

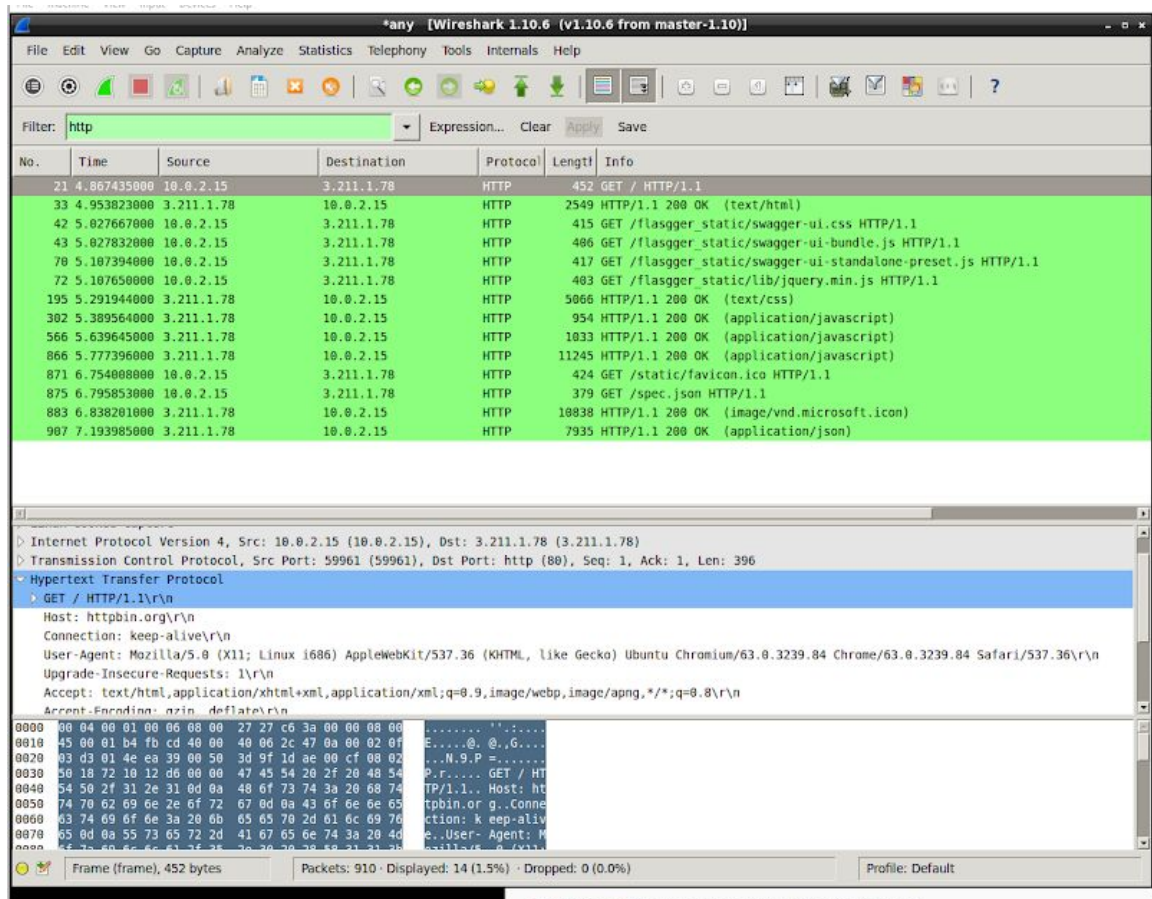
Prelab 2:

1. 5 HTTP status codes:
  - a. 100: Continue. This code indicates that everything is ok so far and the client is allowed to continue the request if it is still in progress.
  - b. 200: OK. Used when the request has succeeded, where the meaning of success depends on the HTTP method used.
  - c. 201: Created. A request has completed successfully and a new resource has been created from said request.
  - d. 202: Accepted. The request has been received but no action has been taken in response to said request.
  - e. 300: Multiple Choice: Request can have more than one response, which is determined by the user or user-agent
2. 8 HTTP 1.1 Methods
  - a. GET: Retrieve information from requested server using an URI without affecting said data.
  - b. HEAD: Retrieves the status line and header section from the given server using an URI.
  - c. POST: Send data to the server
  - d. PUT: Updates target resource with supplied content
  - e. DELETE: Removes representation of target resource given by URI
  - f. CONNECT: Create a tunnel to server from URI
  - g. OPTIONS: Describes communication options for a target resource
  - h. TRACE: Send and receive message along path to target resource as a ping
3. Last Modified date was 17 October 2019. Return status was 200 OK from `curl -I example.com`
4. It plays the Star Wars in ascii
5. The DNS RR is a type of data used to build host names and IP addresses used for name-resolution services. The resource record from the command gives the IP address of the site.
6. This command returns the 13 root name servers .
7. Multiple applications can run on the same machine with a single Ip address being uniquely identified by using port addresses, where each application uses a different port.
8. Windowing, the window mechanism in TCP, is used to control the flow of packets between two computers or network hosts. This method allows for multiple packets of data to be affirmed with single acknowledgement.
9. MTU is the maximum transmission unit. If a packet is larger than the MTU, the packet becomes fragmented in order to be sent in parts, which causes some latency and inefficiency in network communications.

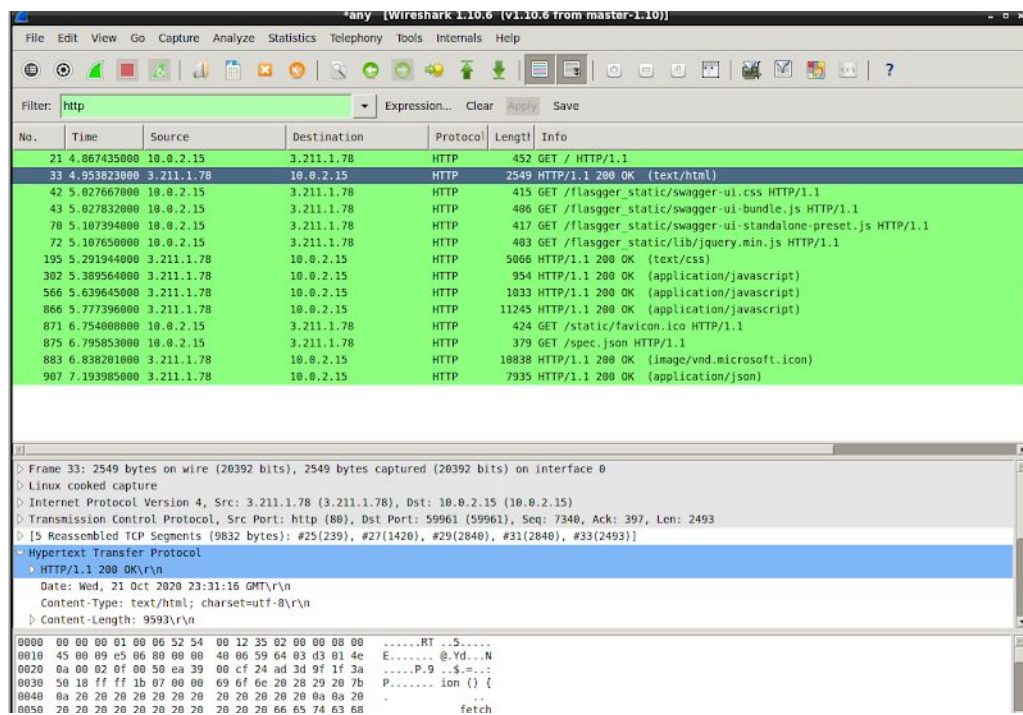
Lab 2:

Part 1:

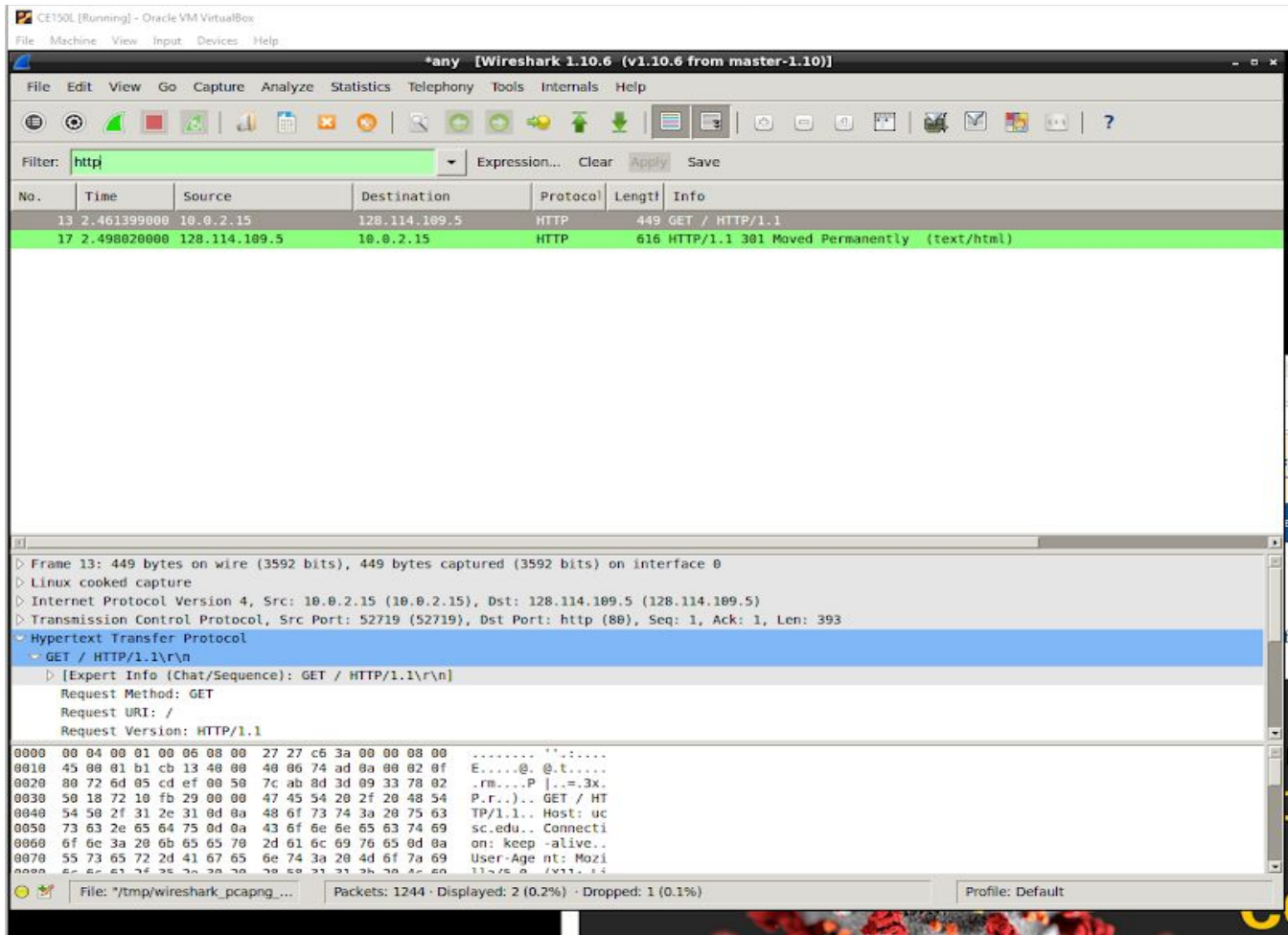
1. The HTTP method used was GET



2. The initial response the server made in response to my request was status code 200 OK. The content type is text/html



- For the initial request, the only difference is the length of the packet, the responses from the servers are different. For UCSC.edu, the response is a 301 Moved permanently status code rather than a 200 OK code. Additionally, the communication no longer continues after loading the page for ucsc while the previous page made multiple requests and responses aside from the initial.



- Running `curl http://example.com --head` in the terminal will make a http packet using the HEAD method. Was unable to install curl in mininet.

