Luis Valdivia - 1462385
levaldiv@ucsc.edu
PreLab & Lab 2 - 10/30/19

Pre-Lab Questions:

HTTP Questions

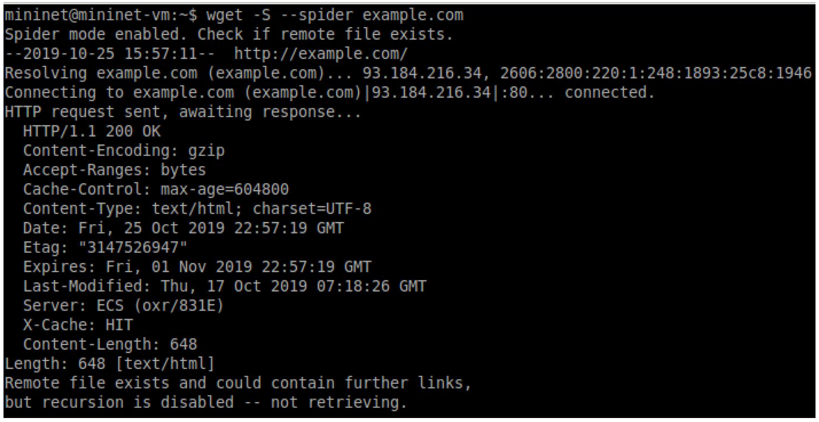
1. **Choose 5 HTTP status codes and describe each one**

401 Unauthorized- Request requires user authentication. The response must include a WWW-Authenticate header field containing a challenge applicable to the requested resource.
403 Forbidden- Server understood the request, but is refusing to fulfill it. In other other words, authorization will not help and the request should not be repeated.
404 Not Found- Server did not find anything matching the requested-URI (Uniform Resource Identifiers). Commonly used when the server does not wish to reveal exactly why the request has been refused
500 Internal Server Error- The server encountered an unexpected condition which prevented it from fulfilling the request
502 Bad Gateway- The server, while acting as a gateway or proxy, received an invaild response from the upstream server it accessed in attempting to fulfill the request

2. **List the 8 HTTP 1.1 methods and explain what they do**

Options - This method describes the communication options for the target resource
Get - It is used to retrieve information from the given server using a given URI.
Head - This method is the same as GET but it transfers the status line and header section only
Post - It is a request that is used to send data to the server (i.e. customer information, file upload using HTML forms)
Put - It replaces all current representations of the target resource with the upload content
Delete - It removes all current representations of the target resource given by a URI
Trace - This method performs a message loop-back test along the path to the target resource
Connect - This establishes a tunnel to the server identified by a given URI

3. **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?**

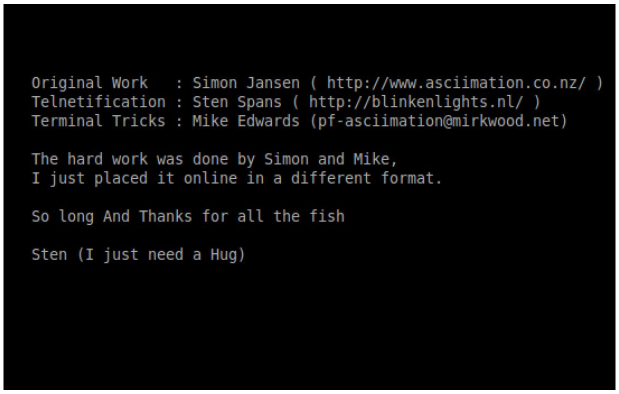


```
mininet@mininet-vm:~$ wget -S --spider example.com
Spider mode enabled. Check if remote file exists.
--2019-10-25 15:57:11--  http://example.com/
Resolving example.com (example.com)... 93.184.216.34, 2606:2800:220:1:248:1893:25c8:1946
Connecting to example.com (example.com)|93.184.216.34|:80... connected.
HTTP request sent, awaiting response...
  HTTP/1.1 200 OK
  Content-Encoding: gzip
  Accept-Ranges: bytes
  Cache-Control: max-age=604800
  Content-Type: text/html; charset=UTF-8
  Date: Fri, 25 Oct 2019 22:57:19 GMT
  Etag: "3147526947"
  Expires: Fri, 01 Nov 2019 22:57:19 GMT
  Last-Modified: Thu, 17 Oct 2019 07:18:26 GMT
  Server: ECS (oxr/831E)
  X-Cache: HIT
  Content-Length: 648
Length: 648 [text/html]
Remote file exists and could contain further links,
but recursion is disabled -- not retrieving.
```

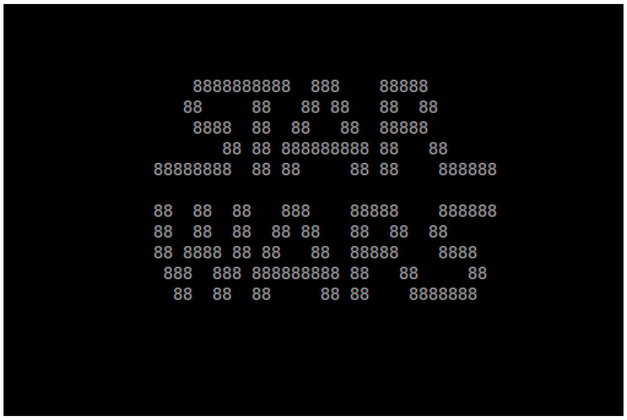
Luis Valdivia - 1462385
levaldiv@ucsc.edu
PreLab & Lab 2 - 10/30/19

Using the command ***wget -S --spider example.com***, I was able to find out when the when the webpage was last modified, which was Oct. 17 2019. And the HTTP return status code was **200 ok**, which means that the request has been succeeded. The --spider allowed me to be able to look at the contents without downloading anything.

4. **Look up the *telnet* command. Use *telnet* to connect to *towel.blinkenlights.nl*. What does this telnet server do?**



```
Original Work   : Simon Jansen ( http://www.asciimation.co.nz/ )
Telnetification : Sten Spans ( http://blinkenlights.nl/ )
Terminal Tricks : Mike Edwards (pf-asciimation@mirkwood.net)

The hard work was done by Simon and Mike,
I just placed it online in a different format.

So long And Thanks for all the fish

Sten (I just need a Hug)
```



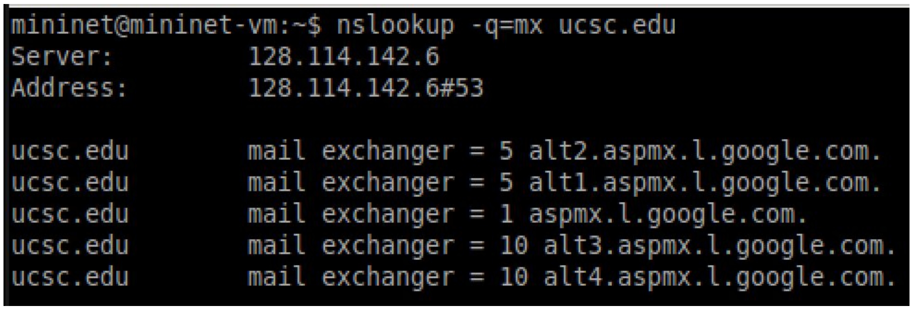
```
    8888888888  888    88888
   88     88    88 88    88  88
    8888  88  88    88  88888
       88 88 888888888 88    88
88888888  88 88      88 88     888888

88  88  88    888      88888    888888
88  88  88   88 88    88  88  88
88 8888 88 88    88  88888    8888
 888  888 888888888 88   88     88
  88  88  88      88 88    8888888
```

The *telnet* command is a network protocol and the application that uses the protocol. It can be used to connect to remote computers and issue commands. Using the command *telnet towel.blinkenlights.nl,* this server shows the complete Star Wars (Episode IV) movie in ASCII characters.

