

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.

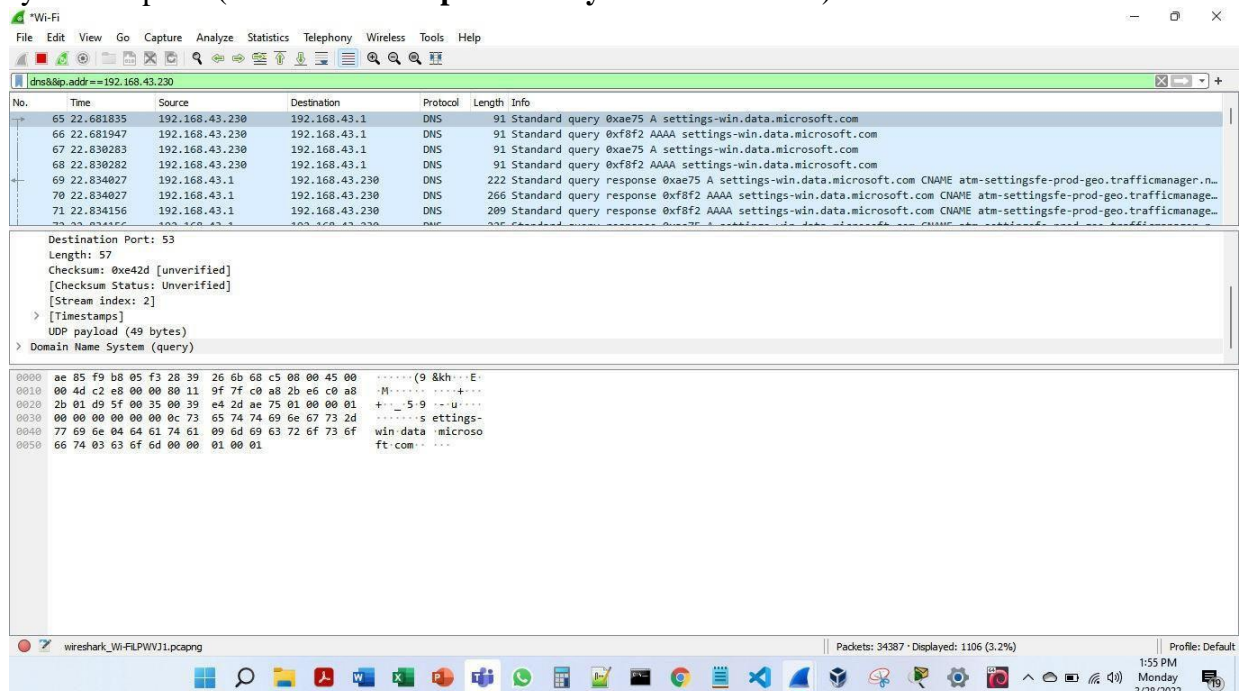
Note: Take screenshots wherever necessary.

Step 1: UDP and DNS

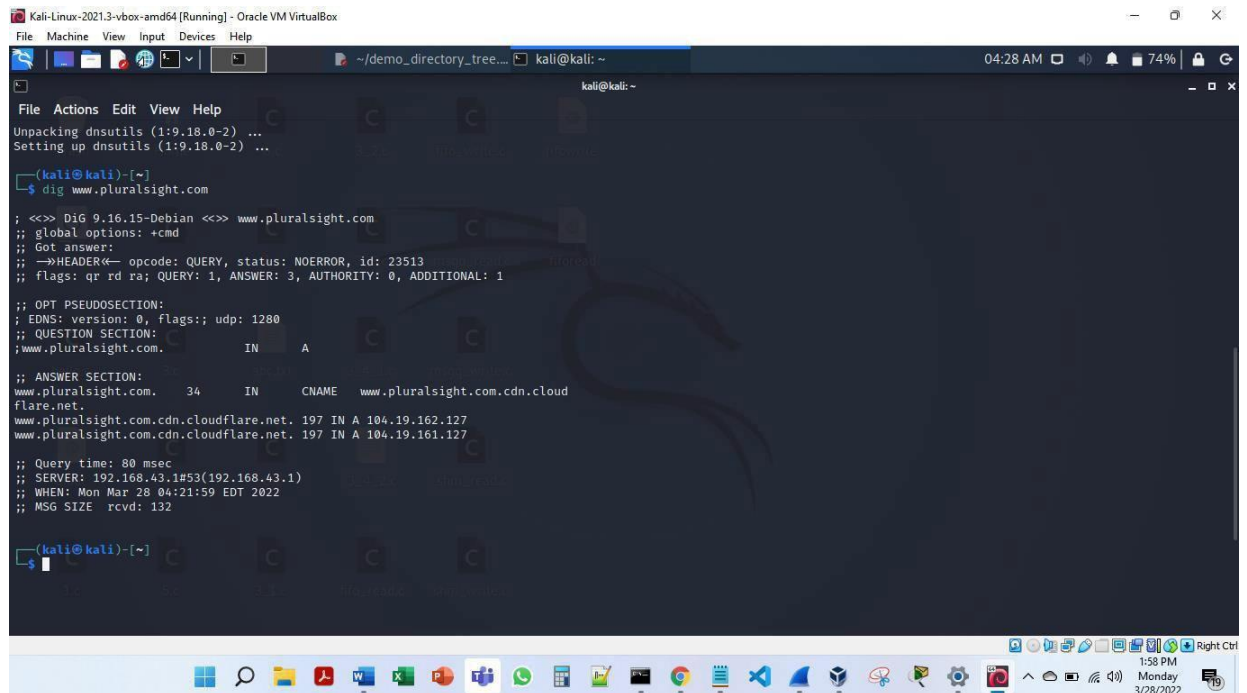
Let's start by examining a few UDP segments. UDP is a streamlined, no-frills transport protocol. All state information is conveyed in each individual UDP segment. In Lab 4, we used dig to generate DNS traffic with the intent of examining the DNS protocol. In this lab, we will use dig to generate DNS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



```
Kali-Linux-2021.3-vbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
~/demo_directory_tree... kali@kali: ~
kali@kali: ~
File Actions Edit View Help
Unpacking dnstools (1:9.18.0-2) ...
Setting up dnstools (1:9.18.0-2) ...

(kali@kali)-[~]
$ dig www.pluralsight.com

;<>> DiG 9.16.15-Debian <>> www.pluralsight.com
;; global options: +cmd
;; Got answer:
;;->HEADER<- opcode: QUERY, status: NOERROR, id: 23513
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 1

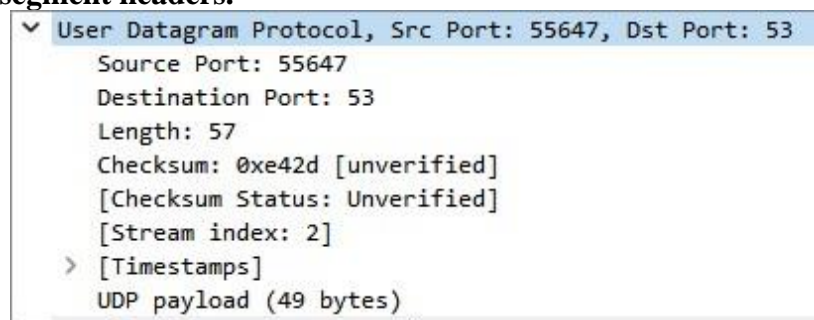
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1280
;; QUESTION SECTION:
;www.pluralsight.com.      IN      A

;; ANSWER SECTION:
www.pluralsight.com.      34      IN      CNAME  www.pluralsight.com.cdn.cloudflare.net.
www.pluralsight.com.cdn.cloudflare.net. 197 IN A 104.19.162.127
www.pluralsight.com.cdn.cloudflare.net. 197 IN A 104.19.161.127

;; Query time: 80 msec
;; SERVER: 192.168.43.1#53(192.168.43.1)
;; WHEN: Mon Mar 28 04:21:59 EDT 2022
;; MSG SIZE rcvd: 132

(kali@kali)-[~]
```

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



```
▼ User Datagram Protocol, Src Port: 55647, Dst Port: 53
  Source Port: 55647
  Destination Port: 53
  Length: 57
  Checksum: 0xe42d [unverified]
  [Checksum Status: Unverified]
  [Stream index: 2]
  > [Timestamps]
  UDP payload (49 bytes)
```

- 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?

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.