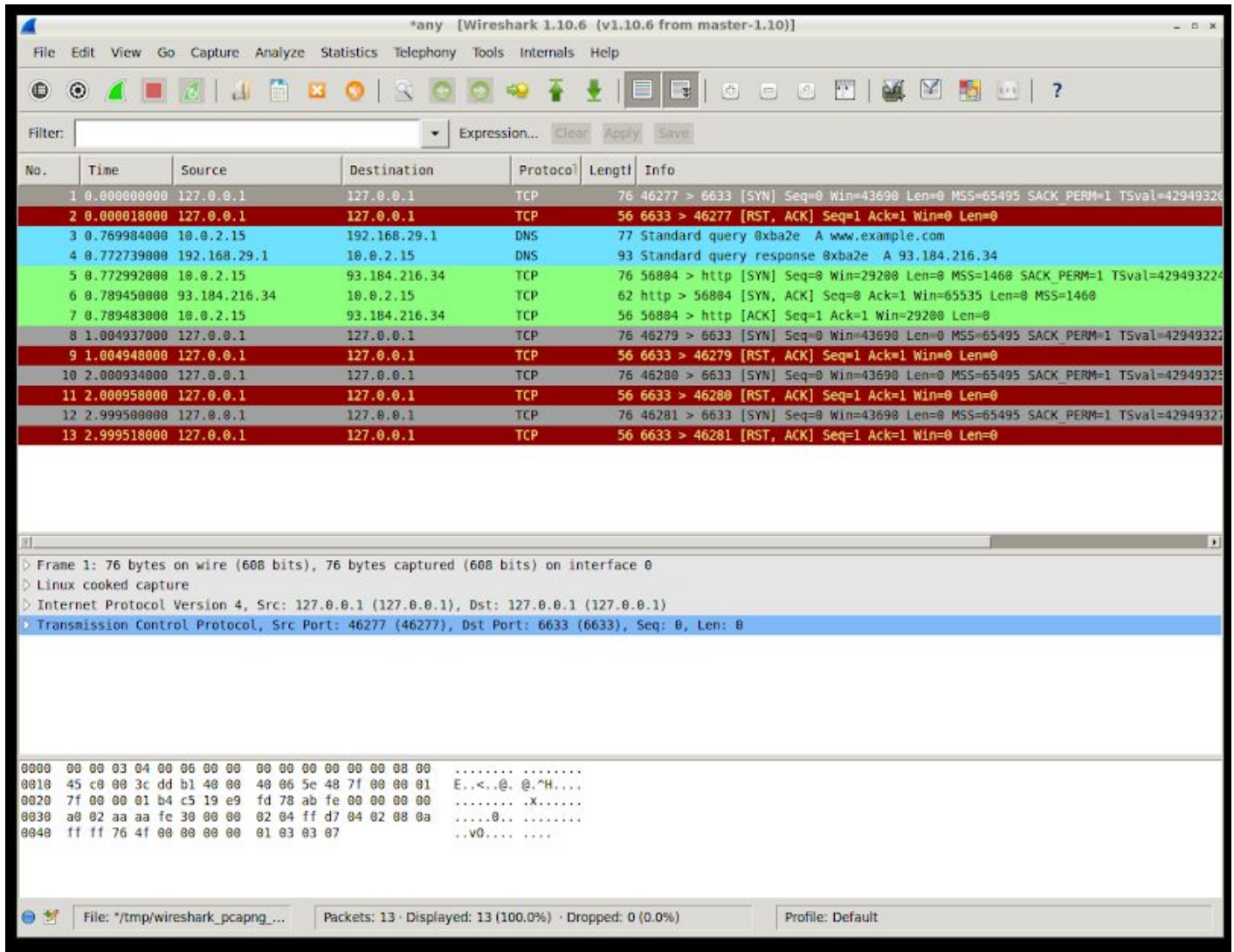
```
-bash-4.2$ curl http://example.com --head
HTTP/1.1 200 OK
Content-Encoding: gzip
Accept-Ranges: bytes
Age: 593151
Cache-Control: max-age=604800
Content-Type: text/html; charset=UTF-8
Date: Thu, 22 Oct 2020 02:07:08 GMT
Etag: "3147526947"
Expires: Thu, 29 Oct 2020 02:07:08 GMT
Last-Modified: Thu, 17 Oct 2019 07:18:26 GMT
Server: ECS (oxr/8322)
X-Cache: HIT
Content-Length: 648

-bash-4.2$
```

