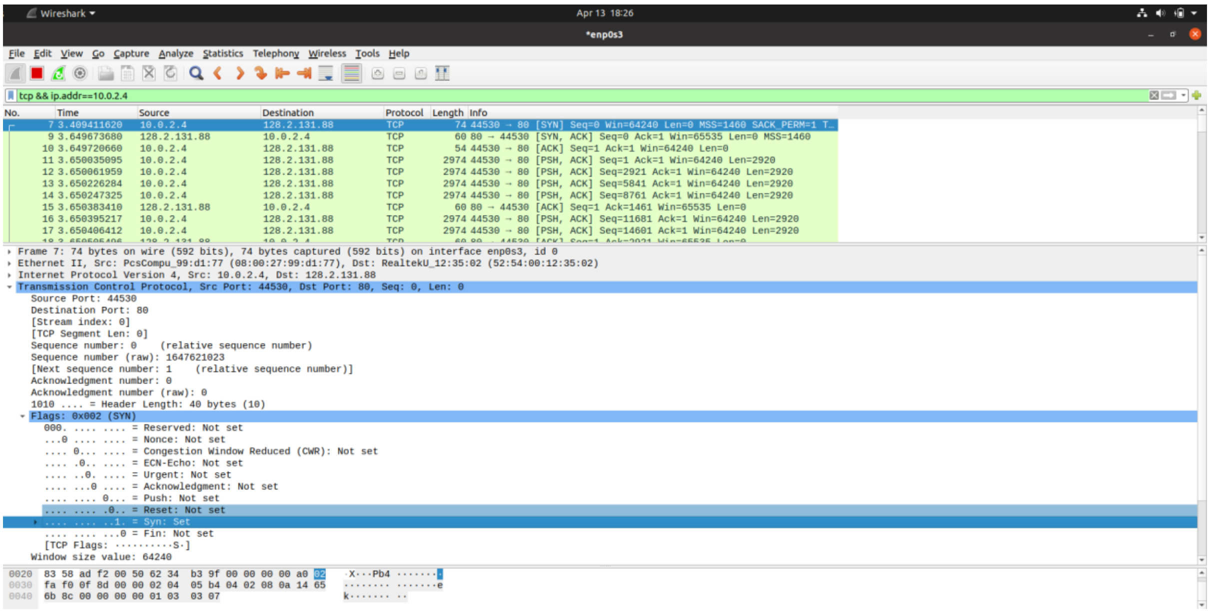
What is the IP address of the server?

IP ADDRESS OF THE SERVER IS 128.2.131.88

On what port number is it sending and receiving TCP segments for this transfer of the file?

IT IS RECEIVING FROM PORT 44524 AND SENDING THROUGH PORT 80

13) Since this lab is about TCP rather than HTTP, let’s change Wireshark’s “listing of captured packets” window so that it shows information about the TCP segments containing the HTTP messages, rather than about the HTTP messages. To have Wireshark do this, select Analyze → Enabled Protocols. Then uncheck the HTTP box and select OK. You should now see a Wireshark window that looks like:



This is what we’re looking for - a series of TCP segments sent between your computer and www.ece.cmu.edu. We will use the packet trace that you have captured to study TCP behaviour in the rest of this lab.

Step 2b: TCP Basics

Answer the following questions for the TCP segments:

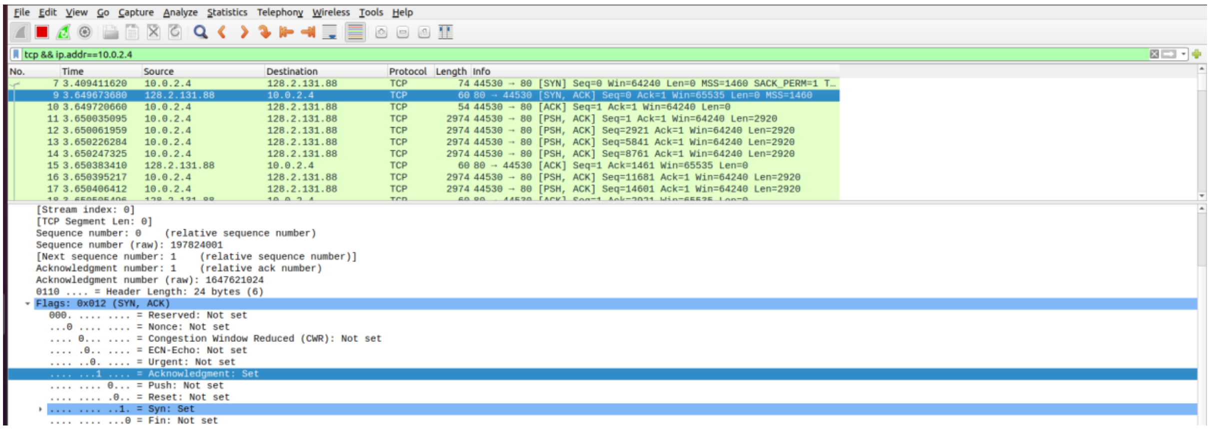
14) What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection? RELATIVE SEQUENCE NUMBER =0 AND RAW SEQUENCE NUMBER IS 1647621023

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

THE FLAG FIELD

Wireshark uses relative sequence numbers by default. Can you obtain absolute sequence numbers instead? How?YES, by going to Edit > Preferences > Protocols > TCP

You can use relative sequence numbers to answer the remaining questions.

