Prelab 2 - HTTP, DNS, and TCP:
Suggested Resources:
https://www.ietf.org/rfc/rfc2616.txt
https://www.ietf.org/rfc/rfc1035.txt
https://linux.die.net/man/
http://www.tcpipguide.com/free/
HTTP Questions
1. [7 pts] Choose 5 HTTP status codes and describe each one.

- 200 OK - request succeeded, requested object later in this msg
- 301 Moved Permanently - Requested object moved, new location specified later in this msg(Location:)
- 400 Bad Request - Request msg not understood by server
- 404 Not Found - requested document not found on this server
- 505 HTTP - Version Not Supported

2. [7 pts] List the 8 HTTP 1.1 methods and explain what they do.

GET - Used to retrieve whatever is stored or produced by the resource located at the requested-URL

POST - Used to submit data to the resource located at the specified Request-URI.

HEAD - The HEAD method is identical to the GET method except that an HTTP 1.1 server should not return a message-body in the response.

PUT - The PUT method allows for data to be transferred to an HTTP server and stored at the location identified by the Request-URI.

OPTIONS - The OPTIONS method represents a request for information about the communication options available on the request/response chain identified by the Request-URI.

DELETE - The DELETE method requests that the origin server delete the resource identified by the Request-URI.

TRACE - The TRACE method is used to invoke a remote, application-layer loop-back of the request message.

CONNECT - The CONNECT message type is used to specify a proxy connection to the resource identified by the Request-URI.

wget and telnet are two commonly known command line tools for testing and debugging. Answer the following questions by using your Mininet VM's terminal or the Unix timeshare (see Lab 1 for instructions on connecting to the timeshare).

3. [7 pts] Use wget on example.com to view the last modified date of the webpage. What was the HTTP return status given and what command was used to do this? (The command should not download the file! Hint: Look into the wget man page.)

The return status was 200 OK

Wget –server-response –spider example.com

4. [7 pts] Look up the telnet command. Use telnet to connect to www.telehack.com, then type starwars What does this telnet server do?

This telnet server is playing Starwar using Ascii characters.

DNS Questions
5. [7 pts] In your own words describe what a DNS resource record (RR) is. Now using

the command line tool nslookup find the MX resource record of ucsc.edu. What does this resource record mean?

A DNS resource record(RR) is a type of data stored in a DNS server. This maps a domain name (URL) to a specific IP address. The resource record means the ucsc.edu domain has a mail exchanger, which directs email to mail servers.

6. [7 pts] What does the command nslookup -type=ns . do? Explain its output. (Note: the . is part of the command!)

The command is used to query the root name servers for the DNS. When this command is executed, the nslookup tool will send a DNS query to one of the root name servers, requesting the list of name servers responsible for the root zone.

TCP Questions

7. [10 pts] How can multiple application services running on a single machine with a single IP address be uniquely identified?

Multiple applications can run on a single machine with a single IP address with different port numbers uniquely identified the application.

8. [9 pts] What is the purpose of the window mechanism in TCP?

The purpose of the window mechanism in TCP is to control the amount of data sent in a transmission.

9. [9 pts] What is an MTU? What happens when a packet is larger than the MTU?

MTU is Maximum Transmission Unit, when a packet is larger than the MTU it will fragment into smaller packets also known as IP fragmentation. Once the packets arrive at the end user, they are reassembled into the original packet by the receiving host.

Lab 2 - HTTP, DNS, and TCP:

Suggested Resources:

http://packetbomb.com/understanding-the-tcptrace-time-sequence-graph-in-wireshark/

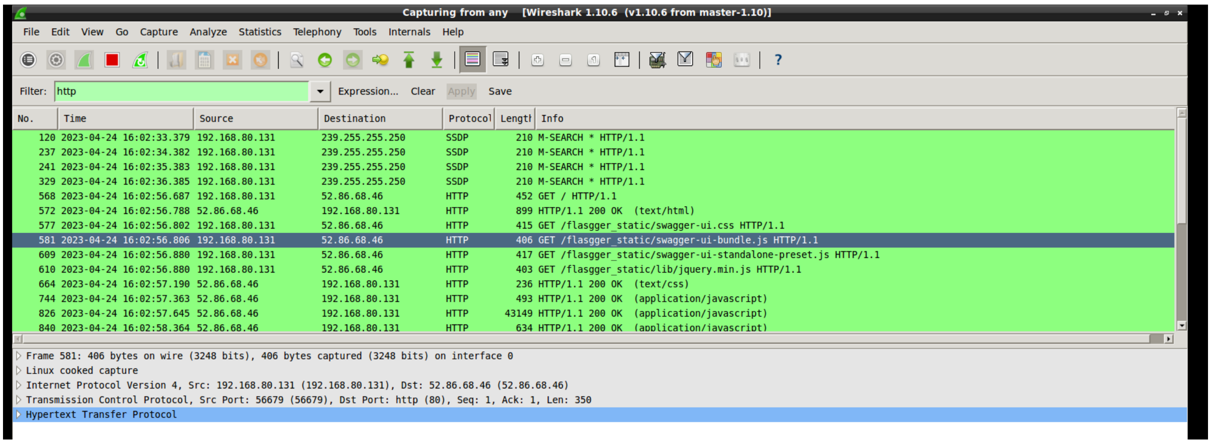
https://wiki.linuxfoundation.org/networking/netem

Part 1: HTTP

In this section, we will observe how the HTTP protocol operates. We will do this by using the Mininet VM. Begin by opening Wireshark and listening on the ‘any’ interface.

Open Chromium and navigate to http://httpbin.org

1. [10 pts] Find the HTTP packet that corresponds to the initial request that your computer made. Take a screenshot of this packet. What HTTP method did your computer use to make this request?



