Name: Adithya M **Section: K**

SRN:PES1UG20CS621

Lab 5 – Understanding Transport and Network Layer using Wireshark

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.

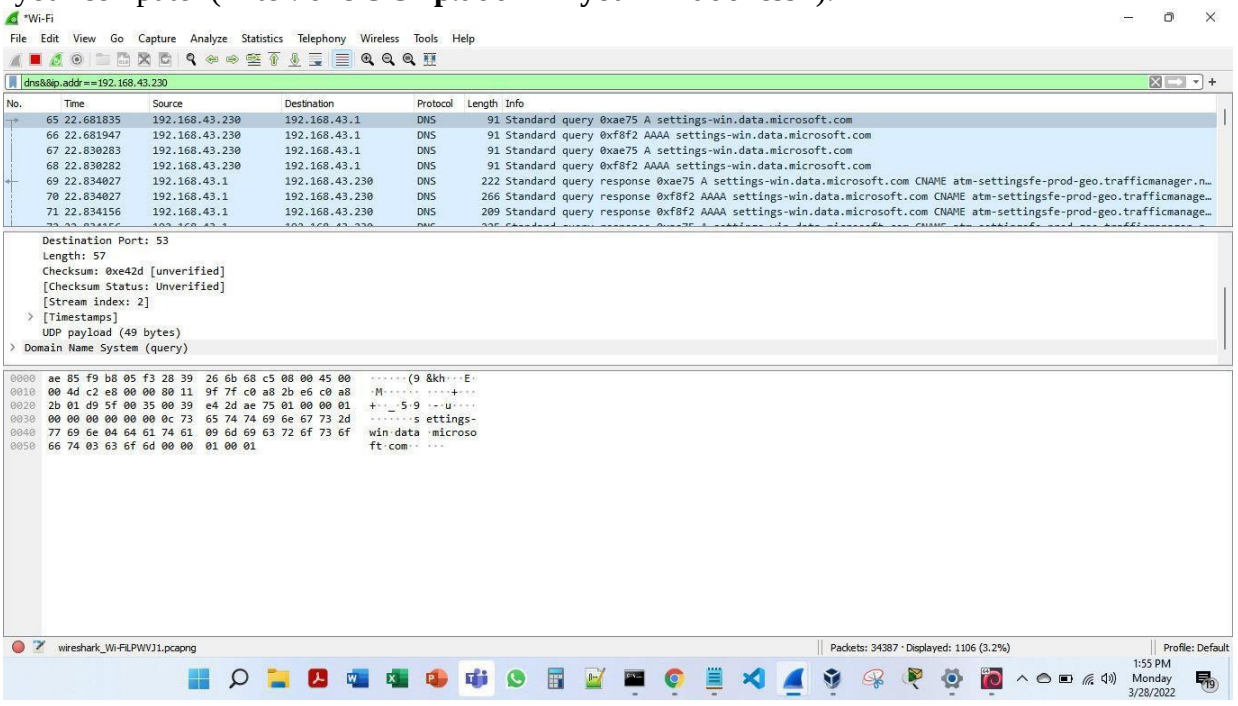
Note: Take screenshots wherever necessary.

Step 1: UDP and DNS

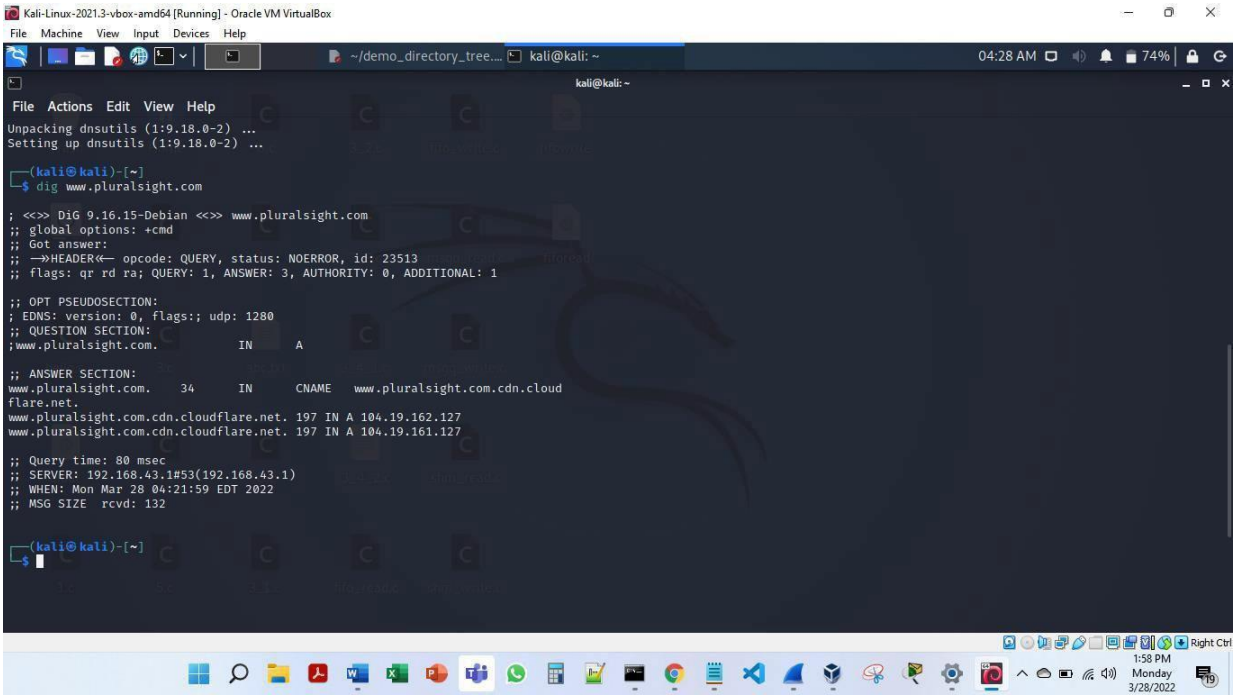
Let's start by examining a few UDP segments. UDP is a streamlined, no-frills transport protocol. All state information is conveyed in each individual UDP segment. In Lab 4, we used dig to generate DNS traffic with the intent of examining the DNS protocol. In this lab, we will use dig to generate DNS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

Procedures

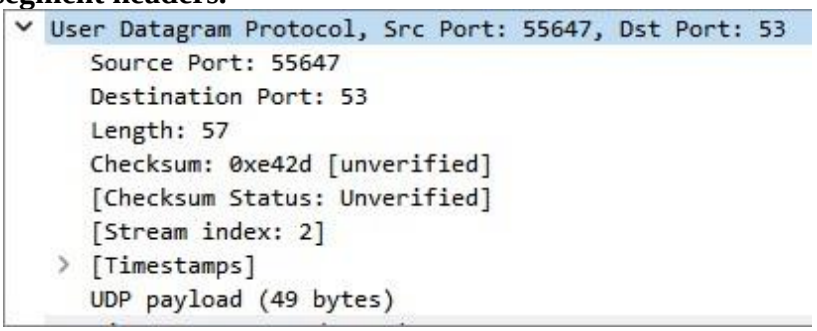
1) Open Wireshark and set up our privacy filter so that you display only DNS traffic to or from your computer (Filter: **dns && ip.addr==<your IP address>**).



2) Use dig to generate a DNS query to lookup the domain name "**www.pluralsight.com**". Then, stop the capture.

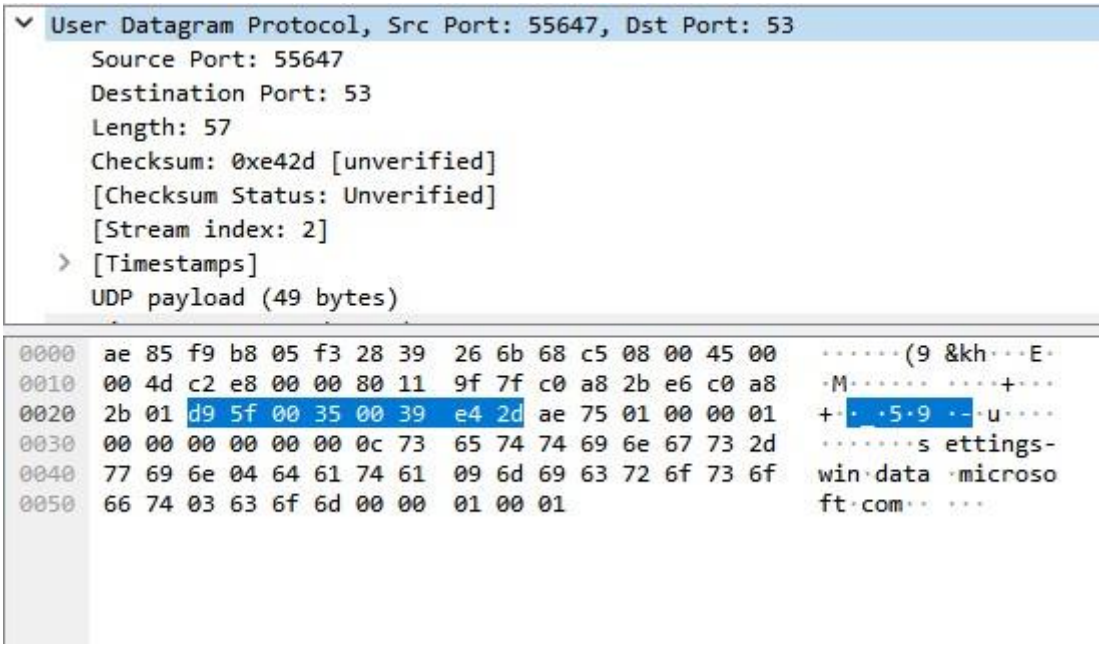


3) **Before you look at the packets in Wireshark, think for a minute about what you expect to see as the UDP segment headers.**

