

Prelab 2 - HTTP, DNS, and TCP:

HTTP Questions

1. [7 pts] Choose 5 HTTP status codes and describe each one.

1. 200 OK: The request has succeeded. The information returned with the response is dependent on the method (e.g., GET, HEAD, POST) used in the request.
2. 307 Temporary Redirect: The requested resource resides temporarily under a different URI. Because the redirection may be altered on occasion, the client should continue to use the Request-URI for future requests.
3. 400 Bad Request: The request could not be understood by the server due to malformed syntax. The client should not repeat the request without modifications.
4. 404 Not Found: The server has not found anything matching the Request-URI.
5. 500 Internal Server Error: The server encountered an unexpected condition which prevented it from fulfilling the request.

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., & Berners-Lee, T. (1999). Hypertext Transfer Protocol -- HTTP/1.1. (RFC 2616). Internet Engineering Task Force. <https://www.ietf.org/rfc/rfc2616.txt>

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

1. OPTIONS: This method requests information about the communication options available on the request/response chain identified by the Request-URI. It allows the client to determine the options and/or requirements associated with a resource, or the capabilities of a server, without implying a resource action or initiating a resource retrieval.
2. GET: This method retrieves the information (in the form of an entity) identified by the Request-URI. The GET method's semantics may change to a "conditional GET" or a "partial GET" depending on the presence of conditional or range header fields in the request message. Responses to a GET request can be cacheable.
3. HEAD: This method is identical to GET except that the server must not return a message-body in the response. It is used to obtain meta-information about the entity implied by the request without transferring the entity-body itself, often for testing hypertext links for validity, accessibility, and recent modification.
4. POST: This method requests that the origin server accept the enclosed entity as a new subordinate of the resource identified by the Request-URI. It is a uniform method for functions like annotating existing resources, posting messages to bulletin boards, providing data blocks to data-handling processes, and extending databases through append operations. Responses to this method are not cacheable.

5. PUT: This method requests that the enclosed entity be stored under the supplied Request-URI. If the Request-URI refers to an existing resource, the enclosed entity should be considered as a modified version of the one residing on the origin server. If the Request-URI does not point to an existing resource, the origin server can create the resource with that URI. Responses to this method are not cacheable.
6. DELETE: This method requests that the origin server delete the resource identified by the Request-URI. The operation may be overridden by human intervention or other means on the origin server. The client cannot be guaranteed that the operation has been carried out, even if the status code indicates that the action has been completed successfully. Responses to this method are not cacheable.
7. TRACE: This method is used to invoke a remote, application-layer loop-back of the request message. The final recipient of the request should reflect the message received back to the client as the entity-body of a 200 (OK) response. TRACE allows the client to see what is being received at the other end of the request chain and use that data for testing or diagnostic information. Responses to this method must not be cached.
8. CONNECT: This method name is reserved for use with a proxy that can dynamically switch to being a tunnel, such as SSL tunneling.

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., & Berners-Lee, T. (1999). Hypertext Transfer Protocol -- HTTP/1.1. (RFC 2616). Internet Engineering Task Force. <https://www.ietf.org/rfc/rfc2616.txt>

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.)

After running command `wget --spider --server-response example.com`, the server response headers includes the "Last-Modified" date of "Thu, 17 Oct 2019 07:18:26 GMT" and the HTTP return status of "200 OK".

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?

The www.telehack.com `telnet` server is a simulation of a stylized interface for ARPANET and Usenet, circa 1985-1990. After connecting to www.telehack.com, and then, running `starwars`, it starts playing an ASCII text-based animation of the movie *Star Wars, Episode IV: A New Hope*.

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 Domain Name System (DNS) resource record (RR) is an entry in a DNS database that maps a domain name or a subdomain to specific information (e.g., IP addresses, mail servers, name servers, and other data) related to that domain. After running command `nslookup -`

query=MX ucsc.edu, the output shows mail exchange (MX) records for ucsc.edu, which indicates the mail servers responsible for receiving email messages on behalf of the domain.

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

After running `nslookup -type=ns .`, the output displays a list of the root name servers. The root name servers are responsible for directing queries to the appropriate top-level domain (TLD) name servers, which then forward the queries to the appropriate authoritative name servers for specific domain names.

TCP Questions

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

Multiple application services running on a single machine with a single IP address can be uniquely identified by using different port numbers (16-bit unsigned integer, ranging from 0 to 65535). Each application service is assigned a unique port number, and the combination of the IP address and the port number (known as a socket) allows the operating system and the network devices to distinguish between different services on the same machine.

Kozierok, C. M. (n.d.). *TCP/IP sockets and socket pairs: Process and connection identification*. The TCP/IP Guide. Retrieved May 2, 2023, from http://www.tcpipguide.com/free/t_TCPIPSocketsandSocketPairsProcessandConnectionIden-2.htm

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

The purpose of the window mechanism in TCP is to implement flow control between the sender and the receiver. Flow control ensures that the sender does not overwhelm the receiver with too much data at once, preventing buffer overflow and ensuring efficient and reliable data transmission. The window size indicates the amount of data that the receiver can accept at a given time. The sender uses this information to adjust the rate at which it sends data. As the receiver processes the incoming data, it updates the window size and sends it back to the sender, allowing the sender to adjust its transmission rate accordingly.

Kozierok, C. M. (n.d.). *TCP Window Size Adjustment and Flow Control*. The TCP/IP Guide. Retrieved May 2, 2023, from http://www.tcpipguide.com/free/t_TCPWindowSizeAdjustmentandFlowControl.htm

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

MTU stands for Maximum Transmission Unit. It is the largest size of a packet that a network protocol can transmit without requiring fragmentation. When a packet is larger than the MTU, it needs to be fragmented into smaller pieces before being transmitted. The sender divides the packet into smaller fragments, each having its own IP header with the necessary information for reassembly at the destination. The receiver reassembles the fragments to obtain the original packet. Fragmentation can introduce overhead and reduce the efficiency of data transmission. If the "Don't Fragment" (DF) flag is set in the IP header and the packet size exceeds the MTU, the packet is dropped, and an ICMP "Fragmentation Needed" message is sent back to the sender.

Kozierok, C. M. (n.d.). *IP Datagram Size, the Maximum Transmission Unit (MTU), and Fragmentation Overview*. The TCP/IP Guide. Retrieved May 2, 2023, from http://www.tcpipguide.com/free/t_IPDatagramSizetheMaximumTransmissionUnitMTUandFrag-4.htm

