

Lab 2 - DNS and TCP

Total Points: 100

Suggested Resources:

<https://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html>

<http://packetbomb.com/understanding-the-tcp-trace-time-sequence-graph-in-wireshark/>

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

For all the questions below that require a screenshot, make sure that a date timestamp is visible next to your results (you can have a portion of another terminal open with the date command). Make sure to highlight as necessary in the screenshots to get full credit.

Part 1: DNS [50 pts]

In this lab, we'll make extensive use of the command line tool *dig*. To run *dig* in Linux/Unix, you just type the command on the command line. In its most basic operation, the *dig* tool allows the host running the tool to query any specified DNS server for a DNS record. The queried DNS server can be a root DNS server, a top-level-domain DNS server, an authoritative DNS server, or an intermediate DNS server (see the textbook for definitions of these terms).

Example commands:

`dig [domain name] [record type:optional]`

`dig www.google.com`

`dig www.google.com AAAA`

Read man page for more information

A. [35 pts] DNS Warmup

1. [2 pts] In your own words describe DNS resource records (RR) and how they are used by DNS.

DNS resource records are used to define DNS queries' name, value, type, and ttl.

There are different types and are the basic building blocks of host-name and IP information.

2. [3 pts] Run *dig* to find the *MX* resource record of google.com. What command did you use to obtain the MX resource records for the domain? What information does this resource record provide? Which name is easier for you to remember, google.com or the

canonical hostname?

I used the command *dig google.com mx*. The mx record provides the alias of the email domain: google.com and the value is the name of the mail server for google: smtp.google.com. Google.com is easier for me to remember.

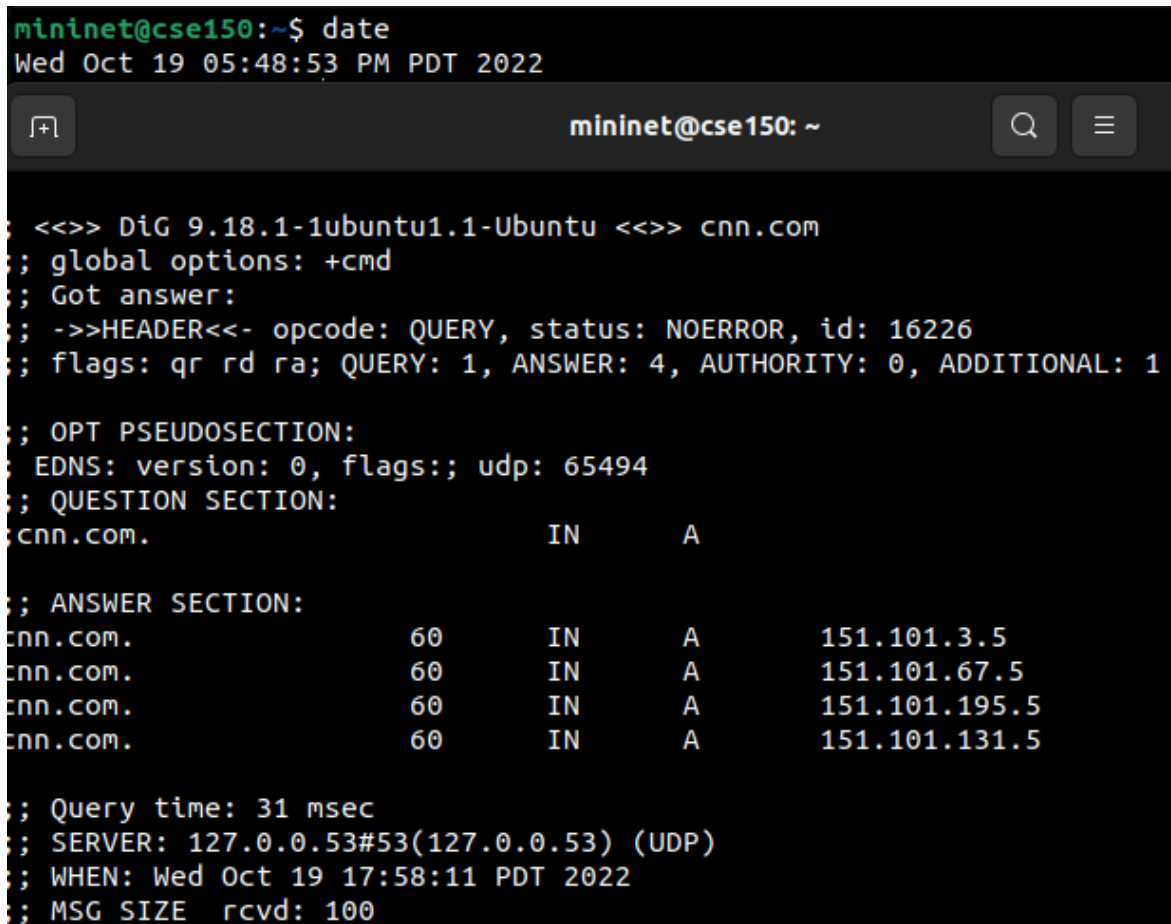
3. [2 pts] Run the command *dig cnn.com* and take a screenshot.

Why is more than one IP address returned?

More than one IP address is returned because the domain hosts multiple sites that could be in different countries or locations.

What type of RR is returned and what is its function?

The RR returned is type "A" and it gives the name as the hostname: cnn.com and the values are the IP addresses.



```
mininet@cse150:~$ date
Wed Oct 19 05:48:53 PM PDT 2022

<<>> DiG 9.18.1-1ubuntu1.1-Ubuntu <<>> cnn.com
; global options: +cmd
; Got answer:
; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 16226
; flags: qr rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 0, ADDITIONAL: 1

; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
; QUESTION SECTION:
cnn.com.                IN      A

; ANSWER SECTION:
cnn.com.                60      IN      A      151.101.3.5
cnn.com.                60      IN      A      151.101.67.5
cnn.com.                60      IN      A      151.101.195.5
cnn.com.                60      IN      A      151.101.131.5

; Query time: 31 msec
; SERVER: 127.0.0.53#53(127.0.0.53) (UDP)
; WHEN: Wed Oct 19 17:58:11 PDT 2022
; MSG SIZE rcvd: 100
```

4. [5 pts] CNAME records are used to map domains to aliases.