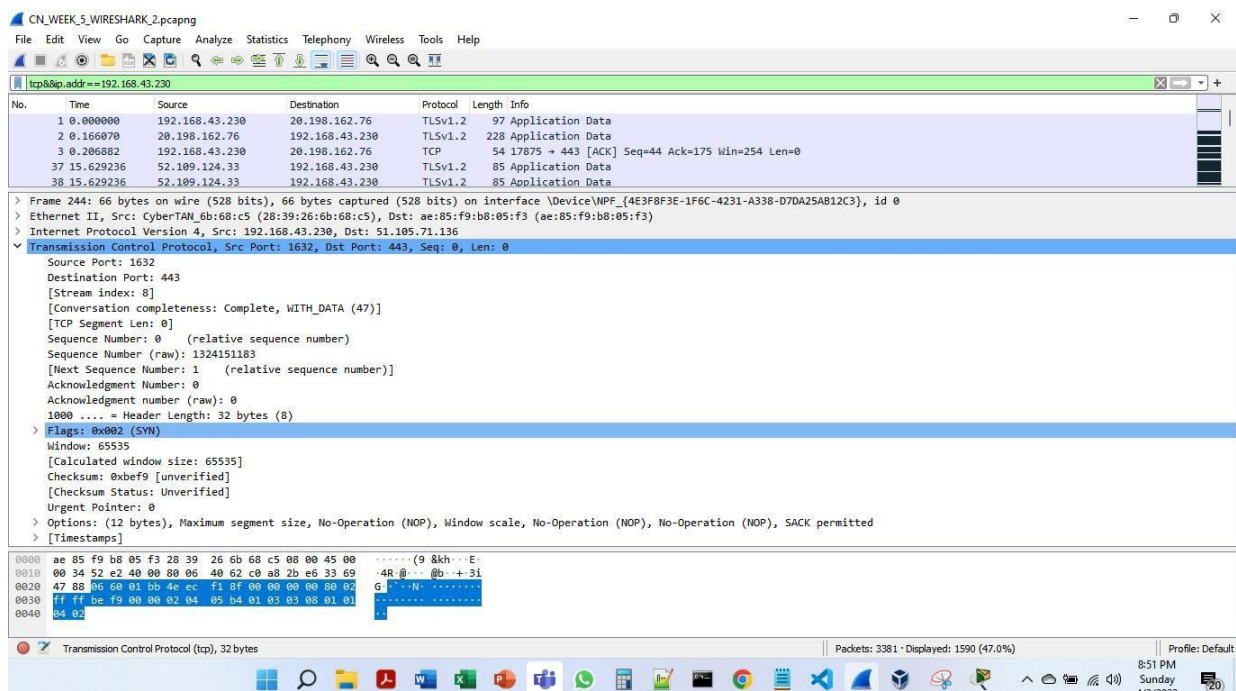
- 8) Download and save a copy of Geoffrey Chaucer's Canterbury Tales and Other Poems from the Project Gutenberg website¹. Grab the Plain Text UTF-8 version:

<http://www.gutenberg.org/ebooks/2383.txt.utf-8>

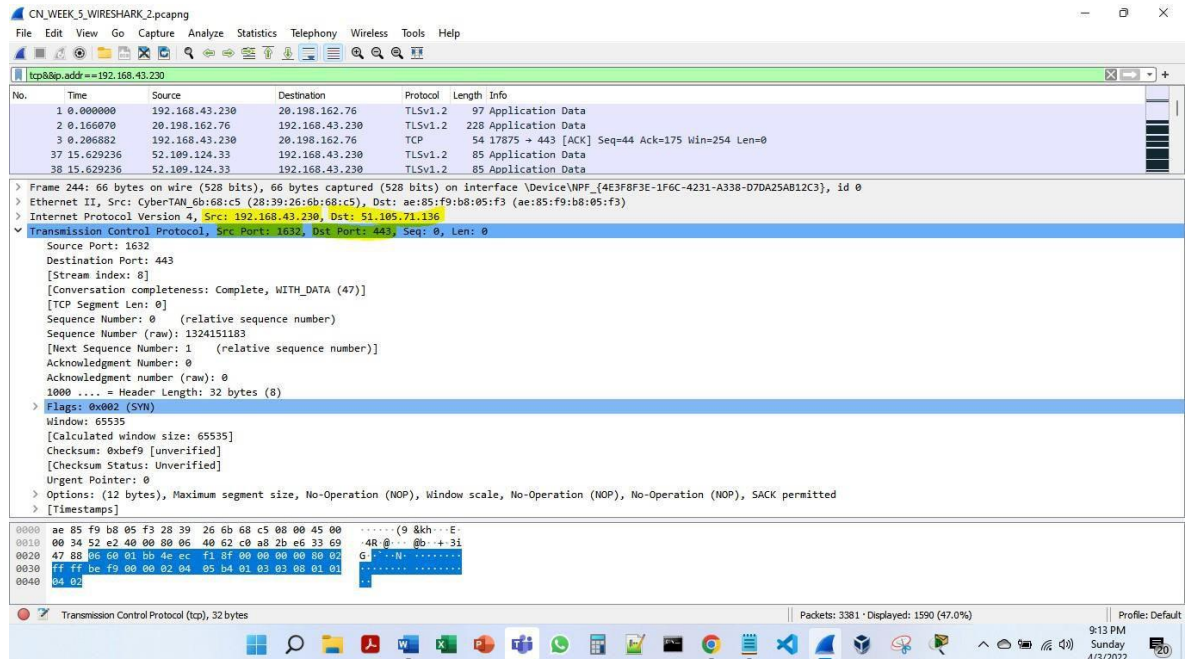
- 9) Clear out Wireshark and start a new capture.

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.

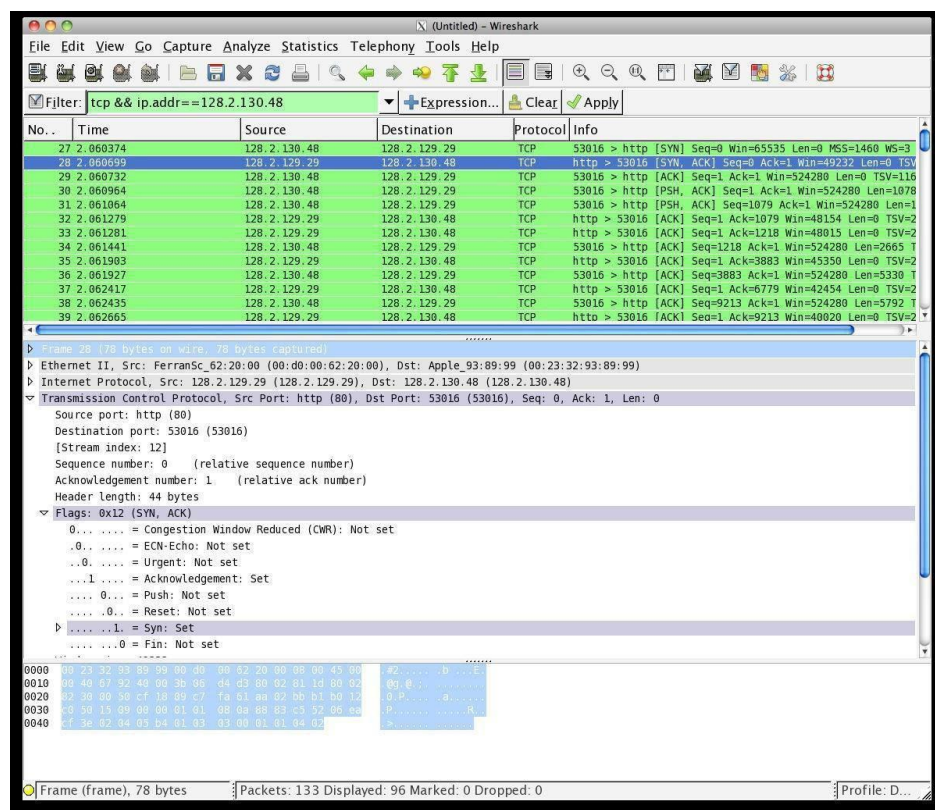
12) Let's look at what you captured. First, filter the results to look for TCP packets and to only look at those going to and from your computer with the filter "**tcp && ip.addr == <your IP address>**". If you have other services running on your computer, you might want to further filter so you only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



The image shows a Wireshark packet capture window titled 'CN_WEEK_5_WIRESHARK_2.pcapng'. The filter bar at the top shows 'tcp&&ip.addr==192.168.43.230'. The packet list pane shows several packets, with packet 117 selected. The packet details pane shows the 'Transmission Control Protocol' section for packet 117, which is a TCP segment from 192.168.43.230 to 20.198.162.76. The segment has a source port of 17875 and a destination port of 443. The sequence number is 1 (relative sequence number) and the length is 43. The packet bytes pane shows the raw data of the segment, which is a TLSv1.2 application data packet.