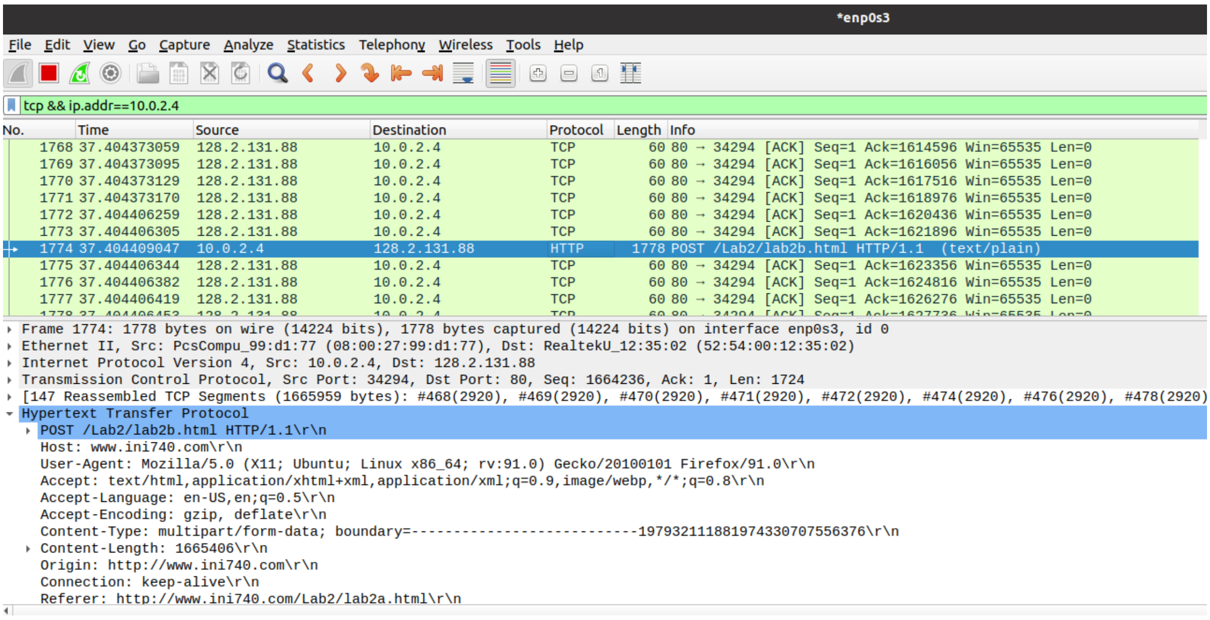


15) What is the sequence number of the SYNACK segment sent by the server in reply to the SYN? RELATIVE SEQUENCE NUMBER IS 0 AND ABSOLUTE SEQUENCE NUMBER IS 197824001

What is the value of the Acknowledgement field in the SYNACK segment? 1

How did the server determine that value?THE SERVER HAS A RELATIVE VALUE OF ZERO INITIALLY BUT BECOMES 1 UPON ACKNOWLEDGEMENT.

What element in the segment identifies it as a SYNACK segment? IF THE FLAG ELEMENT HAS SYN BIT AND ACK BIT SET TO 1,THE SEGMENT IDENTIFIES IT.



16) What is the sequence number of the TCP segment containing the HTTP POST command?THE SEQUENCE NUMBER IS 1664236

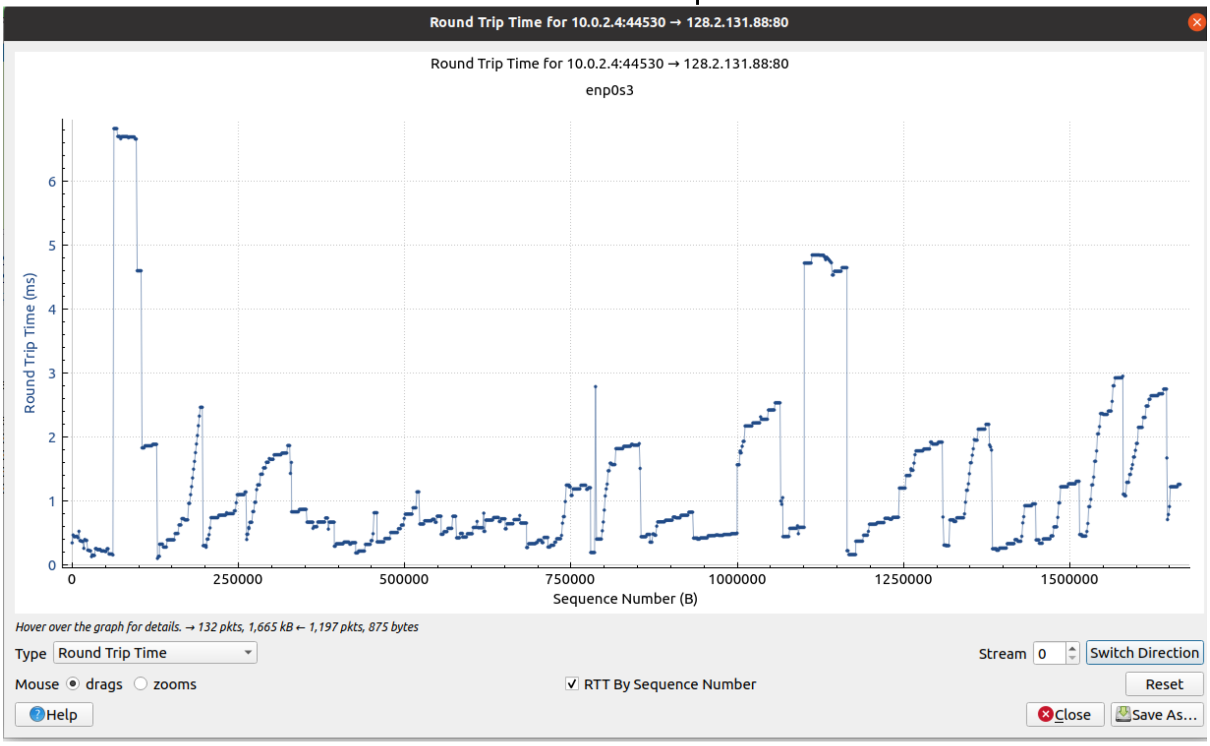
Note that in order to find the POST command, you'll need to dig into the packet content field at the bottom of the Wireshark window, looking for a segment with a "POST" within its DATA field.