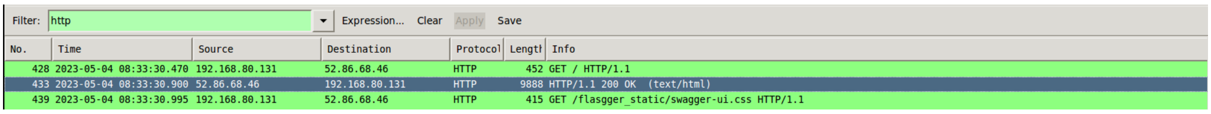
Filter: http

No.	Time	Source	Destination	Protocol	Length	Info
120	2023-04-24 16:02:33.379	192.168.80.131	239.255.255.250	SSDP	210	M-SEARCH * HTTP/1.1
237	2023-04-24 16:02:34.382	192.168.80.131	239.255.255.250	SSDP	210	M-SEARCH * HTTP/1.1
241	2023-04-24 16:02:35.383	192.168.80.131	239.255.255.250	SSDP	210	M-SEARCH * HTTP/1.1
329	2023-04-24 16:02:36.385	192.168.80.131	239.255.255.250	SSDP	210	M-SEARCH * HTTP/1.1
568	2023-04-24 16:02:56.687	192.168.80.131	52.86.68.46	HTTP	452	GET / HTTP/1.1
572	2023-04-24 16:02:56.788	52.86.68.46	192.168.80.131	HTTP	899	HTTP/1.1 200 OK (text/html)
577	2023-04-24 16:02:56.802	192.168.80.131	52.86.68.46	HTTP	415	GET /flasgger_static/swagger-ui.css HTTP/1.1
581	2023-04-24 16:02:56.806	192.168.80.131	52.86.68.46	HTTP	406	GET /flasgger_static/swagger-ui-bundle.js HTTP/1.1
609	2023-04-24 16:02:56.880	192.168.80.131	52.86.68.46	HTTP	417	GET /flasgger_static/swagger-ui-standalone-preset.js HTTP/1.1
610	2023-04-24 16:02:56.880	192.168.80.131	52.86.68.46	HTTP	403	GET /flasgger_static/lib/jquery.min.js HTTP/1.1
664	2023-04-24 16:02:57.190	52.86.68.46	192.168.80.131	HTTP	236	HTTP/1.1 200 OK (text/css)
744	2023-04-24 16:02:57.363	52.86.68.46	192.168.80.131	HTTP	493	HTTP/1.1 200 OK (application/javascript)
826	2023-04-24 16:02:57.645	52.86.68.46	192.168.80.131	HTTP	43149	HTTP/1.1 200 OK (application/javascript)
840	2023-04-24 16:02:58.364	52.86.68.46	192.168.80.131	HTTP	634	HTTP/1.1 200 OK (application/javascript)

```
▷ Frame 581: 406 bytes on wire (3248 bits), 406 bytes captured (3248 bits) on interface 0
▷ Linux cooked capture
▷ Internet Protocol Version 4, Src: 192.168.80.131 (192.168.80.131), Dst: 52.86.68.46 (52.86.68.46)
▷ Transmission Control Protocol, Src Port: 56679 (56679), Dst Port: http (80), Seq: 1, Ack: 1, Len: 350
▷ Hypertext Transfer Protocol
```

My computer used the GET method to make this request.

2. [10 pts] Find the HTTP packet that corresponds to the initial response the server made to your request. Take a screenshot of this packet. What HTTP status code did the server return? What is the content type of the response the server is sending back?



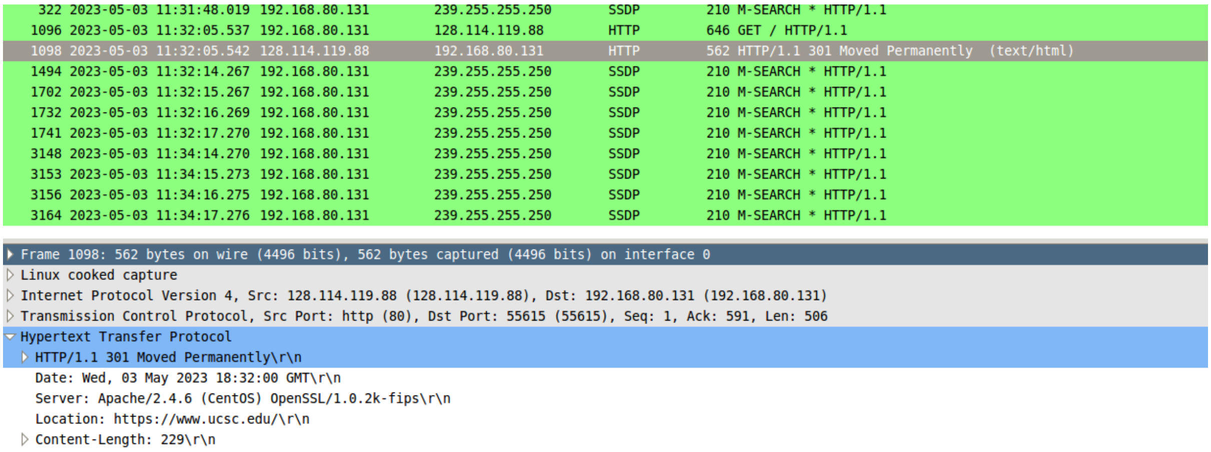
Filter: http

No.	Time	Source	Destination	Protocol	Length	Info
428	2023-05-04 08:33:30.470	192.168.80.131	52.86.68.46	HTTP	452	GET / HTTP/1.1
433	2023-05-04 08:33:30.900	52.86.68.46	192.168.80.131	HTTP	9888	HTTP/1.1 200 OK (text/html)
439	2023-05-04 08:33:30.995	192.168.80.131	52.86.68.46	HTTP	415	GET /flasgger_static/swagger-ui.css HTTP/1.1

The server returned a HTTP status code of 200 OK. The content type of the response is text/html.

Using Chromium and navigate to http://ucsc.edu

3. [10 pts] Find the HTTP packets that correspond to the initial request and response that your computer made. Take a screenshot of these packets. What’s different? Explain.