DNS Questions

5. **In your own words describe what a DNS Resource Record (RR) is. Now using the command line tool *nslookup* find the MX-resource of ucsc.edu. What does this resource mean?**

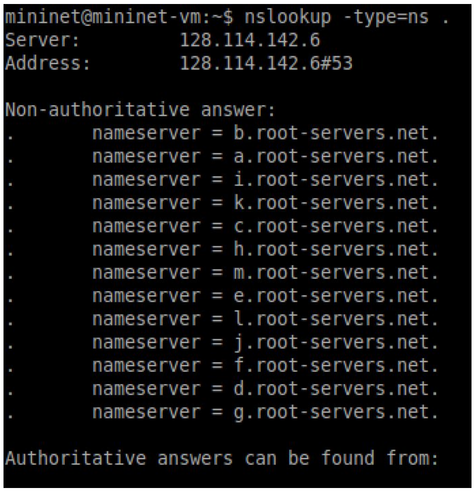
A Resource Record (RR) is the unit of information entry in the DNS (Domain Name System) zone files. RRs are the "basic building" blocks of host-name and IP information, and are used to resolve all DNS queries.



```
mininet@mininet-vm:~$ nslookup -q=mx ucsc.edu
Server:         128.114.142.6
Address:        128.114.142.6#53

ucsc.edu        mail exchanger = 5 alt2.aspmx.l.google.com.
ucsc.edu        mail exchanger = 5 alt1.aspmx.l.google.com.
ucsc.edu        mail exchanger = 1 aspmx.l.google.com.
ucsc.edu        mail exchanger = 10 alt3.aspmx.l.google.com.
ucsc.edu        mail exchanger = 10 alt4.aspmx.l.google.com.
```

Using the command ***nslookup -q=mx ucsc.edu***, I was able to find the MX-resources of ucsc. It's describing the characteristics of the ucsc domain.

6. **What does the command *nslookup -type=ns .* do? Explain its output.**



```
mininet@mininet-vm:~$ nslookup -type=ns .
Server:         128.114.142.6
Address:        128.114.142.6#53

Non-authoritative answer:
.       nameserver = b.root-servers.net.
.       nameserver = a.root-servers.net.
.       nameserver = i.root-servers.net.
.       nameserver = k.root-servers.net.
.       nameserver = c.root-servers.net.
.       nameserver = h.root-servers.net.
.       nameserver = m.root-servers.net.
.       nameserver = e.root-servers.net.
.       nameserver = l.root-servers.net.
.       nameserver = j.root-servers.net.
.       nameserver = f.root-servers.net.
.       nameserver = d.root-servers.net.
.       nameserver = g.root-servers.net.

Authoritative answers can be found from:
```

Using the command ***-type=ns*** to query NS (Name Server) to identify the nameservers of a given domain. Adding a "." to the end instead of a domain gives us 13 root servers with details.

TCP Questions

7. **How can multiple application services running on a single machine with a single IP address be uniquely identified?**

Multiple applications running on a single machine with a single IP can be uniquely identified using ***port addresses***. Each application has a different port address which allows us to distinguish them if they are on one computer.

8. **What is the purpose of the window mechanism in TCP?**

Windowing is done to ensure how many packets are sent at a time that depends on the size of the window and how each packet is acknowledged. It can also be used to control the flow of packets (data) between two networks as this method is used to get acknowledgement for each packet received at the receiver side.

9. **What is an MTU? What happens when a packet is larger than the MTU?**

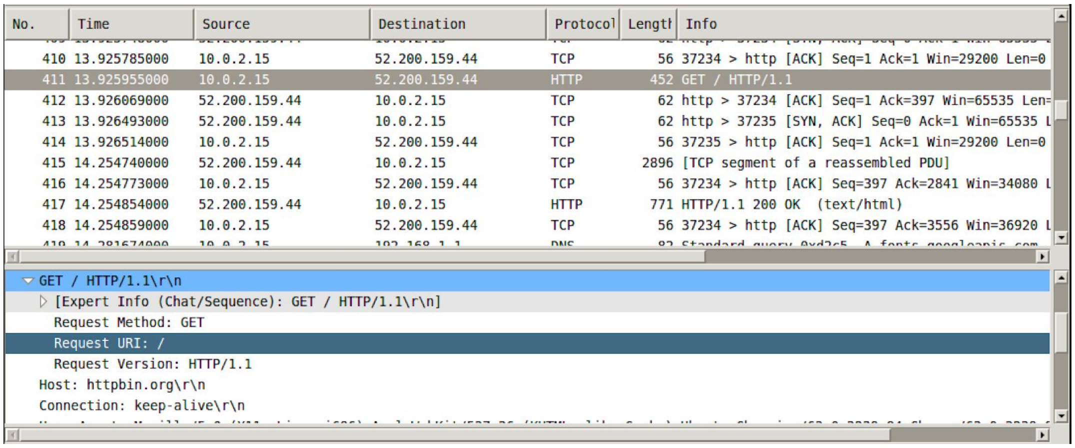
MTU or Maximum Transmission Unit, is the largest size packet or frame, that can be sent in a packet- or frame-based network such as the internet. In other words, TCP uses MTU to determine the maximum size of each packet in any transmission. If a packet is larger than the MTU then it might get retransmissioned if the packet encounters a router that cant handle that large packet.