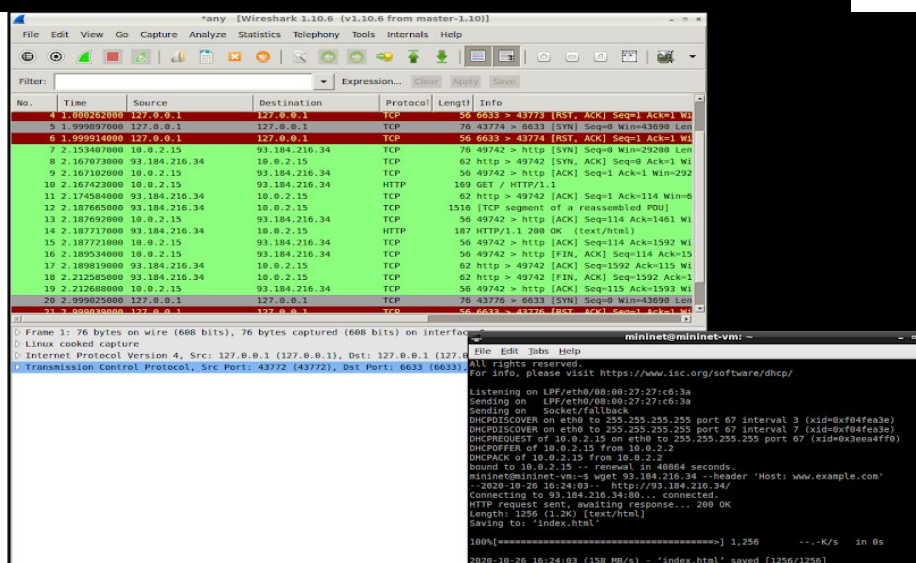


Part 2:

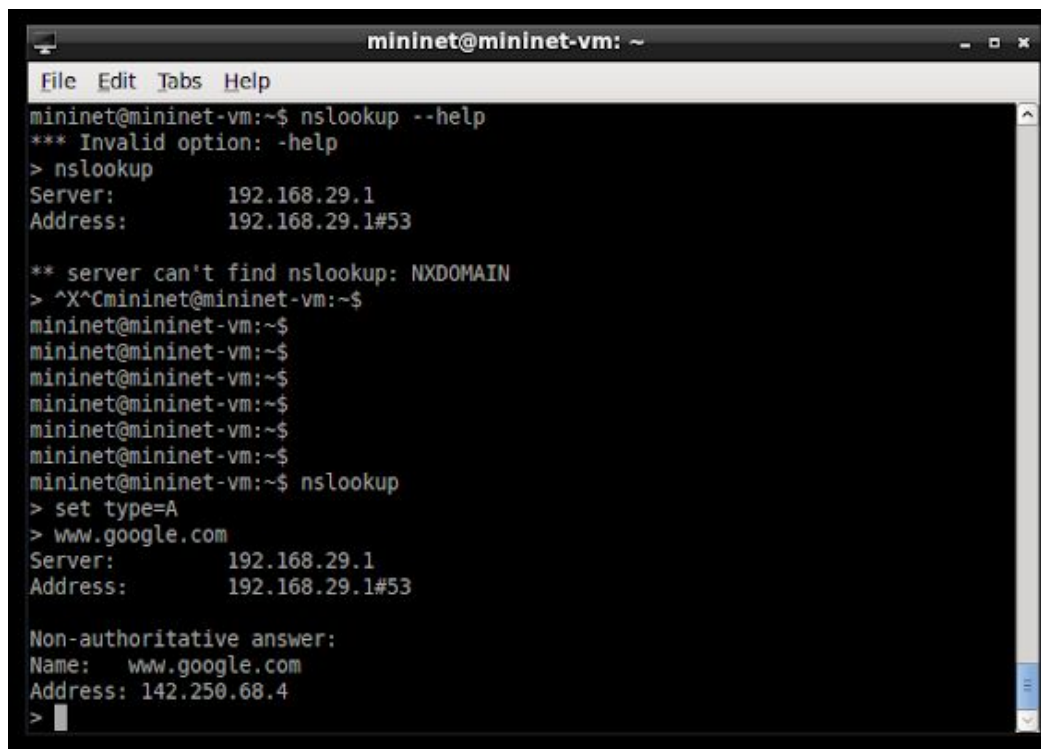
- Before the page is loaded, there is a DNS protocol making a query to the site, followed by their response and then SYN and ACK TCP protocols. I think the DNS protocol (blue) packets were the ones that locates where to find the site data and the TCP protocols load the data. The IP address of the site is 93.184.216.34