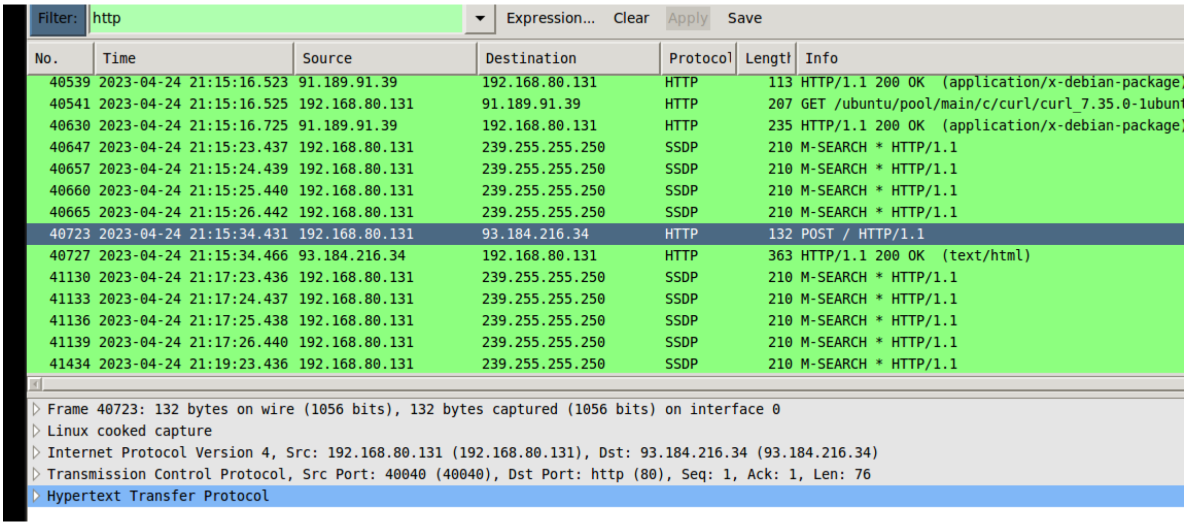


```
  322 2023-05-03 11:31:48.019 192.168.80.131    239.255.255.250   SSDP    210 M-SEARCH * HTTP/1.1
 1096 2023-05-03 11:32:05.537 192.168.80.131    128.114.119.88    HTTP    646 GET / HTTP/1.1
 1098 2023-05-03 11:32:05.542 128.114.119.88    192.168.80.131    HTTP    562 HTTP/1.1 301 Moved Permanently  (text/html)
 1494 2023-05-03 11:32:14.267 192.168.80.131    239.255.255.250   SSDP    210 M-SEARCH * HTTP/1.1
 1702 2023-05-03 11:32:15.267 192.168.80.131    239.255.255.250   SSDP    210 M-SEARCH * HTTP/1.1
 1732 2023-05-03 11:32:16.269 192.168.80.131    239.255.255.250   SSDP    210 M-SEARCH * HTTP/1.1
 1741 2023-05-03 11:32:17.270 192.168.80.131    239.255.255.250   SSDP    210 M-SEARCH * HTTP/1.1
 3148 2023-05-03 11:34:14.270 192.168.80.131    239.255.255.250   SSDP    210 M-SEARCH * HTTP/1.1
 3153 2023-05-03 11:34:15.273 192.168.80.131    239.255.255.250   SSDP    210 M-SEARCH * HTTP/1.1
 3156 2023-05-03 11:34:16.275 192.168.80.131    239.255.255.250   SSDP    210 M-SEARCH * HTTP/1.1
 3164 2023-05-03 11:34:17.276 192.168.80.131    239.255.255.250   SSDP    210 M-SEARCH * HTTP/1.1

▶ Frame 1098: 562 bytes on wire (4496 bits), 562 bytes captured (4496 bits) on interface 0
▷ Linux cooked capture
▷ Internet Protocol Version 4, Src: 128.114.119.88 (128.114.119.88), Dst: 192.168.80.131 (192.168.80.131)
▷ Transmission Control Protocol, Src Port: http (80), Dst Port: 55615 (55615), Seq: 1, Ack: 591, Len: 506
▽ Hypertext Transfer Protocol
  ▷ HTTP/1.1 301 Moved Permanently\r\n
    Date: Wed, 03 May 2023 18:32:00 GMT\r\n
    Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips\r\n
    Location: https://www.ucsc.edu/\r\n
  ▷ Content-Length: 229\r\n
```

The difference between the initial request and response is http and https. It’s asking you to connect to https by responding to a status code of 301, with the new location provided as https://www.ucsc.edu .

Using Chromium (or any other Linux utility you are comfortable with), find a way to make a HTTP packet with a method other than GET.

4. [10 pts] Take a screenshot of your packet, and explain what you did to create it.



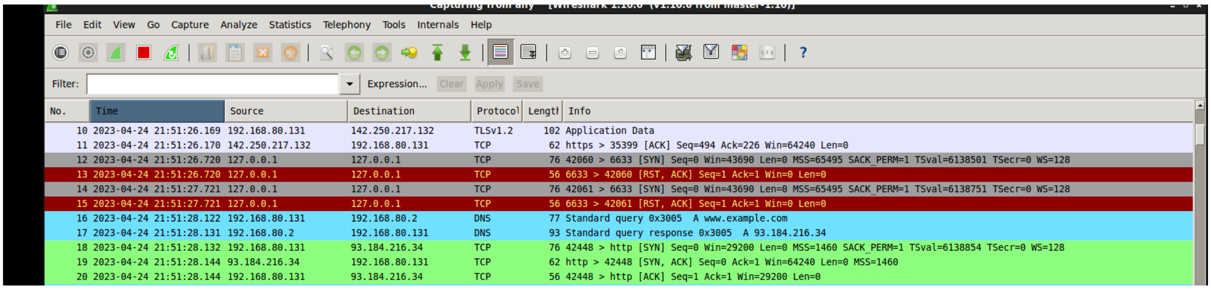
```
Filter: http

No.   Time                    Source            Destination       Protocol Length Info
40539 2023-04-24 21:15:16.523 91.189.91.39      192.168.80.131    HTTP     113 HTTP/1.1 200 OK  (application/x-debian-package)
40541 2023-04-24 21:15:16.525 192.168.80.131    91.189.91.39      HTTP     207 GET /ubuntu/pool/main/c/curl/curl_7.35.0-1ubunt
40630 2023-04-24 21:15:16.725 91.189.91.39      192.168.80.131    HTTP     235 HTTP/1.1 200 OK  (application/x-debian-package)
40647 2023-04-24 21:15:23.437 192.168.80.131    239.255.255.250   SSDP     210 M-SEARCH * HTTP/1.1
40657 2023-04-24 21:15:24.439 192.168.80.131    239.255.255.250   SSDP     210 M-SEARCH * HTTP/1.1
40660 2023-04-24 21:15:25.440 192.168.80.131    239.255.255.250   SSDP     210 M-SEARCH * HTTP/1.1
40665 2023-04-24 21:15:26.442 192.168.80.131    239.255.255.250   SSDP     210 M-SEARCH * HTTP/1.1
40723 2023-04-24 21:15:34.431 192.168.80.131    93.184.216.34     HTTP     132 POST / HTTP/1.1
40727 2023-04-24 21:15:34.466 93.184.216.34     192.168.80.131    HTTP     363 HTTP/1.1 200 OK  (text/html)
41130 2023-04-24 21:17:23.436 192.168.80.131    239.255.255.250   SSDP     210 M-SEARCH * HTTP/1.1
41133 2023-04-24 21:17:24.437 192.168.80.131    239.255.255.250   SSDP     210 M-SEARCH * HTTP/1.1
41136 2023-04-24 21:17:25.438 192.168.80.131    239.255.255.250   SSDP     210 M-SEARCH * HTTP/1.1
41139 2023-04-24 21:17:26.440 192.168.80.131    239.255.255.250   SSDP     210 M-SEARCH * HTTP/1.1
41434 2023-04-24 21:19:23.436 192.168.80.131    239.255.255.250   SSDP     210 M-SEARCH * HTTP/1.1

▷ Frame 40723: 132 bytes on wire (1056 bits), 132 bytes captured (1056 bits) on interface 0
▷ Linux cooked capture
▷ Internet Protocol Version 4, Src: 192.168.80.131 (192.168.80.131), Dst: 93.184.216.34 (93.184.216.34)
▷ Transmission Control Protocol, Src Port: 40040 (40040), Dst Port: http (80), Seq: 1, Ack: 1, Len: 76
▷ Hypertext Transfer Protocol
```