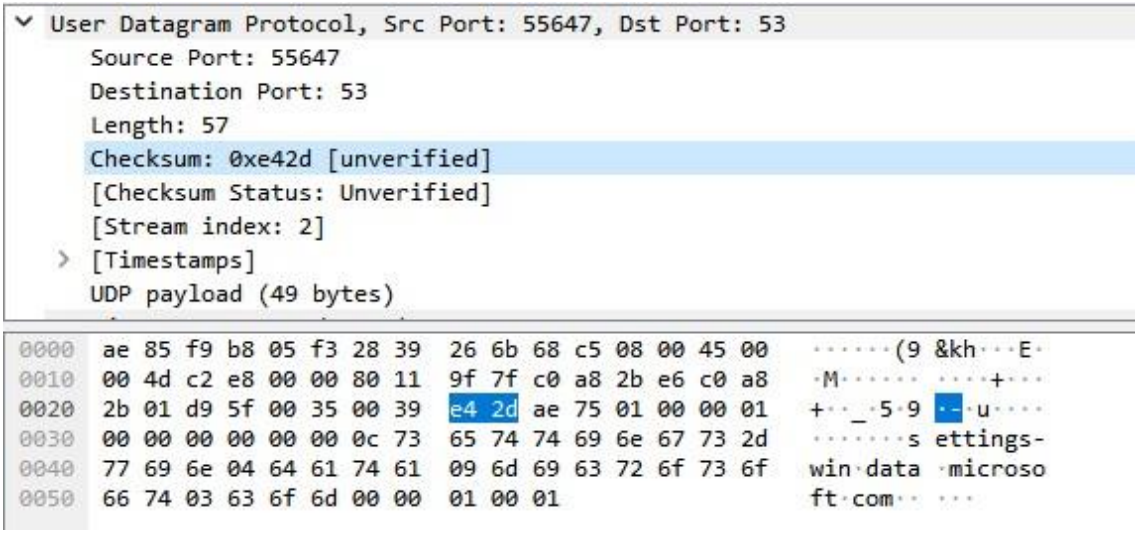


4) **What can you reasonably predict, and what could you figure out if you had some time and a calculator handy? Use your knowledge of UDP to inform your predictions.**
A)We can calculate the checksum value in advance.

5) **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?**



6) **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?**



A) On systems that support checksum offloading, IP, TCP, and UDP checksums are calculated on the NIC just before they're transmitted on the wire. In Wireshark these show up as outgoing packets marked black with red Text and the note *[incorrect, should be xxxx (maybe caused by "TCP checksum offload"?)]*.

Wireshark captures packets before they are sent to the network adapter. It won't see the correct checksum because it has not been calculated yet. Even worse, most OSes don't bother initialize this data so you're probably seeing little chunks of memory that you shouldn't.

New installations of Wireshark 1.2 and above disable IP, TCP, and UDP checksum validation by default. You can disable checksum validation in each of those dissectors by hand if needed.

7) Save your capture file. Restrict the range of saved packets to only those in the DNS query.

Step 2: TCP

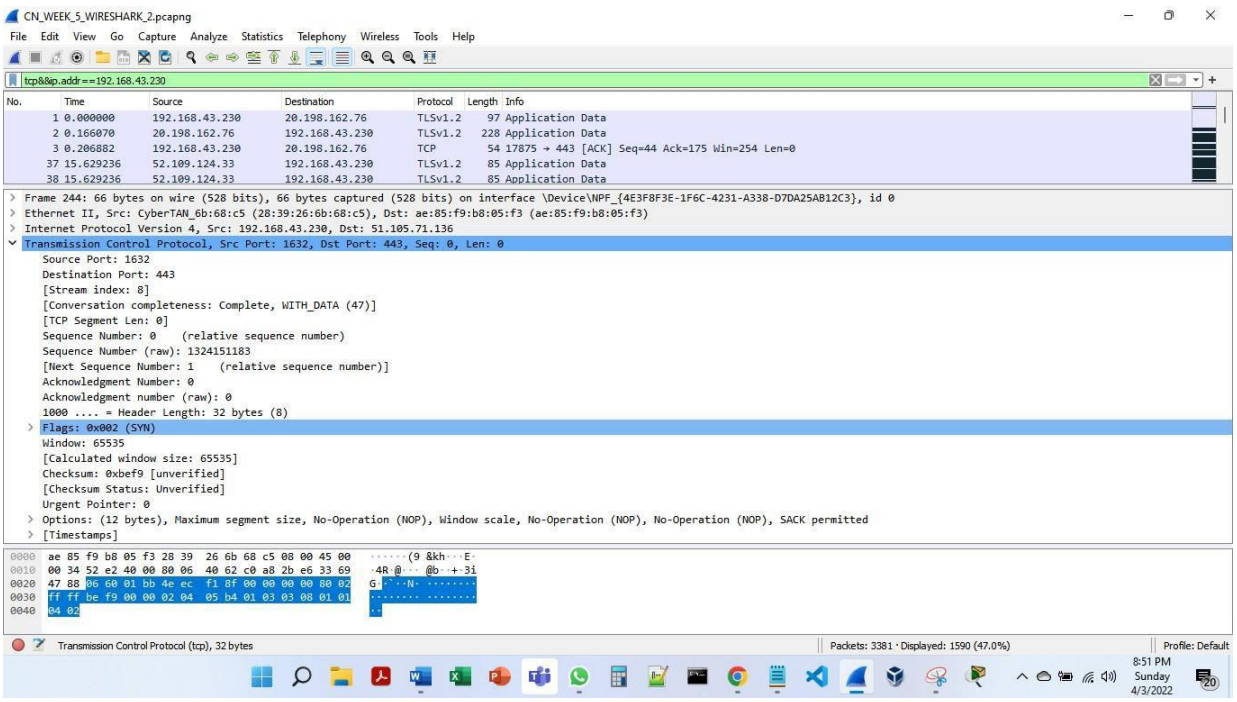
Now, let's look at another transport protocol, TCP. We will use HTTP to invoke the sort of TCP behaviours we want to study -> I trust that you understand HTTP well enough by now.

8) 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**

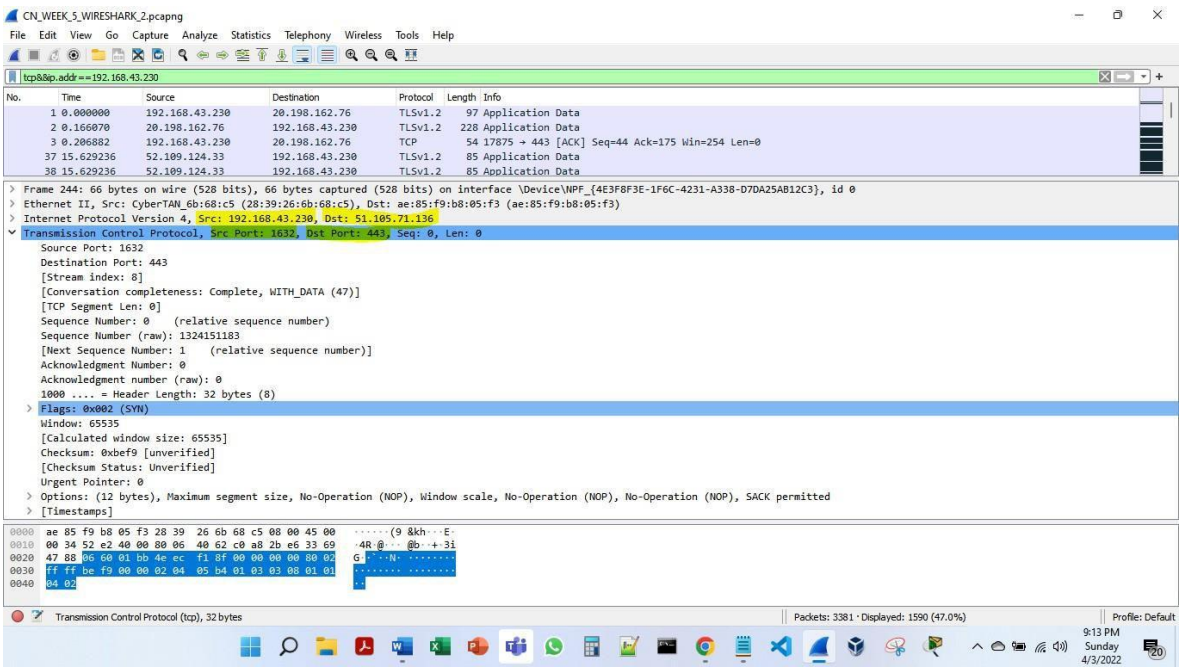
9) Clear out Wireshark and start a new capture.

10) 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) a 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** 11)Stop the Wireshark capture.

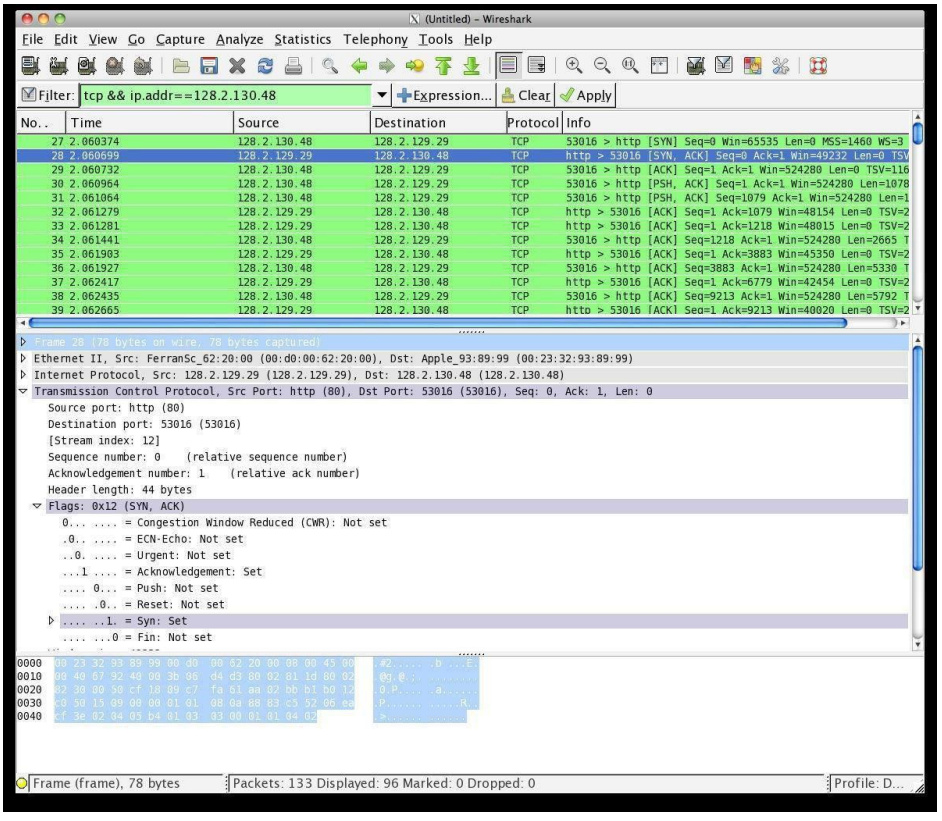
12)Let's look at what you captured. First, filter the results to look for TCP packets and to only look at those going to and from your computer with the filter **"tcp && ip.addr == <your IP address>**". If you have other services running on your computer, you might want to further filter so you only 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.