No.	Time	Source	Destination	Protocol	Length	Info
111	90.216528	192.168.43.230	20.198.162.76	TCP	54	17875 → 443 [ACK] Seq=130 Ack=523 Win=258 Len=0
112	103.278051	192.168.43.230	20.198.162.76	TLSv1.2	154	Application Data
113	103.578539	192.168.43.230	20.198.162.76	TCP	154	[TCP Retransmission] 17874 → 443 [PSH, ACK] Seq=102 Ack=172 Win=257 Len=100
114	103.879371	192.168.43.230	20.198.162.76	TCP	154	[TCP Retransmission] 17874 → 443 [PSH, ACK] Seq=102 Ack=172 Win=257 Len=100
115	104.479684	192.168.43.230	20.198.162.76	TCP	154	[TCP Retransmission] 17874 → 443 [PSH, ACK] Seq=102 Ack=172 Win=257 Len=100
116	105.680604	192.168.43.230	20.198.162.76	TCP	154	[TCP Retransmission] 17874 → 443 [PSH, ACK] Seq=102 Ack=172 Win=257 Len=100
117	105.738763	192.168.43.230	20.198.162.76	TLSv1.2	89	Application Data

Transmission Control Protocol, Src Port: 17875, Dst Port: 443, Seq: 1, Ack: 1, Len: 43

Source Port: 17875
Destination Port: 443
[Stream index: 0]
[Conversation completeness: Incomplete (12)]
[TCP Segment Len: 43]
Sequence Number: 1 (relative sequence number)
Sequence Number (raw): 16521913
[Next Sequence Number: 44 (relative sequence number)]

0000 ae 85 f0 b8 05 f3 20 39 26 6b 68 c5 00 00 45 00(9 &kh...E
0010 00 53 82 90 40 00 00 06 d4 73 c0 a8 2b e6 14 c6 ..S..:..s+...
0020 a2 4c 45 d3 01 bb 00 fc 1a b9 33 91 4e 9e 50 18 ..LE.....:3-N-P
0030 00 ff be 7c 00 00 17 03 03 00 26 00 00 00 00 00 ...:.....&.....
0040 00 00 45 93 81 a9 04 18 e8 fa f1 73 0e d1 0e 6b ..E.....:s:k
0050 9d 09 3a 01 d5 de 36 13 85 77 1b dd 2d 98 ae 25 ...:..6...w...X
0060 04

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

No.	Time	Source	Destination	Protocol	Length	Info
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
Frame 1: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface 0						
Ethernet II, Src: Apple_1f:d4:56 (b8:e8:56:1f:d4:56), Dst: Tp-LinkT_f8:6d:f9 (a0:f3:c1:f8:6d:f9)						
Internet Protocol Version 4, Src: 192.168.1.8 (192.168.1.8), Dst: 128.119.245.12 (128.119.245.12)						
Transmission Control Protocol, Src Port: 60706 (60706), Dst Port: http (80), Seq: 0, Len: 0						
Source port: 60706 (60706)						
Destination port: http (80)						
[Stream index: 0]						
Sequence number: 0 (relative sequence number)						
Header length: 44 bytes						
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						
window size value: 65535						
[calculated window size: 65535]						

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

No.	Time	Source	Destination	Protocol	Length	Info
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

Frame 4: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
 Ethernet II, Src: Tp-LinkTf8:6d:f9 (a0:f3:c1:f8:6d:f9), Dst: Apple_1f:d4:56 (b8:e8:56:1f:d4:56)
 Internet Protocol Version 4, Src: 128.119.245.12 (128.119.245.12), Dst: 192.168.1.8 (192.168.1.8)
 Transmission Control Protocol, Src Port: http (80), Dst Port: 60706 (60706), Seq: 0, Ack: 1, Len: 0

Source port: http (80)
 Destination port: 60706 (60706)
 [Stream index: 0]
 Sequence number: 0 (relative sequence number)
 Acknowledgment number: 1 (relative ack number)
 Header length: 40 bytes

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
 window size value: 5792
 [calculated window size: 5792]

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 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.

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

Filter: tcp		Expression...		Clear	Apply	Save
No.	Time	Source	Destination	Protocol	Length	Info
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
Frame 6: 644 bytes on wire (5152 bits), 644 bytes captured (5152 bits) on interface 0 Ethernet II, Src: Apple_1f:d4:56 (b8:e8:56:1f:d4:56), Dst: Tp-LinkT_f8:6d:f9 (a0:f3:c1:f8:6d:f9) Internet Protocol Version 4, Src: 192.168.1.8 (192.168.1.8), Dst: 128.119.245.12 (128.119.245.12) Transmission Control Protocol, Src Port: 60706 (60706), Dst Port: http (80), Seq: 1, Ack: 1, Len: 578 Source port: 60706 (60706) Destination port: http (80) [Stream index: 0] Sequence number: 1 (relative sequence number) [Next sequence number: 579 (relative sequence number)] Acknowledgment number: 1 (relative ack number) Header length: 32 bytes Flags: 0x018 (PSH, ACK) 000. = Reserved: Not set ...0 = Nonce: Not set0... = Congestion window Reduced (CWR): Not set0... = ECN-Echo: Not set0... = Urgent: Not set1... = Acknowledgment: Set1... = Push: Set0... = Reset: Not set						
0000	a0 f3 c1 f8 6d f9 b8 e8	56 1f d4 56 08 00 45 00	...	m...	V..V..E.	
0010	02 76 f6 5a 40 00 40 06	0a f3 c0 a8 01 08 80 77	...	V.Z@.W	
0020	f5 0c ed 22 00 50 1f e9	a7 e8 79 47 80 0a 80 18P..	..yG...	
0030	20 2b bf 08 00 00 01 01	08 0a 05 16 f8 ee 86 ca	+	
0040	ee 56 50 4f 53 54 20 2f	77 69 72 65 73 68 61 72	.	VPOST	/ wireshar	
0050	6b 2d 6c 61 62 73 2f 6c	61 62 33 2d 31 2d 72 65	k-TAB5/l	ab3-l-re		
0060	70 6c 79 2e 68 74 6d 20	48 54 54 50 2f 31 2e 31	ply.htm	HTTP/1.1		
0070	0d 0a 48 6f 73 74 3a 20	67 61 69 61 2e 63 73 2e	.Host:	gaia.cs.		
0080	75 6d 61 73 73 2e 65 64	75 0d 0a 43 6f 6e 74 65	umass.ed	u..Conte		
0090	6e 74 2d 54 79 70 65 3a	20 6d 75 6c 74 69 70 61	nt-Type:	multipa		
00a0	72 74 2f 66 6f 72 6d 2d	64 61 74 61 3b 20 62 6f	rt/form-	data; bo		