I typed curl -X POST http://example.com into the terminal

Part 2: DNS

In this section, we will observe how the DNS protocol operates. We will do this by using the Mininet VM. Begin by opening Wireshark and listening on the ‘any’ interface. Open Chromium and navigate to www.example.com.

5. [10 pts] Were any steps taken by your computer before the web page was loaded? If so, using your captured packets in Wireshark, find the packets that allowed your computer to successfully load http://www.example.com. Take a screenshot of these packets, and explain why you think these are the correct packets. What’s the IP address of www.example.com?



No.	Time	Source	Destination	Protocol	Length	Info
10	2023-04-24 21:51:26.169	192.168.80.131	142.250.217.132	TLSv1.2	102	Application Data
11	2023-04-24 21:51:26.170	142.250.217.132	192.168.80.131	TCP	62	https > 35399 [ACK] Seq=494 Ack=226 Win=64240 Len=0
12	2023-04-24 21:51:26.720	127.0.0.1	127.0.0.1	TCP	76	42060 > 6633 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=6138501 TSecr=0 WS=128
13	2023-04-24 21:51:26.720	127.0.0.1	127.0.0.1	TCP	56	6633 > 42060 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
14	2023-04-24 21:51:27.721	127.0.0.1	127.0.0.1	TCP	76	42061 > 6633 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=6138751 TSecr=0 WS=128
15	2023-04-24 21:51:27.721	127.0.0.1	127.0.0.1	TCP	56	6633 > 42061 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
16	2023-04-24 21:51:28.122	192.168.80.131	192.168.80.2	DNS	77	Standard query 0x3005 A www.example.com
17	2023-04-24 21:51:28.131	192.168.80.2	192.168.80.131	DNS	93	Standard query response 0x3005 A 93.184.216.34
18	2023-04-24 21:51:28.132	192.168.80.131	93.184.216.34	TCP	76	42448 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=6138854 TSecr=0 WS=128
19	2023-04-24 21:51:28.144	93.184.216.34	192.168.80.131	TCP	62	http > 42448 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
20	2023-04-24 21:51:28.144	192.168.80.131	93.184.216.34	TCP	56	42448 > http [ACK] Seq=1 Ack=1 Win=29200 Len=0

The host is querying the DNS server to get the ip address that matches the name of example.com. These are correct packet because the computer needs to know its ip address

The IP address of www.example.com is 93.184.216.34

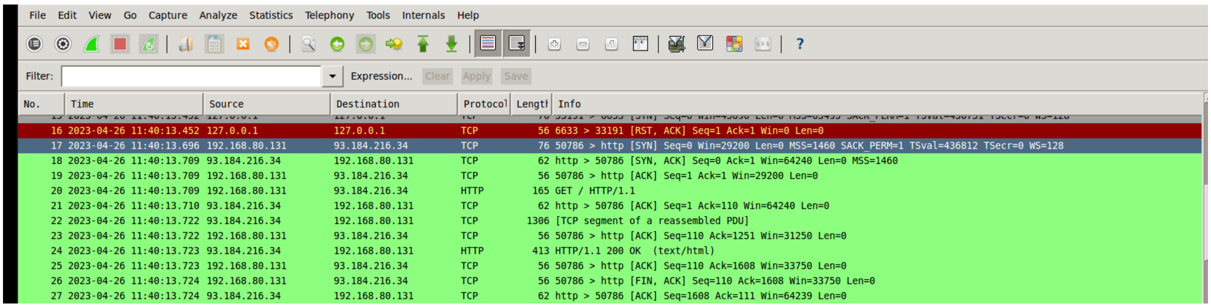
6. [10 pts] Open a terminal window. Execute the command to flush your DNS cache:

```
sudo /etc/init.d/networking restart
```

Using wget, download the same content of www.example.com with its IP address you discovered in question 5, without sending DNS requests.
What command did you use to accomplish that? Take a screenshot of related packets and explain why you think these are the correct packets.

```
wget http://93.184.216.34 –header “Host: example.com”
```