Luis Valdivia - 1462385
levaldiv@ucsc.edu
PreLab & Lab 2 - 10/30/19

Lab Questions:

Part 1: HTTP

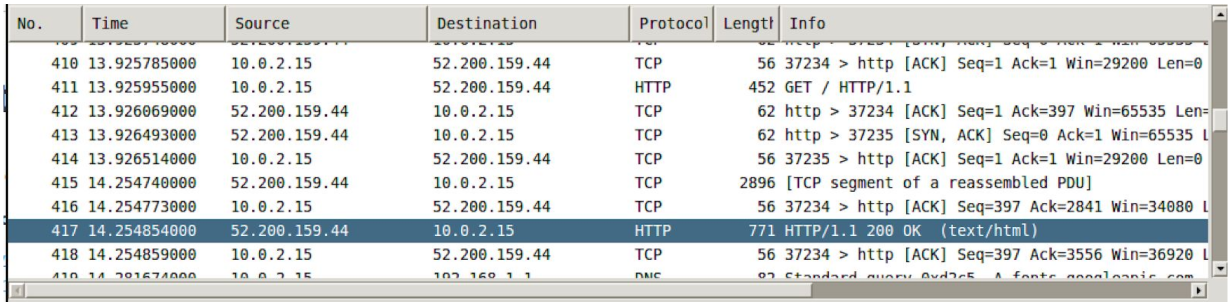
1. **Find the HTTP packet that corresponds to the initial request that your computer made. What HTTP method did your computer use to make this request? What URI did your computer request from the server?**



```
No.  Time          Source         Destination    Protocol Length Info
 410 13.925785000  10.0.2.15      52.200.159.44  TCP        56 37234 > http [ACK] Seq=1 Ack=1 Win=29200 Len=0
 411 13.925955000  10.0.2.15      52.200.159.44  HTTP      452 GET / HTTP/1.1
 412 13.926069000  52.200.159.44  10.0.2.15      TCP        62 http > 37234 [ACK] Seq=1 Ack=397 Win=65535 Len=
 413 13.926493000  52.200.159.44  10.0.2.15      TCP        62 http > 37235 [SYN, ACK] Seq=0 Ack=1 Win=65535 L
 414 13.926514000  10.0.2.15      52.200.159.44  TCP        56 37235 > http [ACK] Seq=1 Ack=1 Win=29200 Len=0
 415 14.254740000  52.200.159.44  10.0.2.15      TCP      2896 [TCP segment of a reassembled PDU]
 416 14.254773000  10.0.2.15      52.200.159.44  TCP        56 37234 > http [ACK] Seq=397 Ack=2841 Win=34080 L
 417 14.254854000  52.200.159.44  10.0.2.15      HTTP      771 HTTP/1.1 200 OK  (text/html)
 418 14.254859000  10.0.2.15      52.200.159.44  TCP        56 37234 > http [ACK] Seq=397 Ack=3556 Win=36920 L

▽ GET / HTTP/1.1\r\n
  ▷ [Expert Info (Chat/Sequence): GET / HTTP/1.1\r\n]
    Request Method: GET
    Request URI: /
    Request Version: HTTP/1.1
  Host: httpbin.org\r\n
  Connection: keep-alive\r\n
```

The packet that corresponds to the initial HTTP is packet number 411. The HTTP method my computer used to make the request was the GET method, which is the method to retrieve information from the given server using a given URI. The URI requested by the computer is "/"

2. **Find the HTTP packet that corresponds to the initial response the server made to your request. What HTTP status code did the server return? What is the content type of the response the server is sending back?**

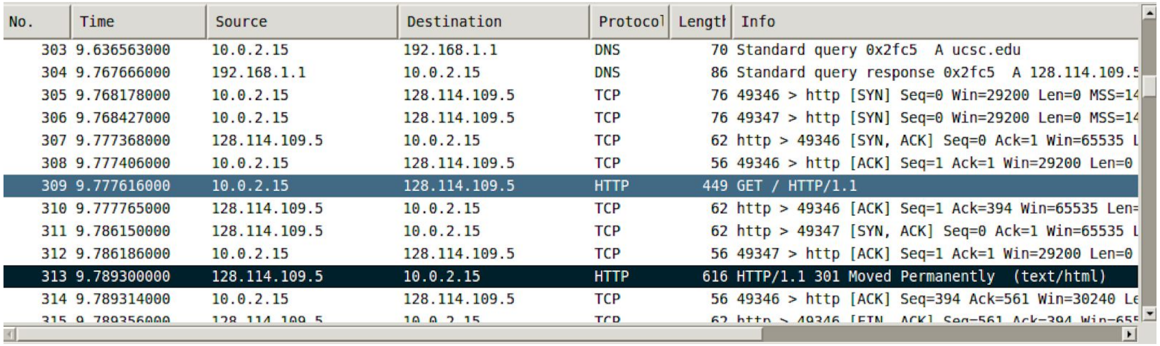


```
No.  Time          Source         Destination    Protocol Length Info
 410 13.925785000  10.0.2.15      52.200.159.44  TCP        56 37234 > http [ACK] Seq=1 Ack=1 Win=29200 Len=0
 411 13.925955000  10.0.2.15      52.200.159.44  HTTP      452 GET / HTTP/1.1
 412 13.926069000  52.200.159.44  10.0.2.15      TCP        62 http > 37234 [ACK] Seq=1 Ack=397 Win=65535 Len=
 413 13.926493000  52.200.159.44  10.0.2.15      TCP        62 http > 37235 [SYN, ACK] Seq=0 Ack=1 Win=65535 L
 414 13.926514000  10.0.2.15      52.200.159.44  TCP        56 37235 > http [ACK] Seq=1 Ack=1 Win=29200 Len=0
 415 14.254740000  52.200.159.44  10.0.2.15      TCP      2896 [TCP segment of a reassembled PDU]
 416 14.254773000  10.0.2.15      52.200.159.44  TCP        56 37234 > http [ACK] Seq=397 Ack=2841 Win=34080 L
 417 14.254854000  52.200.159.44  10.0.2.15      HTTP      771 HTTP/1.1 200 OK  (text/html)
 418 14.254859000  10.0.2.15      52.200.159.44  TCP        56 37234 > http [ACK] Seq=397 Ack=3556 Win=36920 L
```

The packet corresponding to the initial HTTP response the server made to my request was packet number 417. The status code returned was 200 ok. The content type the response server is sending back is text/html.

3. Find the HTTP packets that correspond to the initial request and response that your computer made. What's different? Explain.