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

No.	Time	Source	Destination	Protocol	Length	Info
4	0.26949200	192.168.1.8	192.168.1.8	TCP	74	http > 60706 [EST, ACK] Seq=0 Ack=1 win=0 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
Frame 6: 644 bytes on wire (5152 bits), 644 bytes captured (5152 bits) on interface 0						
Ethernet II, Src: Apple_1f:d4:56 (b8:e8:56:1f:d4:56), Dst: Tp-LinkT_f8:6d:f9 (a0:f3:c1:f8:6d:f9)						
Internet Protocol Version 4, Src: 192.168.1.8 (192.168.1.8), Dst: 128.119.245.12 (128.119.245.12)						
Transmission Control Protocol, Src Port: 60706 (60706), Dst Port: http (80), Seq: 1, Ack: 1, Len: 578						
Source port: 60706 (60706)						
Destination port: http (80)						
[Stream index: 0]						
Sequence number: 1 (relative sequence number)						
[Next sequence number: 579 (relative sequence number)]						
Acknowledgment number: 1 (relative ack number)						
Header length: 32 bytes						
Flags: 0x018 (PSH, ACK)						
000. = Reserved: Not set						
...0 = Nonce: Not set						
.... 0... = Congestion window Reduced (CWR): Not set						
.... 0... = ECN-Echo: Not set						
0000	a0	f3	c1	f8	6d	f9 b8 e8 56 1f d4 56 08 00 45 00 ...m... V.V..E.
0010	02	76	f6	5a	40	00 40 06 0a f3 c0 a8 01 08 80 77 ...V.Z@.0.w
0020	f5	0c	ed	22	00	50 1f e9 a7 e8 79 47 80 0a 80 18 ...P..yG....
0030	20	2b	bf	08	00	00 01 08 0a 05 16 f8 ee 86 ca ..7.....
0040	ee	56	50	4f	53	54 20 2f 77 69 72 65 73 68 61 72 ..VPOST/ wireshar
0050	6b	2d	6c	61	62	73 2f 6c 61 62 33 2d 31 2d 72 65 k-labs/ ab3-1-re
0060	70	6c	79	2e	68	74 6d 20 48 54 54 50 2f 31 2e 31 ply.htm HTTP/1.1
0070	0d	0a	48	6f	73	74 3a 20 67 61 69 61 2e 63 73 2e ..Host: gaia.cs.
0080	75	6d	61	73	73	2e 65 64 75 0d 0a 43 6f 6e 74 65 umass.ed u..conte
0090	6e	74	2d	54	79	70 65 3a 20 6d 75 6c 74 69 70 61 nt-type: multipa
00a0	72	74	2f	66	6f	72 6d 2d 64 61 74 61 3b 20 62 6f rt/form- data; bo

Segments 1-6

No.	Time	Source	Destination	Protocol	Length	Info	
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	
16	0.36995200	128.119.245.12	192.168.1.8	TCP	66	http > 60706 [ACK] Seq=1 Ack=3612 win=13952 Len=0 TSval=	
17	0.37006300	192.168.1.8	128.119.245.12	TCP	1514	60706 > http [ACK] Seq=7956 Ack=1 win=131760 Len=1448 TS	
18	0.37006400	192.168.1.8	128.119.245.12	TCP	1514	60706 > http [ACK] Seq=9404 Ack=1 win=131760 Len=1448 TS	
19	0.47996500	128.119.245.12	192.168.1.8	TCP	66	http > 60706 [ACK] Seq=1 Ack=5060 win=16896 Len=0 TSval=	
20	0.48010500	192.168.1.8	128.119.245.12	TCP	1514	60706 > http [ACK] Seq=10852 Ack=1 win=131760 Len=1448 T	
21	0.48010600	192.168.1.8	128.119.245.12	TCP	1514	60706 > http [ACK] Seq=12300 Ack=1 win=131760 Len=1448 T	
22	0.48249200	128.119.245.12	192.168.1.8	TCP	66	http > 60706 [ACK] Seq=1 Ack=6508 win=19712 Len=0 TSval=	
Frame 10: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0							
Ethernet II, Src: Tp-LinkT_f8:6d:f9 (a0:f3:c1:f8:6d:f9), Dst: Apple_1f:d4:56 (b8:e8:56:1f:d4:56)							
Internet Protocol Version 4, Src: 128.119.245.12 (128.119.245.12), Dst: 192.168.1.8 (192.168.1.8)							
Transmission Control Protocol, Src Port: http (80), Dst Port: 60706 (60706), Seq: 1, Ack: 579, Len: 0							
Source port: http (80)							
Destination port: 60706 (60706)							
[Stream index: 0]							
Sequence number: 1 (relative sequence number)							
Acknowledgment number: 579 (relative ack number)							
Header length: 32 bytes							
Flags: 0x010 (ACK)							
000. = Reserved: Not set							
...0 = Nonce: Not set							
n = Congestion window reduced (CWR): Not set							
0000	b8	e8	56	1f	d4	56 a0 f3 c1 f8 6d f9 08 00 45 00 ...V..V.. m...E.	
0010	00	34	6f	2d	40	00 31 06 a3 62 80 77 f5 0c c0 a8 ..40-@.1. .b.w...	
0020	01	08	00	50	ed	22 79 47 80 0a 1f e9 aa 2a 80 10 ...P..yG	
0030	00	37	1a	82	00	00 01 01 08 0a 86 ca ef 27 05 16 ..7.....	
0040	f8	ee					..

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: $\text{EstimatedRTT} = 0.875 * \text{EstimatedRTT} + 0.125 * \text{SampleRTT}$

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:

$$\text{EstimatedRTT} = 0.875 * 0.095674 + 0.125 * 0.095864 = 0.09569775 \text{ s}$$

EstimatedRTT after the receipt of the ACK of segment 3:

$$\text{EstimatedRTT} = 0.875 * 0.09569775 + 0.125 * 0.09682 = 0.09583803125 \text{ s}$$

EstimatedRTT after the receipt of the ACK of segment 4:

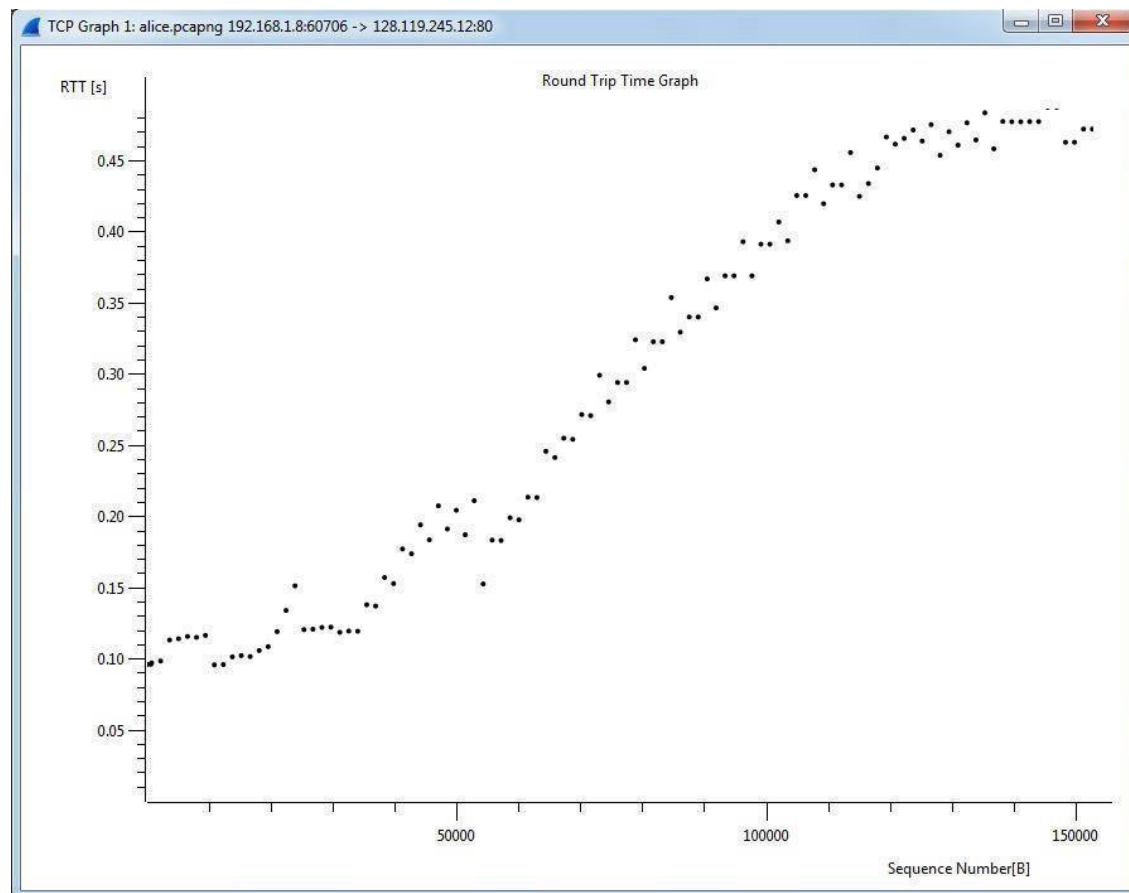
$$\text{EstimatedRTT} = 0.875 * 0.09583803125 + 0.125 * 0.098154 = 0.09612752734 \text{ s}$$

EstimatedRTT after the receipt of the ACK of segment 5:

$$\text{EstimatedRTT} = 0.875 * 0.09612752734 + 0.125 * 0.112884 = 0.09822208642 \text{ s}$$

EstimatedRTT after the receipt of the ACK of segment 6:

$$\text{EstimatedRTT} = 0.875 * 0.09822208642 + 0.125 * 0.113781 = 0.10016695061 \text{ s}$$



Round Trip Time Graph

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

Answer

No.	Time	Source	Destination	Protocol	Length	Info
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=8
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
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 TS
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=2
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=2
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

Options: (12 bytes), No-operation (NOP), No-operation (NOP), Timestamps

- No-operation (NOP)
- No-operation (NOP)
- Timestamps: TSval 85391598, TSecr 2261446230
 - Kind: Timestamp (8)
 - Length: 10
 - Timestamp value: 85391598
 - Timestamp echo reply: 2261446230

[SEQ/ACK analysis]

Data (578 bytes)

Data: 504f5354202f77697265736861726b2d6c6162732f6c6162...