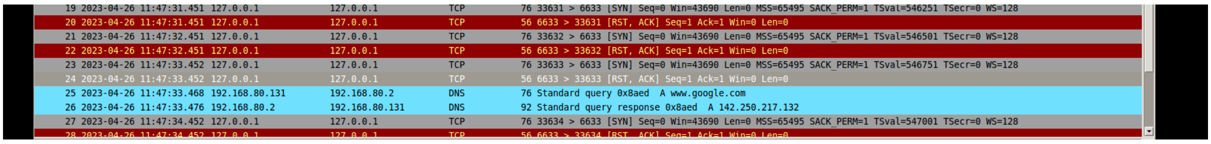
I think these are correct packets since they are sent and received from 93.184.216.36



```
No.  Time                     Source          Destination     Protocol Length Info
  16 2023-04-26 11:40:13.452 127.0.0.1       127.0.0.1       TCP        56 6633 > 33191 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
  17 2023-04-26 11:40:13.696 192.168.80.131  93.184.216.34   TCP        76 50786 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=436812 TSecr=0 WS=128
  18 2023-04-26 11:40:13.709 93.184.216.34   192.168.80.131  TCP        62 http > 50786 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
  19 2023-04-26 11:40:13.709 192.168.80.131  93.184.216.34   TCP        56 50786 > http [ACK] Seq=1 Ack=1 Win=29200 Len=0
  20 2023-04-26 11:40:13.709 192.168.80.131  93.184.216.34   HTTP      165 GET / HTTP/1.1
  21 2023-04-26 11:40:13.710 93.184.216.34   192.168.80.131  TCP        62 http > 50786 [ACK] Seq=1 Ack=110 Win=64240 Len=0
  22 2023-04-26 11:40:13.722 93.184.216.34   192.168.80.131  TCP      1306 [TCP segment of a reassembled PDU]
  23 2023-04-26 11:40:13.722 192.168.80.131  93.184.216.34   TCP        56 50786 > http [ACK] Seq=110 Ack=1251 Win=31250 Len=0
  24 2023-04-26 11:40:13.723 93.184.216.34   192.168.80.131  HTTP      413 HTTP/1.1 200 OK  (text/html)
  25 2023-04-26 11:40:13.723 192.168.80.131  93.184.216.34   TCP        56 50786 > http [ACK] Seq=110 Ack=1608 Win=33750 Len=0
  26 2023-04-26 11:40:13.724 192.168.80.131  93.184.216.34   TCP        56 50786 > http [FIN, ACK] Seq=110 Ack=1608 Win=33750 Len=0
  27 2023-04-26 11:40:13.724 93.184.216.34   192.168.80.131  TCP        62 http > 50786 [ACK] Seq=1608 Ack=111 Win=64239 Len=0
```

Open a terminal window. Using nslookup, find the A records for www.google.com. (If you can’t access Google, for example, you are in China, you could replace the domain name with www.baidu.com)

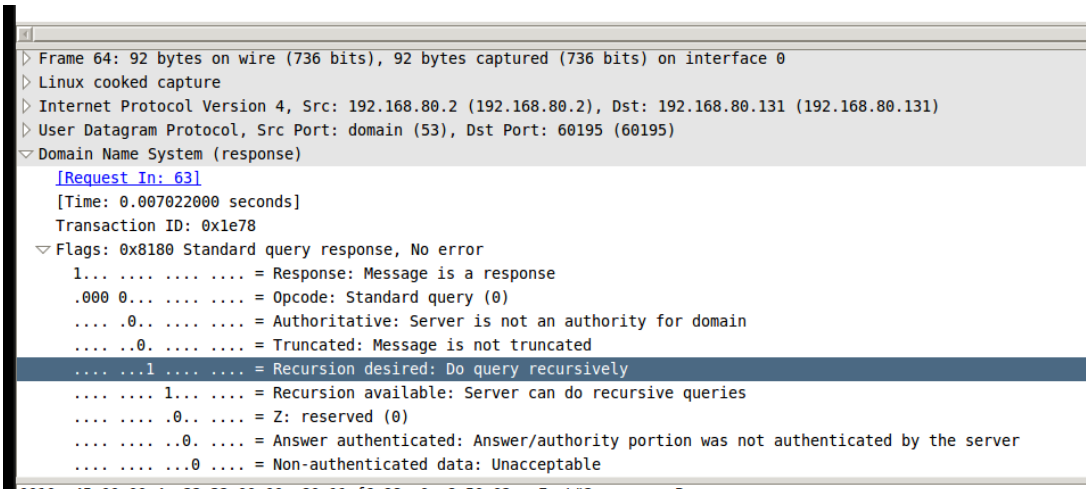
7. [10 pts] Take a screenshot of the packets corresponding to your request, and the response from the server. If the request was resolved, what is the IP address you were given for www.google.com?



```
  19 2023-04-26 11:47:31.451 127.0.0.1       127.0.0.1       TCP      76 33631 > 6633 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=546251 TSecr=0 WS=128
  20 2023-04-26 11:47:31.451 127.0.0.1       127.0.0.1       TCP      56 6633 > 33631 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
  21 2023-04-26 11:47:32.451 127.0.0.1       127.0.0.1       TCP      76 33632 > 6633 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=546501 TSecr=0 WS=128
  22 2023-04-26 11:47:32.451 127.0.0.1       127.0.0.1       TCP      56 6633 > 33632 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
  23 2023-04-26 11:47:33.452 127.0.0.1       127.0.0.1       TCP      76 33633 > 6633 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=546751 TSecr=0 WS=128
  24 2023-04-26 11:47:33.452 127.0.0.1       127.0.0.1       TCP      56 6633 > 33633 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
  25 2023-04-26 11:47:33.468 192.168.80.131  192.168.80.2    DNS      76 Standard query 0x8aed  A www.google.com
  26 2023-04-26 11:47:33.476 192.168.80.2    192.168.80.131  DNS      92 Standard query response 0x8aed  A 142.250.217.132
  27 2023-04-26 11:47:34.452 127.0.0.1       127.0.0.1       TCP      76 33634 > 6633 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=547001 TSecr=0 WS=128
  28 2023-04-26 11:47:34.452 127.0.0.1       127.0.0.1       TCP      56 6633 > 33634 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
```

The IP address I was given for www.google.com is 142.250.217.132

8. [10 pts] Did your computer want to complete the request recursively? How do you know? Take a screenshot proving your answer.