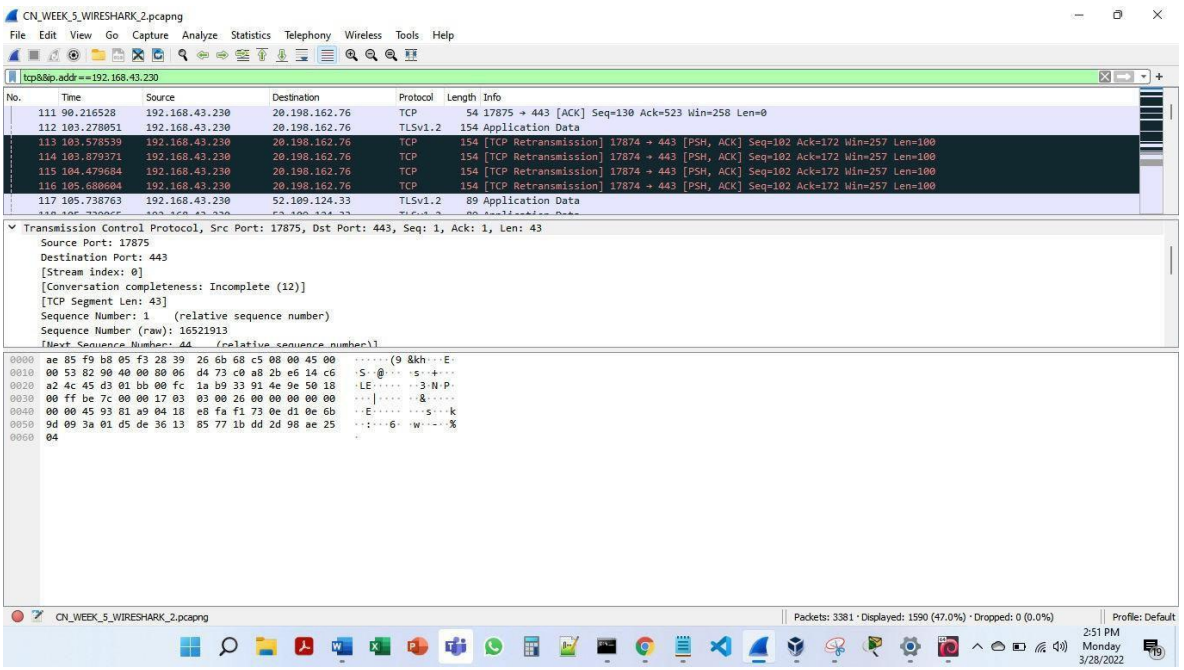


13) What is the IP address and TCP port number used by your computer (client) to transfer the file? What is the IP address of the server? On what port number is it sending and receiving TCP segments for this transfer of the file?



14) 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:



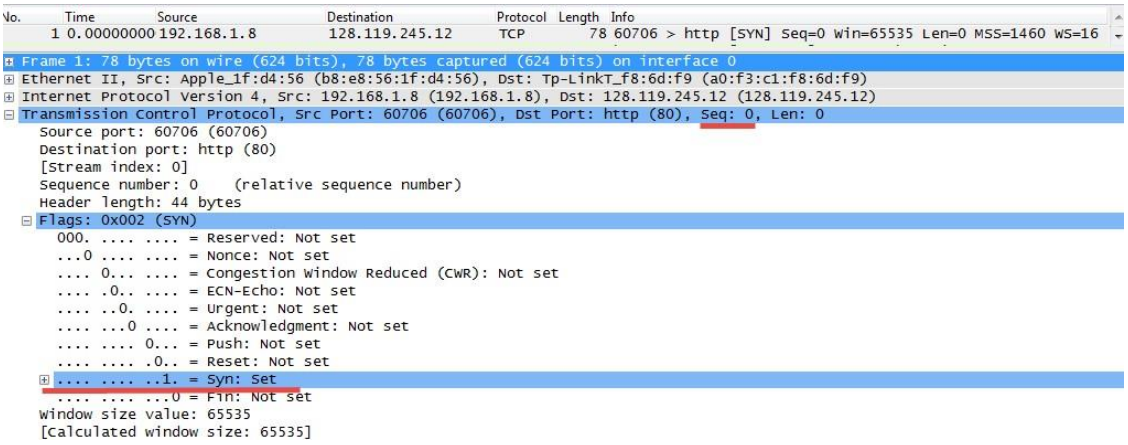


This is what we're looking for - a series of TCP segments sent between your computer and www.ece.cmu.edu. We will use the packet trace that you have captured to study TCP behaviour in the rest of this lab.

Step 2b: TCP Basics

1. 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?

Answer

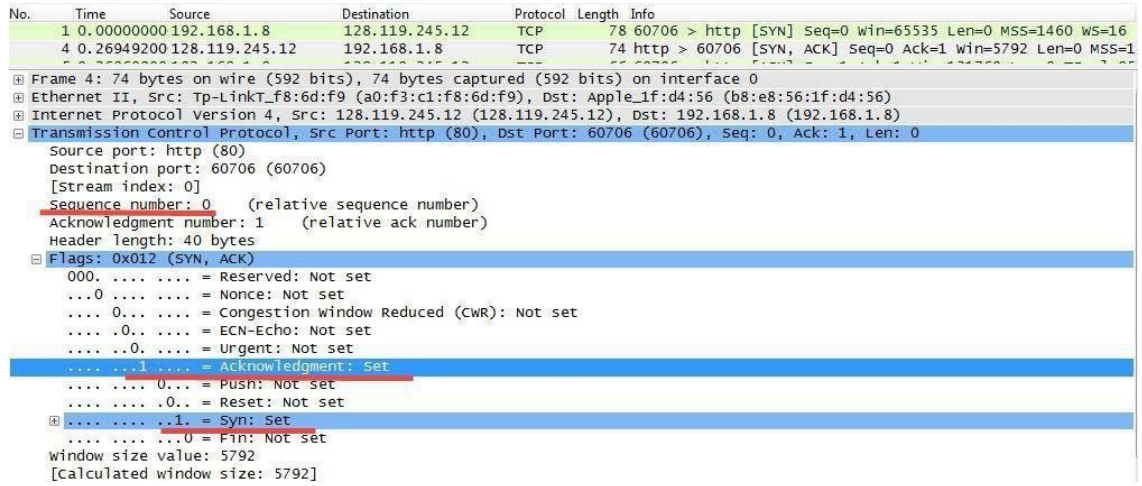


The sequence number of the TCP SYN segment is 0 since it is used to imitate the TCP connection between the client computer and gaia.cs.umass.edu.

According to above figure, in the Flags section, the Syn flag is set to 1 which indicates that this segment is a SYN segment.

2. What is the sequence number of the SYNACK 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?

Answer



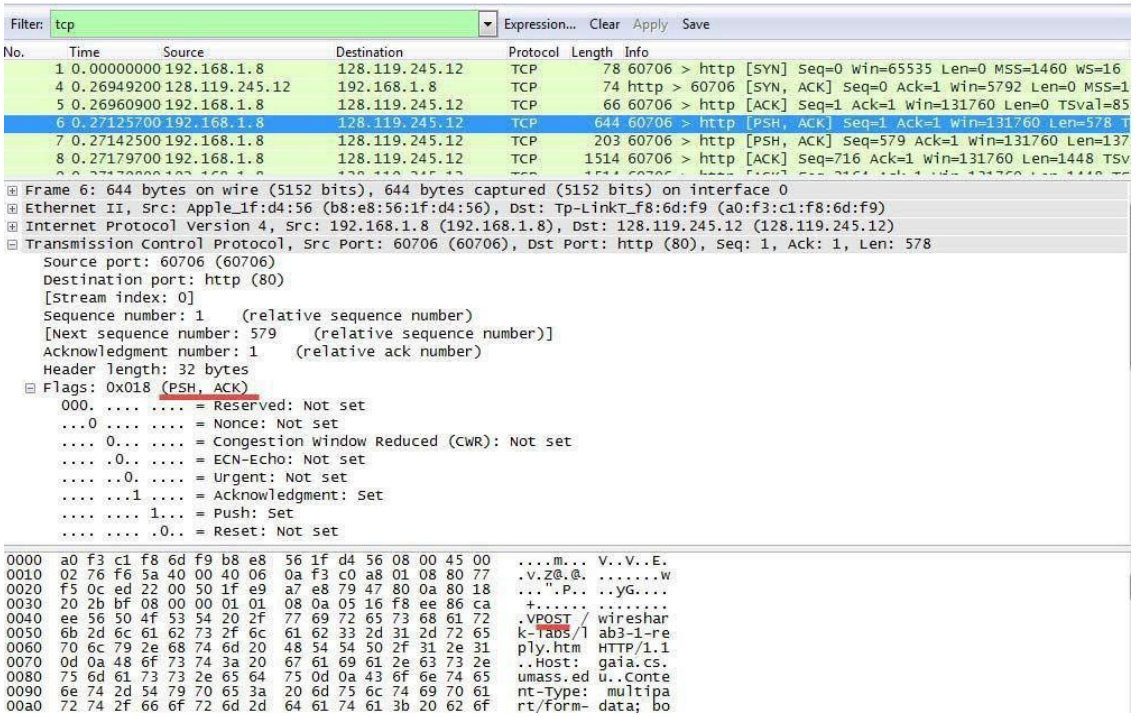
According to the above figure, the sequence number of the SYNACK segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN is 0.