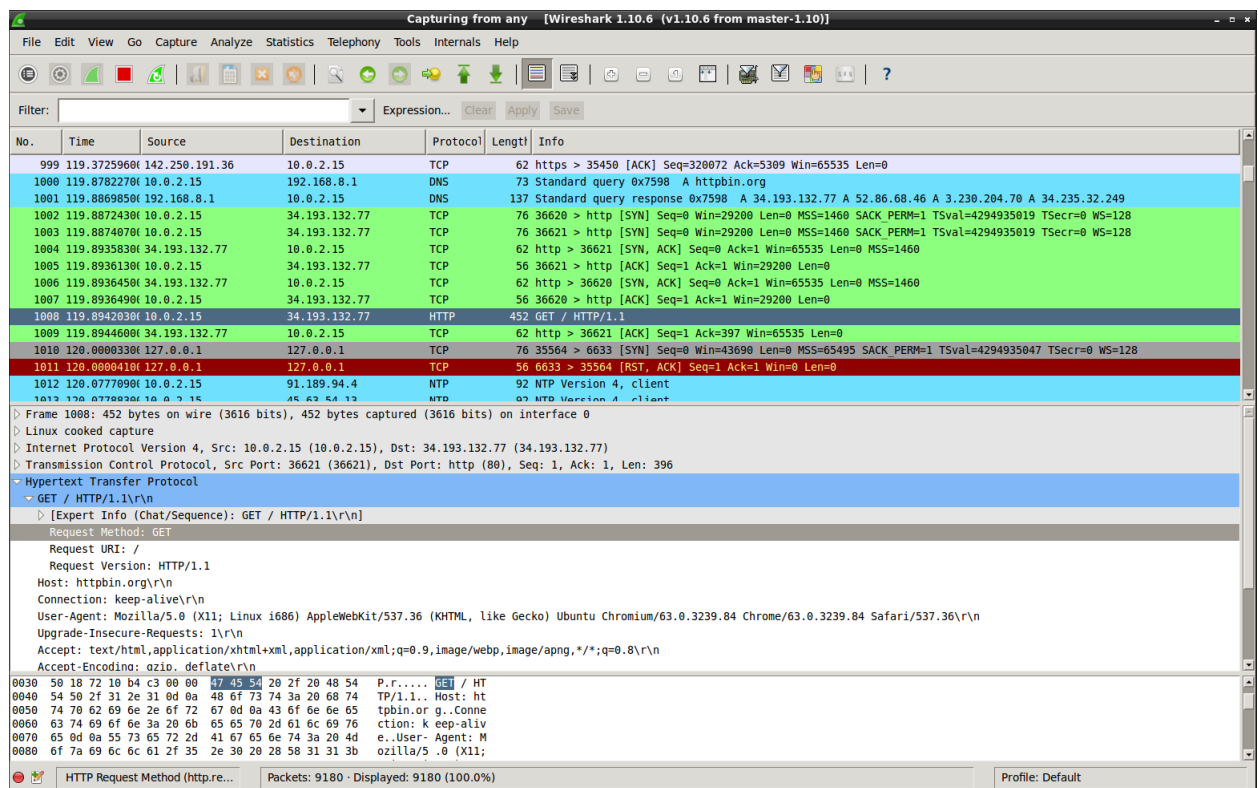
Lab 2 - HTTP, DNS, and TCP:

Part 1: HTTP

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

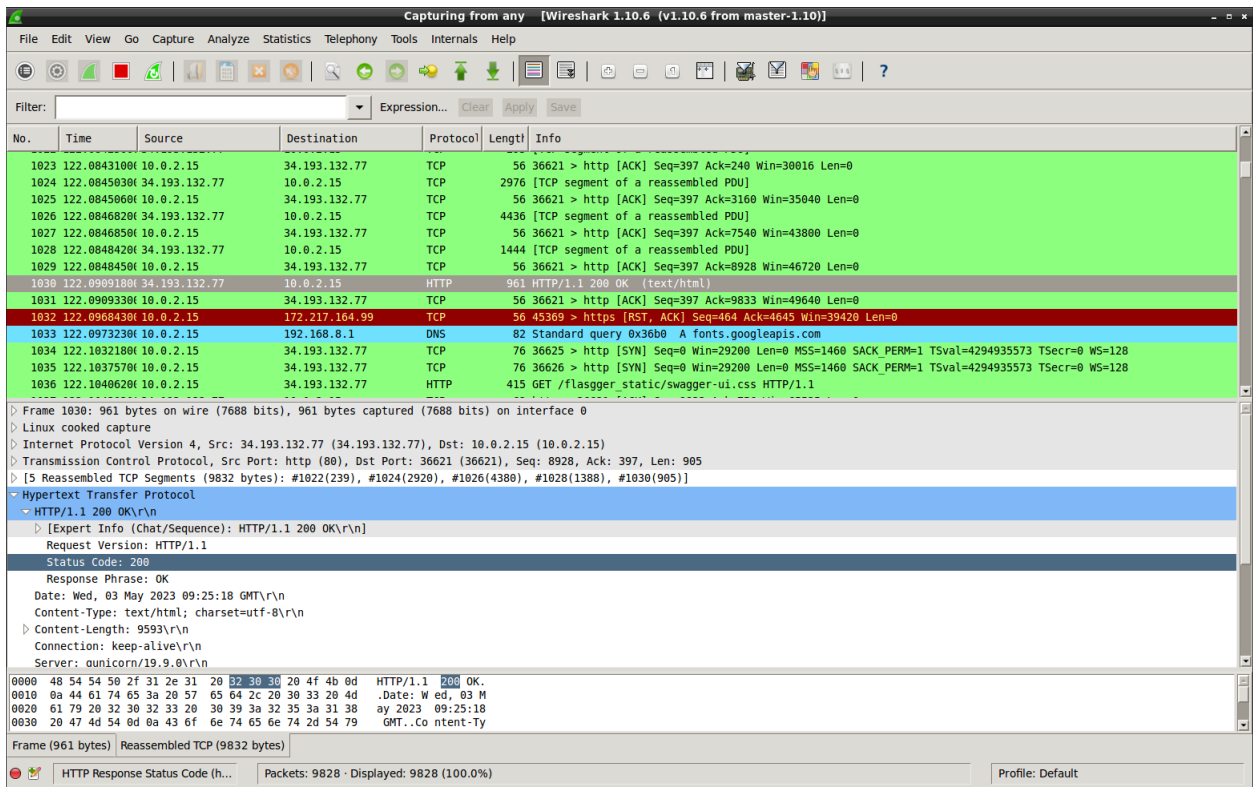
1. [10 pts] Find the HTTP packet that corresponds to the initial request that your computer made. Take a screenshot of this packet. What HTTP method did your computer use to make this request?

My computer used the "GET" method to make the request. Below is a screenshot of the initial request that my computer made:



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?

The server returned HTTP status code "200 OK". The content-type of the response the server sent back is "text/html; charset=utf-8". Below is a screenshot of this HTTP packet:



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

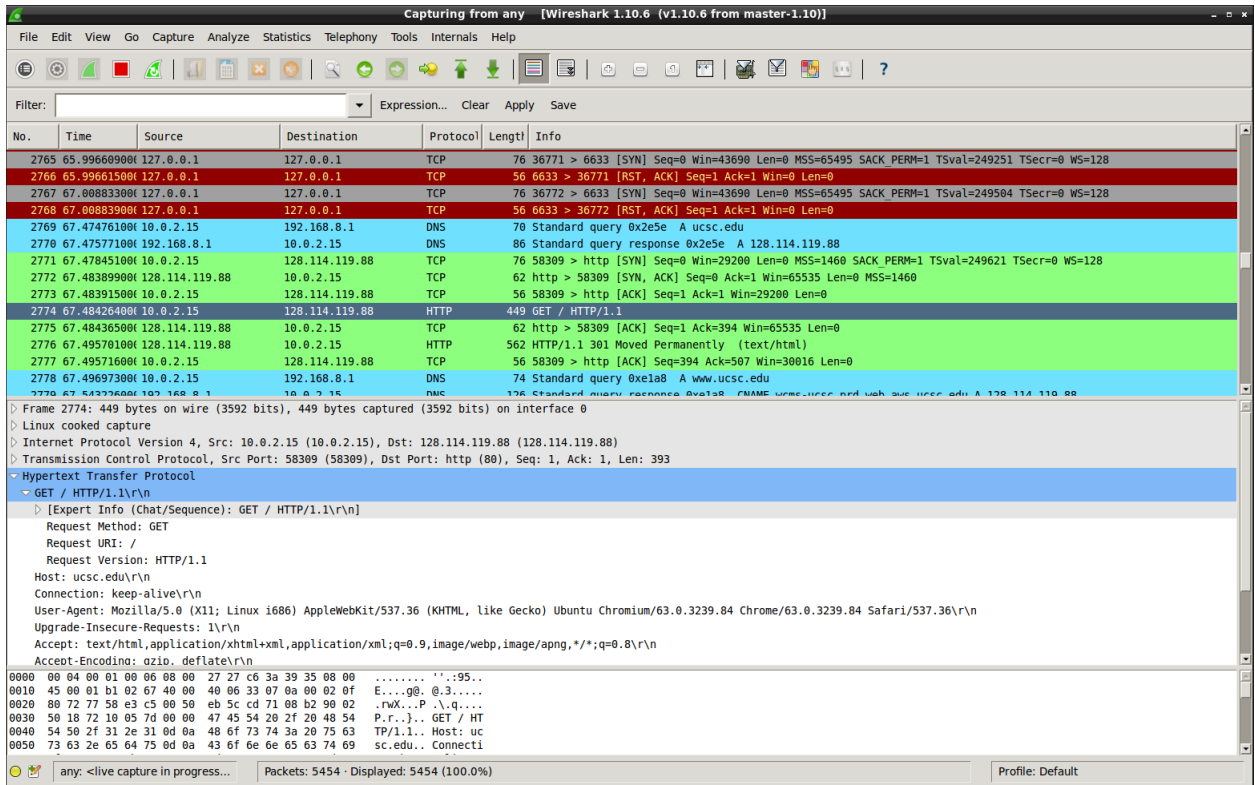
3. [10 pts] Find the HTTP packets that correspond to the initial request and response that your computer made. Take a screenshot of these packets. What's different? Explain.