- Command used was `wget 93.184.216.34 --header 'Host: www.example.com'`. I think this is correct because wireshark shows no references to DNS protocols while still downloading the same content from the previous question

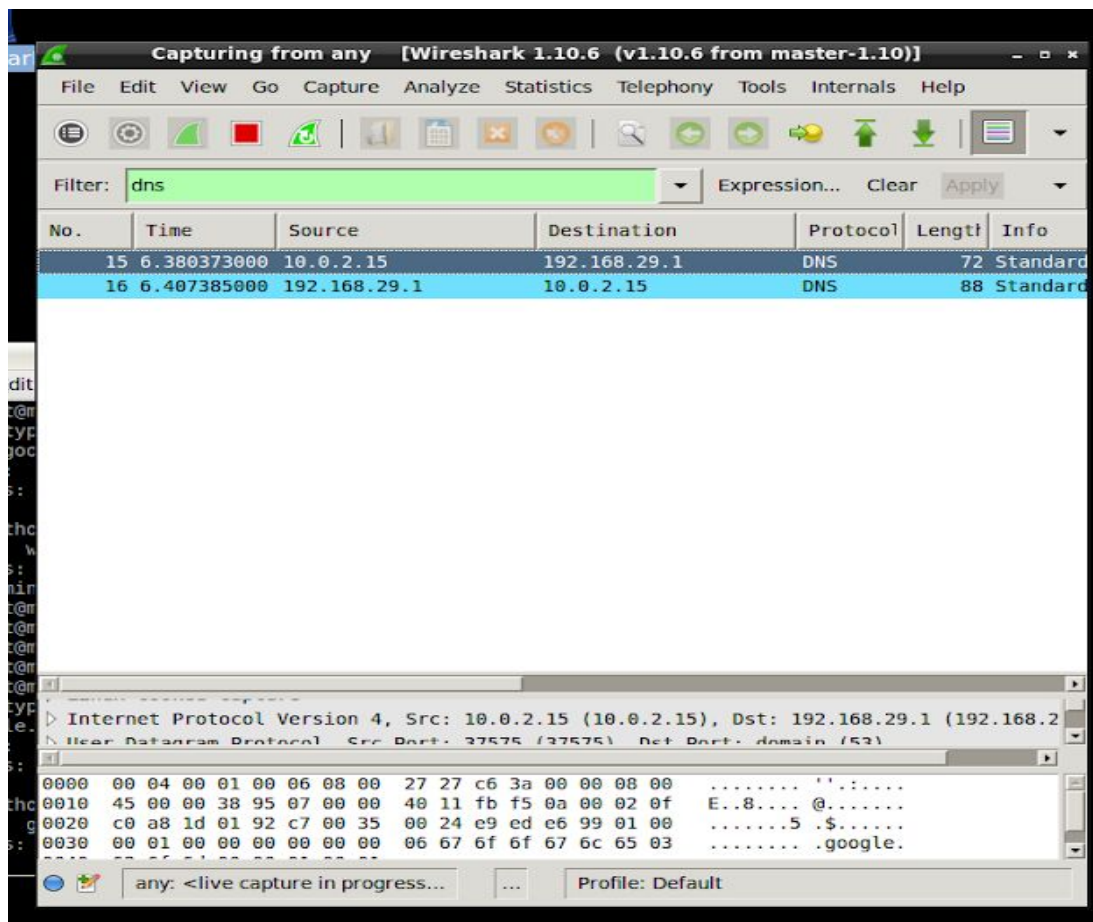


7. Address got was 142.250.68.4

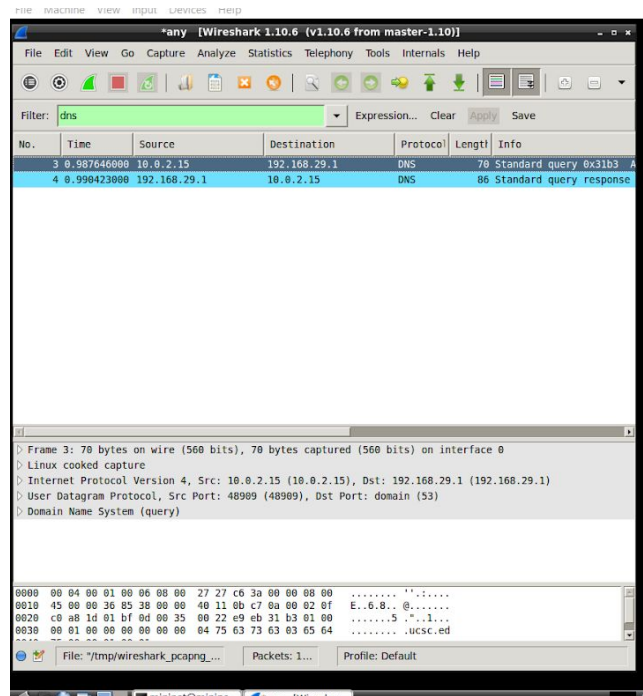


```
mininet@mininet-vm: ~  
File Edit Tabs Help  
mininet@mininet-vm:~$ nslookup --help  
*** Invalid option: -help  
> nslookup  
Server:          192.168.29.1  
Address:         192.168.29.1#53  
  