17) 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.

Segment	Sent Time	ACK received time	RTT	ESTIMATED RTT
1	37.404409047	37.788304363	0.383895316	0.383895316
2	79.902899660	79.921637263	0.018737603	0.3382506019
3	79.923188164	79.941142533	0.017954369	0.2982135728
4	188.253388937	188.551822797	0.29843386	0.2982411087

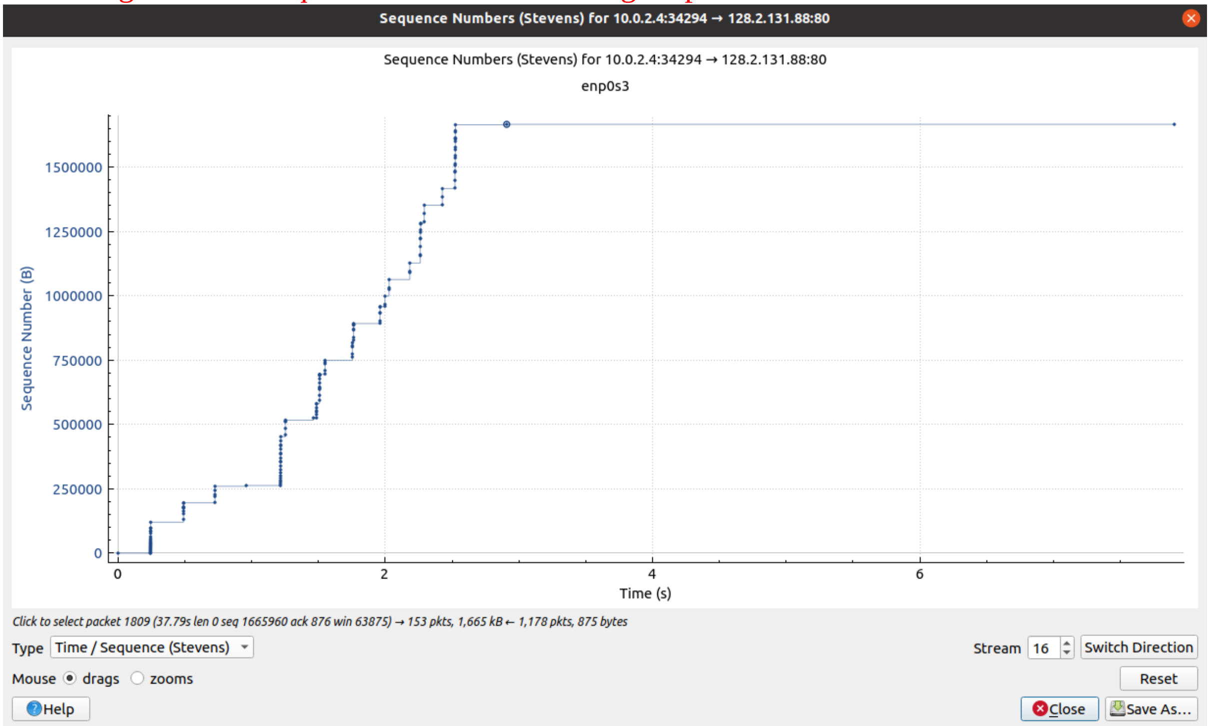
EstimatedRTT = 0.875 * EstimatedRTT + 0.125 *sampleRTT



18) 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?
The minimum amount of available buffer space advertised at the receiver for the entire trace is 64240.
No,the sender doesn't get throttled.

```
.... .... .0.. = Reset: Not set
.... .... ..1. = Syn: Set
.... .... ...0 = Fin: Not set
[TCP Flags: ··········S·]
Window size value: 64240
[Calculated window size: 64240]
Checksum: 0x0f8d [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0
```

19) Are there any retransmitted segments? What did you check for (in the trace) to answer this question? No,there are no retransmitted segments.The stevens plot illustrates the increasing nature of sequence numbers indicating no possible retransmissions.