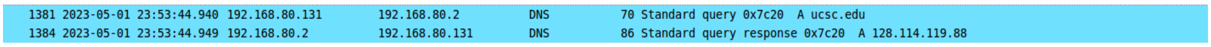


```
▷ Frame 64: 92 bytes on wire (736 bits), 92 bytes captured (736 bits) on interface 0
▷ Linux cooked capture
▷ Internet Protocol Version 4, Src: 192.168.80.2 (192.168.80.2), Dst: 192.168.80.131 (192.168.80.131)
▷ User Datagram Protocol, Src Port: domain (53), Dst Port: 60195 (60195)
▽ Domain Name System (response)
    [Request In: 63]
    [Time: 0.007022000 seconds]
    Transaction ID: 0x1e78
  ▽ Flags: 0x8180 Standard query response, No error
      1... .... .... .... = Response: Message is a response
      .000 0... .... .... = Opcode: Standard query (0)
      .... .0.. .... .... = Authoritative: Server is not an authority for domain
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... 1... .... = Recursion available: Server can do recursive queries
      .... .... .0.. .... = Z: reserved (0)
      .... .... ..0. .... = Answer authenticated: Answer/authority portion was not authenticated by the server
      .... .... ...0 .... = Non-authenticated data: Unacceptable
```

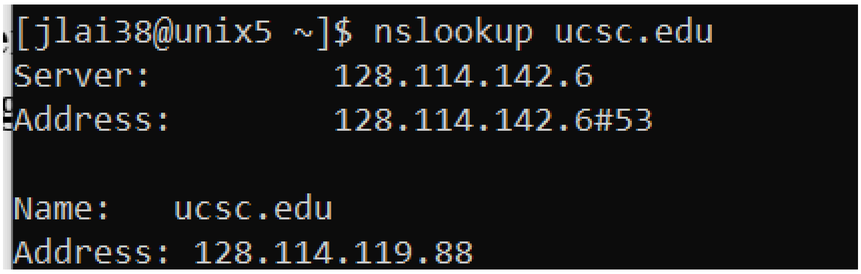
The computer wants to complete the request recursively, I knew that based on the true statement on recursion desired.

Using nslookup, find the A records for ucsc.edu.

9. [10 pts] Take a screenshot of the packets corresponding to your request, and the response from the server. If the request was resolved, what is the IP address you were given for ucsc.edu?



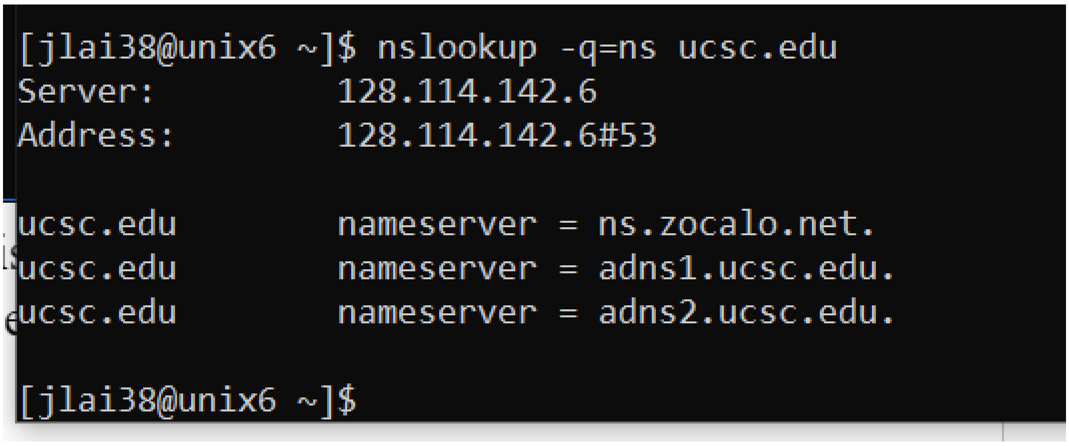
```
1381 2023-05-01 23:53:44.940 192.168.80.131       192.168.80.2        DNS        70 Standard query 0x7c20  A ucsc.edu
1384 2023-05-01 23:53:44.949 192.168.80.2         192.168.80.131      DNS        86 Standard query response 0x7c20  A 128.114.119.88
```



```
[jlai38@unix5 ~]$ nslookup ucsc.edu
Server:         128.114.142.6
Address:        128.114.142.6#53

Name:   ucsc.edu
Address: 128.114.119.88
```

The IP address I was given for ucsc.edu is 128.114.119.88

10. [10 pts] What is the authoritative name server for the ucsc.edu domain? How do you know? Take a screenshot proving your answer.

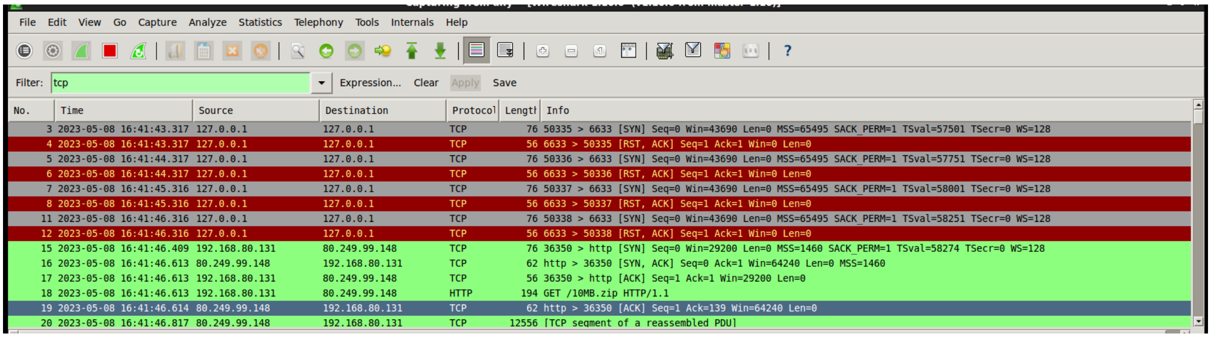


```
[jlai38@unix6 ~]$ nslookup -q=ns ucsc.edu
Server:         128.114.142.6
Address:        128.114.142.6#53

ucsc.edu        nameserver = ns.zocalo.net.
ucsc.edu        nameserver = adns1.ucsc.edu.
ucsc.edu        nameserver = adns2.ucsc.edu.

[jlai38@unix6 ~]$
```

The authoritative name server for the ucsc.edu are Adns1.ucsc.edu, adns1.ucsc.edu, adns2.ucsc.edu
I know that by typing nslookup -q=ns ucsc.edu, the results are shown above.