The value of the acknowledgement field in the SYNACK segment is 1. The value of the ACKnowledgement field in the SYNACK segment is determined by the server gaia.cs.umass.edu. The server adds 1 to the initial sequence number of SYN segment form 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.

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

3. 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.

Answer

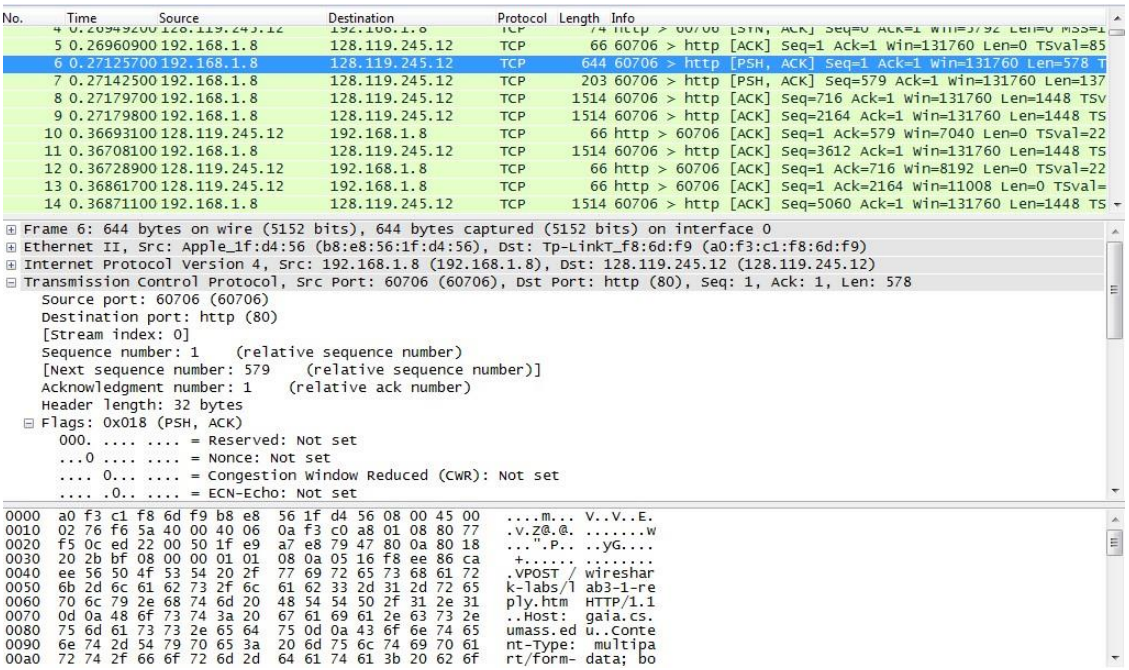


According to above figure, the segment No.6 contains the HTTP POST command, the sequence number of this segment is 1.

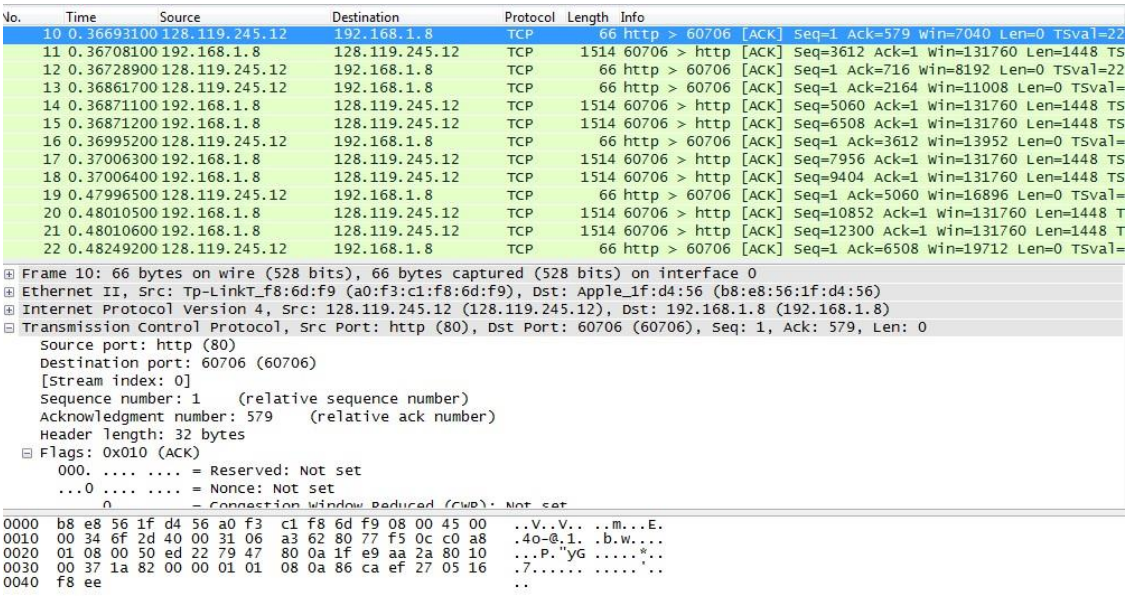
4. Consider the TCP segment containing the HTTP POST as the first segment in the TCP connection. What are the sequence numbers of the first six segments in the TCP connection (including the segment containing the HTTP POST)? At what time was each segment sent? When was the ACK for each segment received? 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 Estimated RTT value (see Section 3.5.3, page 239 in text) after the receipt of each ACK? Assume that the value of the Estimated RTT is equal to the measured RTT for the first segment, and then is computed using the Estimated RTT 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.

Answer



Segments 1-6



ACK of segments 1-6

According to above figures, the segments 1-6 are No. 6, 7, 8, 9, 11 and 14. The ACK of segments 1-6 are No. 10, 12, 13, 16, 19 and 22.

Segment 1 sequence number is 1

Segment 2 sequence number is 579

Segment 3 sequence number is 716

Segment 4 sequence number is 2164

Segment 5 sequence number is 3612

Segment 6 sequence number is 5060

Recording the sending time and received time of ACKs:

	Sent time	**ACK received time**	**RTT**
Segment 1	0.271257000	0.366931000	0.095674
Segment 2	0.271425000	0.367289000	0.095864
Segment 3	0.271797000	0.368617000	0.09682
Segment 4	0.271798000	0.369952000	0.098154
Segment 5	0.367081000	0.479965000	0.112884
Segment 6	0.368711000	0.482492000	0.113781

According to the formula: EstimatedRTT = 0.875 * EstimatedRTT + 0.125 * SampleRTT

EstimatedRTT after the receipt of the ACK of segment 1:

EstimatedRTT = RTT for Segment 1 = 0.095674 s

EstimatedRTT after the receipt of the ACK of segment 2:

EstimatedRTT = 0.875 * 0.095674 + 0.125 * 0.095864= 0.09569775 s

EstimatedRTT after the receipt of the ACK of segment 3:

EstimatedRTT = 0.875 * 0.09569775 + 0.125 * 0.09682= 0.09583803125 s

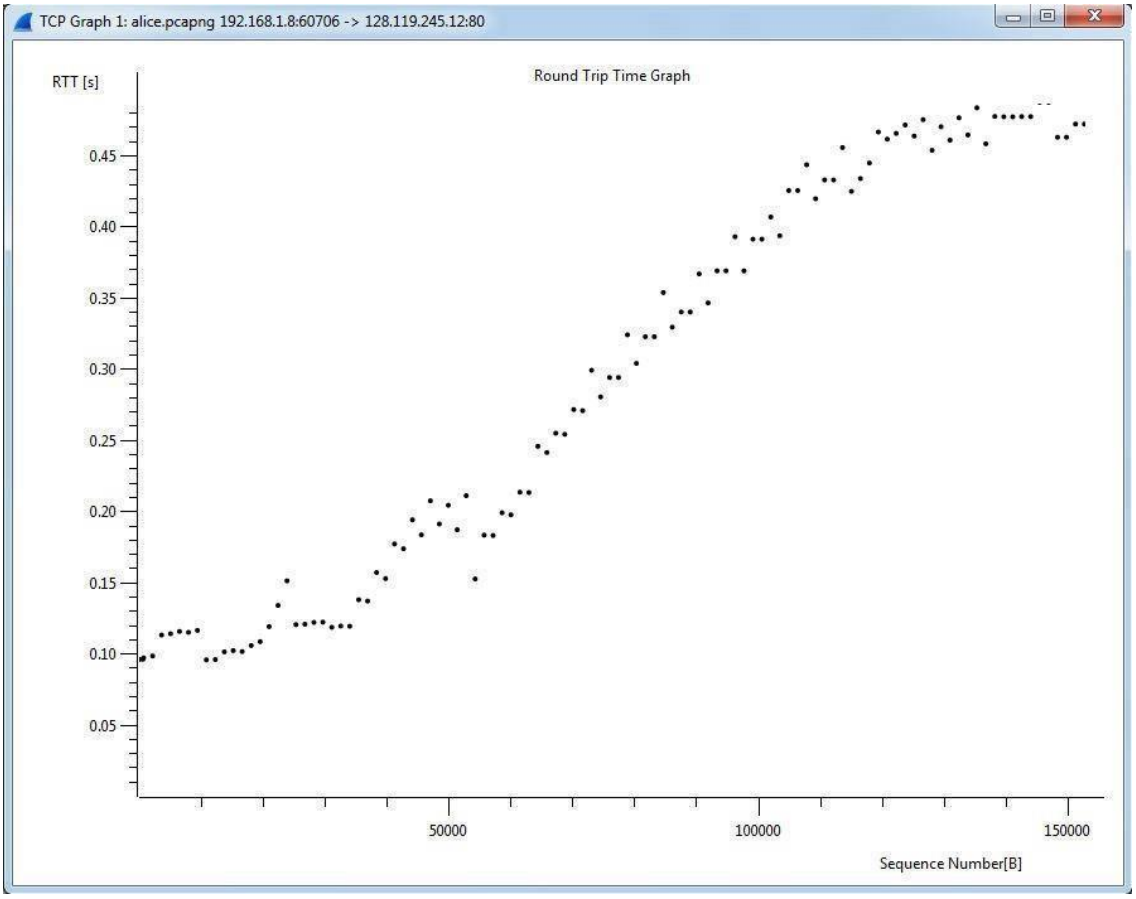
EstimatedRTT after the receipt of the ACK of segment 4:

EstimatedRTT = 0.875 * 0.09583803125 + 0.125 * 0.098154= 0.09612752734 s

EstimatedRTT after the receipt of the ACK of segment 5:

EstimatedRTT = 0.875 * 0.09612752734 + 0.125 * 0.112884= 0.09822208642 s EstimatedRTT after the receipt of the ACK of segment 6:

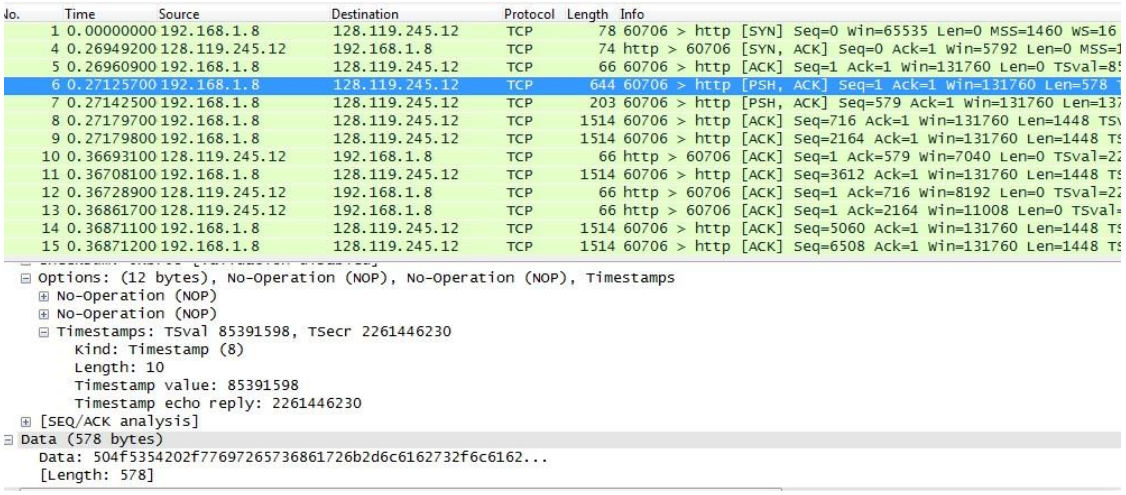
EstimatedRTT = 0.875 *0.09822208642 + 0.125 * 0.113781= 0.10016695061 s



Round Trip Time Graph

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

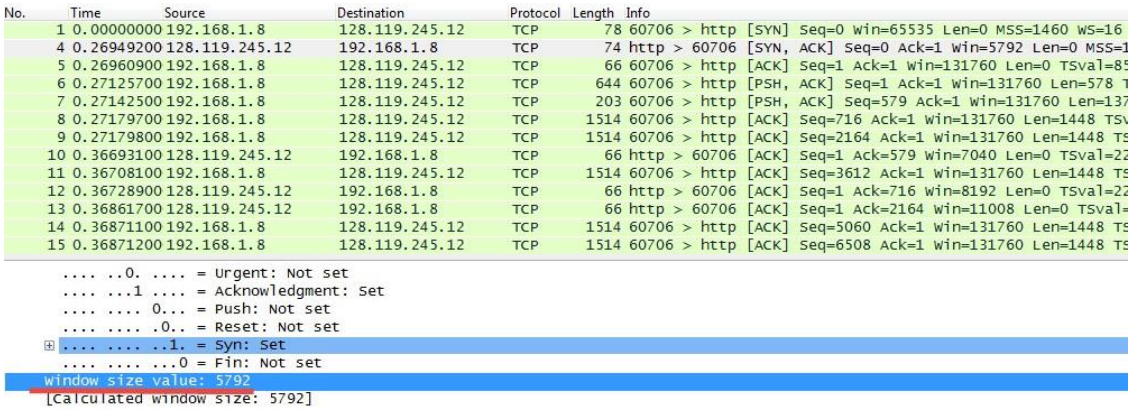
Answer



The length of the first TCP segment is 578 bytes, the length of the second TCP segment is 137 bytes. The length of each of the following five TCP segments is 1448 bytes.

6. 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?

Answer

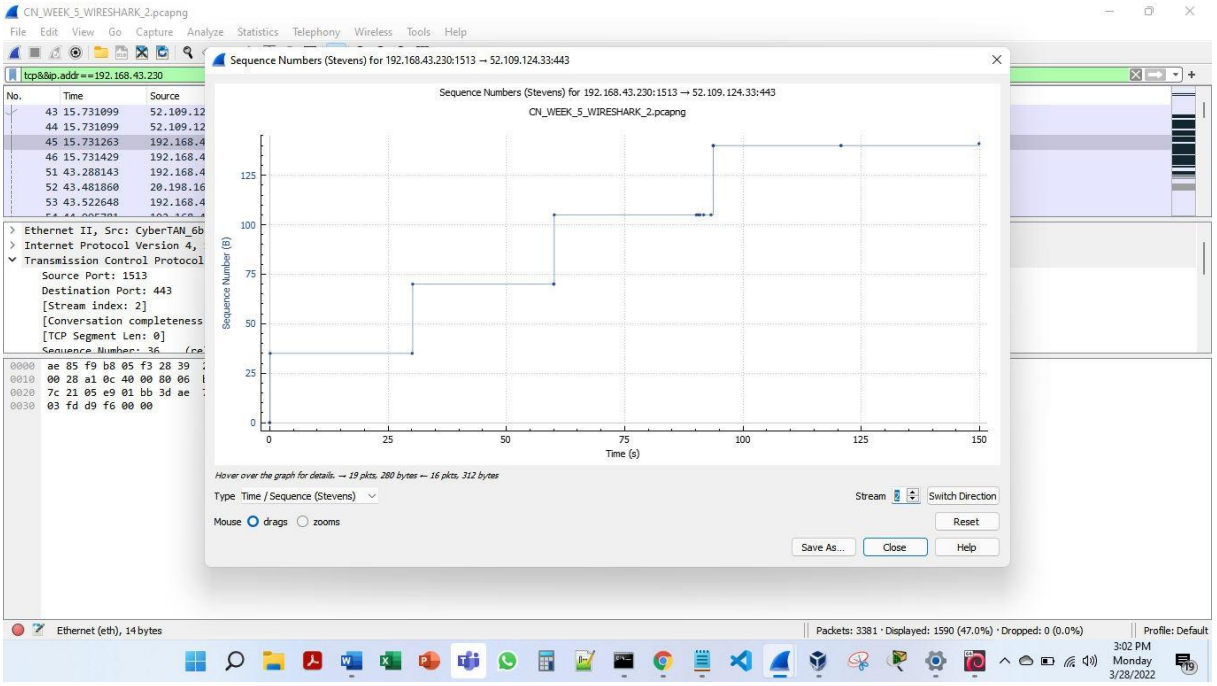


The minimum amount of available buffer space advertised at the received for the entire trace is indicated first ACK from the server, its value is 5792 bytes (shown in above figure).

This reviver window grows until it reaches the maximum receiver buffer size of 62780 bytes. According to the trace, the sender is never throttled due to lacking of receiver buffer space.

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

Answer



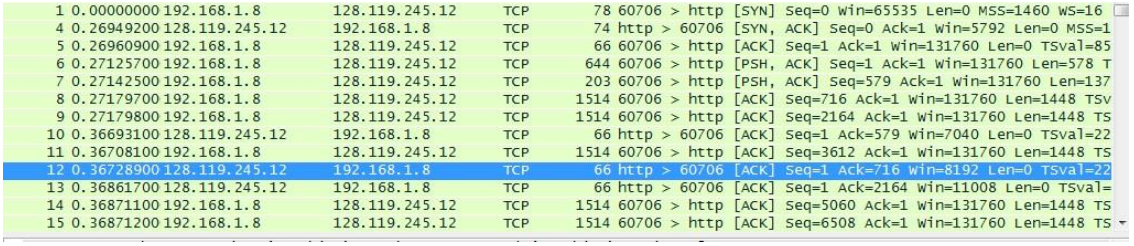
There are retransmitted segments in the trace file since in the time sequence graph (stevens), all sequence numbers are increasing in a staircase manner.

8. How much data does the receiver typically acknowledge in an ACK? Can you identify cases where the receiver is ACKing every other received segment (see Table 3.2 on page 247 in the text).

Answer

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 No. 13 acknowledged data with 1430 bytes.



1	0.00000000	192.168.1.8	128.119.245.12	TCP	78 60706 > http [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=16
4	0.26949200	128.119.245.12	192.168.1.8	TCP	74 http > 60706 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1
5	0.26960900	192.168.1.8	128.119.245.12	TCP	66 60706 > http [ACK] Seq=1 Ack=1 Win=131760 Len=0 TSval=85
6	0.27125700	192.168.1.8	128.119.245.12	TCP	644 60706 > http [PSH, ACK] Seq=1 Ack=1 Win=131760 Len=578 T
7	0.27142500	192.168.1.8	128.119.245.12	TCP	203 60706 > http [PSH, ACK] Seq=579 Ack=1 Win=131760 Len=137
8	0.27179700	192.168.1.8	128.119.245.12	TCP	1514 60706 > http [ACK] Seq=716 Ack=1 Win=131760 Len=1448 TSv
9	0.27179800	192.168.1.8	128.119.245.12	TCP	1514 60706 > http [ACK] Seq=2164 Ack=1 Win=131760 Len=1448 TS
10	0.36693100	128.119.245.12	192.168.1.8	TCP	66 http > 60706 [ACK] Seq=1 Ack=579 Win=7040 Len=0 TSval=22
11	0.36708100	192.168.1.8	128.119.245.12	TCP	1514 60706 > http [ACK] Seq=3612 Ack=1 Win=131760 Len=1448 TS
12	0.36728900	128.119.245.12	192.168.1.8	TCP	66 http > 60706 [ACK] Seq=1 Ack=716 Win=8192 Len=0 TSval=22
13	0.36861700	128.119.245.12	192.168.1.8	TCP	66 http > 60706 [ACK] Seq=1 Ack=2164 Win=11008 Len=0 TSval=
14	0.36871100	192.168.1.8	128.119.245.12	TCP	1514 60706 > http [ACK] Seq=5060 Ack=1 Win=131760 Len=1448 TS
15	0.36871200	192.168.1.8	128.119.245.12	TCP	1514 60706 > http [ACK] Seq=6508 Ack=1 Win=131760 Len=1448 TS