20) 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.
2921-1461 bytes=1460bytes

21) 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 16,65,192 bytes, and the download time is 3.429156 second. Therefore, the throughput for the TCP connection is computed as
16,65,192 /3.429156 =485598.2205 bytes/second.

Step 2c: Statistics

Wireshark has some fairly robust reporting abilities, most of which are accessed via the Statistics menu. Spend a few minutes messing around with the options on that menu, trying to figure out what each report is telling you. Then, answer the following questions about the Canterbury Tales capture:

Topic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate	Burst start
Packet Lengths	4207	752.96	42	36182	0.0001	100%	2.4300	37.307
0-19	0	-	-	-	0.0000	0.00%	-	-
20-39	0	-	-	-	0.0000	0.00%	-	-
40-79	2843	57.71	42	79	0.0001	67.58%	2.2300	37.307
80-159	547	104.23	81	159	0.0000	13.00%	0.2200	22.979
160-319	234	203.87	160	319	0.0000	5.56%	0.1100	28.268
320-639	95	488.21	341	634	0.0000	2.26%	0.0900	23.002
640-1279	58	865.19	640	1262	0.0000	1.38%	0.0500	23.089
1280-2559	92	1541.50	1308	2193	0.0000	2.19%	0.2700	46576.532
2560-5119	176	3362.10	2582	5094	0.0000	4.18%	0.5900	46576.477
5120 and greater	162	12770.60	5350	36182	0.0000	3.85%	0.4200	46576.532

22) What is the most common TCP packet length range? 40-79 bytes

What is the second most common TCP packet length range? 80-159 bytes

Why is the ratio of TCP packets of length < 40 bytes equal to zero?

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.

Describe what actions you took to get answers to these questions from Wireshark.

Statistics>packet lengths

23) 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.

Details

File

Name: /tmp/wireshark_enp0s3_20220413193532_y2aigl.pcapng
Length: 3,306 kB
Hash (SHA256): 15539d00aababc79790bb37edb7bdc403e11e9e5347284b8569070fa3efe0c6f
Hash (RIPEMD160): 71ee03b10e84a78a9ae4e60261452617ae25f5c8
Hash (SHA1): 46c9734565bc329d13548bfcf7ad59477dd95d4b
Format: Wireshark/... - pcapng
Encapsulation: Ethernet

Time

First packet: 2022-04-13 19:36:13
Last packet: 2022-04-14 09:16:53
Elapsed: 13:40:39

Capture

Hardware: 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz (with SSE4.2)
OS: Linux 5.13.0-28-generic
Application: Dumpcap (Wireshark) 3.2.3 (Git v3.2.3 packaged as 3.2.3-1)

Interfaces

Interface	Dropped packets	Capture filter	Link type	Packet size limit
enp0s3	0 (0.0%)	none	Ethernet	262144 bytes

Statistics

Measurement	Captured	Displayed	Marked
Packets	4207	3412 (81.1%)	—
Time span, s	49239.646	49216.643	—
Average pps	0.1	0.1	—
Average packet size, B	753	904	—
Bytes	3167720	3085744 (97.4%)	0
Average bytes/s	64	62	—
Average bits/s	514	501	—

Capture file comments

The average throughput=3167720/49239.646=514bits=514*10^-6 Mbps

24) 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?

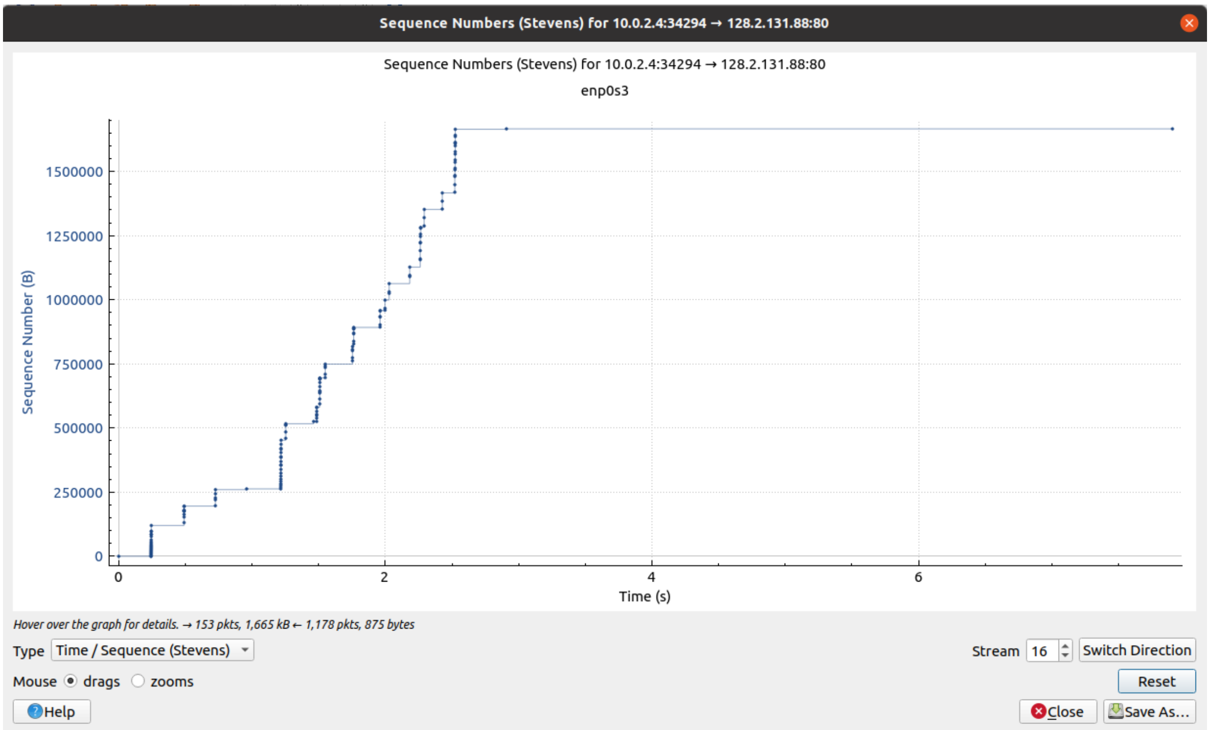
Ethernet · 5 | IPv4 · 36 | IPv6 · 2 | TCP · 62 | UDP · 252