In this section, we will observe how the TCP protocol operates. We will do this by using the Mininet VM. Begin by opening Wireshark and listening on the ‘any’ interface.
Open a terminal window. Using wget, download the file http://ipv4.download.thinkbroadband.com/10MB.zip

11. [10 pts] Find the packets corresponding with the SYN, SYN-ACK, and ACK that initiated the TCP connection for this file transfer. Take a screenshot of these packets. What was the initial window size that your computer advertised to the server? What was the initial window size that the server advertised to you?

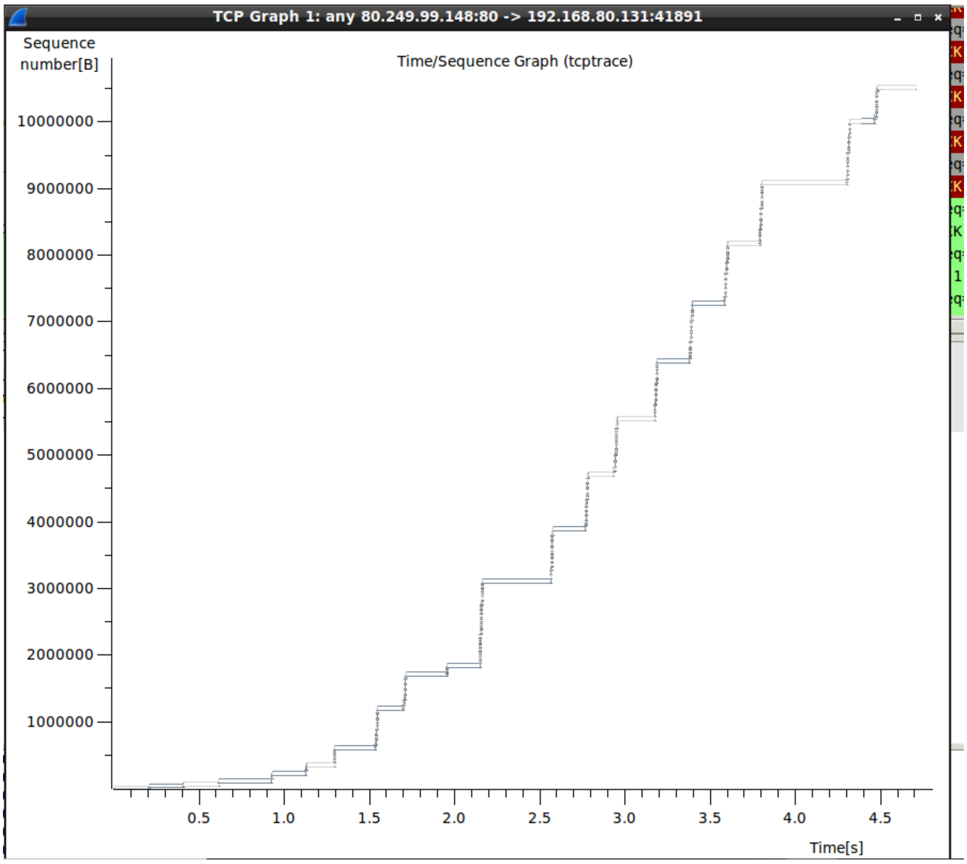


Filter: tcp

No.	Time	Source	Destination	Protocol	Length	Info
3	2023-05-08 16:41:43.317	127.0.0.1	127.0.0.1	TCP	76	50335 > 6633 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=57501 TSecr=0 WS=128
4	2023-05-08 16:41:43.317	127.0.0.1	127.0.0.1	TCP	56	6633 > 50335 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
5	2023-05-08 16:41:44.317	127.0.0.1	127.0.0.1	TCP	76	50336 > 6633 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=57751 TSecr=0 WS=128
6	2023-05-08 16:41:44.317	127.0.0.1	127.0.0.1	TCP	56	6633 > 50336 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
7	2023-05-08 16:41:45.316	127.0.0.1	127.0.0.1	TCP	76	50337 > 6633 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=58001 TSecr=0 WS=128
8	2023-05-08 16:41:45.316	127.0.0.1	127.0.0.1	TCP	56	6633 > 50337 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
11	2023-05-08 16:41:46.316	127.0.0.1	127.0.0.1	TCP	76	50338 > 6633 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=58251 TSecr=0 WS=128
12	2023-05-08 16:41:46.316	127.0.0.1	127.0.0.1	TCP	56	6633 > 50338 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
15	2023-05-08 16:41:46.409	192.168.80.131	80.249.99.148	TCP	76	36350 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=58274 TSecr=0 WS=128
16	2023-05-08 16:41:46.613	80.249.99.148	192.168.80.131	TCP	62	http > 36350 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
17	2023-05-08 16:41:46.613	192.168.80.131	80.249.99.148	TCP	56	36350 > http [ACK] Seq=1 Ack=1 Win=29200 Len=0
18	2023-05-08 16:41:46.613	192.168.80.131	80.249.99.148	HTTP	194	GET /10MB.zip HTTP/1.1
19	2023-05-08 16:41:46.614	80.249.99.148	192.168.80.131	TCP	62	http > 36350 [ACK] Seq=1 Ack=139 Win=64240 Len=0
20	2023-05-08 16:41:46.817	80.249.99.148	192.168.80.131	TCP	12556	[TCP segment of a reassembled PDU]

The initial window size that my computer advertised to the server is 29200. The server advertised back to me as 64240.

12. [10 pts] Find a packet from the download with a source of the server and a destination of your computer. Create a tcptrace graph with this packet selected. Take a screenshot of the graph and explain what it is showing. Look into the Wireshark documentation if you need assistance making this graph.