[Length: 578]

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

No.	Time	Source	Destination	Protocol	Length	Info
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=8
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
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 TS
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=2
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=2
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

.... ..0. = Urgent: Not set

.... ..1. = Acknowledgment: Set

.... ..0. = Push: Not set

.... ..0. = Reset: Not set

.... ..1. = Syn: Set

.... ..0. = Fin: Not set

window size value: 5792

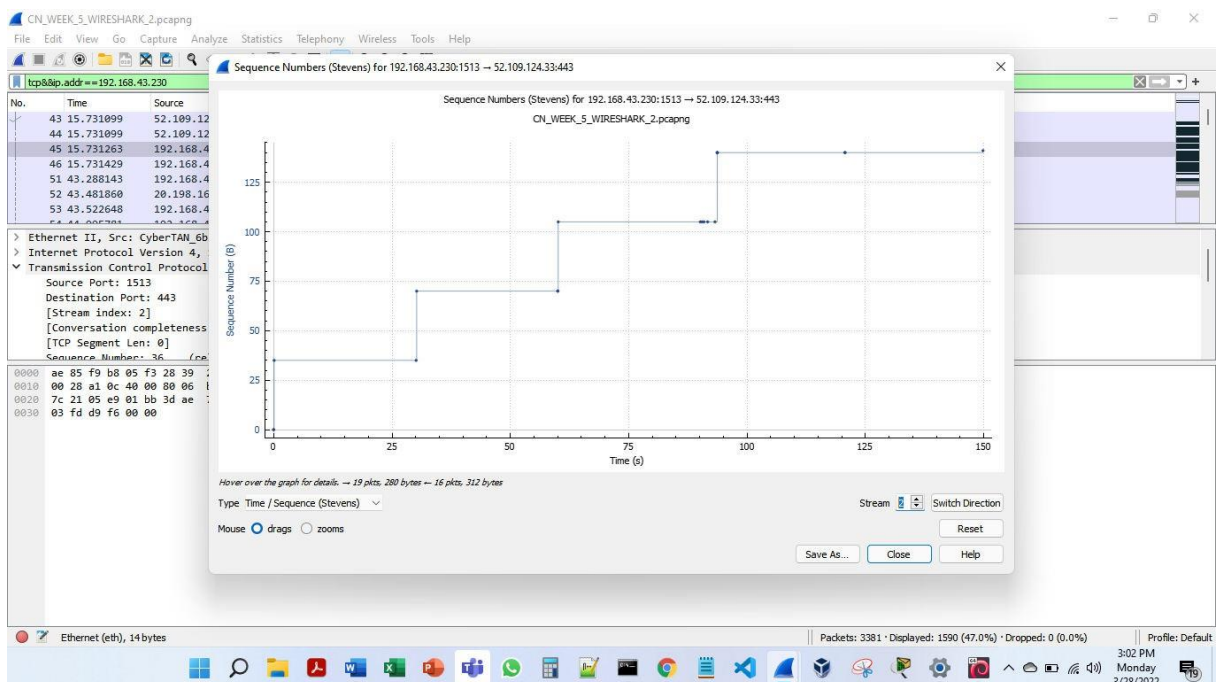
[calculated window size: 5792]

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 receiver 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.

Home Page: <http://uniteng.com>

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

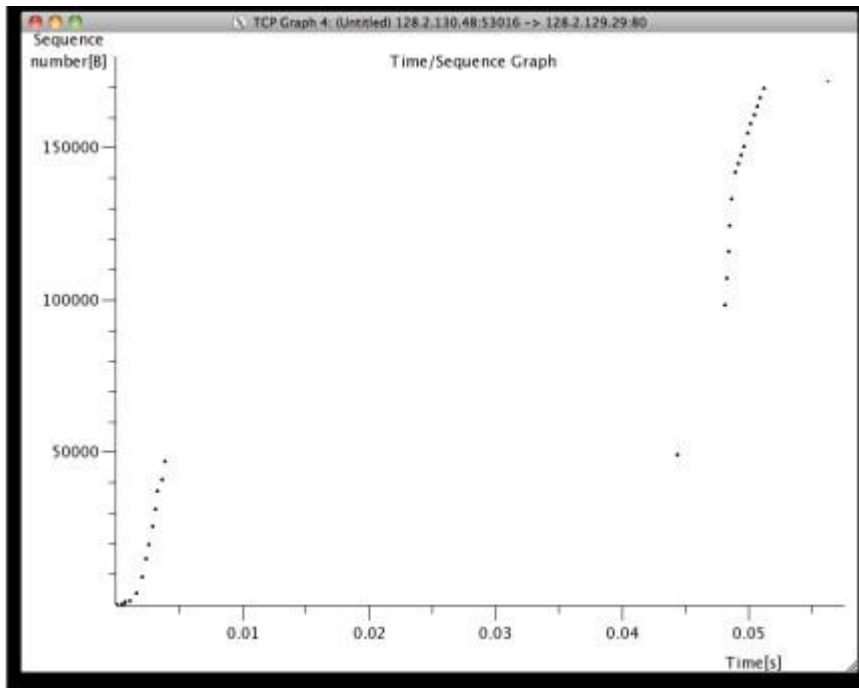
9. What is the throughput (bytes transferred per unit time) for the TCP connection? Explain how you calculated this value.

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).

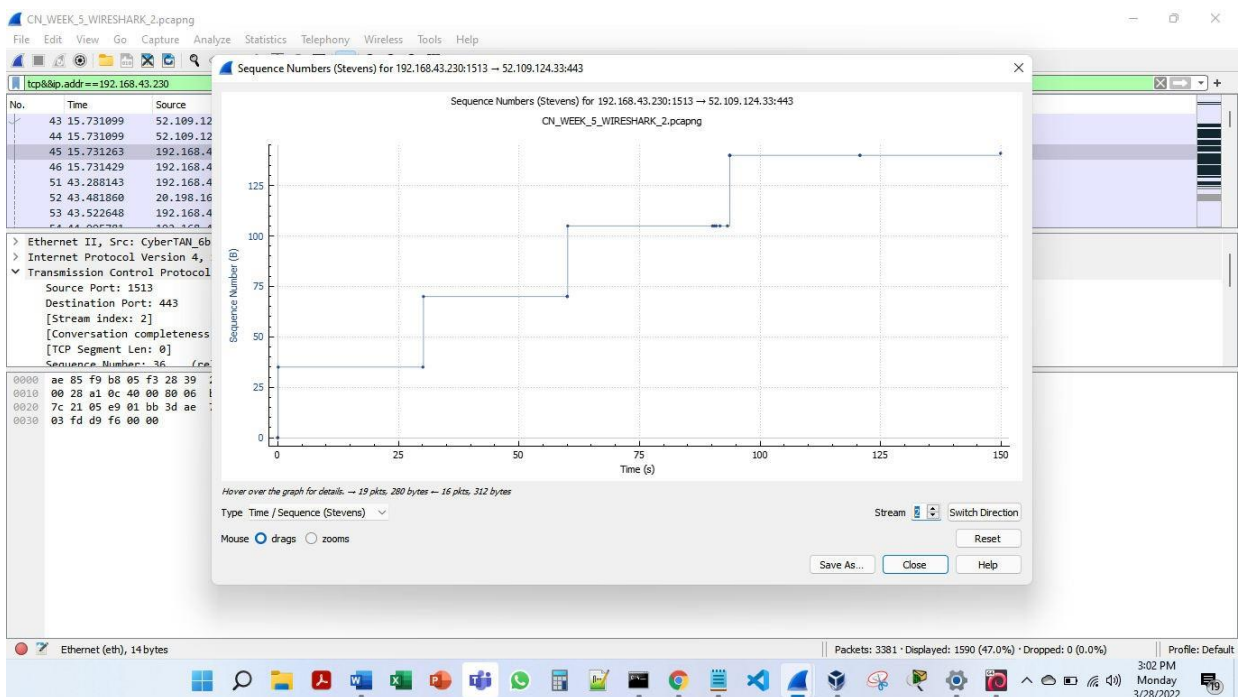


You should see a plot that looks like the following plot (though the individual plotted values may differ quite a bit). Here, each dot represents a TCP segment sent, plotting the sequence number of the segment versus the time at which it was sent. Note that a set of dots stacked above each other represents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