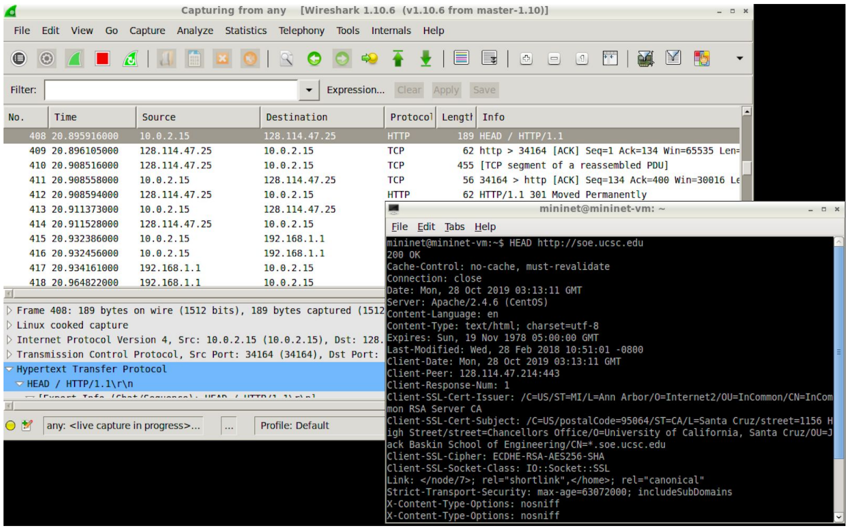
Using Chromium and navigating to http://ucsc.edu



```
No.  Time         Source          Destination     Protocol Length Info
 303 9.636563000  10.0.2.15       192.168.1.1     DNS        70 Standard query 0x2fc5  A ucsc.edu
 304 9.767666000  192.168.1.1     10.0.2.15       DNS        86 Standard query response 0x2fc5  A 128.114.109.5
 305 9.768178000  10.0.2.15       128.114.109.5   TCP        76 49346 > http [SYN] Seq=0 Win=29200 Len=0 MSS=14
 306 9.768427000  10.0.2.15       128.114.109.5   TCP        76 49347 > http [SYN] Seq=0 Win=29200 Len=0 MSS=14
 307 9.777368000  128.114.109.5   10.0.2.15       TCP        62 http > 49346 [SYN, ACK] Seq=0 Ack=1 Win=65535 L
 308 9.777406000  10.0.2.15       128.114.109.5   TCP        56 49346 > http [ACK] Seq=1 Ack=1 Win=29200 Len=0
 309 9.777616000  10.0.2.15       128.114.109.5   HTTP      449 GET / HTTP/1.1
 310 9.777765000  128.114.109.5   10.0.2.15       TCP        62 http > 49346 [ACK] Seq=1 Ack=394 Win=65535 Len=
 311 9.786150000  128.114.109.5   10.0.2.15       TCP        62 http > 49347 [SYN, ACK] Seq=0 Ack=1 Win=65535 L
 312 9.786186000  10.0.2.15       128.114.109.5   TCP        56 49347 > http [ACK] Seq=1 Ack=1 Win=29200 Len=0
 313 9.789300000  128.114.109.5   10.0.2.15       HTTP      616 HTTP/1.1 301 Moved Permanently  (text/html)
 314 9.789314000  10.0.2.15       128.114.109.5   TCP        56 49346 > http [ACK] Seq=394 Ack=561 Win=30240 Le
 315 9.789356000  128.114.109.5   10.0.2.15       TCP        62 http > 49346 [FIN, ACK] Seq=561 Ack=394 Win=655
```

Packet 309 corresponds to the initial HTTP request made from my computer, which was the GET method, and the same URI was requested "/" The difference is that the server returned a 301 status code and is used for permanent URL redirection, meaning that current links using the URL that the response is received for should be updated.

4. Using Chromium find a way to make a HTTP packet with a method other than GET. Take a sc of your packet and explain what you did to create it.


Capturing from any   [Wireshark 1.10.6 (v1.10.6 from master-1.10)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save

```
No.  Time          Source          Destination     Protocol Length Info
 408 20.895916000  10.0.2.15       128.114.47.25   HTTP      189 HEAD / HTTP/1.1
 409 20.896105000  128.114.47.25   10.0.2.15       TCP        62 http > 34164 [ACK] Seq=1 Ack=134 Win=65535 Len=
 410 20.908516000  128.114.47.25   10.0.2.15       TCP       455 [TCP segment of a reassembled PDU]
 411 20.908558000  10.0.2.15       128.114.47.25   TCP        56 34164 > http [ACK] Seq=134 Ack=400 Win=30016 Le
 412 20.908594000  128.114.47.25   10.0.2.15       HTTP       62 HTTP/1.1 301 Moved Permanently
 413 20.911373000  10.0.2.15       128.114.47.25
 414 20.911528000  128.114.47.25   10.0.2.15
 415 20.932386000  10.0.2.15       192.168.1.1
 416 20.932456000  10.0.2.15       192.168.1.1
 417 20.934161000  192.168.1.1     10.0.2.15
 418 20.964822000  192.168.1.1     10.0.2.15
```

```
▷ Frame 408: 189 bytes on wire (1512 bits), 189 bytes captured (1512
▷ Linux cooked capture
▷ Internet Protocol Version 4, Src: 10.0.2.15 (10.0.2.15), Dst: 128.
▷ Transmission Control Protocol, Src Port: 34164 (34164), Dst Port:
▽ Hypertext Transfer Protocol
  ▽ HEAD / HTTP/1.1\r\n
```

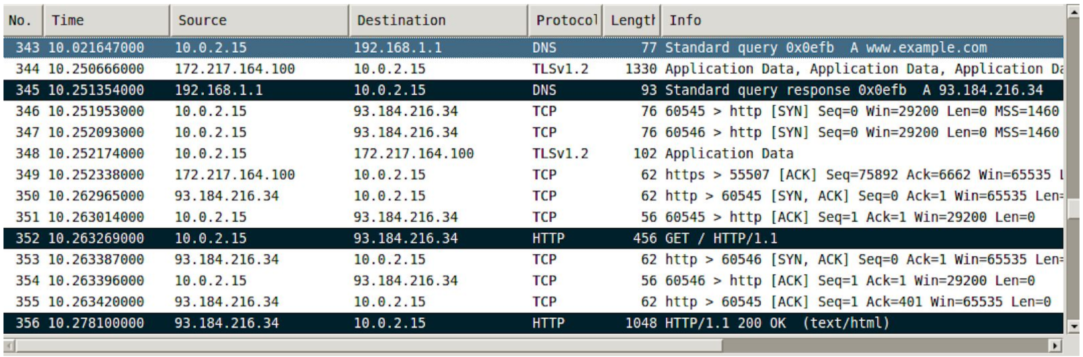
```
mininet@mininet-vm:~$ HEAD http://soe.ucsc.edu
200 OK
Cache-Control: no-cache, must-revalidate
Connection: close
Date: Mon, 28 Oct 2019 03:13:11 GMT
Server: Apache/2.4.6 (CentOS)
Content-Language: en
Content-Type: text/html; charset=utf-8
Expires: Sun, 19 Nov 1978 05:00:00 GMT
Last-Modified: Wed, 28 Feb 2018 10:51:01 -0800
Client-Date: Mon, 28 Oct 2019 03:13:11 GMT
Client-Peer: 128.114.47.214:443
Client-Response-Num: 1
Client-SSL-Cert-Issuer: /C=US/ST=MI/L=Ann Arbor/O=Internet2/OU=InCommon/CN=InCom
mon RSA Server CA
Client-SSL-Cert-Subject: /C=US/postalCode=95064/ST=CA/L=Santa Cruz/street=1156 H
igh Street/street=Chancellors Office/O=University of California, Santa Cruz/OU=J
ack Baskin School of Engineering/CN=*.soe.ucsc.edu
Client-SSL-Cipher: ECDHE-RSA-AES256-SHA
Client-SSL-Socket-Class: IO::Socket::SSL
Link: </node/7>; rel="shortlink",</home>; rel="canonical"
Strict-Transport-Security: max-age=63072000; includeSubDomains
X-Content-Type-Options: nosniff
X-Content-Type-Options: nosniff
```

In order for me to get the HTTP method HEAD (this method is the same as GET but it transfers the status line and header section only) I used terminal and typed "HEAD http://soe.ucsc.edu" which then created a packet in the screenshot above. The server has the status code 301 moved permanently.