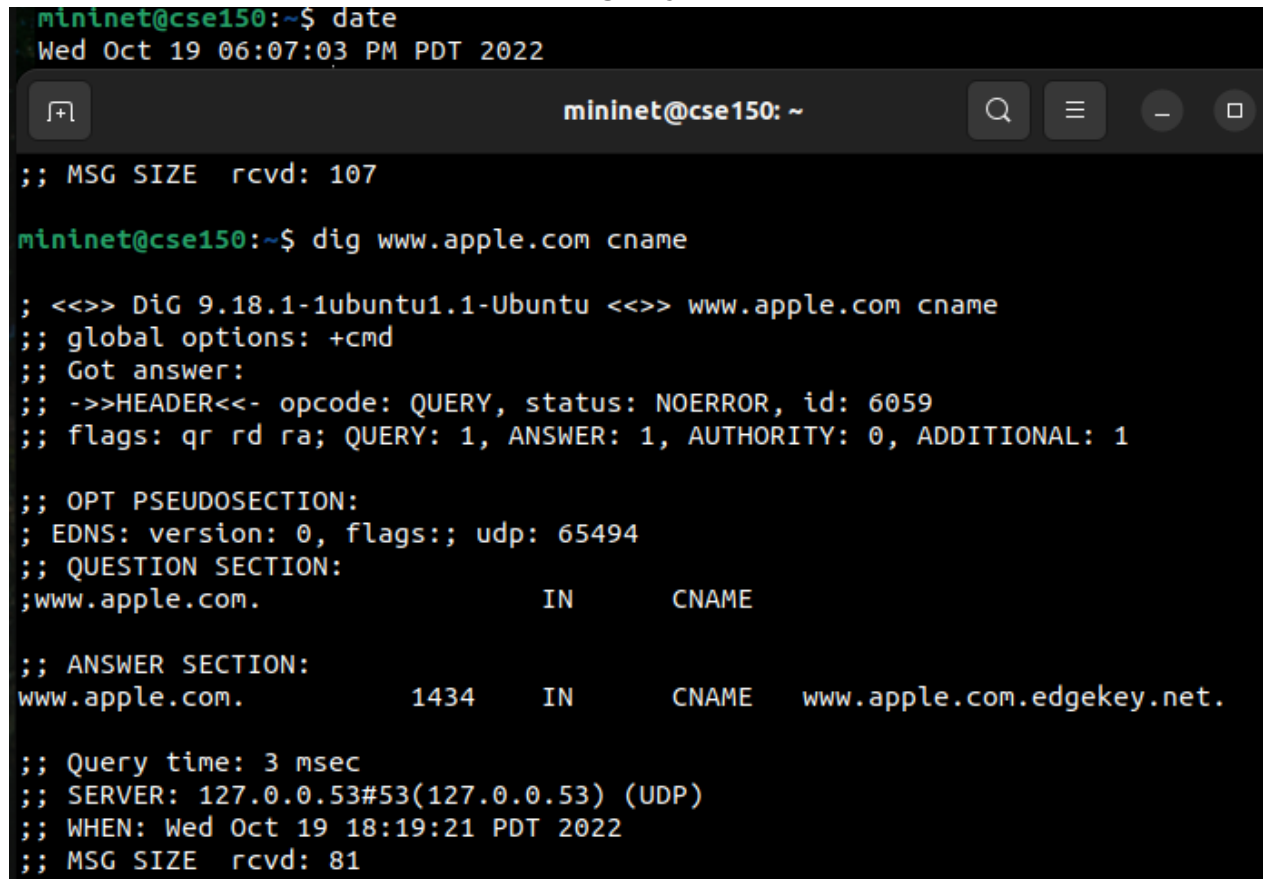
a. Why are domain aliases used on the Internet?

Domain aliases allow hosts to direct users to the same website using multiple domain names.

b. Use *dig* to obtain a CNAME record for a domain of your choice. Take a screenshot and highlight to support your answer. Which name is the *alias*? Which name is the *canonical name*?