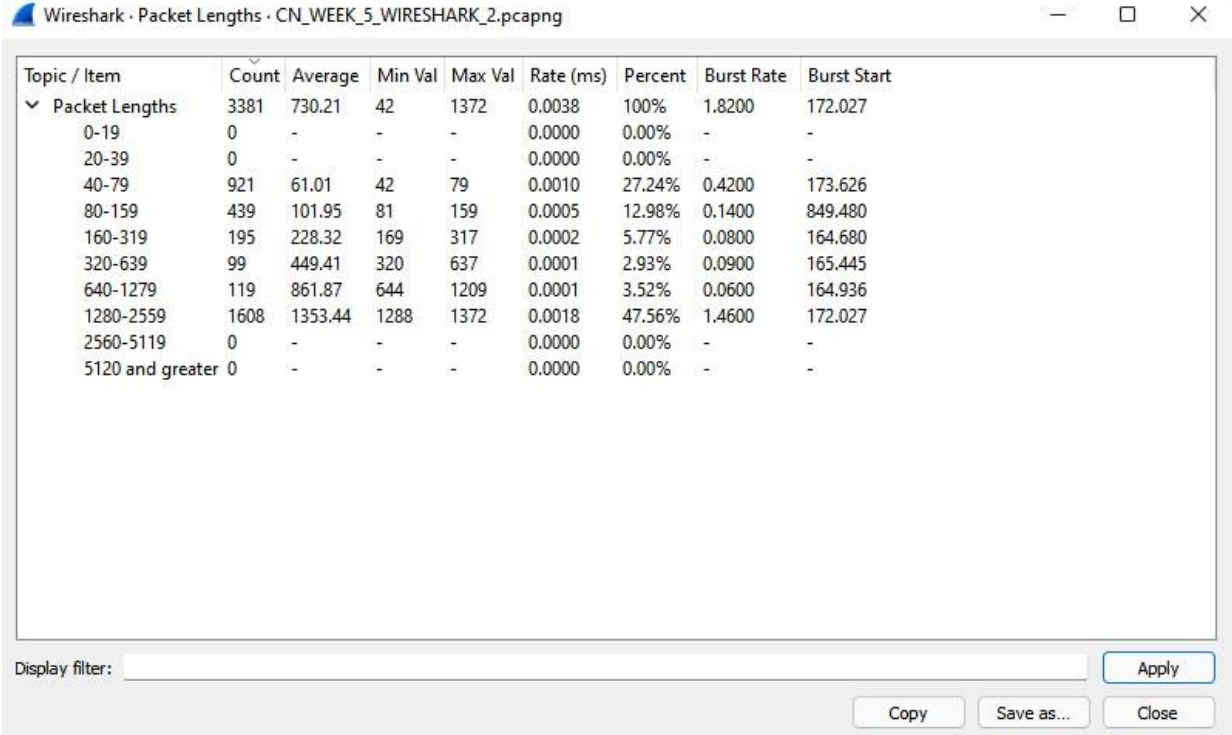
The slow start of the TCP seems to begin at about 0.27 seconds and then ends at about 0.35 seconds. Congestion avoidance takes over at about 0.7 seconds because it cut down the amount being sent.



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.**



Wireshark · Packet Lengths · CN_WEEK_5_WIRESHARK_2.pcapng

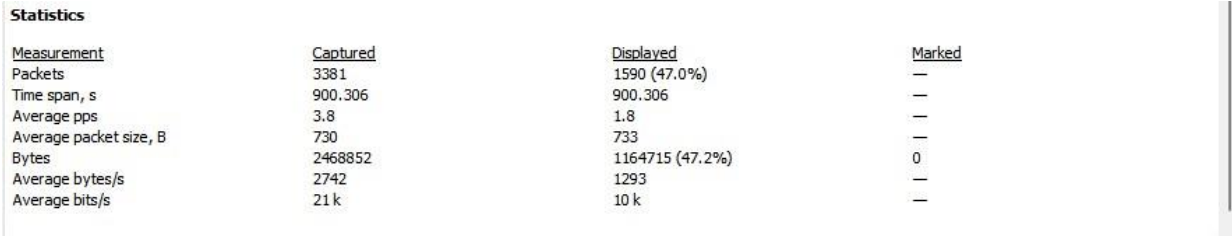
Topic / Item	Count	Average	Min Val	Max Val	Rate (ms)	Percent	Burst Rate	Burst Start
Packet Lengths	3381	730.21	42	1372	0.0038	100%	1.8200	172.027
0-19	0	-	-	-	0.0000	0.00%	-	-
20-39	0	-	-	-	0.0000	0.00%	-	-
40-79	921	61.01	42	79	0.0010	27.24%	0.4200	173.626
80-159	439	101.95	81	159	0.0005	12.98%	0.1400	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%	-	-

Display filter:

GO to Statistics->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.

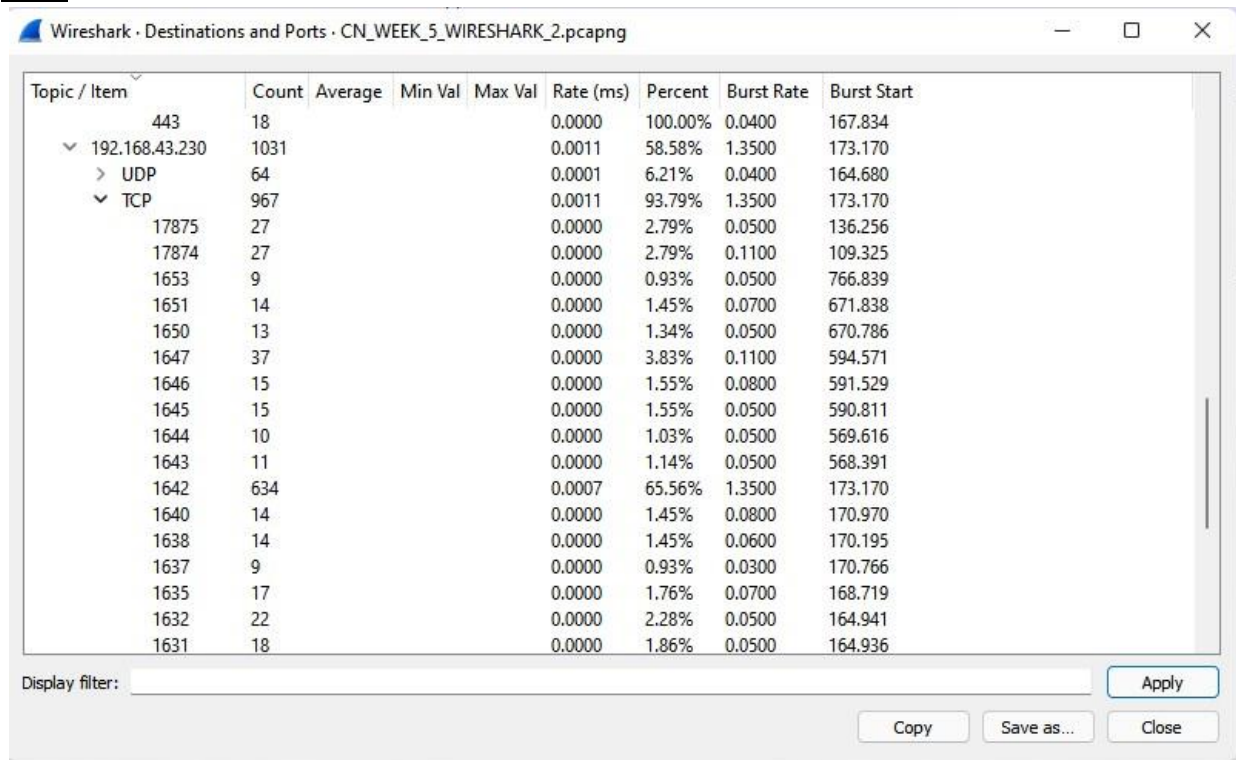


Measurement	Captured	Displayed	Marked
Packets	3381	1590 (47.0%)	—
Time span, s	900.306	900.306	—
Average pps	3.8	1.8	—
Average packet size, B	730	733	—
Bytes	2468852	1164715 (47.2%)	0
Average bytes/s	2742	1293	—
Average bits/s	21 k	10 k	—