Luis Valdivia - 1462385
levaldiv@ucsc.edu
PreLab & Lab 2 - 10/30/19

Part 2: DNS

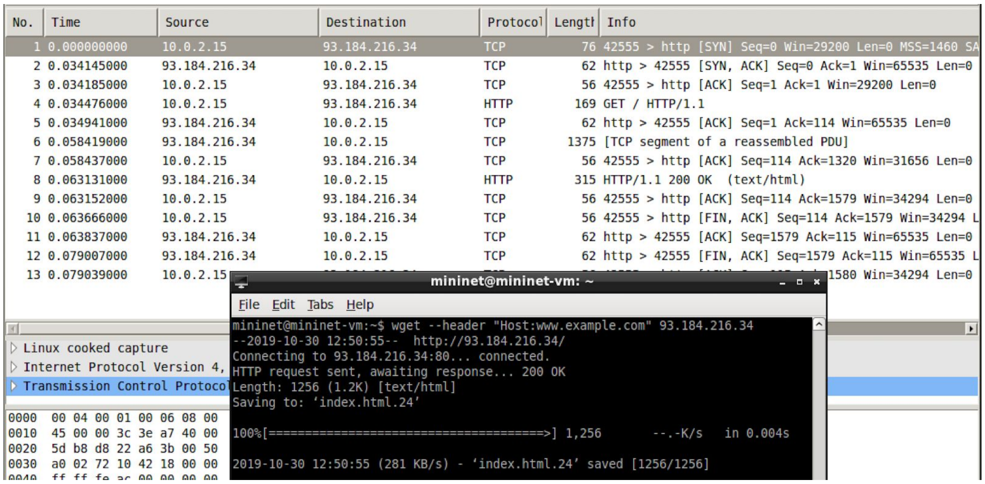
5. **Were any steps taken by your computer before the webpage was loaded? If so, using your captured packets in wireshark, find the packets that allows your computer to successfully load http://example.com**



No.	Time	Source	Destination	Protocol	Length	Info
343	10.021647000	10.0.2.15	192.168.1.1	DNS	77	Standard query 0x0efb A www.example.com
344	10.250666000	172.217.164.100	10.0.2.15	TLSv1.2	1330	Application Data, Application Data, Application Da
345	10.251354000	192.168.1.1	10.0.2.15	DNS	93	Standard query response 0x0efb A 93.184.216.34
346	10.251953000	10.0.2.15	93.184.216.34	TCP	76	60545 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460
347	10.252093000	10.0.2.15	93.184.216.34	TCP	76	60546 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460
348	10.252174000	10.0.2.15	172.217.164.100	TLSv1.2	102	Application Data
349	10.252338000	172.217.164.100	10.0.2.15	TCP	62	https > 55507 [ACK] Seq=75892 Ack=6662 Win=65535 L
350	10.262965000	93.184.216.34	10.0.2.15	TCP	62	http > 60545 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=
351	10.263014000	10.0.2.15	93.184.216.34	TCP	56	60545 > http [ACK] Seq=1 Ack=1 Win=29200 Len=0
352	10.263269000	10.0.2.15	93.184.216.34	HTTP	456	GET / HTTP/1.1
353	10.263387000	93.184.216.34	10.0.2.15	TCP	62	http > 60546 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=
354	10.263396000	10.0.2.15	93.184.216.34	TCP	56	60546 > http [ACK] Seq=1 Ack=1 Win=29200 Len=0
355	10.263420000	93.184.216.34	10.0.2.15	TCP	62	http > 60545 [ACK] Seq=1 Ack=401 Win=65535 Len=0
356	10.278100000	93.184.216.34	10.0.2.15	HTTP	1048	HTTP/1.1 200 OK (text/html)

The steps taken by my computer can be shown in DNS packets 343 and 345. The DNS performs a standard query (packet 343) to retrieve the IP address of www.example.com. It then sends a standard query response (packet 345) and that's when the TCP connections start. The TCP connection sends the HTTP 1.1 file (GET method) and we get a status code 200 ok. Looking at the source and destination IP addresses, we can verify that these are the correct packets. The IP address for www.example.com is: 93.184.216.34

6. **Use the command "sudo /etc/init.d/networking restart. Now, using wget, download the same content of www.example.com with its IP address you discovered in question 5, without sending DNS request.**



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	93.184.216.34	TCP	76	42555 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SA
2	0.034145000	93.184.216.34	10.0.2.15	TCP	62	http > 42555 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0
3	0.034185000	10.0.2.15	93.184.216.34	TCP	56	42555 > http [ACK] Seq=1 Ack=1 Win=29200 Len=0
4	0.034476000	10.0.2.15	93.184.216.34	HTTP	169	GET / HTTP/1.1
5	0.034941000	93.184.216.34	10.0.2.15	TCP	62	http > 42555 [ACK] Seq=1 Ack=114 Win=65535 Len=0
6	0.058419000	93.184.216.34	10.0.2.15	TCP	1375	[TCP segment of a reassembled PDU]
7	0.058437000	10.0.2.15	93.184.216.34	TCP	56	42555 > http [ACK] Seq=114 Ack=1320 Win=31656 Len=0
8	0.063131000	93.184.216.34	10.0.2.15	HTTP	315	HTTP/1.1 200 OK (text/html)
9	0.063152000	10.0.2.15	93.184.216.34	TCP	56	42555 > http [ACK] Seq=114 Ack=1579 Win=34294 Len=0
10	0.063666000	10.0.2.15	93.184.216.34	TCP	56	42555 > http [FIN, ACK] Seq=114 Ack=1579 Win=34294 L
11	0.063837000	93.184.216.34	10.0.2.15	TCP	62	http > 42555 [ACK] Seq=1579 Ack=115 Win=65535 Len=0
12	0.079007000	93.184.216.34	10.0.2.15	TCP	62	http > 42555 [FIN, ACK] Seq=1579 Ack=115 Win=65535 L
13	0.079039000	10.0.2.15				1580 Win=34294 Len=0