The alias name is: www.apple.com

The canonical name is: www.apple.com.edgekey.net



```
mininet@cse150:~$ date
Wed Oct 19 06:07:03 PM PDT 2022

mininet@cse150:~$ dig www.apple.com cname

;; MSG SIZE rcvd: 107

; <<>> DiG 9.18.1-1ubuntu1.1-Ubuntu <<>> www.apple.com cname
;; global options: +cmd
;; Got answer:
;; ->HEADER<<- opcode: QUERY, status: NOERROR, id: 6059
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

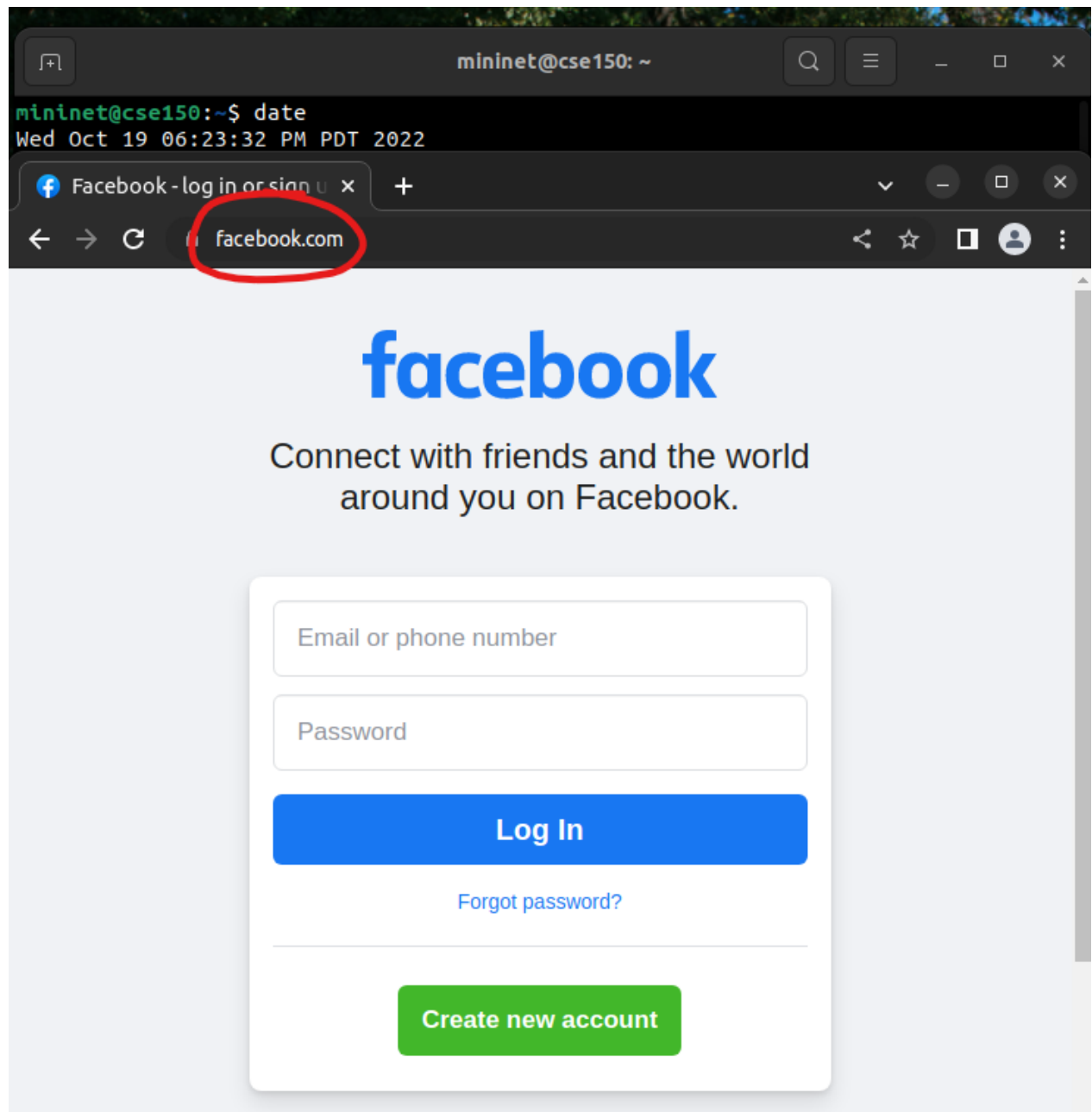
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;www.apple.com.                IN      CNAME

;; ANSWER SECTION:
www.apple.com.                1434    IN      CNAME   www.apple.com.edgekey.net.

;; Query time: 3 msec
;; SERVER: 127.0.0.53#53(127.0.0.53) (UDP)
;; WHEN: Wed Oct 19 18:19:21 PDT 2022
;; MSG SIZE rcvd: 81
```

5. [2 pts] Find another domain name for www.fb.com. What name did you find and how did you determine it? Take a screenshot and highlight to support your answer.

Another domain name is facebook.com. When clicking the first link the name in the browser search bar changed to facebook.com



6. [7 pts] Authoritative name servers

a. What is an Authoritative name server and why is it important?

The Authoritative name server is used to provide answers to DNS queries about where specific websites can be found. It contains the IP address for each domain.

b. Run *dig* to determine the authoritative DNS servers for a university in South America. What is the name of the university you chose? Provide a screenshot highlighting your results (the Authoritative nameserver).

I chose the University of Chile and the Authoritative name server is circled

below.

```
mininet@cse150:~$ date
Fri Oct 21 06:49:02 PM PDT 2022
mininet@cse150:~$ |

mininet@cse150:~$ dig uchile.cl cname

; <<>> DiG 9.18.1-1ubuntu1.1-Ubuntu <<>> uchile.cl cname
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 12369
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;uchile.cl.                IN      CNAME

;; AUTHORITY SECTION:
uchile.cl.                 300     IN      SOA     ns1.uchile.cl. named.uchile.cl.
2018123039 300 60 2419200 300

;; Query time: 268 msec
;; SERVER: 127.0.0.53#53(127.0.0.53) (UDP)
;; WHEN: Fri Oct 21 18:48:22 PDT 2022
;; MSG SIZE rcvd: 84
```

7. [4 pts] Use dig to return 2 different resource records [RR] (not MX or CNAME) of your choice. Explain the RRs returned and what specific information you learned from the record. Take a screenshot and highlight to support your answer.

The RR in the first screenshot is type A, which shows the hostname of the University of Chile in the first column and the IP address in the value column.

The RR in the second screenshot is type NS which shows the domain in the name column and the hostnames of the authoritative name servers in the domain. It seems the University of Chile has multiple hostnames for the authoritative name server.