The difference between the initial request and response is that instead of receiving the content of the requested page, my computer received a "301 Moved Permanently" status, indicating that the resource has been moved to a new location.

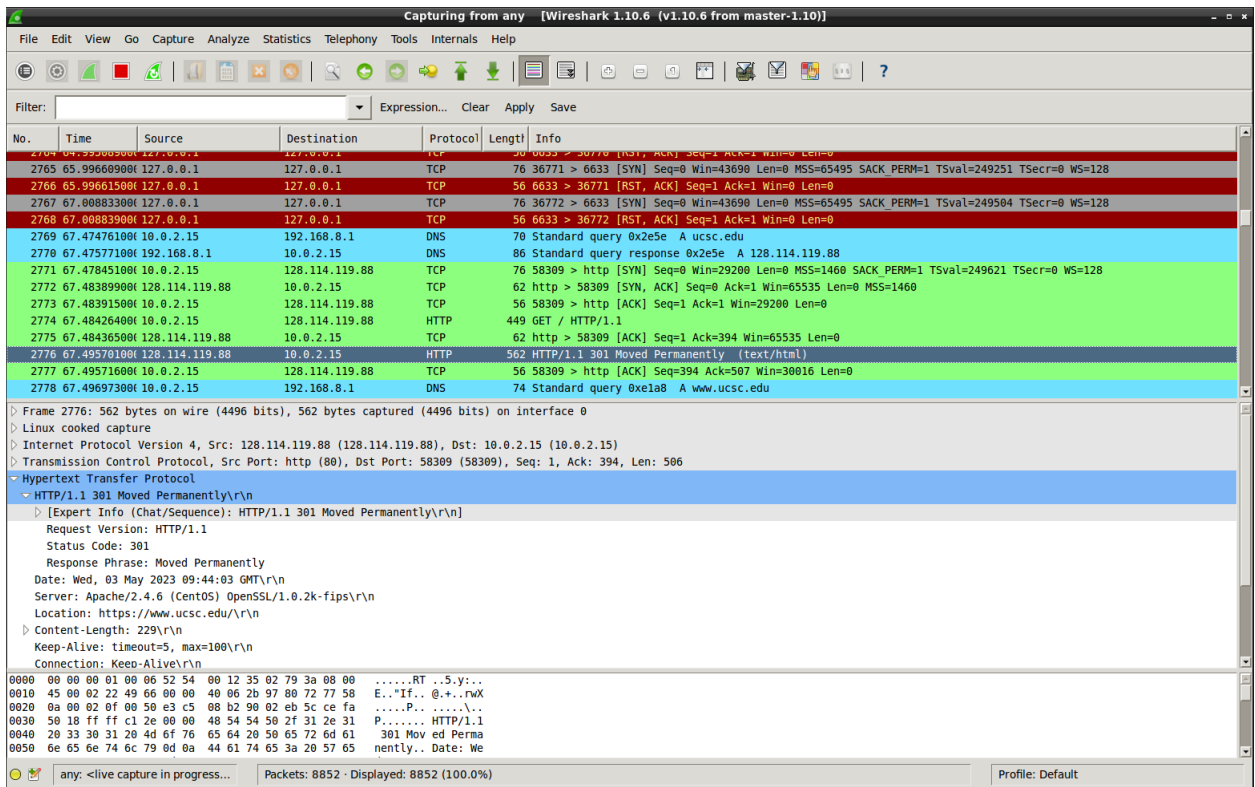
In the initial request, my computer sent a GET request for the main page ("/") of the UCSC website. In response, the server didn't send the content of the main page. Instead, it informed my browser that the requested resource has moved permanently to a new location.

The new location is in the "Location" header in the response packet, which contains the URL. My browser automatically follow the new URL and request the resource again, ultimately receiving the content of the desired page.

Below is a screenshot of the initial request packet that my computer made:



Below is a screenshot of the initial response that server sends back to my computer in response to the initial request:



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.

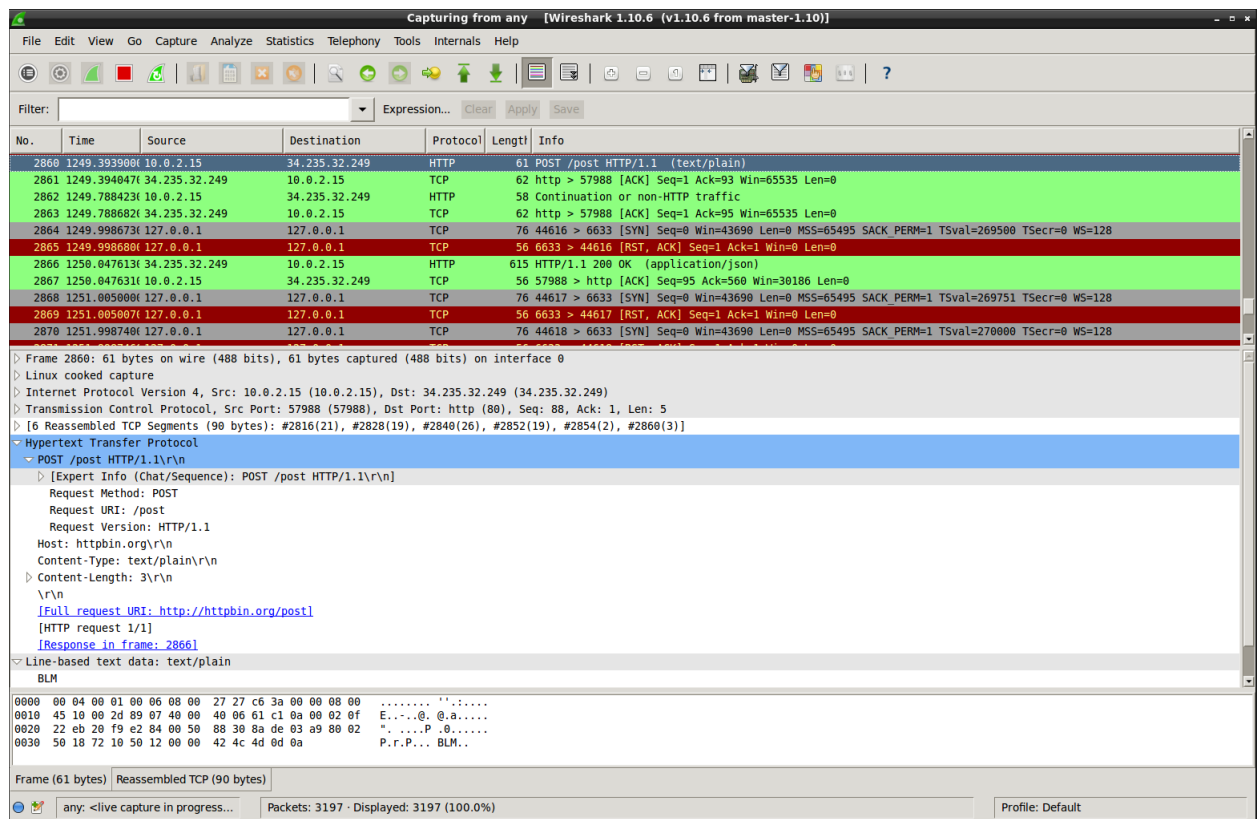
After running command `telnet httpbin.org 80`, I manually created a POST request by typing in the following lines and pressing enter after each line:

```
POST /post HTTP/1.1
Host: httpbin.org
Content-Type: text/plain
Content-Length: 3
```