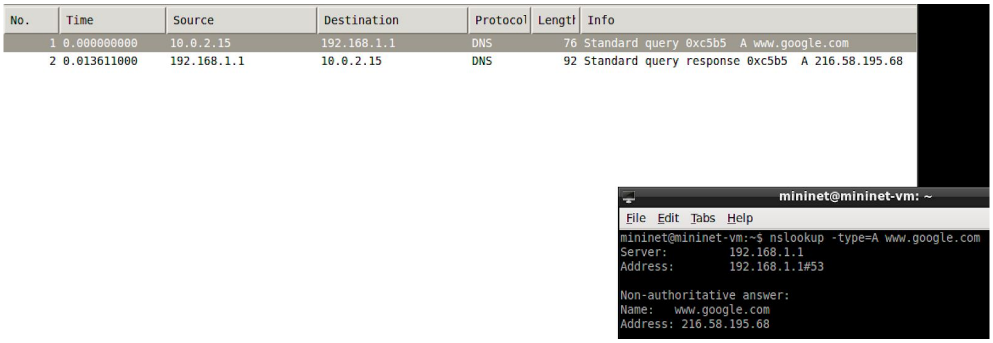
```
▷ Linux cooked capture
▷ Internet Protocol Version 4,
▷ Transmission Control Protocol
```

```
0000   00 04 00 01 00 06 08 00
0010   45 00 00 3c 3e a7 40 00
0020   5d b8 d8 22 a6 3b 00 50
0030   a0 02 72 10 42 18 00 00
0040   ff ff fe ac 00 00 00 00
```

```
mininet@mininet-vm:~$ wget --header "Host:www.example.com" 93.184.216.34
--2019-10-30 12:50:55--  http://93.184.216.34/
Connecting to 93.184.216.34:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1256 (1.2K) [text/html]
Saving to: ‘index.html.24’

100%[===================================>] 1,256       --.-K/s   in 0.004s

2019-10-30 12:50:55 (281 KB/s) - ‘index.html.24’ saved [1256/1256]
```

Using the command "wget --header "Host:www.example.com" 93.184.216.34" we were able to get the packets from above. We are only using the URL and the IP address, hence the bind. We only got those packets without sending DNS request.

7. **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?**

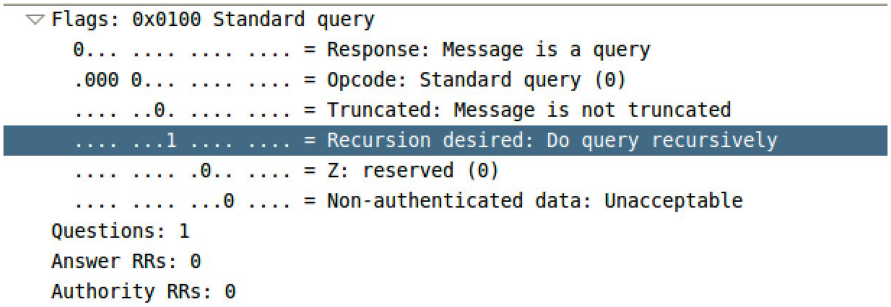


No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	192.168.1.1	DNS	76	Standard query 0xc5b5 A www.google.com
2	0.013611000	192.168.1.1	10.0.2.15	DNS	92	Standard query response 0xc5b5 A 216.58.195.68

```
mininet@mininet-vm:~$ nslookup -type=A www.google.com
Server:         192.168.1.1
Address:        192.168.1.1#53

Non-authoritative answer:
Name:   www.google.com
Address: 216.58.195.68
```

Using the command “nslookup -type=A www.google.com” the request was resolved. The IP address given was:
216.58.195.68; This is displayed in the terminal and in packet 2

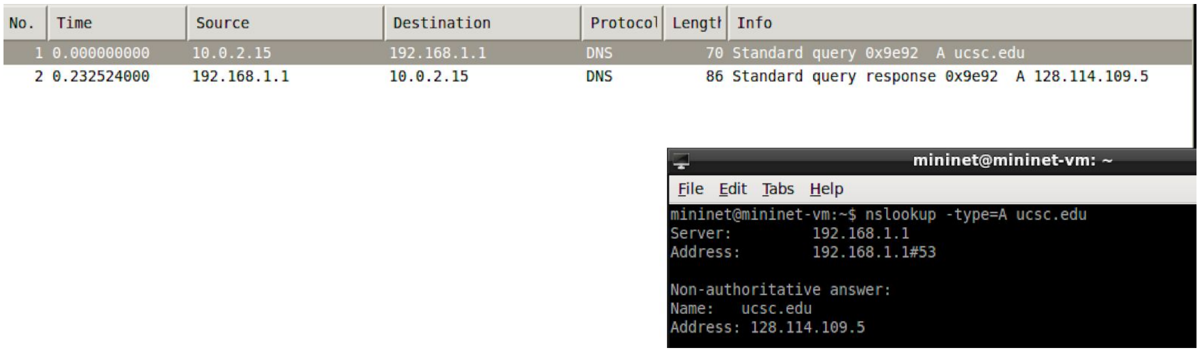
8. **Did your computer want to complete the request recursively? How do you know? Take a screenshot proving your answer**



```
▽ Flags: 0x0100 Standard query
    0... .... .... .... = Response: Message is a query
    .000 0... .... .... = Opcode: Standard query (0)
    .... ..0. .... .... = Truncated: Message is not truncated
    .... ...1 .... .... = Recursion desired: Do query recursively
    .... .... .0.. .... = Z: reserved (0)
    .... .... ...0 .... = Non-authenticated data: Unacceptable
  Questions: 1
  Answer RRs: 0
  Authority RRs: 0
```

Looking at the sc above, we can see that my computer did complete the request recursively. I checked under the Flags tab and under the “Recursion desired” we can see that it has a 1, meaning that it used recursion.

9. 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?

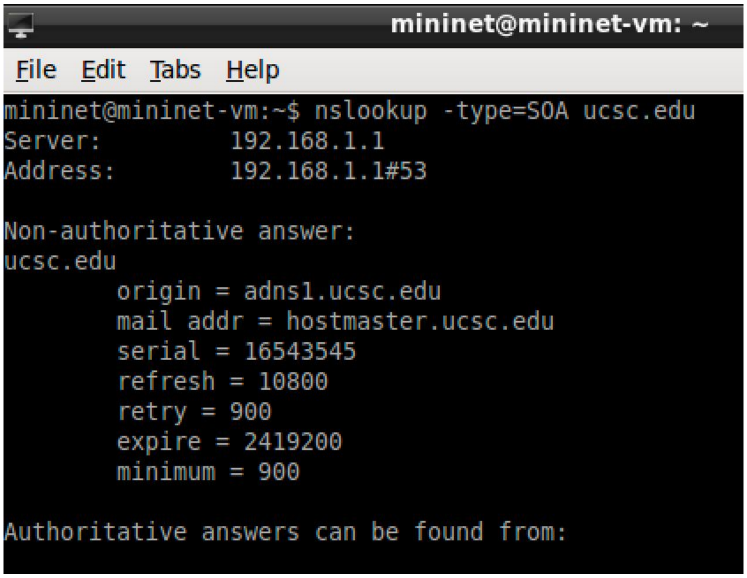


No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	192.168.1.1	DNS	70	Standard query 0x9e92 A ucsc.edu
2	0.232524000	192.168.1.1	10.0.2.15	DNS	86	Standard query response 0x9e92 A 128.114.109.5

```
mininet@mininet-vm:~$ nslookup -type=A ucsc.edu
Server:         192.168.1.1
Address:        192.168.1.1#53

Non-authoritative answer:
Name:   ucsc.edu
Address: 128.114.109.5
```