Address A	Port A	Address B	Port 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
10.0.2.4	33156	13.33.171.92	443	31	6,855	16	1,807	15	5,048	23.003368	170.9399	84	
10.0.2.4	32878	34.107.221.82	80	47	3,886	24	1,914	23	1,972	23.003524	172.8986	88	
10.0.2.4	41162	54.230.90.55	443	43	9,020	22	2,170	21	6,850	23.003649	116.5926	148	
10.0.2.4	41164	54.230.90.55	443	27	9,088	14	2,888	13	6,200	23.003780	0.1709	135 k	
10.0.2.4	36514	128.30.52.100	80	10	1,500	5	712	5	788	23.003950	2.4208	2,352	
10.0.2.4	42082	142.250.195.138	443	31	11 k	16	2,344	15	8,785	23.004081	2.1383	8,769	
10.0.2.4	32880	34.107.221.82	80	47	3,726	24	1,918	23	1,808	23.056663	172.8458	88	
10.0.2.4	34292	128.2.131.88	80	10	1,881	5	712	5	1,169	23.550643	5.7740	986	
10.0.2.4	38380	34.215.40.77	443	34	7,367	17	2,463	17	4,904	26.123145	1833.5701	10	
10.0.2.4	40982	34.120.237.76	443	67	41 k	34	4,595	33	37 k	26.123383	172.8702	212	
10.0.2.4	42652	34.120.208.123	443	21	6,023	11	1,406	10	4,617	26.157827	2.1349	5,268	
10.0.2.4	42654	34.120.208.123	443	74	13 k	37	6,290	37	6,858	26.157905	172.8393	291	
10.0.2.4	41166	54.230.90.55	443	45	12 k	23	2,299	22	10 k	29.185366	116.8795	157	
10.0.2.4	55626	103.16.203.232	80	33	3,201	17	1,359	16	1,842	29.217571	116.8473	93	
10.0.2.4	49090	142.250.195.195	80	33	3,017	17	1,362	16	1,655	29.217590	115.8018	94	
10.0.2.4	56868	117.18.237.29	80	41	5,999	21	2,420	20	3,579	32.313274	115.6146	167	
10.0.2.4	34294	128.2.131.88	80	1,331	1,745 k	153	1,674 k	1,178	71 k	34.879528	7.9057	1,694 k	
10.0.2.4	34230	34.122.121.32	80	13	1,041	8	599	5	442	173.004740	15.5475	308	
10.0.2.4	51404	185.125.188.60	443	36	36 k	17	8,487	19	28 k	190.506733	5.9338	11 k	
10.0.2.4	59820	35.224.170.84	80	10	819	5	377	5	442	473.020332	0.5642	5,345	
10.0.2.4	40986	34.120.237.76	443	60	35 k	30	4,416	30	31 k	738.664819	171.1141	206	
10.0.2.4	56786	35.232.111.17	80	10	819	5	377	5	442	772.992932	0.4784	6,304	
10.0.2.4	59822	35.224.170.84	80	10	819	5	377	5	442	1073.043206	0.4774	6,317	
10.0.2.4	56788	35.232.111.17	80	11	873	6	431	5	442	1372.990885	0.4809	7,169	
10.0.2.4	56790	35.232.111.17	80	10	819	5	377	5	442	1673.040499	0.4870	6,193	
10.0.2.4	42084	142.250.195.138	443	21	6,588	11	2,232	10	4,356	1827.634934	0.1006	177 k	
10.0.2.4	39138	34.210.39.83	443	23	6,634	12	2,090	11	4,544	1850.466946	45101.6029	0	
10.0.2.4	56870	117.18.237.29	80	11	1,859	6	766	5	1,093	1853.842867	0.2619	23 k	
10.0.2.4	41168	54.230.90.55	443	21	7,046	11	1,701	10	5,345	1853.843330	0.2715	50 k	
10.0.2.4	59824	35.224.170.84	80	10	819	5	377	5	442	1972.998121	0.5404	5,580	
10.0.2.4	40988	34.120.237.76	443	51	7,360	26	4,406	25	2,954	2076.412174	44661.8344	0	

Step 3: Congestion Control

Let's now examine the amount of data sent per unit time from the client to the server. Rather than (tediously!) calculating this from the raw data in the Wireshark window, we'll use one of Wireshark's TCP graphing utilities - Time-Sequence-Graph (Stevens) - to plot our data.