** server can't find nslookup: NXDOMAIN  
> ^X^Cmininet@mininet-vm:~$  
mininet@mininet-vm:~$  
mininet@mininet-vm:~$  
mininet@mininet-vm:~$  
mininet@mininet-vm:~$  
mininet@mininet-vm:~$  
mininet@mininet-vm:~$ nslookup  
> set type=A  
> www.google.com  
Server:          192.168.29.1  
Address:         192.168.29.1#53  
  
Non-authoritative answer:  
Name:   www.google.com  
Address: 142.250.68.4  
>
```

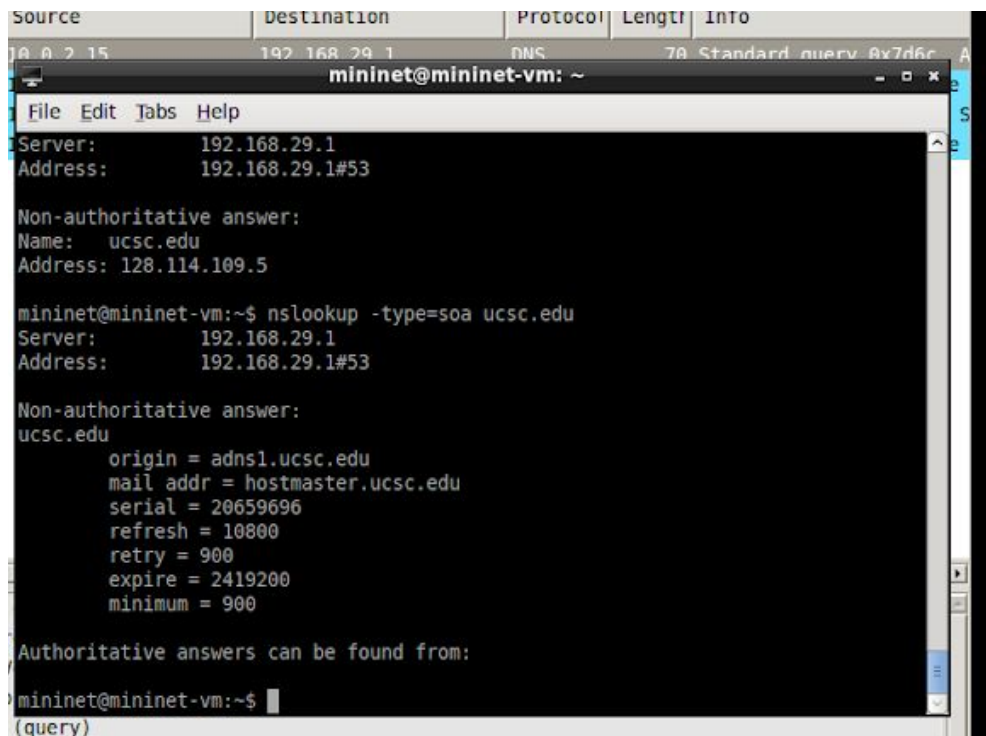
8. The command is recursive because there is only one DNS query instead of an iterative set of queries and referrals.