(SCREENSHOTS BELOW)

```
mininet@cse150: ~  
mininet@cse150:~$ date  
Fri Oct 21 06:49:02 PM PDT 2022  
mininet@cse150:~$  
;; MSG SIZE rcvd: 84  
mininet@cse150:~$ dig uchile.cl a  
;  
;<> DiG 9.18.1-1ubuntu1.1-Ubuntu <> uchile.cl a  
;; global options: +cmd  
;; Got answer:  
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 41575  
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1  
;  
;; OPT PSEUDOSECTION:  
; EDNS: version: 0, flags:; udp: 65494  
;; QUESTION SECTION:  
; uchile.cl. IN A  
;  
;; ANSWER SECTION:  
uchile.cl. 300 IN A 200.89.76.36  
;  
;; Query time: 199 msec  
;; SERVER: 127.0.0.53#53(127.0.0.53) (UDP)  
;; WHEN: Fri Oct 21 19:03:52 PDT 2022  
;; MSG SIZE rcvd: 54
```

```
mininet@cse150: ~
mininet@cse150:~$ date
Fri Oct 21 06:49:02 PM PDT 2022
mininet@cse150:~$ |

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;uchile.cl.                IN      NS

;; ANSWER SECTION:
uchile.cl.                 274     IN      NS      ns1.uchile.cl.
uchile.cl.                 274     IN      NS      ns2.uchile.cl.
uchile.cl.                 274     IN      NS      ns4.uchile.cl.
uchile.cl.                 274     IN      NS      ns5.uchile.cl.

;; ADDITIONAL SECTION:
ns2.uchile.cl.             3573    IN      A        200.89.70.70
ns1.uchile.cl.             3574    IN      A        200.89.70.3
ns5.uchile.cl.             3574    IN      A        35.83.170.25
ns4.uchile.cl.             3574    IN      A        146.83.185.209

;; Query time: 23 msec
;; SERVER: 127.0.0.53#53(127.0.0.53) (UDP)
;; WHEN: Fri Oct 21 19:04:44 PDT 2022
;; MSG SIZE rcvd: 174
```

8. [5 pts] Open Wireshark and listen on the 'any' interface.
Perform a DNS lookup twice using `dig` to `www.example.fr` from a terminal window.
Does the DNS query take the same amount of time to run both times? Provide an explanation for the difference in lookup time.
- From my results, the DNS query took the same amount of time to run both times. The first time was 0.0295 seconds and the second time was 0.0294 seconds.**
9. [5 pts] What are root name servers? What command would we use in order to find the root name servers? Run the command and screenshot the results.
- Root name servers get IP addresses of Top Level Domain servers like a .com DNS server and directly answer requests for records in the root zone.**

B. [15 pts] DNS - digging in with Wireshark

In this section, we will observe how the DNS protocol operates at the packet level. We will do this by using the Mininet VM and Wireshark. Begin by doing the following: •

Open Wireshark and listen on the 'any' interface without any filter

- Open Chromium, clear your web cache and navigate to <http://www2.cs.uh.edu/~gnawali/courses/cosc4377-s12/hw2/http.html>

1. [5 pts] **Transport protocol and Wireshark capture:**

- a. [2 pts] Locate the DNS query and response messages. Are they sent over UDP or TCP? Show a screenshot in Wireshark identifying (circle) the transport layer protocol.

The DNS query and response messages are sent over via UDP.

mininet@cse150:~\$ date
Fri Oct 21 09:46:14 PM PDT 2022

Wireshark interface showing a packet capture on the 'any' interface. The filter is 'dns'. The packet list shows several DNS packets. Packet 7 is highlighted in red, indicating it is the selected packet. The packet details pane for packet 7 shows the following structure:

- Frame 5: 87 bytes on wire (696 bits), 87 bytes captured (696 bits) on interface
- Linux cooked capture v1
- Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.53
- User Datagram Protocol, Src Port: 54460, Dst Port: 53** (circled in red)
- Domain Name System (query)** (circled in red)
 - Transaction ID: 0xb5fd
 - Flags: 0x0120 Standard query
 - Questions: 1
 - Answer RRs: 0
 - Authority RRs: 0
 - Additional RRs: 1
 - Queries
 - www2.cs.uh.edu: type A, class IN
 - Name: www2.cs.uh.edu

- b. [2 pts] What is the reasoning behind this choice of transport layer protocol for this type of messaging?

UDP is faster than TCP, since TCP requires the three-way handshake.
DNS requests are also fairly small and are best suited for UDP.

- c. [1 pt] What is the source port of DNS response messages? How do you know? Show a screenshot in Wireshark and highlight the port.

The DNS response messages use source port 53.

The screenshot shows a terminal window at the top with the command `date` and the output `Fri Oct 21 09:46:14 PM PDT 2022`. Below the terminal is a Wireshark capture window titled `*any`. The filter bar shows `dns`. The packet list table is as follows:

No.	Time	Source	Destination	Protocol
3	0.130034520	127.0.0.1	127.0.0.53	DNS
4	0.130229930	10.0.2.15	192.168.1.254	DNS
5	0.132300415	127.0.0.1	127.0.0.53	DNS
6	0.202577081	192.168.1.254	10.0.2.15	DNS
7	0.202785085	127.0.0.53	127.0.0.1	DNS
8	0.202824917	127.0.0.53	127.0.0.1	DNS
29	0.440702360	127.0.0.1	127.0.0.53	DNS
30	0.440725952	127.0.0.1	127.0.0.53	DNS
31	0.440829841	10.0.2.15	192.168.1.254	DNS
32	0.440881290	10.0.2.15	192.168.1.254	DNS
33	0.440962658	127.0.0.1	127.0.0.53	DNS
34	0.441008961	10.0.2.15	192.168.1.254	DNS
39	0.513579003	192.168.1.254	10.0.2.15	DNS

Packet 6 is selected. The packet details pane shows the following structure:

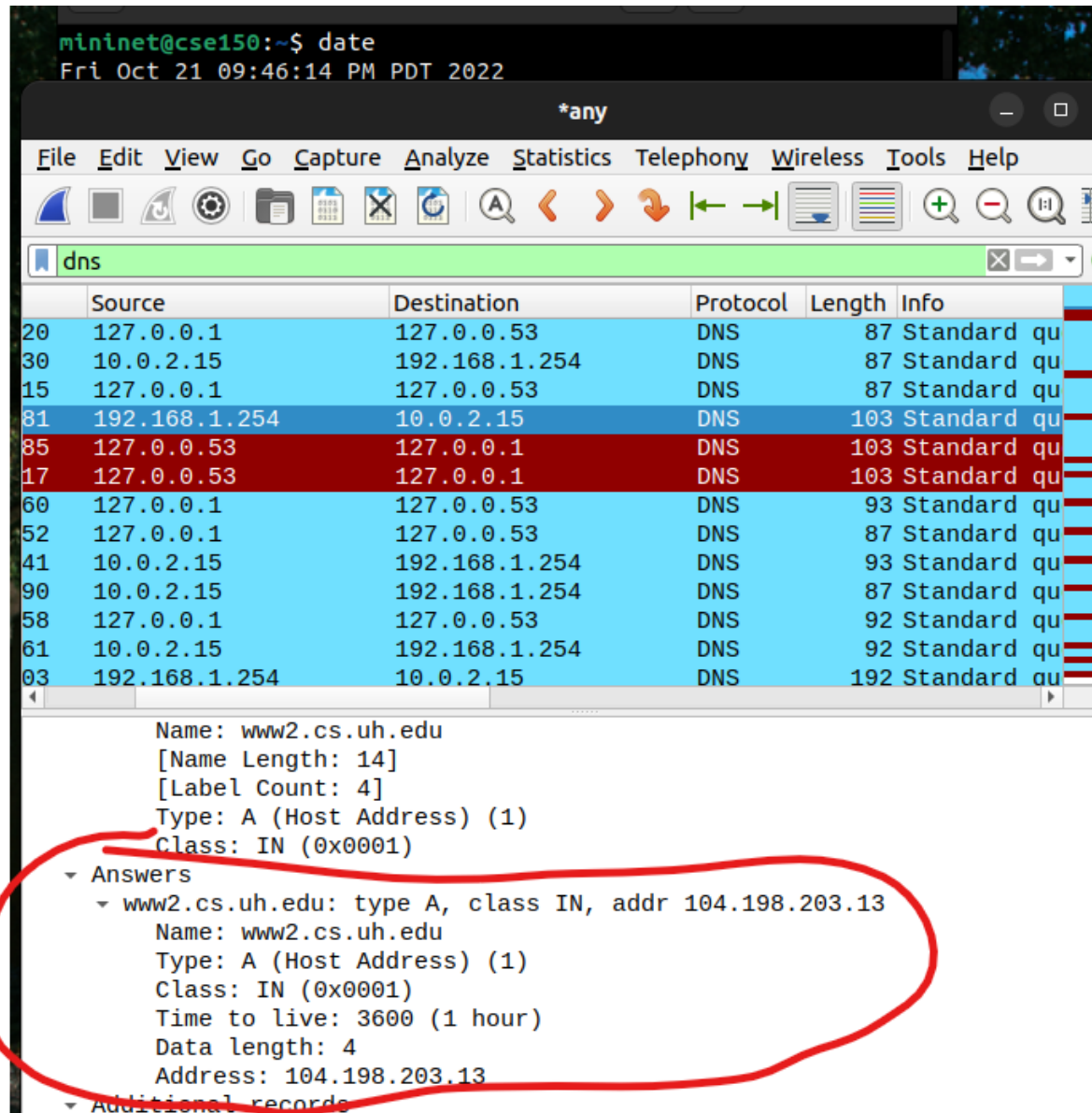
- Frame 6: 103 bytes on wire (824 bits), 103 bytes captured (824 bits) on interface
- Linux cooked capture v1
- Internet Protocol Version 4, Src: 192.168.1.254, Dst: 10.0.2.15
- User Datagram Protocol, Src Port: 53, Dst Port: 51216
- Domain Name System (response)
 - Transaction ID: 0x41ff
 - Flags: 0x8180 Standard query response, No error
 - Questions: 1
 - Answer RRs: 1
 - Authority RRs: 0
 - Additional RRs: 1
 - Queries
 - www2.cs.uh.edu: type A, class IN
 - Name: www2.cs.uh.edu

2. [10 pts] **Loading the webpage and Wireshark capture:**

- a. [3 pts] Examine the DNS response message. How many “answers” are

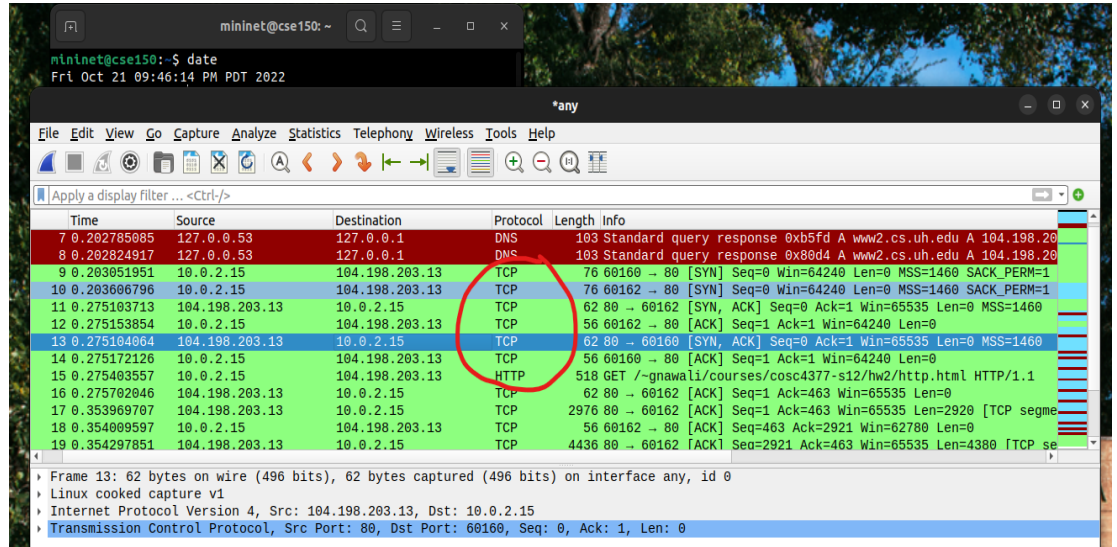
provided? What information is provided in each of these “answers”? Do you see a final answer or only referrals?