25) 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).



Here, each dot represents a TCP segment sent, plotting the sequence number of the segment versus the time at which it was sent. Note that a set of dots stacked above each other represents a series of packets that were sent back-to-back by the sender. Don't be distraught if your graph doesn't look like that shown above. Recall that the particular algorithms for managing congestion control can be implemented (or not) based on the OS you are running.

26) Use the Time-Sequence-Graph (Stevens) plotting tool to view the sequence number versus time plot of segments being sent. Can you identify where TCP's slowstart phase begins and ends, and where congestion avoidance takes over? Comment on ways in which the measured data differs from the idealized behavior of TCP that we've studied in the text. Make sure to include a copy of the plot in your report.

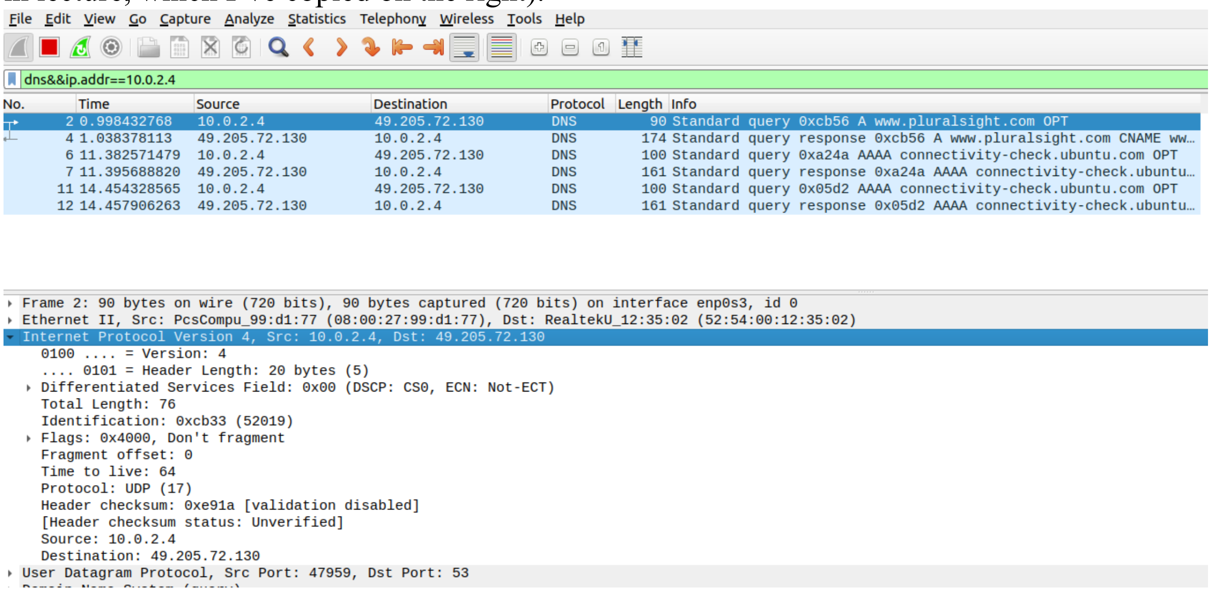
It can be observed that the sequence number graph starts at a time not equal to 0,indicating that some time was spent prior for handshaking.

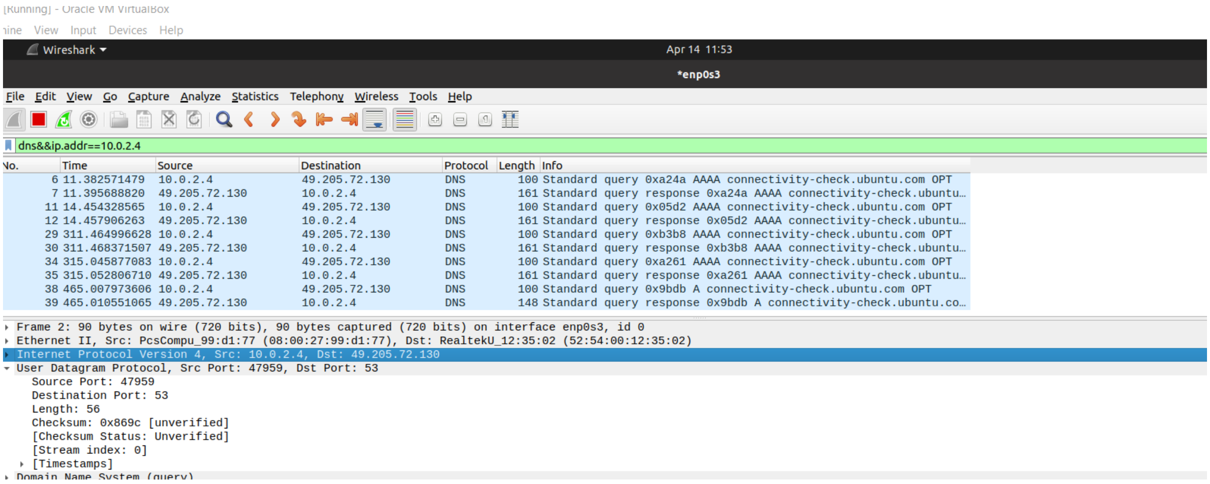
Step 4: The Network Layer

Let's take this opportunity to check out a bit of IP traffic. We don't have to capture any additional traffic, as everything we've seen today is carried over IP packets.