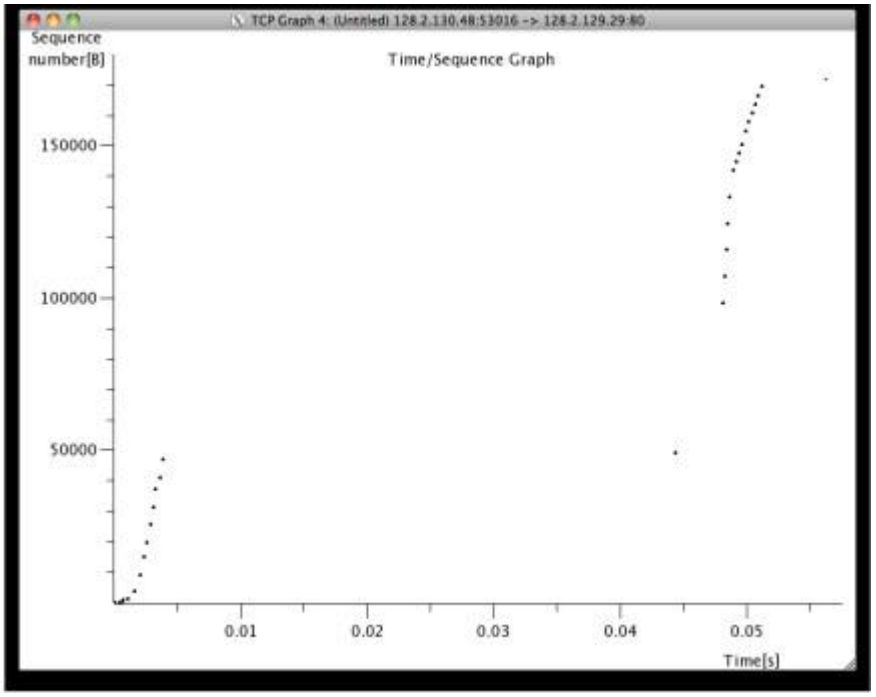
9. What is the throughput (bytes transferred per unit time) for the TCP connection? Explain how you calculated this value.

Answer

The alice.txt on the hard drive is 152,138 bytes, and the download time is 1.578736000 (First TCP segment) - 0.271257000 (last ACK) = 1.307479 second. Therefore, the throughput for the TCP connection is computed as 152,138/1.307479=116359.803867 bytes/second.

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).

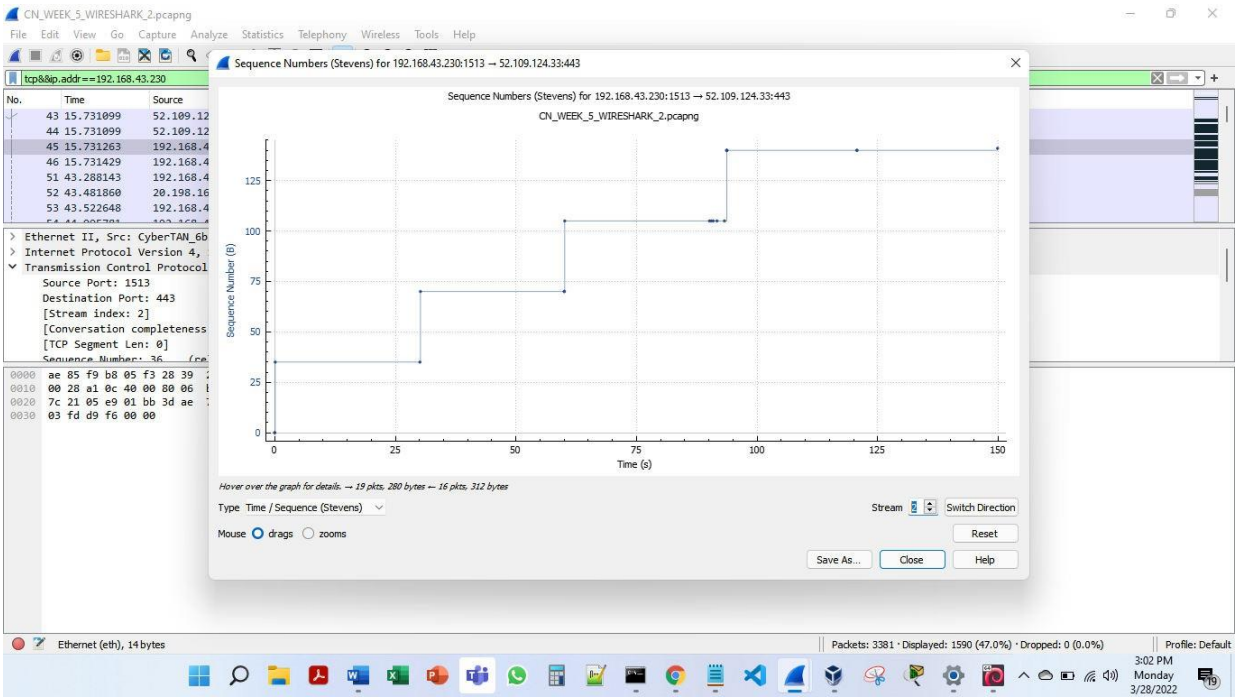


You should see a plot that looks like the following plot (though the individual plotted values may differ quite a bit). Here, each dot represents a TCP segment sent, plotting the sequence number of the segment versus the time at which it was sent. Note that a set of dots stacked above each other represents 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.

10. Use the Time-Sequence-Graph(Stevens) plotting tool to view the sequence number versus time plot of segments being sent from the client to the gaia.cs.umass.edu server. 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.

Answer

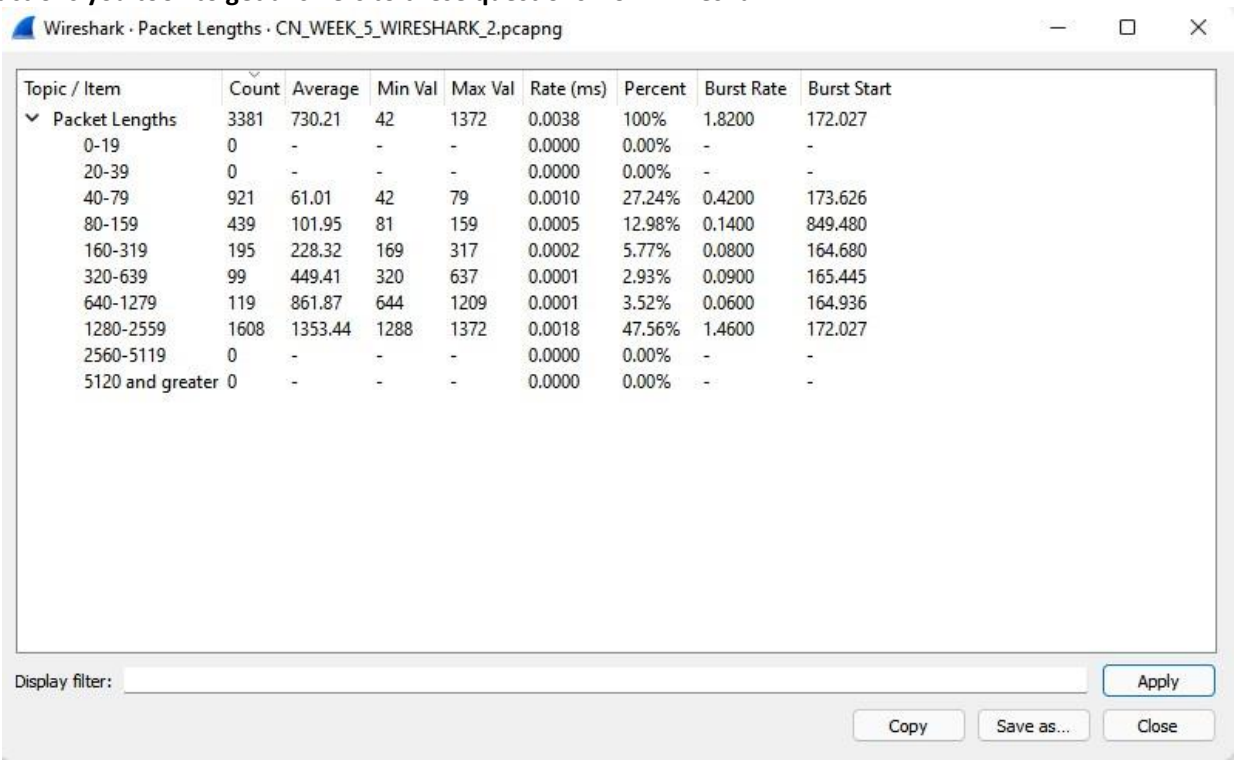
The slow start of the TCP seems to begin at about 0.27 seconds and then ends at about 0.35 seconds. Congestion avoidance takes over at about 0.7 seconds because it cut down the amount being sent.



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:

1) **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**.

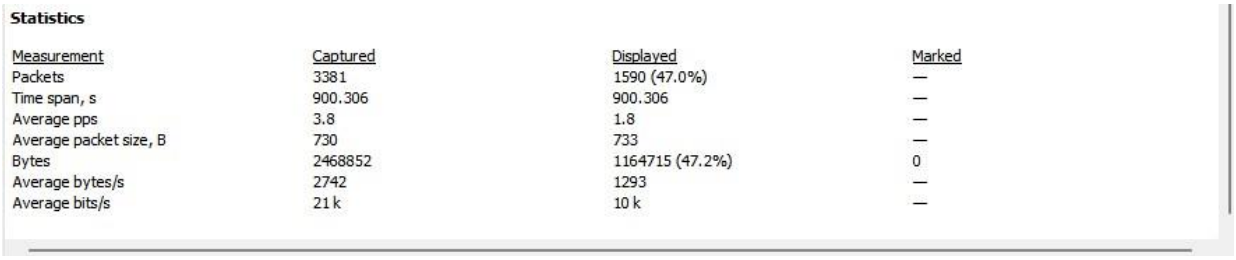


Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
Packet Lengths	3381	730.21	42	1372	0.0038	100%	1.8200	172.027
0-19	0	-	-	-	0.0000	0.00%	-	-
20-39	0	-	-	-	0.0000	0.00%	-	-
40-79	921	61.01	42	79	0.0010	27.24%	0.4200	173.626
80-159	439	101.95	81	159	0.0005	12.98%	0.1400	849.480
160-319	195	228.32	169	317	0.0002	5.77%	0.0800	164.680
320-639	99	449.41	320	637	0.0001	2.93%	0.0900	165.445
640-1279	119	861.87	644	1209	0.0001	3.52%	0.0600	164.936
1280-2559	1608	1353.44	1288	1372	0.0018	47.56%	1.4600	172.027
2560-5119	0	-	-	-	0.0000	0.00%	-	-
5120 and greater	0	-	-	-	0.0000	0.00%	-	-

GO to Statsistics->Packet Lengths->This dialog box appears

From here we can observe that the packets length from(1280-2559) are most common with a count of 1608 and packets length from (40-79) are second most frequent with a count of 921. Ratio of TCP packets of length < 40 bytes is equal to zero since these packets are TCP Packets and their minimum size of TCP Header is 40 bytes.