In this response message, there is only one answer. This answer gives me the name of the hostname, the type of DNS response, and it’s IP address. This seems like a final answer.



- b. [3 pts] According to Wireshark, after DNS is finally resolved, what are the next steps taken before the HTTP request is sent? Attach a screenshot highlighting these steps.

The client must establish a TCP connection before exchanging an HTTP request/response.



- c. [4 pts] Find the HTTP GET request and highlight it in your screenshot. Find the destination IP address for this request – is it the same IP address that was returned by DNS. Include screenshots verifying the address match.

The destination IP address is the same address that was returned by the DNS response. (SCREENSHOTS BELOW)

```
mininet@cse150:~$ date
Fri Oct 21 09:46:14 PM PDT 2022
```

*any

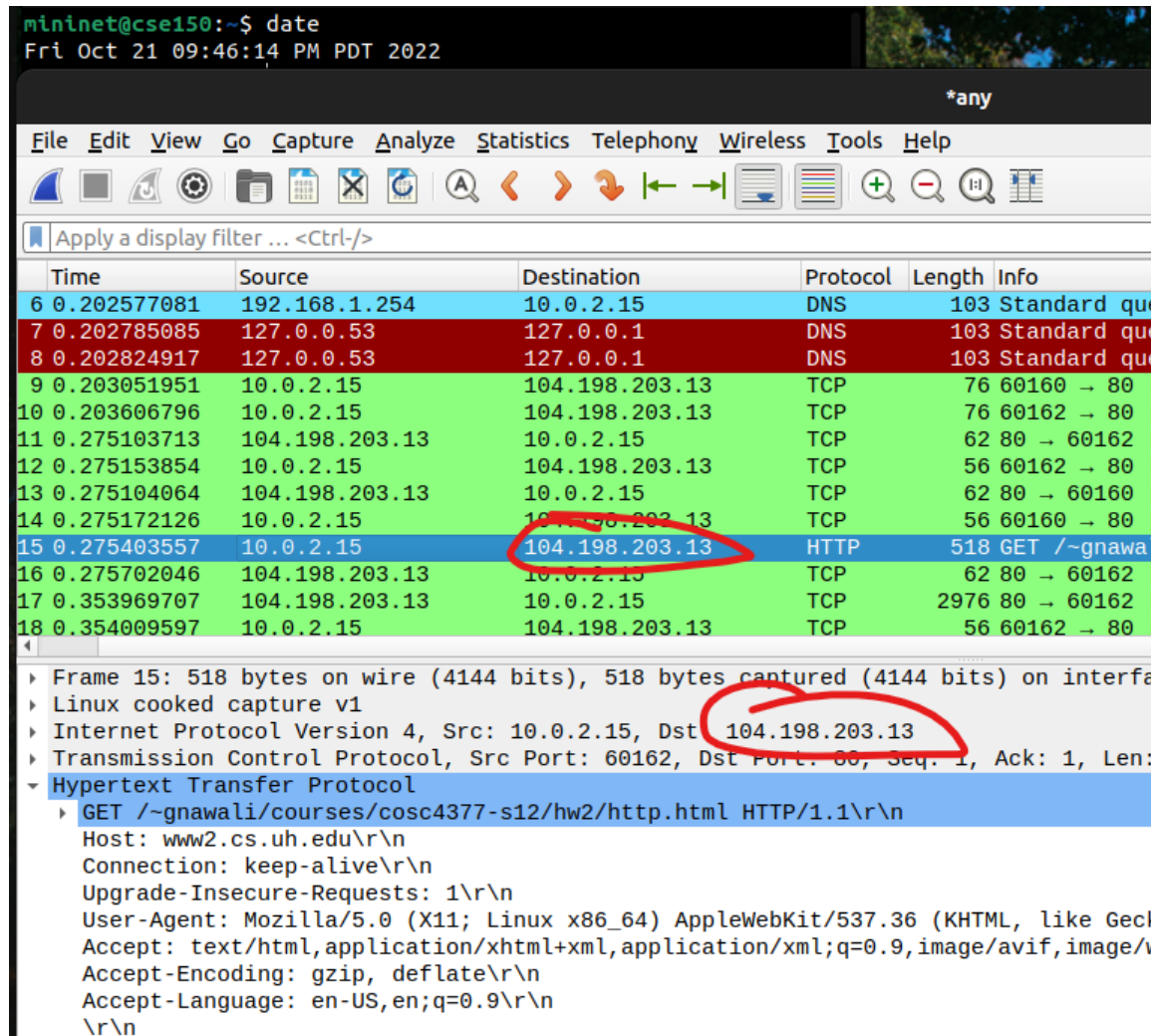
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
3	0.130034520	127.0.0.1	127.0.0.53	DNS	87	Standard query
4	0.130229930	10.0.2.15	192.168.1.254	DNS	87	Standard query
5	0.132300415	127.0.0.1	127.0.0.53	DNS	87	Standard query
6	0.202577081	192.168.1.254	10.0.2.15	DNS	103	Standard query
7	0.202785085	127.0.0.53	127.0.0.1	DNS	103	Standard query
8	0.202824917	127.0.0.53	127.0.0.1	DNS	103	Standard query
9	0.203051951	10.0.2.15	104.198.203.13	TCP	76	60160 → 80 [S
10	0.203606796	10.0.2.15	104.198.203.13	TCP	76	60162 → 80 [S
11	0.275103713	104.198.203.13	10.0.2.15	TCP	62	80 → 60162 [S
12	0.275153854	10.0.2.15	104.198.203.13	TCP	56	60162 → 80 [S
13	0.275104064	104.198.203.13	10.0.2.15	TCP	62	80 → 60160 [S
14	0.275172126	10.0.2.15	104.198.203.13	TCP	56	60160 → 80 [S
15	0.275403557	10.0.2.15	104.198.203.13	HTTP	518	GET /~anawall