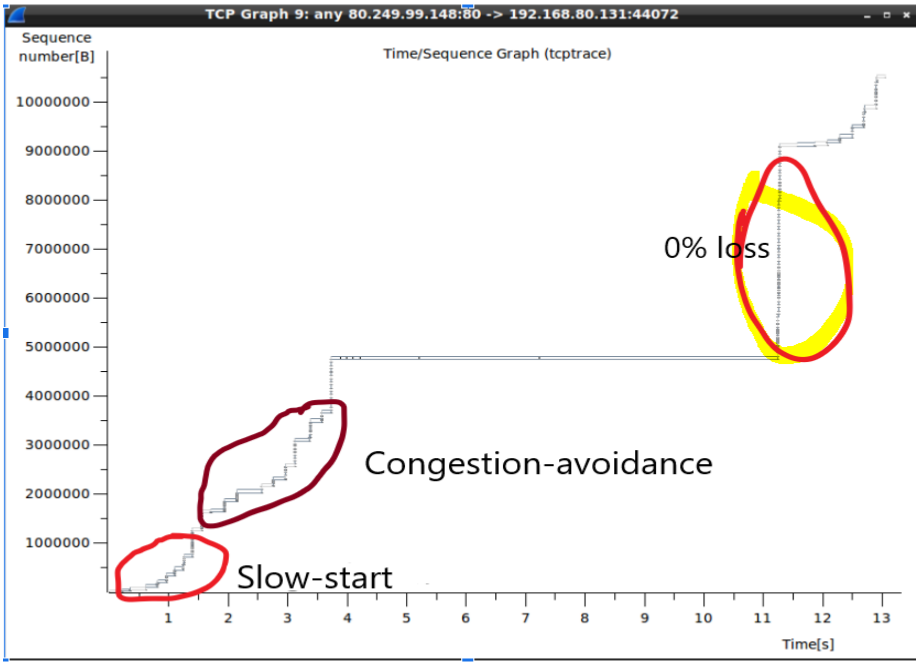
This graph shows the transmission rate of packets. The dark line represents data, and the sequence number represents the number of bytes being sent. As it goes up into the right that represents the sequence numbers over time.

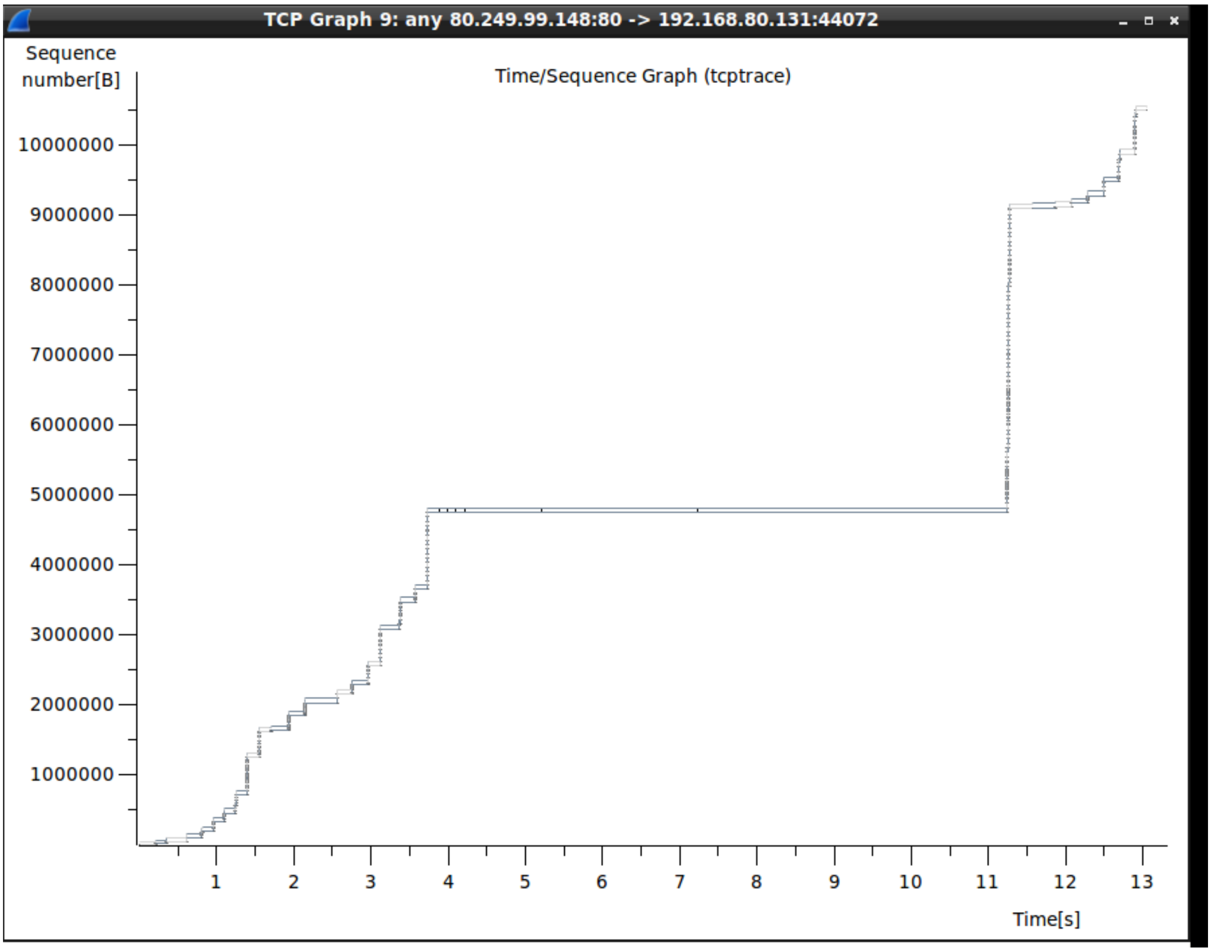
In the next section, we will be simulating loss, the command tc qdisc will be needed. When you first use the command you should use add dev for the device you plan on changing. It only needs to be set on the sender's side. After adding the device use change dev.

Example:
sudo tc qdisc add dev eth0 root netem loss 0%
sudo tc qdisc change dev eth0 root netem loss 100%

Read through the following paragraph before starting the next step. Open 2 terminals and have the commands typed and ready before you begin. In one terminal, download the 10MB.zip file again. While the download is in progress, change loss to 100%. After a few seconds, change loss to 0%.

13. [10 pts] Find a packet from the download with a source of the server and a destination of your computer. Create a tcptrace graph with this packet selected. Take a screenshot of the graph and explain what it is showing. Using an image editting program, circle the areas where the 0% loss is shown, as well as where TCP is in slow-start and congestion-avoidance



TCP Graph 9: any 80.249.99.148:80 -> 192.168.80.131:44072
Time/Sequence Graph (tcptrace)
Sequence number[B]
10000000
9000000
8000000
7000000
6000000
5000000
4000000
3000000
2000000
1000000
1
2
3
4
5
6
7
8
9
10
11
12
13
Time[s]