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



Wireshark · Destinations and Ports · CN_WEEK_5_WIRESHARK_2.pcapng

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

Display filter: [] Apply

Copy Save as... Close

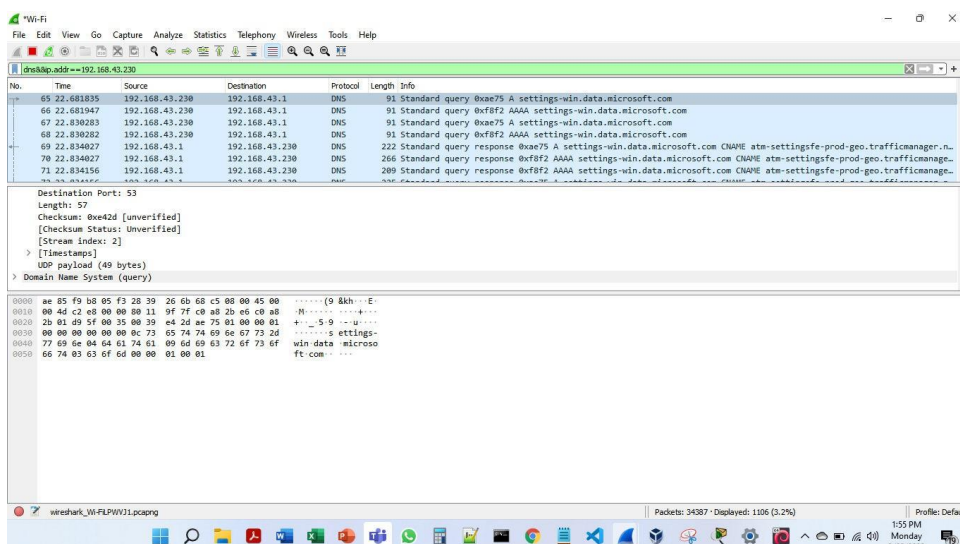
DESTINATION

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.



Wireshark · Wi-Fi · Capture · Analyze · Statistics · Telephony · Wireless · Tools · Help

Filter: dns[dstip.addr==192.168.43.230]

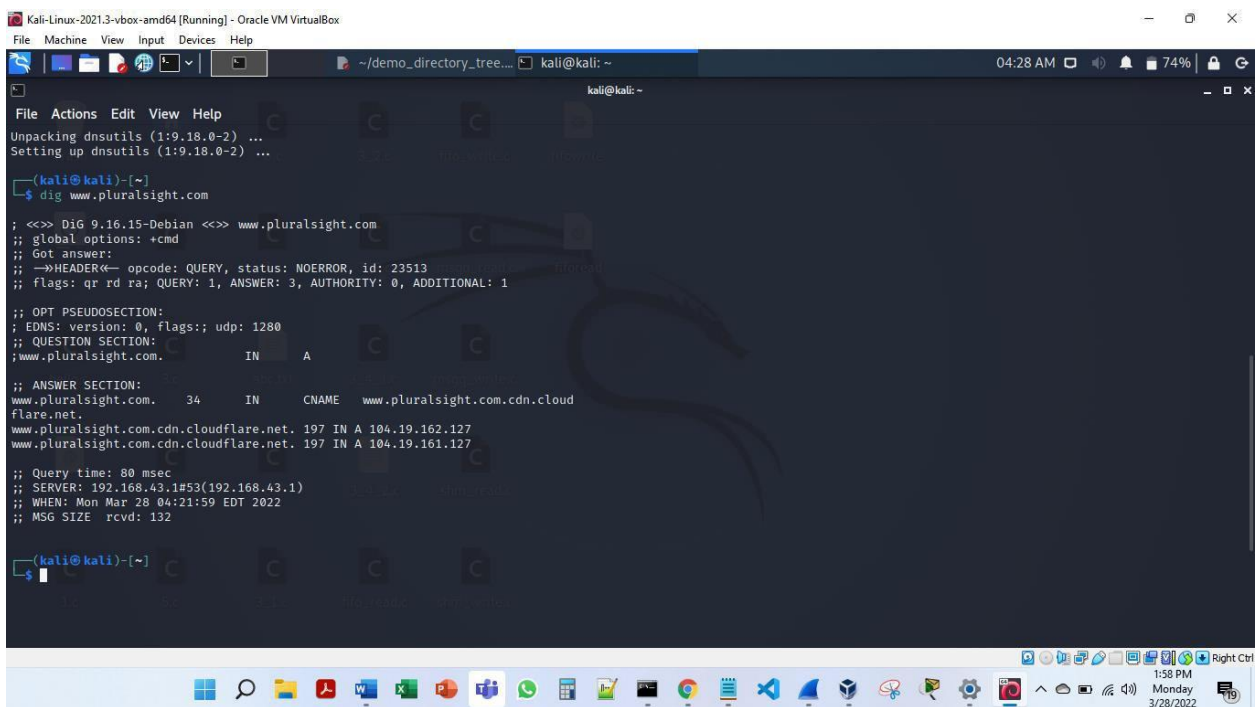
No.	Time	Source	Destination	Protocol	Length	Info
65	22.681835	192.168.43.230	192.168.43.1	DNS	91	Standard query 0xae75 A settings-win.data.microsoft.com
66	22.681947	192.168.43.230	192.168.43.1	DNS	91	Standard query 0xfbf2 AAAA settings-win.data.microsoft.com
67	22.830283	192.168.43.230	192.168.43.1	DNS	91	Standard query 0xae75 A settings-win.data.microsoft.com
68	22.830282	192.168.43.230	192.168.43.1	DNS	91	Standard query 0xfbf2 AAAA settings-win.data.microsoft.com
69	22.834827	192.168.43.1	192.168.43.230	DNS	222	Standard query response 0xae75 A settings-win.data.microsoft.com CNAME atm-settingsfe-prod-geo.trafficmanager-n...
70	22.834827	192.168.43.1	192.168.43.230	DNS	266	Standard query response 0xfbf2 AAAA settings-win.data.microsoft.com CNAME atm-settingsfe-prod-geo.trafficmanage...
71	22.834156	192.168.43.1	192.168.43.230	DNS	289	Standard query response 0xfbf2 AAAA settings-win.data.microsoft.com CNAME atm-settingsfe-prod-geo.trafficmanage...

Destination Port: 53
Length: 57
Checksum: 0xae42d [unverified]
[Checksum Status: Unverified]
[Stream Index: 2]
[Timestamps]
UDP payload (49 bytes)
Domain Name System (query)

0000 ae 85 f9 b8 05 f3 28 39 26 6b 68 c5 00 00 45 00(9 &kh...E...
0010 00 4d c2 e8 00 00 11 9f 7f c0 a8 2b e6 c0 a8 M.....
0020 2b 01 d9 5f 00 05 00 39 e4 2d ae 75 01 00 00 01 t...5 9...
0030 00 00 00 00 00 00 73 65 74 74 69 6d 67 73 2ds ettings-
0040 77 69 6e 04 64 61 74 61 09 6d 69 63 72 6f 73 6f win data microso
0050 66 74 03 63 6f 6d 00 00 01 00 01 ft.com....