Class: IN (0x0001)

- Answers
 - www2.cs.uh.edu: type A, class IN, addr 104.198.203.13
 - Name: www2.cs.uh.edu
 - Type: A (Host Address) (1)
 - Class: IN (0x0001)
 - Time to live: 3600 (1 hour)
 - Data length: 4
 - Address: 104.198.203.13
- Additional records
 - <Root>: type OPT
 - Name: <Root>
 - Type: OPT (41)
 - UDP payload size: 4096
 - Higher bits in extended RCODE: 0x00



Part 2: TCP [50 pts]

A. [35 pts] In this exercise we look at a few ways to transfer files and observe the transfers over Wireshark.

1. [9 pts] In this problem we will download a file using `wget`. Please do the following:

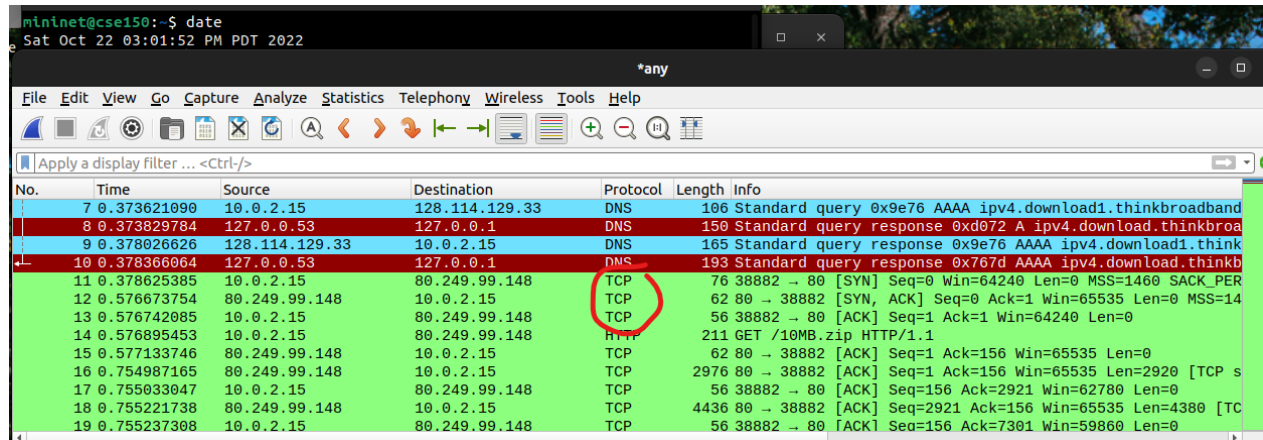
- Open Wireshark and listen on the 'any' interface.
- Open a terminal window. Using `wget`, download this <http://ipv4.download.thinkbroadband.com/10MB.zip>

a. Enumerate and describe all of the different protocols you see in your capture

related to this download. There are

In this capture there are 5 DNS queries and 5 DNS responses. There is one HTTP request/response pair for the request URI: /10MB.zip. The rest of the protocols for each frame is TCP, establishing connections before the HTTP GET request and in between the GET request and response and after the response. There were about 1232 TCP frames.

- b. Find the three-way handshake for TCP connection establishment. Take a screenshot of these packets and highlight all of them in the screenshot.



- c. Calculate the average throughput based on the times in Wireshark. Show all of your work and include screenshots showing values used in your calculation. How does this compare to the MB/s value reported by *wget*?

The total file data length was 10485760 bytes and the time since HTTP request was about 4.83 seconds. That means the throughput was about 2170964.8 bytes/sec, which is about 2.17 MB/sec. The throughput I got from *wget* was 2.16MB/s, so the values were very similar. (SCREENSHOTS BELOW)

```
mininet@cse150:~$ date
Sat Oct 22 03:01:52 PM PDT 2022
```

*any

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

http

No.	Time	Source	Destination	Protocol	Length	Info
14	0.576895453	10.0.2.15	80.249.99.148	HTTP	211	GET /10MB.zip HTTP/1.1
1239	5.402048682	80.249.99.148	10.0.2.15	HTTP	15420	HTTP/1.1 200 OK

Content-Type: application/zip\r\n

Content-Length: 10485760\r\n

Last-Modified: Mon, 02 Jun 2008 15:30:33 GMT\r\n

Connection: keep-alive\r\n

ETag: "48441219-a00000"\r\n

Access-Control-Allow-Origin: *\r\n

Accept-Ranges: bytes\r\n

\r\n

[HTTP response 1/1]

[Time since request: 4.825153223 seconds]

[Request in frame: 141]

[Request URI: http://ipv4.download.thinkbroadband.com/10MB.zip]

File Data: 10485760 bytes



```
mininet@cse150: ~  
mininet@cse150:~$ date  
Sat Oct 22 03:01:52 PM PDT 2022  
mininet@cse150:~$ wget http://ipv4.download.thinkbroadband.com/10MB.zip  
--2022-10-22 14:52:49-- http://ipv4.download.thinkbroadband.com/10MB.zip  
Resolving ipv4.download.thinkbroadband.com (ipv4.download.thinkbroadband.com).  
80.249.99.148  
Connecting to ipv4.download.thinkbroadband.com (ipv4.download.thinkbroadband.com).  
|80.249.99.148|:80... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 10485760 (10M) [application/zip]  
Saving to: '10MB.zip'  
  
10MB.zip          100%[=====>]  10.00M  2.52MB/s   in 4.6s  
2022-10-22 14:52:54 (2.16 MB/s) - '10MB.zip' saved [10485760/10485760]  
mininet@cse150:~$
```

2. [18 pts] Transfer using netcat

In this problem we will use `netcat` to perform a local file transfer of `10MB.zip`. One terminal is set up for listening and the other for sending.