2) 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.



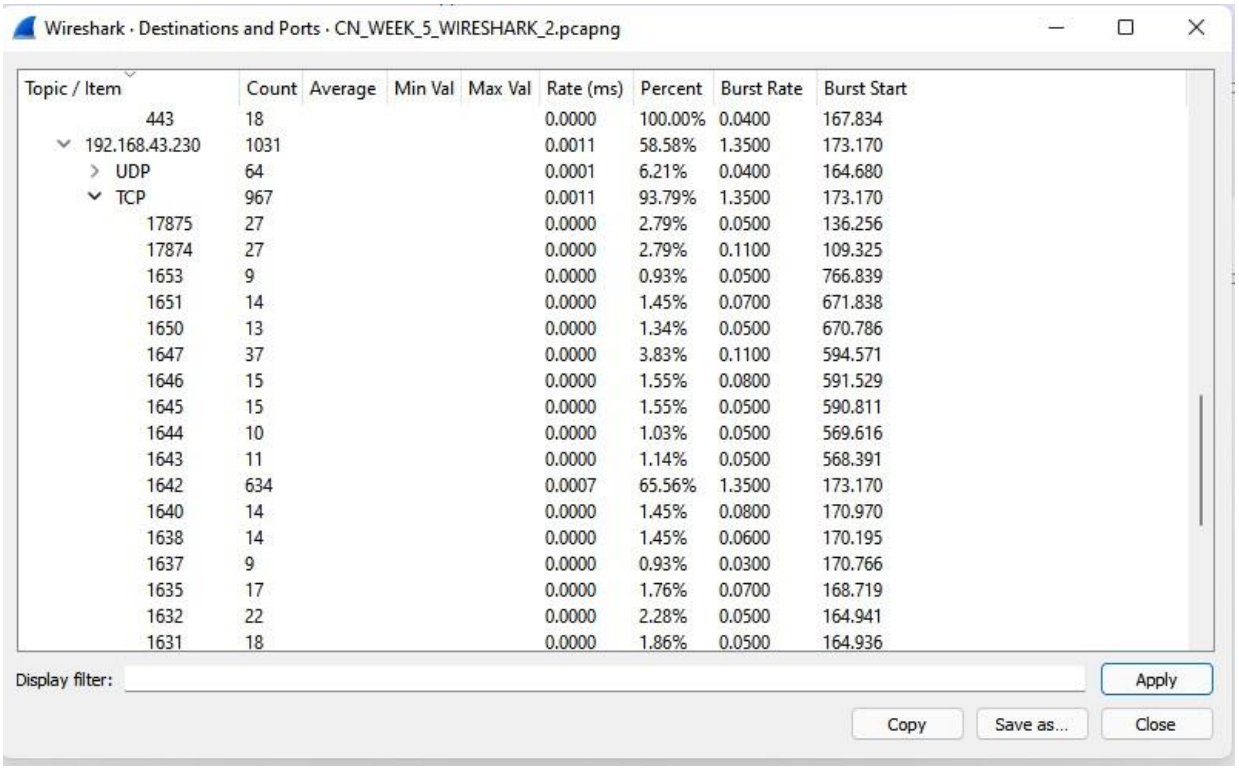
Statistics

Measurement	Captured	Displayed	Marked
Packets	3381	1590 (47.0%)	—
Time span, s	900.306	900.306	—
Average pps	3.8	1.8	—
Average packet size, B	730	733	—
Bytes	2468852	1164715 (47.2%)	0
Average bytes/s	2742	1293	—
Average bits/s	21 k	10 k	—

Go to Statistics->Capture File properties

3) 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?

HOST



Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
443	18				0.0000	100.00%	0.0400	167.834
192.168.43.230	1031				0.0011	58.58%	1.3500	173.170
UDP	64				0.0001	6.21%	0.0400	164.680
TCP	967				0.0011	93.79%	1.3500	173.170
17875	27				0.0000	2.79%	0.0500	136.256
17874	27				0.0000	2.79%	0.1100	109.325
1653	9				0.0000	0.93%	0.0500	766.839
1651	14				0.0000	1.45%	0.0700	671.838
1650	13				0.0000	1.34%	0.0500	670.786
1647	37				0.0000	3.83%	0.1100	594.571
1646	15				0.0000	1.55%	0.0800	591.529
1645	15				0.0000	1.55%	0.0500	590.811
1644	10				0.0000	1.03%	0.0500	569.616
1643	11				0.0000	1.14%	0.0500	568.391
1642	634				0.0007	65.56%	1.3500	173.170
1640	14				0.0000	1.45%	0.0800	170.970
1638	14				0.0000	1.45%	0.0600	170.195
1637	9				0.0000	0.93%	0.0300	170.766
1635	17				0.0000	1.76%	0.0700	168.719
1632	22				0.0000	2.28%	0.0500	164.941
1631	18				0.0000	1.86%	0.0500	164.936

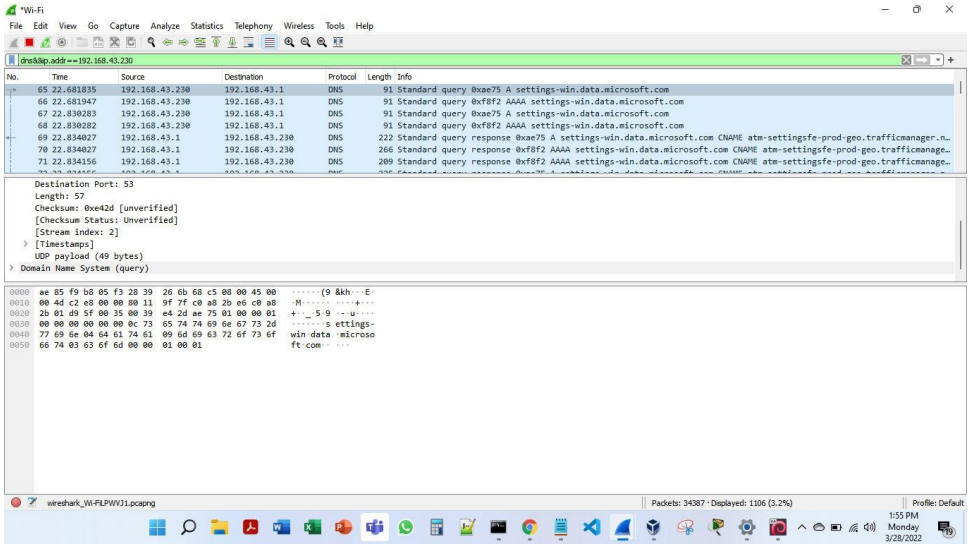
DESTINATION

20.198.162.76	89			0.0001	5.06%	0.0500	109.325
TCP	89			0.0001	100.00%	0.0500	109.325
443	89			0.0001	100.00%	0.0500	109.325

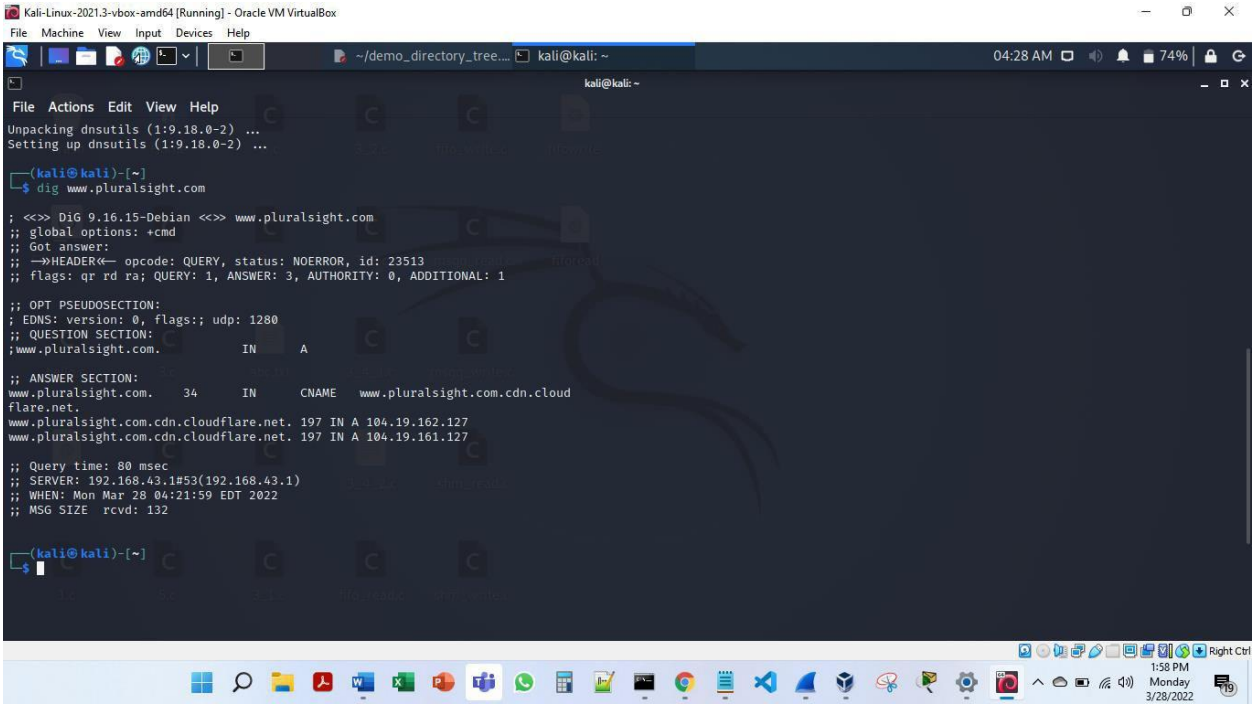
Step 4: The Network Layer

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.