Using the command “nslookup -type=A ucsc.edu” the request was resolved. The given IP for ucsc.edu is 128.114.109.5

10. What is the authoritative name server for the ucsc.edu domain? How do you know?

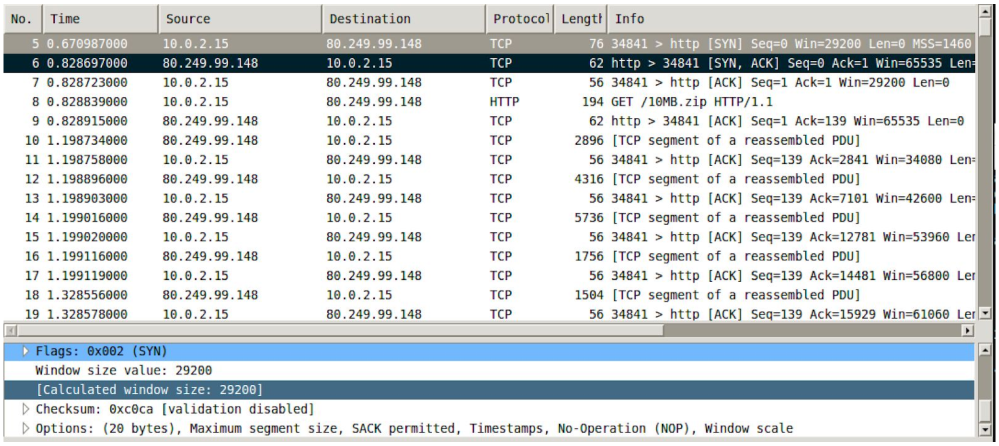


```
mininet@mininet-vm:~$ nslookup -type=SOA ucsc.edu
Server:         192.168.1.1
Address:        192.168.1.1#53

Non-authoritative answer:
ucsc.edu
        origin = adns1.ucsc.edu
        mail addr = hostmaster.ucsc.edu
        serial = 16543545
        refresh = 10800
        retry = 900
        expire = 2419200
        minimum = 900

Authoritative answers can be found from:
```

I used the Resource Record (RR) “SOA” because it specifies authoritative information about a DNS zone, which includes all the information seen above. With that, the authoritative name server for the ucsc.edu domain is: **adns1.ucsc.edu**.

Part 3: TCP

11. 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?

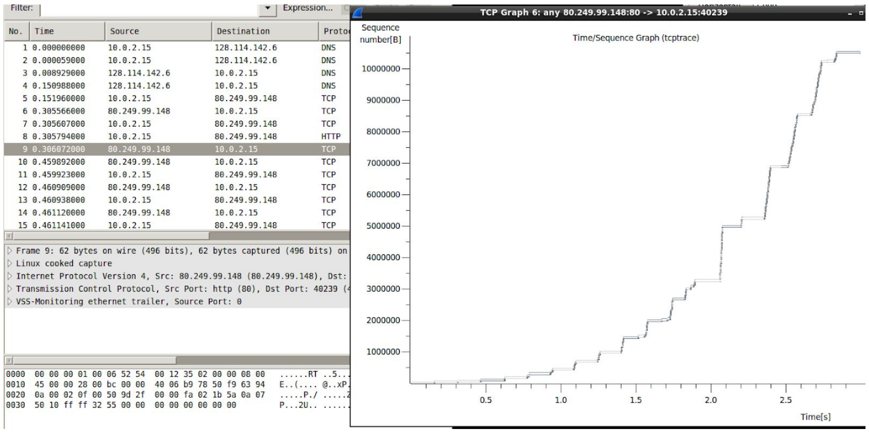


```
No. Time          Source        Destination   Protocol Lengtl Info
  5 0.670987000   10.0.2.15     80.249.99.148 TCP        76 34841 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460
  6 0.828697000   80.249.99.148 10.0.2.15     TCP        62 http > 34841 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=
  7 0.828723000   10.0.2.15     80.249.99.148 TCP        56 34841 > http [ACK] Seq=1 Ack=1 Win=29200 Len=0
  8 0.828839000   10.0.2.15     80.249.99.148 HTTP      194 GET /10MB.zip HTTP/1.1
  9 0.828915000   80.249.99.148 10.0.2.15     TCP        62 http > 34841 [ACK] Seq=1 Ack=139 Win=65535 Len=0
 10 1.198734000   80.249.99.148 10.0.2.15     TCP      2896 [TCP segment of a reassembled PDU]
 11 1.198758000   10.0.2.15     80.249.99.148 TCP        56 34841 > http [ACK] Seq=139 Ack=2841 Win=34080 Len=
 12 1.198896000   80.249.99.148 10.0.2.15     TCP      4316 [TCP segment of a reassembled PDU]
 13 1.198903000   10.0.2.15     80.249.99.148 TCP        56 34841 > http [ACK] Seq=139 Ack=7101 Win=42600 Len=
 14 1.199016000   80.249.99.148 10.0.2.15     TCP      5736 [TCP segment of a reassembled PDU]
 15 1.199020000   10.0.2.15     80.249.99.148 TCP        56 34841 > http [ACK] Seq=139 Ack=12781 Win=53960 Ler
 16 1.199116000   80.249.99.148 10.0.2.15     TCP      1756 [TCP segment of a reassembled PDU]
 17 1.199119000   10.0.2.15     80.249.99.148 TCP        56 34841 > http [ACK] Seq=139 Ack=14481 Win=56800 Ler
 18 1.328556000   80.249.99.148 10.0.2.15     TCP      1504 [TCP segment of a reassembled PDU]
 19 1.328578000   10.0.2.15     80.249.99.148 TCP        56 34841 > http [ACK] Seq=139 Ack=15929 Win=61060 Ler

 ▷ Flags: 0x002 (SYN)
   Window size value: 29200
   [Calculated window size: 29200]
 ▷ Checksum: 0xc0ca [validation disabled]
 ▷ Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale
```

There are a couple of packets that correspond to SYN, SYN-ACK, and ACK that initiated the TCP connection file transfer.
First SYN before the SYN-ACK (initial offering) = 29200
The SYN-ACK (response to the initial offering) = 65535

12. 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. Explain what it is showing.