BLM

Then I pressed entered twice after typing the request body to allow the server to process my request and respond accordingly.

Below is a screenshot of the initial request packet that my computer made:



Part 2: DNS

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?

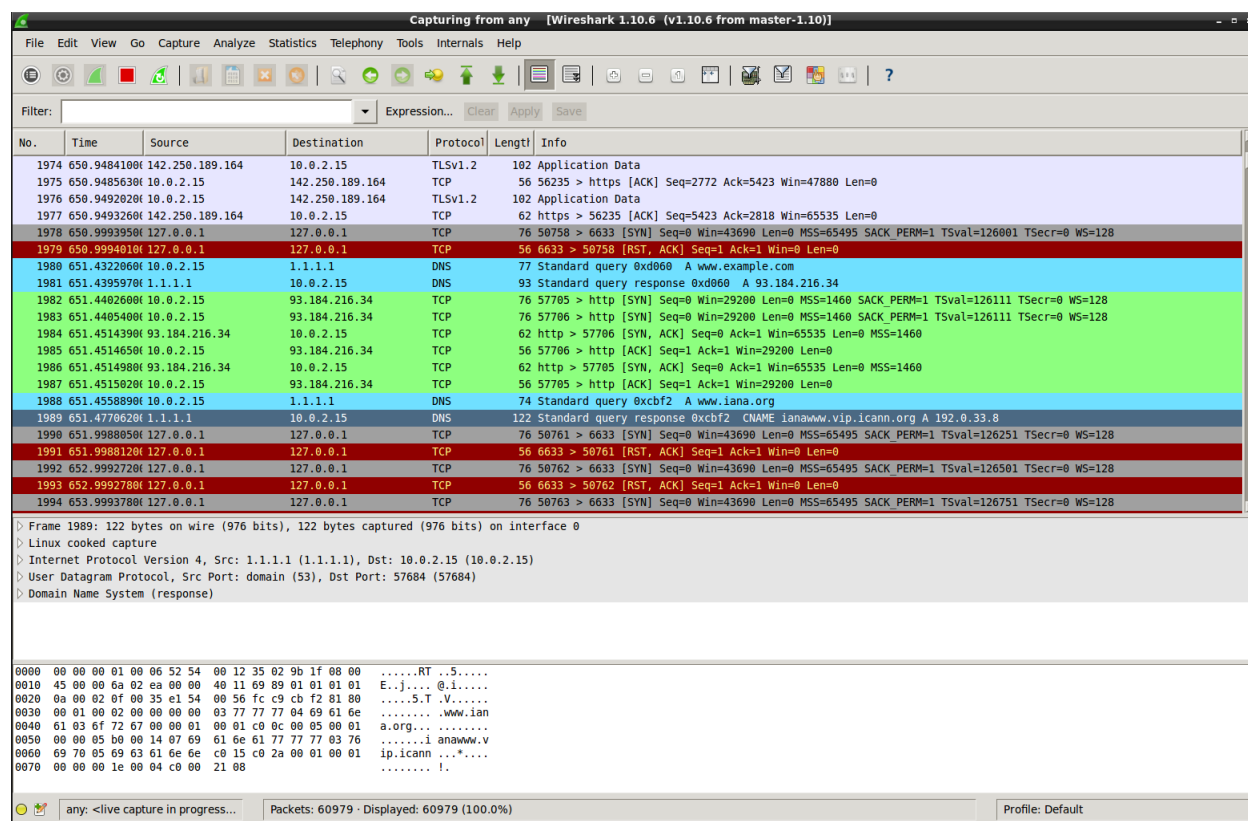
I believe these are the correct packets because the Domain Name System (DNS) query and response packets resolves the domain name to an IP address 93.184.216.34, which is required before any communication with the web server can occur.

Frame 1980 is the DNS query sent by my computer, asking for the IP address of “www.example.com”. The source IP address is 10.0.2.15 (my computer), and the destination IP address is 1.1.1.1 (DNS server).

Frame 1981 is the DNS response from the DNS server, providing the IP address for “www.example.com”. The source IP address is 1.1.1.1 (DNS server), and the destination IP address is 10.0.2.15 (my computer). The response contains the IP address of “www.example.com”, which is 93.184.216.34.

After receiving the IP address, my computer establishes a TCP connection and exchanges HTTP packets with the web server at 93.184.216.34 to load the web page. The series of packets (frame 1982-1987) show the process of establishing the connection, making HTTP requests, and receiving HTTP responses.

Below is a screenshot of these packets:

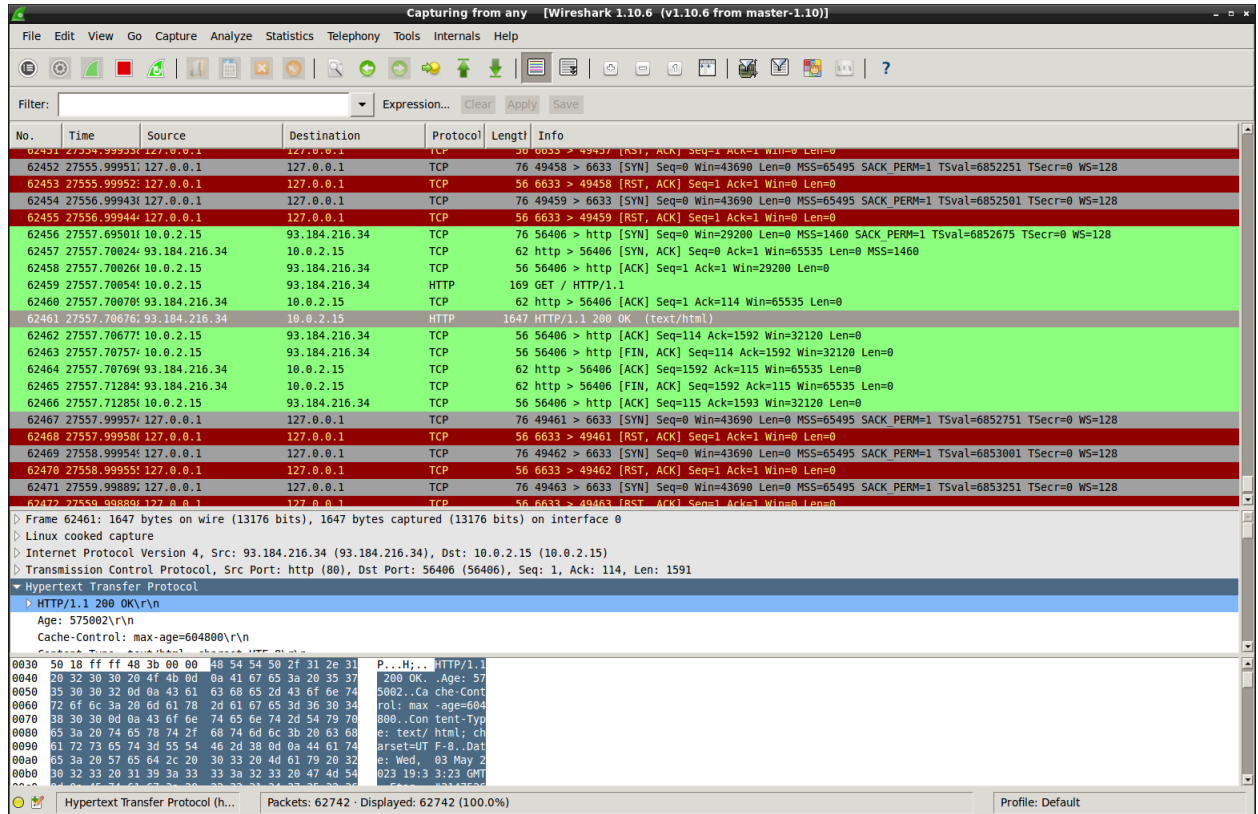


6. [10 pts] Open a terminal window. Execute the command to flush your DNS cache: `sudo /etc/init.d/networking restart`

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.

After running `wget --header "Host: www.example.com" --no-dns-cache 93.184.216.34`, these packets in the screenshot below are displayed without any preceding DNS requests. I used the IP address I discovered in the previous question and set the 'Host' header to the domain name, which is necessary for web servers that have multiple virtual hosts/websites on a single IP address. I believe these are the correct packets because I was able to make an HTTP request and receive a response without sending any DNS requests.

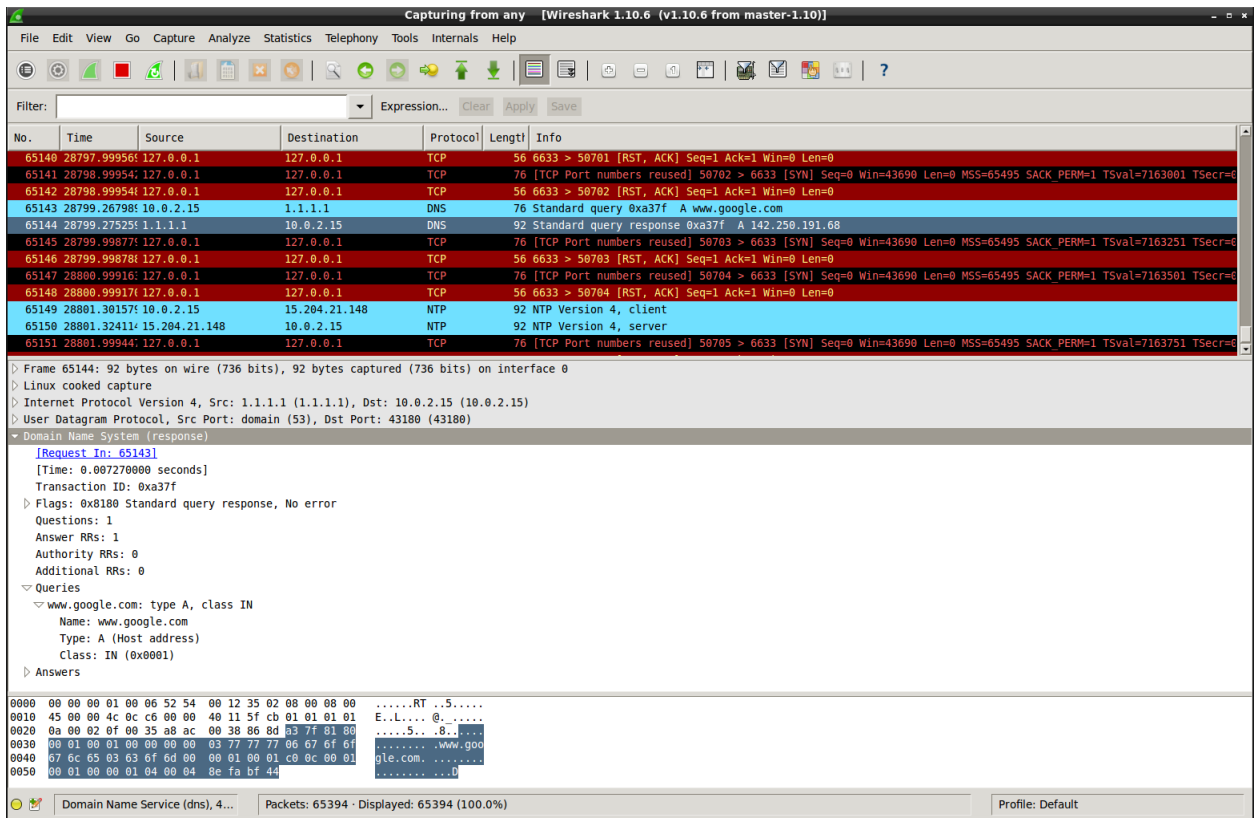


Open a terminal window. Using `nslookup`, find the A records for www.google.com. (If you can't access Google, for example, you are in China, you could replace the domain name with www.baidu.com)

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?

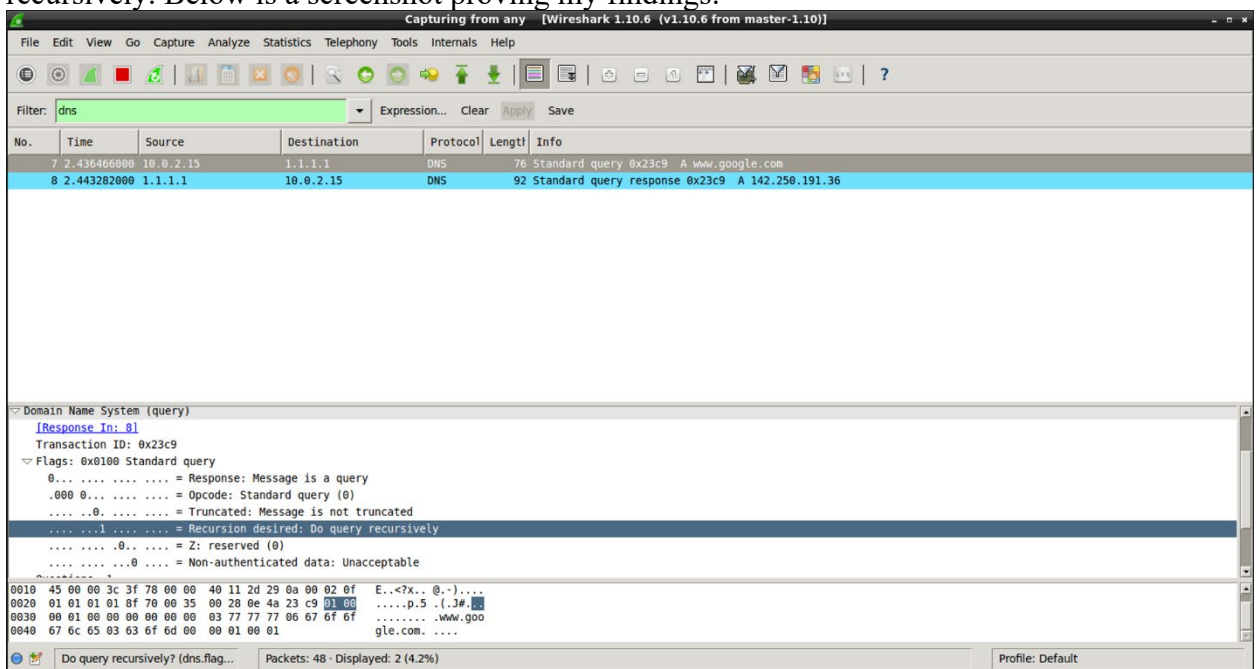
After running `nslookup www.google.com`, the request was resolved, and the given IP address for www.google.com is 142.250.191.68 (Keep in mind that I may receive a different IP address each time I run the `nslookup` command for www.google.com because Google often uses multiple IP addresses to distribute their traffic and improve load balancing.)