27) Load the capture file that you saved in step 1. Recall that this was a simple DNS query, carried in a UDP packet.

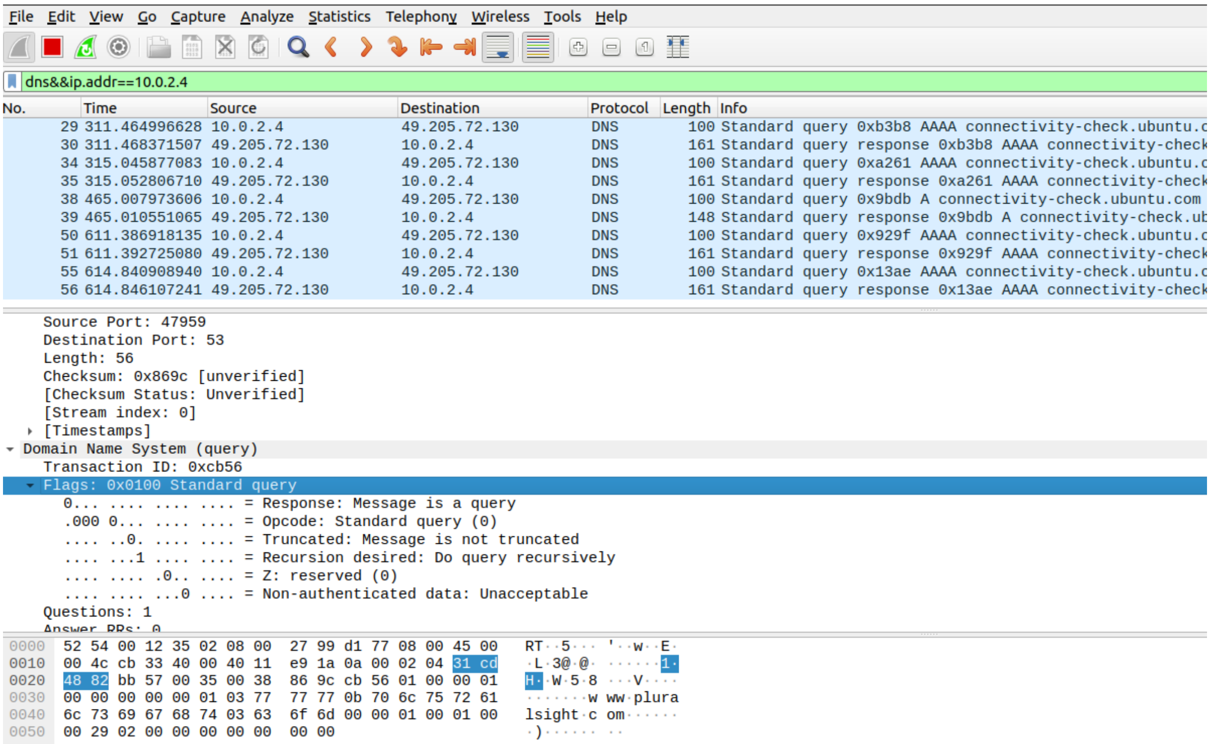
28) Take a look at the IP section of the DNS query (the packet that was generated when you used dig to request the address of **www.pluralsight.com**).
Match up the header fields with the format we discussed in class (don't just look through Wireshark's display -- instead, match the raw bytes with the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





Most of the fields should match up and make perfect sense. Verify the Datagram Length, Upper-layer protocol and the IP address fields. 56,IP,Source address is 10.0.2.4

29) Are there any interesting features of the data in the identifier/flags/offset fields?



30) In class, 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? 64

BTW, please document in this question what your OS and OS version are. Ubuntu 20.04.3 LTS