I will be using this packet:



```
Filter:                                        Expression...
No. Time          Source         Destination    Proto
  1 0.000000000   10.0.2.15      128.114.142.6  DNS
  2 0.000059000   10.0.2.15      128.114.142.6  DNS
  3 0.008929000   128.114.142.6  10.0.2.15      DNS
  4 0.150988000   128.114.142.6  10.0.2.15      DNS
  5 0.151960000   10.0.2.15      80.249.99.148  TCP
  6 0.305566000   80.249.99.148  10.0.2.15      TCP
  7 0.305607000   10.0.2.15      80.249.99.148  TCP
  8 0.305794000   10.0.2.15      80.249.99.148  HTTP
  9 0.306072000   80.249.99.148  10.0.2.15      TCP
 10 0.459892000   80.249.99.148  10.0.2.15      TCP
 11 0.459923000   10.0.2.15      80.249.99.148  TCP
 12 0.460909000   80.249.99.148  10.0.2.15      TCP
 13 0.460938000   10.0.2.15      80.249.99.148  TCP
 14 0.461120000   80.249.99.148  10.0.2.15      TCP
 15 0.461141000   10.0.2.15      80.249.99.148  TCP

▷ Frame 9: 62 bytes on wire (496 bits), 62 bytes captured (496 bits) on
▷ Linux cooked capture
▷ Internet Protocol Version 4, Src: 80.249.99.148 (80.249.99.148), Dst:
▷ Transmission Control Protocol, Src Port: http (80), Dst Port: 40239 (4
▷ VSS-Monitoring ethernet trailer, Source Port: 0

0000  00 00 00 01 00 06 52 54  00 12 35 02 00 00 08 00   ......RT ..5...
0010  45 00 00 28 00 bc 00 00  40 06 b9 78 50 f9 63 94   E..(.... @..xP.
0020  0a 00 02 0f 00 50 9d 2f  00 00 fa 02 1b 5a 0a 07   .....P./ .....Z
0030  50 10 ff ff 32 55 00 00  00 00 00 00 00 00         P...2U.. ......
```

TCP Graph 6: any 80.249.99.148:80 -> 10.0.2.15:40239

Packet 9s source address is the server's address (the file I downloaded) and the destination is my computer's address. The graph indicates that as time increases, the sequence number also increases, which means that we are reducing the number of losses. The sequence number is the amount of data being processed in bytes.

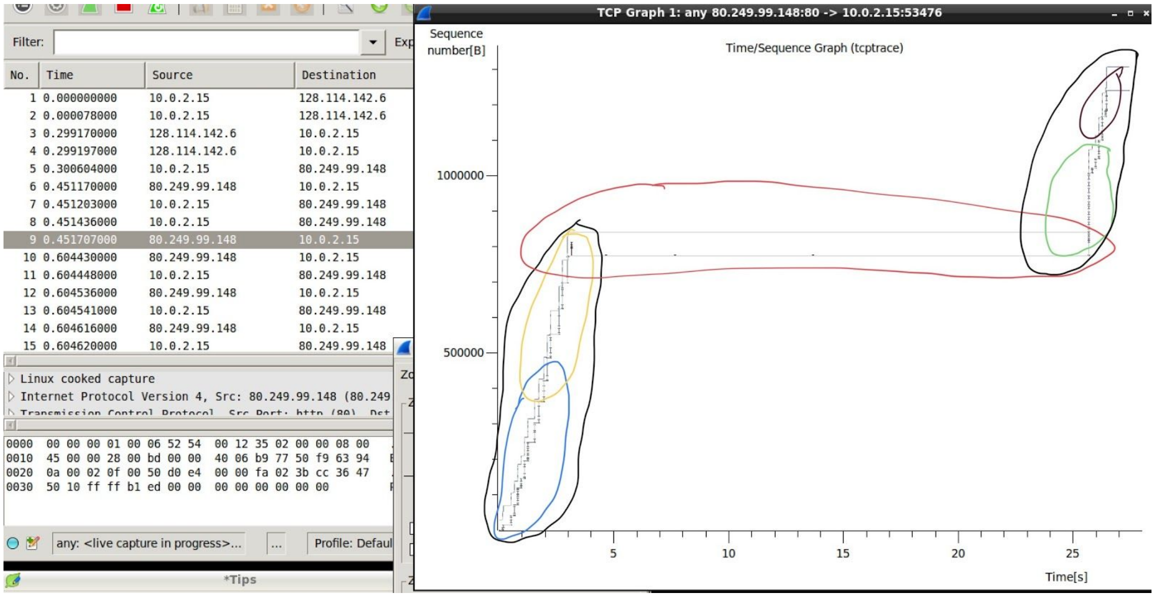
13. Find a packet from the download with a source of the server and a destination of your computer. Create a tcptrace graph with this packet selected. Explain what it is showing. Using an image editing program, circle the areas where the 0% loss is shown, as well as where TCP is in slow-start and congestion-avoidance.

Using the commands:

sudo tc qdisc add dev eth0 root netem loss 100%

sudo tc qdisc change dev eth0 root netem loss 0%

Using the highlighted packet:



Filter:

No.	Time	Source	Destination
1	0.000000000	10.0.2.15	128.114.142.6
2	0.000078000	10.0.2.15	128.114.142.6
3	0.299170000	128.114.142.6	10.0.2.15
4	0.299197000	128.114.142.6	10.0.2.15
5	0.300604000	10.0.2.15	80.249.99.148
6	0.451170000	80.249.99.148	10.0.2.15
7	0.451203000	10.0.2.15	80.249.99.148
8	0.451436000	10.0.2.15	80.249.99.148
9	0.451707000	80.249.99.148	10.0.2.15
10	0.604430000	80.249.99.148	10.0.2.15
11	0.604448000	10.0.2.15	80.249.99.148
12	0.604536000	80.249.99.148	10.0.2.15
13	0.604541000	10.0.2.15	80.249.99.148
14	0.604616000	80.249.99.148	10.0.2.15
15	0.604620000	10.0.2.15	80.249.99.148

```
▷ Linux cooked capture
▷ Internet Protocol Version 4, Src: 80.249.99.148 (80.249
▷ Transmission Control Protocol, Src Port: http (80), Dst
```

```
0000  00 00 00 01 00 06 52 54  00 12 35 02 00 00 08 00
0010  45 00 00 28 00 bd 00 00  40 06 b9 77 50 f9 63 94
0020  0a 00 02 0f 00 50 d0 e4  00 00 fa 02 3b cc 36 47
0030  50 10 ff ff b1 ed 00 00  00 00 00 00 00 00
```

any: <live capture in progress>...

Profile: Defaul

TCP Graph 1: any 80.249.99.148:80 -> 10.0.2.15:53476

Packet 9 contains the source address of the server and the destination of my computer's address. It is showing what happens when there is 100% packet loss, waiting for a couple seconds and then changing it to 0% packet loss. From that we can see that:

- Blue circle = Slow start
- Yellow Circle = Congestion
- Black Circle = 0% Loss
- Red Circle = 100% Loss
- Green Circle = Recovering and slow start
- Burgundy Circle = Congestion avoidance
- 2nd Black Circle = 0% Loss

Essentially this graph is just showing the packets being processed.