9. UCSC address is 128.114.109.5 from Nslookup, whereas the packet came from 192.168.29.1 from the DNS.



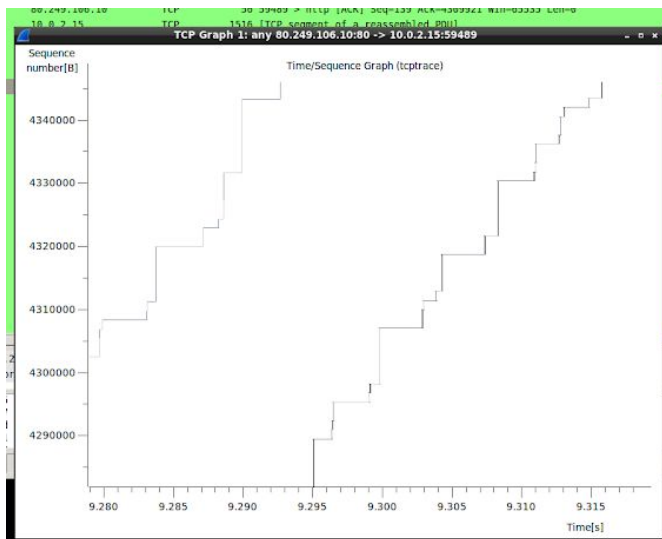
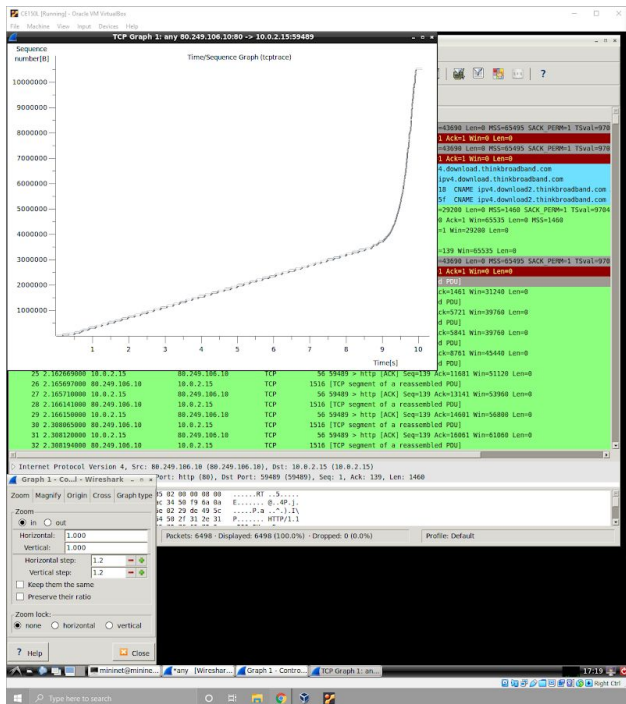
10. The authoritative name server is adns.ucsc.edu, since type=soa responds with the name of the authoritative name server



11. The initial window size by my computer is 29200 while the server advertised 65535

9	1.832718000	10.0.2.15	80.249.106.10	TCP	76	59489 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=9704
10	1.990602000	80.249.106.10	10.0.2.15	TCP	62	http > 59489 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
11	1.990677000	10.0.2.15	80.249.106.10	TCP	56	59489 > http [ACK] Seq=1 Ack=1 Win=29200 Len=0
12	1.991974000	10.0.2.15	80.249.106.10	HTTP	194	GET /10MB.zip HTTP/1.1
13	1.992678000	80.249.106.10	10.0.2.15	TCP	62	http > 59489 [ACK] Seq=1 Ack=139 Win=65535 Len=0
14	2.007572000	127.0.0.1	127.0.0.1	TCP	76	46562 > 6633 [SYN] Seq=0 Win=43698 Len=0 MSS=65493 SACK_PERM=1 TSval=970

12. For the first 9 seconds, the number of bytes sent seems to be linear, but then exponentially increases after 9 seconds until the download is finished. At almost any given point, the TCP segments lies to the right of the ACK segments





13. In this graph, the download proceeds as normal, but the red circle is where netem loss becomes 100%, so no data is downloaded, and the blue circle is where it resumes, and does a little congestion control by slowing the download until it becomes stable.