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



5) 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).

6) Most of the fields should match up and make perfect sense. Verify the Datagram Length, Upper-layer protocol and the IP address fields.

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

8) 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? BTW, please document in this question what your OS and OS version are.

Windows specifications

Edition	Windows 11 Home Single Language
Version	Dev
Installed on	11/22/2021
OS build	21996.1
Experience	Windows Feature Experience Pack 321.14700.0.3

Step 5: ICMP

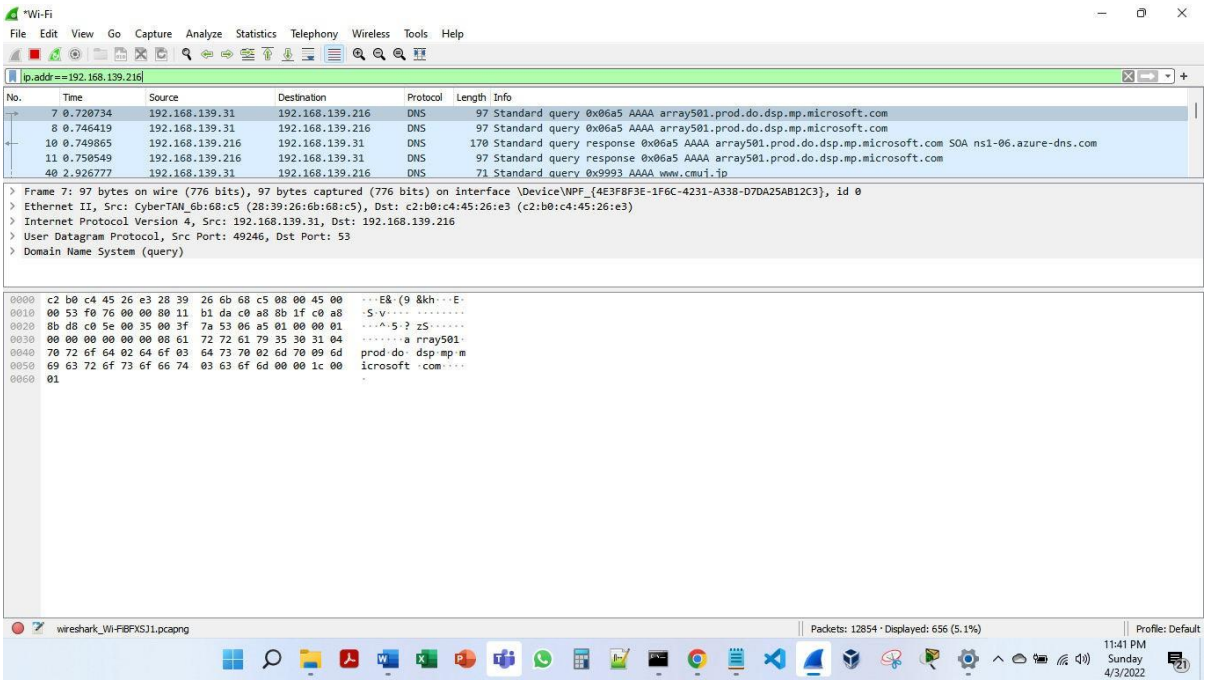
The Network Layer uses ICMP to send information about the network. Some would say that ICMP is a higher-layer protocol, as the actual ICMP packet is carried inside an IP packet. Let's take a look at how that works.

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

10) 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.

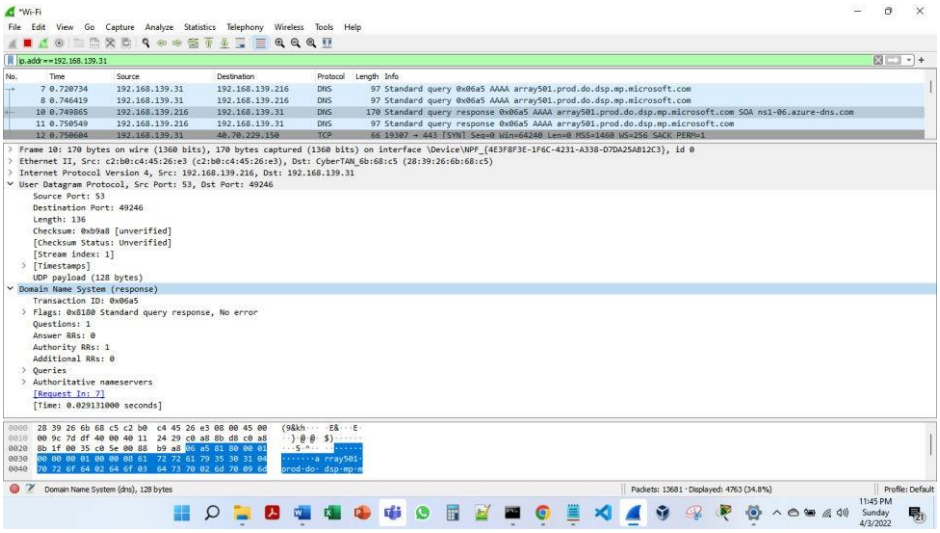
```
Tracing route to cmuj.jp [122.17.163.205]
over a maximum of 30 hops:

  1    40 ms     4 ms     4 ms  192.168.139.216
  2     *        *        *     Request timed out.
  3    44 ms    26 ms    40 ms  10.71.168.66
  4    47 ms    26 ms    37 ms  192.168.65.248
  5    38 ms    26 ms    38 ms  192.168.65.251
  6    40 ms    34 ms    19 ms  172.26.74.20
  7    49 ms    25 ms    38 ms  172.26.77.242
  8    38 ms    30 ms    37 ms  192.168.65.142
  9    49 ms    37 ms    47 ms  192.168.65.143
 10    40 ms    50 ms    44 ms  172.31.2.101
 11    51 ms    38 ms    46 ms  103.198.140.64
 12   248 ms   236 ms   237 ms  103.198.140.15
 13     *      292 ms   342 ms  4.7.26.61
 14     *        *        *     Request timed out.
 15   278 ms   270 ms   266 ms  ae-28.r00.lsanca07.us.bb.gin.ntt.net [129.250.9.93]
 16   292 ms     *      299 ms  ae-6.r25.lsanca07.us.bb.gin.ntt.net [129.250.3.237]
 17   632 ms   349 ms   607 ms  ae-12.r31.tokyjp05.jp.bb.gin.ntt.net [129.250.3.192]
 18   553 ms   594 ms   422 ms  ae-3.r02.tokyjp05.jp.bb.gin.ntt.net [129.250.3.28]
 19   521 ms   639 ms   636 ms  ae-0.ocn.tokyjp05.jp.bb.gin.ntt.net [120.88.53.18]
 20   507 ms   354 ms   601 ms  118.23.168.142
 21   547 ms   359 ms   591 ms  211.129.53.194
 22   381 ms   574 ms   624 ms  122.17.156.2
 23   531 ms   362 ms   588 ms  c15fcohq.mwprem.net [122.17.163.205]

Trace complete.
```

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

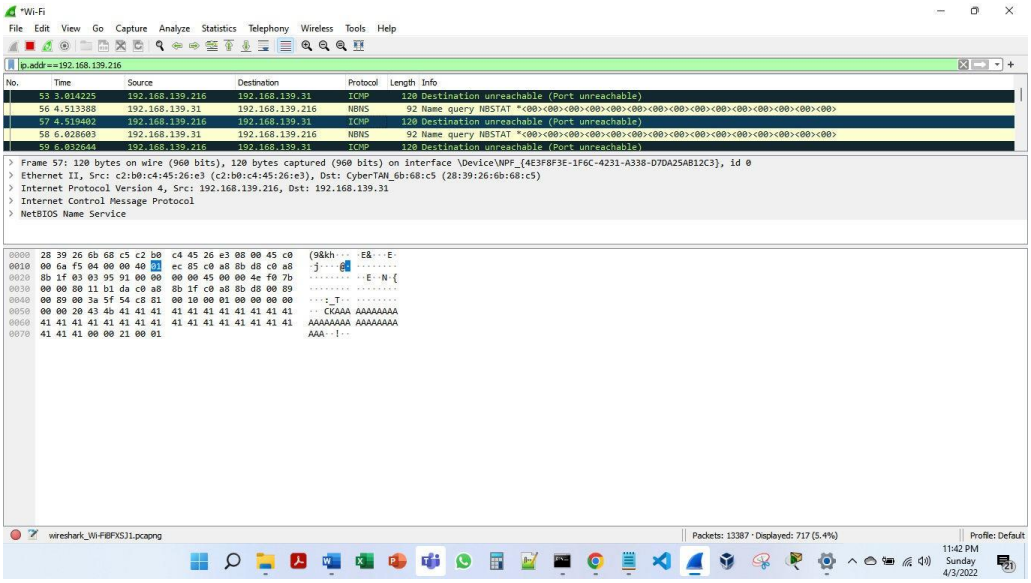


12) What are the transmitted segments like? Describe the important features of the segments you observe. In particular, examine the destination port field. What characteristics do you observe about this port number and why would it be chosen so?

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



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



15) 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?

```
Pinging cmuj.jp [122.17.163.205] with 32 bytes of data:
Reply from 122.17.163.205: bytes=32 time=374ms TTL=41
Reply from 122.17.163.205: bytes=32 time=387ms TTL=41
Reply from 122.17.163.205: bytes=32 time=684ms TTL=41
Reply from 122.17.163.205: bytes=32 time=365ms TTL=41

Ping statistics for 122.17.163.205:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 365ms, Maximum = 684ms, Average = 452ms
```

Finishing up

The report should consist of proper explanation for each question and all the necessary screenshots.