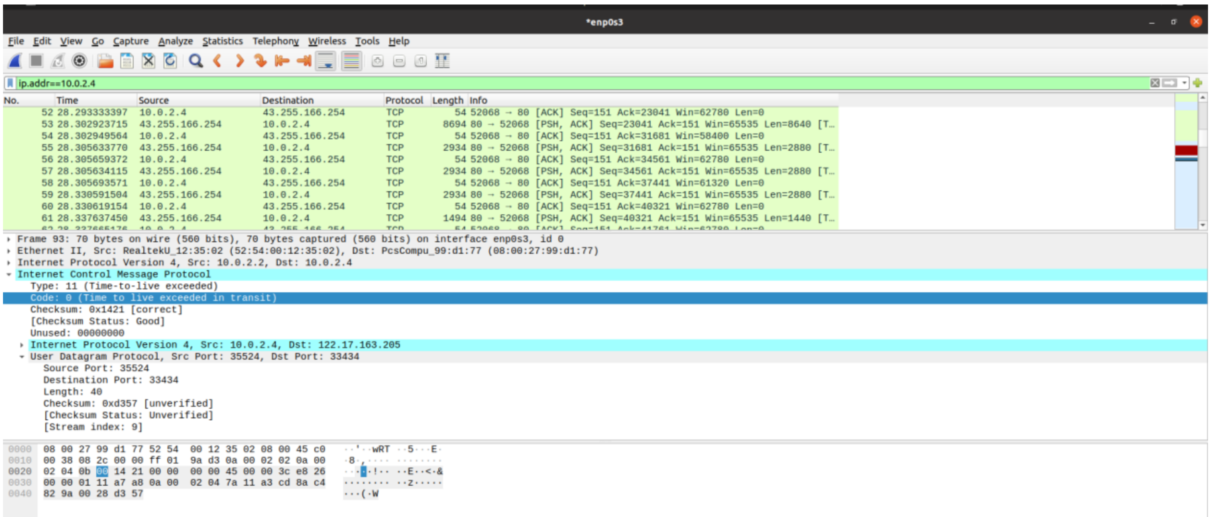
Step 5: ICMP

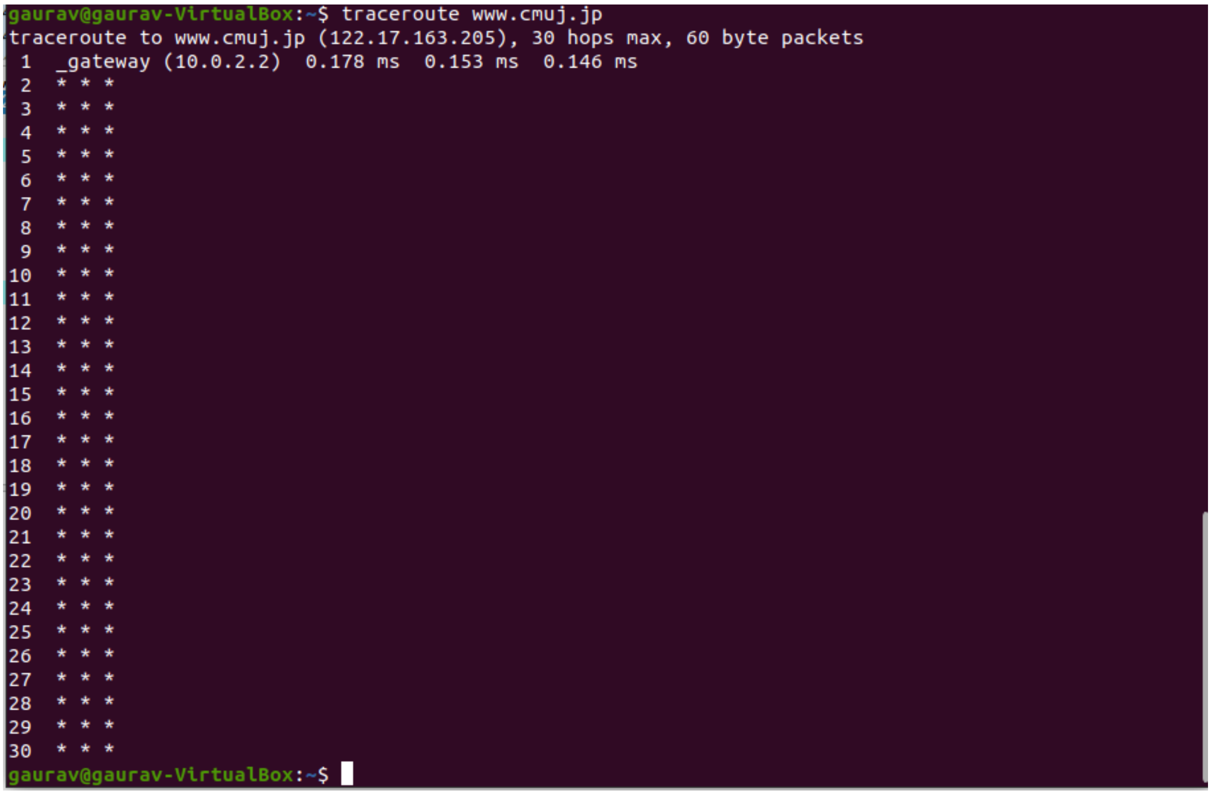
The Network Layer uses ICMP to send information about the network. Some would say that ICMP is a higher-layer protocol, as the actual ICMP packet is carried inside an IP packet. Let's take a look at how that works.

32) Start a new capture, with the display filter showing only packets sent to or from your computer (i.e. "**ip.addr==<ip address>**")

33) In a terminal window, execute the traceroute utility to trace from your computer to **www.cmuj.jp** or **www.regjeringen.no** or some other far-away destination (like we did in our class). If you are having trouble with the weird traceroutes, try this from a non-campus location (your home, a restaurant, etc). Do whatever you can to get a traceroute consisting of about a dozen steps.

34) Stop the capture and take a look at what you found.





35) 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?

The destination port number is 33435.The characteristics observed are that with every hop the destination port number gets incremented by 1.

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

Type: 11 (Time-to-live exceeded)
Code: 0 (Time to live exceeded in transit)
Checksum: 0x1421 [correct]
[Checksum Status: Good]
Unused: 00000000

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

Timestamps

38) 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?

Ping is used to check whether an IP address is accessible or not.

Ping basically works by sending a packet to a specific address and then waits for a reply.

Traceroute's main functionality is to track data packets from your computer to the internet host.

Traceroute works by tracking packets sent by your pc to the destination server.

Ping and traceroute both are primarily used to test the network connectivity issues. Ping does the same with a direct approach to the host or IP while traceroute approaches the host or IP gathering information at each node it passes through.