Below is a screenshot of the packets corresponding to my request and the response from the www.google.com server:



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

Yes, my computer completed the request recursively. I can tell this because the “Recursion Desired” flag value is “1”, which indicates that my computer wanted to complete the request recursively. Below is a screenshot proving my findings:

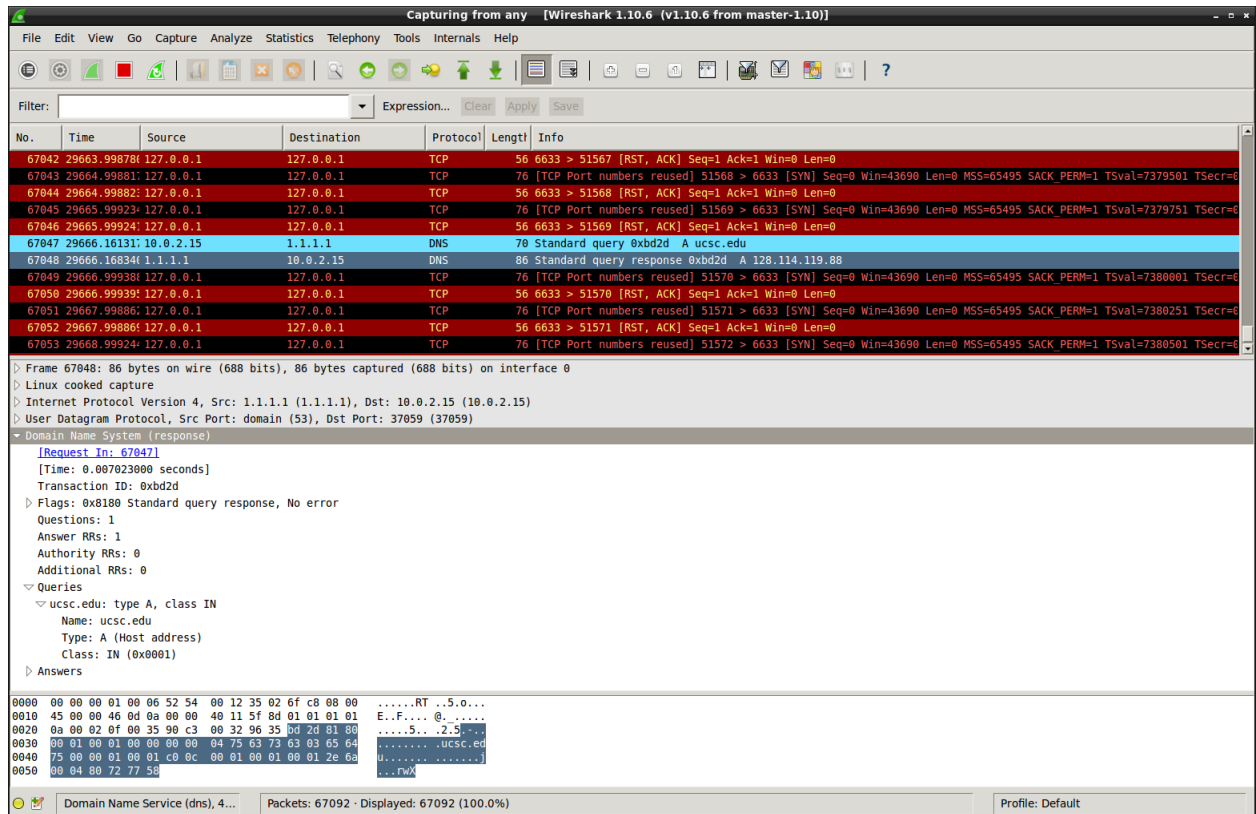


Using nslookup, find the A records for ucsc.edu.

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?

When the request was resolved, the given IP address for ucsc.edu is 128.114.119.88.

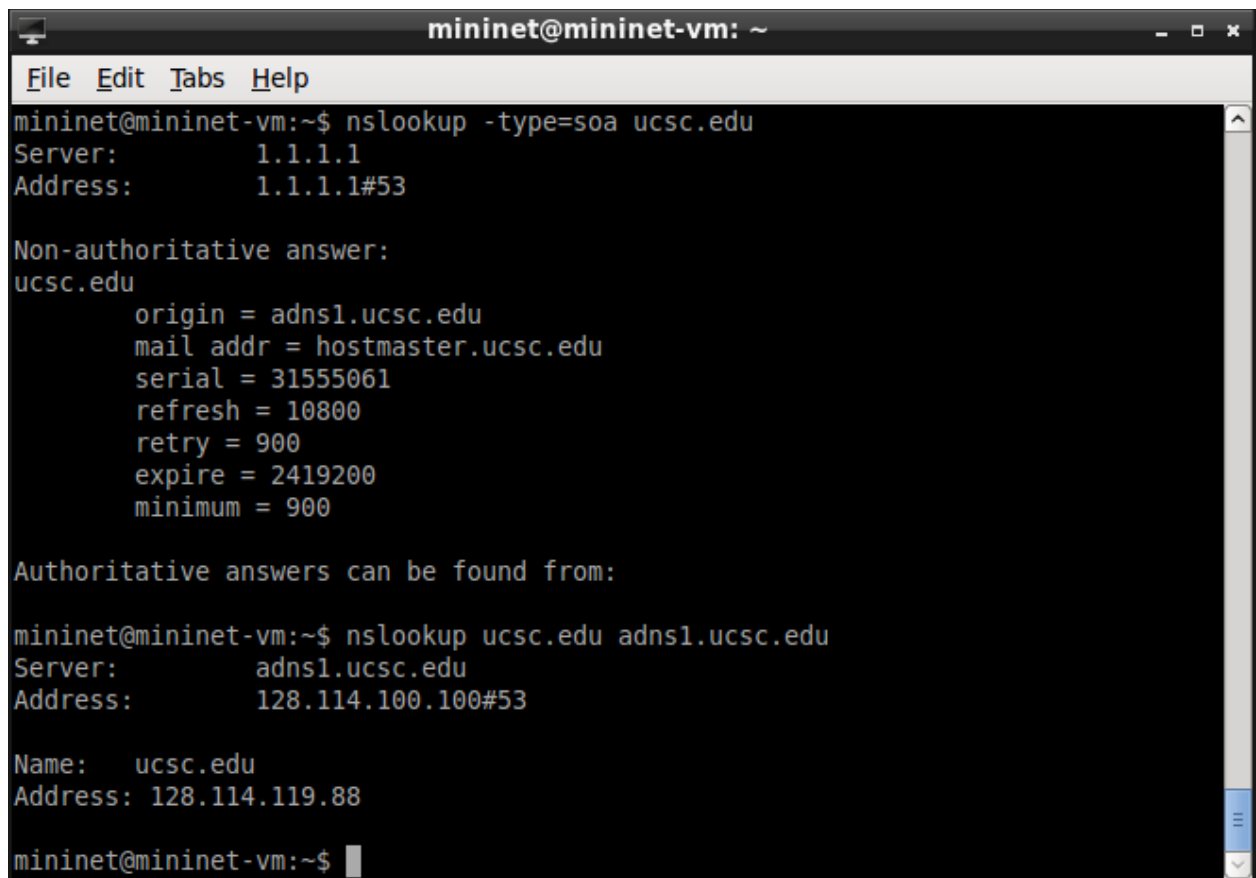
Below is a screenshot of the packets corresponding to my request and the response from the ucsc.edu server:



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

After running, `nslookup -type=soa http://ucsc.edu/` to get the origin name server, run `nslookup ucsc.edu adns1.ucsc.edu` specifying the primary name server as `adns1.ucsc.edu`. The response is the authoritative name server `adns1.usc.edu` and its corresponding IPv4 address `128.114.100.100`.

Below is a screenshot proving my findings:



```
mininet@mininet-vm: ~  
File Edit Tabs Help  
mininet@mininet-vm:~$ nslookup -type=soa ucsc.edu  
Server:      1.1.1.1  
Address:     1.1.1.1#53  
  
Non-authoritative answer:  
ucsc.edu  
    origin = adns1.ucsc.edu  
    mail addr = hostmaster.ucsc.edu  
    serial = 31555061  
    refresh = 10800  
    retry = 900  
    expire = 2419200  
    minimum = 900  
  
Authoritative answers can be found from:  
  
mininet@mininet-vm:~$ nslookup ucsc.edu adns1.ucsc.edu  
Server:      adns1.ucsc.edu  
Address:     128.114.100.100#53  
  
Name:   ucsc.edu  
Address: 128.114.119.88  
  
mininet@mininet-vm:~$
```

Part 3: TCP

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?

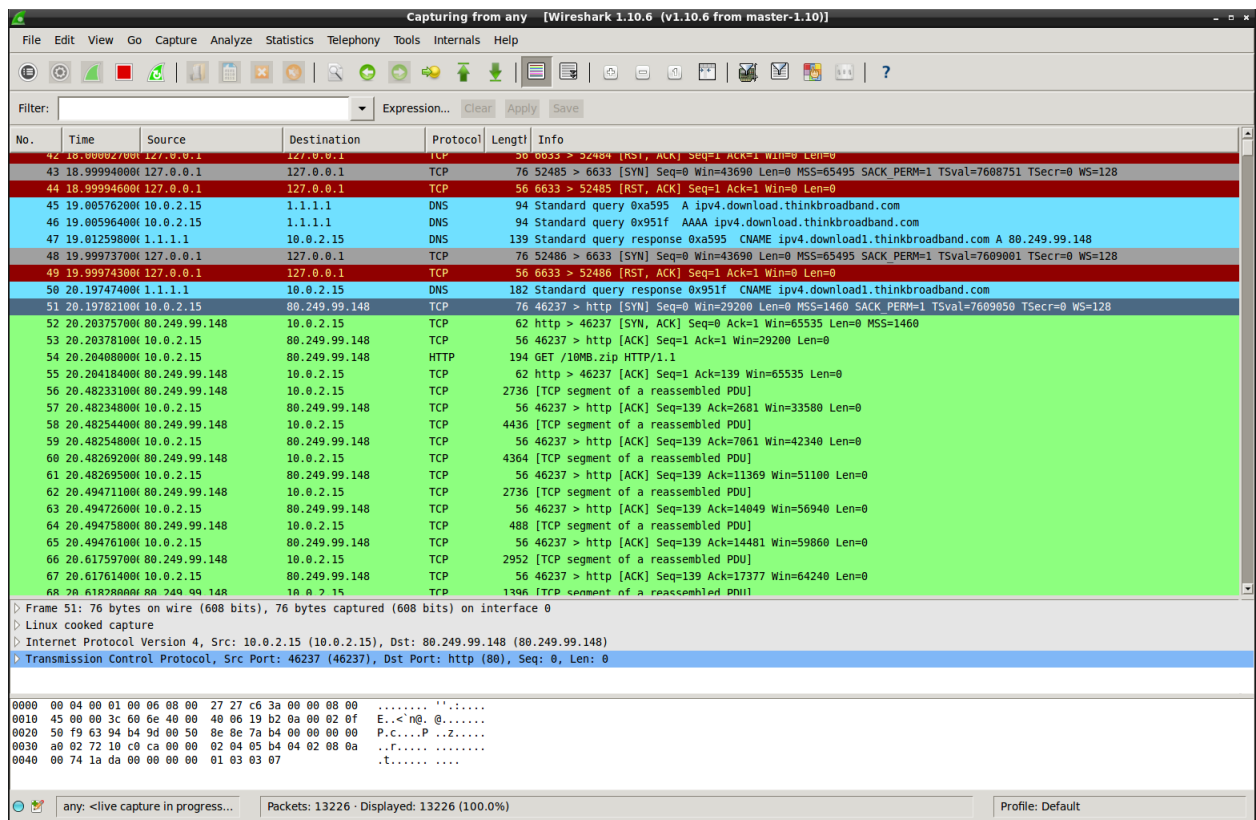
After running wget

<http://ipv4.download.thinkbroadband.com/10MB.zip>,

In Frame 51 (SYN packet), the window size advertised by my computer is 29200.

In Frame 52 (SYN-ACK packet), the window size advertised by the server is 65535.

Below is a screenshot of these packets:

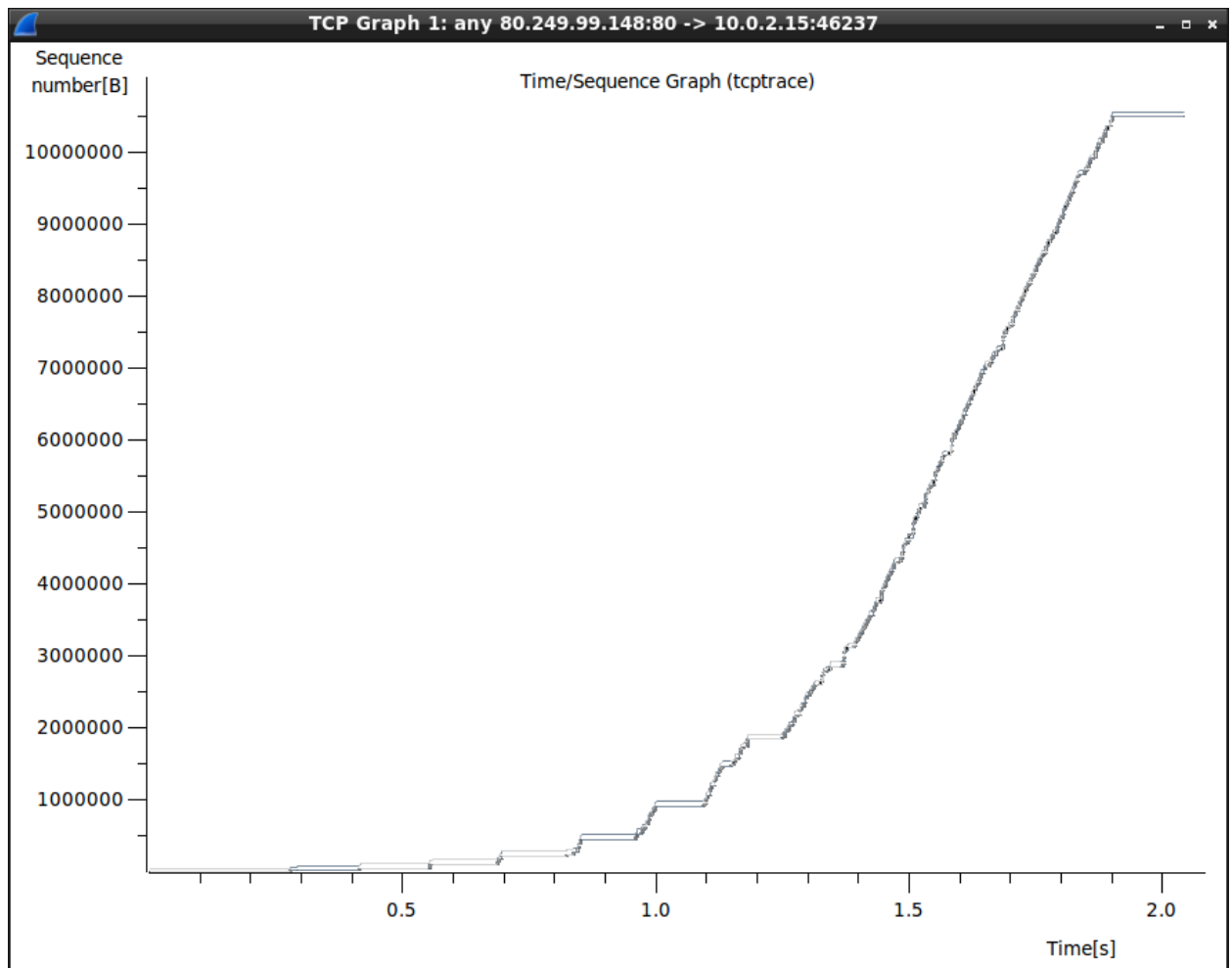


12. [10 pts] Find a packet from the download with a source of the server and a destination of your computer. Create a tcptrace graph with this packet selected. Take a screenshot of the graph and explain what it is showing. Look into the Wireshark documentation if you need assistance making this graph.

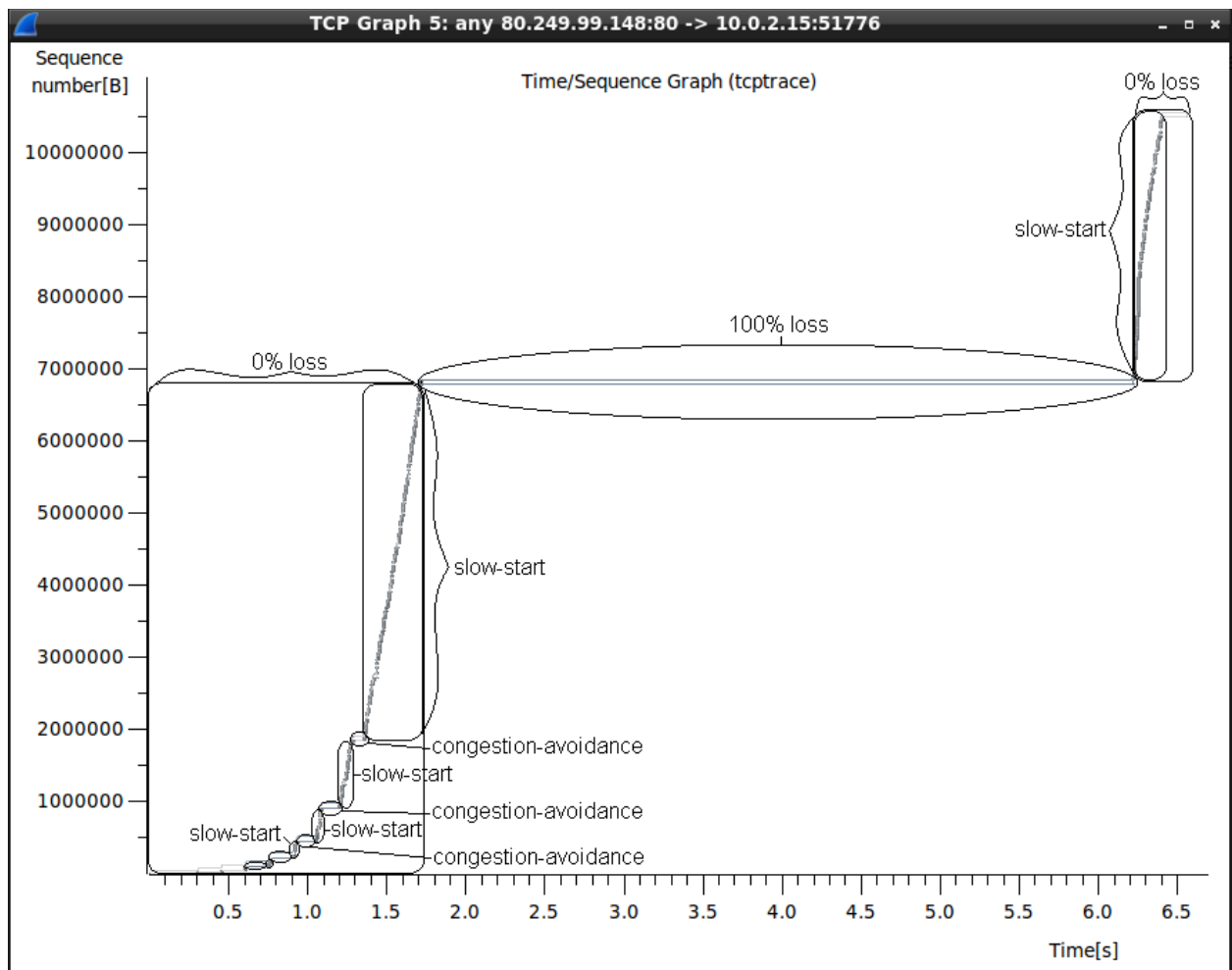
After selecting frame 56 as a suitable candidate (TCP packet), the graph displays the sequence numbers over time for the data transmitted between my computer and the server.

The increasing part of the graph indicates that the transfer rate is increasing, which is expected in TCP connections due to the slow-start mechanism. During this phase, the sender gradually increases the number of packets it sends until it detects congestion or reaches the maximum allowed window size.

Below is a screenshot of the tcptrace graph:



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 editing program, circle the areas where the 0% loss is shown, as well as where TCP is in slow-start and congestion-avoidance.



1. An initial period of 0% loss: The graph shows a steady increase in transfer rate before the loss is set to 100%.
2. Slow-start: During the initial phase of the transfer, TCP will be in the slow-start phase, which is characterized by an increasing line with a steep slope.
3. Congestion-avoidance: The plateaus in the graph are indicative of congestion avoidance. Congestion avoidance is a mechanism used to control the sending rate to avoid overwhelming the network, while slow-start is used to ramp up the sending rate quickly until congestion is detected.
4. Packet loss: After I set the loss rate to 100%, the graph shows a sudden drop in the transfer rate. This is because the sender will reduce its sending rate due to the perceived congestion.
5. Recovery phase: Once I set the loss back to 0%, the graph shows another period of slow-start.