Wireshark_Wi-Fi_Wi-Fi.pcapng Packets: 34387 · Displayed: 1106 (3.2%) Profile: Default
1:55 PM Monday 3/28/2022

- 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).



```
(kali@kali)-[~]
$ dig www.pluralsight.com

;<<>> DiG 9.16.15-Debian <<>> www.pluralsight.com
;; global options: +cmd
;; Got answer:
;;->HEADER<- opcode: QUERY, status: NOERROR, id: 23513
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags: udp: 1280
;; QUESTION SECTION:
;; www.pluralsight.com.      IN      A

;; ANSWER SECTION:
www.pluralsight.com.  34      IN      CNAME   www.pluralsight.com.cdn.cloudflare.net.
www.pluralsight.com.cdn.cloudflare.net. 197 IN A 104.19.162.127
www.pluralsight.com.cdn.cloudflare.net. 197 IN A 104.19.161.127

;; Query time: 80 msec
;; SERVER: 192.168.43.1#53(192.168.43.1)
;; WHEN: Mon Mar 28 04:21:59 EDT 2022
;; MSG SIZE rcvd: 132

(kali@kali)-[~]
$
```

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	21H2.1
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.

```

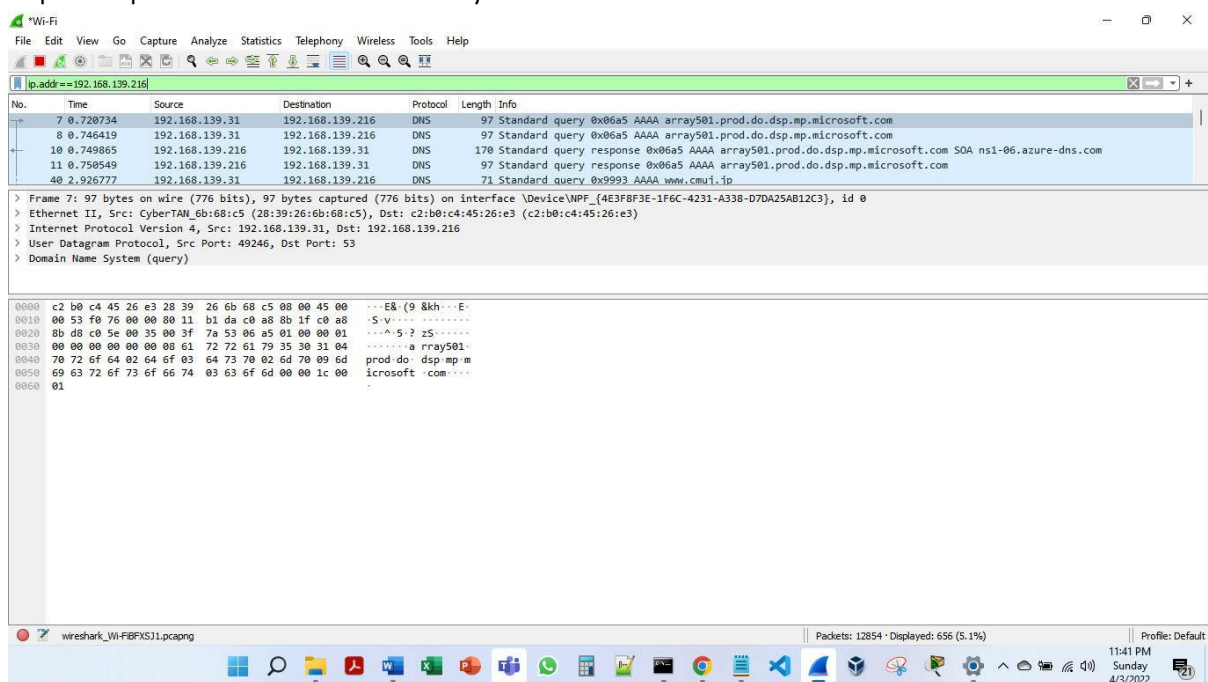
Tracing route to cmuj.jp [122.17.163.205]
over a maximum of 30 hops:

 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.

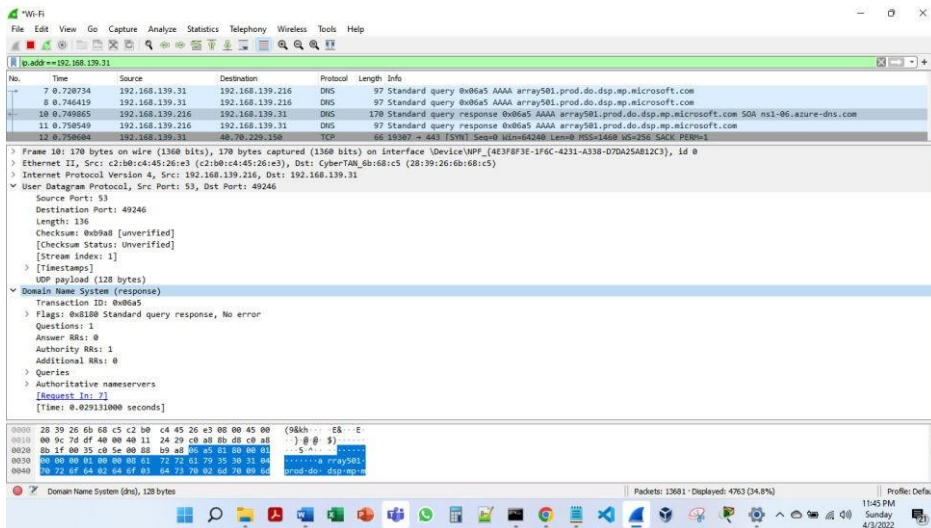
```

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

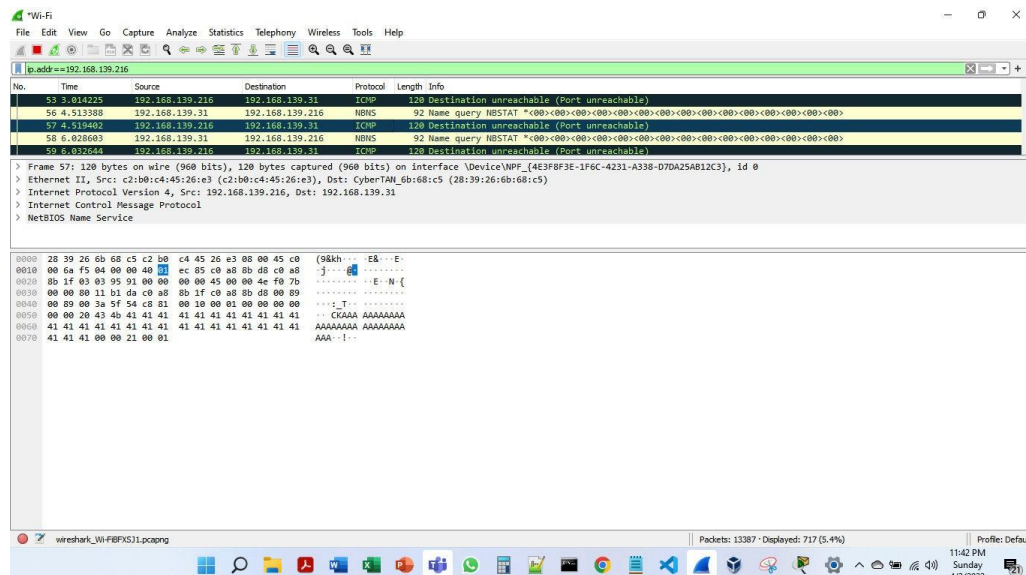


12) What are the transmitted segments like? Describe the important features of the segments you observe. In particular, examine the destination port field. What characteristics do you observe about this port number and why would it be chosen so?

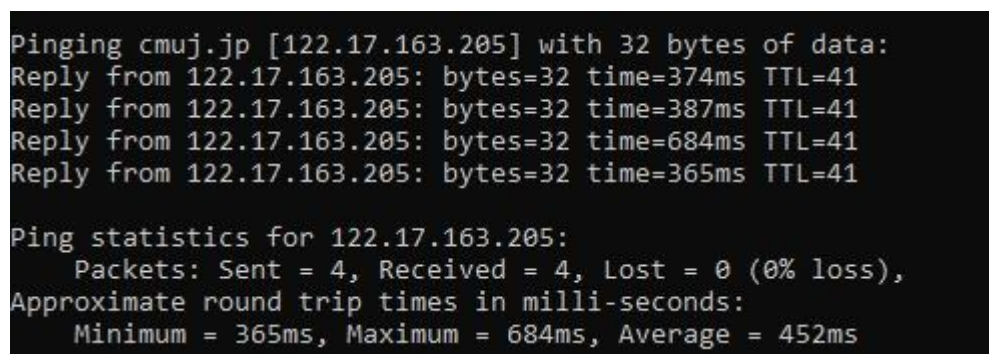
13) What about the return packets? What are the values of the various header fields?



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?



Finishing up

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