- a. [2 pts] Research the *netcat* tool and briefly describe in your own words what it does.

***Netcat* uses both TCP and UDP connections to read and write in a network, also known as port listening and scanning. You can also transfer files through *netcat*.**

Please do the following and follow the procedure outlined below:

- Open Wireshark to capture the packet transfer/exchange and protocol(s)

The following *netcat* commands accomplish this transfer of the downloaded file:

- On terminal 1 window inside directory 1, run:
`netcat -l 9999 > out.file`
- On terminal 2 window inside directory 2, run:
`nc localhost 9999 < 10MB.zip`

- Press Ctrl-C in terminal 1 after a few seconds if netcat does not terminate on its own.
- b. [2 pts] Add screenshots of the commands and output. Screenshot must show that the file was copied from directory 1 to directory 2. (You can run the `ls -l` command on both the directories along with the `date` command before and after the file is transferred to confirm the transfer.)

(SCREENSHOTS BELOW)

```
mininet@cse150:~/dir2$ cd
mininet@cse150:~$ date
Sat Oct 22 06:53:46 PM PDT 2022
mininet@cse150:~$
```

```
mininet@cse150: ~/dir1
mininet@cse150:~$ cd dir1
mininet@cse150:~/dir1$ netcat -l 9999 > out.file
^C
mininet@cse150:~/dir1$ ls
out.file
mininet@cse150:~/dir1$ ls -l
total 10240
-rw-rw-r-- 1 mininet mininet 10485760 Oct 22 18:50 out.file
mininet@cse150:~/dir1$ netcat -l 9999 > out.file
mininet@cse150:~/dir1$ netcat -l 9999 > out.file
^C
```

```
mininet@cse150: ~/dir2
2022-10-22 18:50:00 (3.26 MB/s) - '10MB.zip' saved [10485760/10485760]

mininet@cse150:~/dir2$ nc localhost 9999 < 10MB.zip
mininet@cse150:~/dir2$ nc localhost 9999 < 10MB.zip
^C
mininet@cse150:~/dir2$ get http://ipv4.download.thinkbroadband.com/10MB.zip
Command 'get' not found, but there are 18 similar ones.
mininet@cse150:~/dir2$ wget http://ipv4.download.thinkbroadband.com/10MB.zip
--2022-10-22 18:52:43-- http://ipv4.download.thinkbroadband.com/10MB.zip
Resolving ipv4.download.thinkbroadband.com (ipv4.download.thinkbroadband.com)...
80.249.99.148
Connecting to ipv4.download.thinkbroadband.com (ipv4.download.thinkbroadband.com)
|80.249.99.148|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 10485760 (10M) [application/zip]
Saving to: '10MB.zip'

10MB.zip          100%[=====>] 10.00M  3.32MB/s   in 3.0s

2022-10-22 18:52:47 (3.32 MB/s) - '10MB.zip' saved [10485760/10485760]

mininet@cse150:~/dir1$ nc localhost 9999 < 10MB.zip
mininet@cse150:~/dir2$
```

```
mininet@cse150:~$ date
Sat Oct 22 06:55:53 PM PDT 2022
```

```
mininet@cse150: ~/dir2
mininet@cse150:~/dir2$ nc localhost 9999 < 10MB.zip
mininet@cse150:~/dir2$ nc localhost 9999 < 10MB.zip
^C
mininet@cse150:~/dir2$ get http://ipv4.download.thinkbroadband.com/
Command 'get' not found, but there are 18 similar ones.
mininet@cse150:~/dir2$ wget http://ipv4.download.thinkbroadband.com/
--2022-10-22 18:52:43-- http://ipv4.download.thinkbroadband.com/10
Resolving ipv4.download.thinkbroadband.com (ipv4.download.thinkbroa
80.249.99.148
Connecting to ipv4.download.thinkbroadband.com (ipv4.download.think
)|80.249.99.148|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 10485760 (10M) [application/zip]
Saving to: '10MB.zip'

10MB.zip          100%[=====>] 10.00M  3.32MB/s

2022-10-22 18:52:47 (3.32 MB/s) - '10MB.zip' saved [10485760/104857
mininet@cse150:~/dir2$ nc localhost 9999 < 10MB.zip
mininet@cse150:~/dir2$ ls -l
total 10240
-rw-rw-r-- 1 mininet mininet 10485760 Jun  2  2008 10MB.zip
```

```
mininet@cse150:~$ date
Sat Oct 22 06:55:53 PM PDT 2022

mininet@cse150: ~/dir1
Terminal
mininet@cse150:~$ cd dir1
mininet@cse150:~/dir1$ netcat -l 9999 > out.file
^C
mininet@cse150:~/dir1$ ls
out.file
mininet@cse150:~/dir1$ ls -l
total 10240
-rw-rw-r-- 1 mininet mininet 10485760 Oct 22 18:50 out.file
mininet@cse150:~/dir1$ netcat -l 9999 > out.file
mininet@cse150:~/dir1$ netcat -l 9999 > out.file
^C
mininet@cse150:~/dir1$ ls-l
ls-l: command not found
mininet@cse150:~/dir1$ ls -l
total 10240
-rw-rw-r-- 1 mininet mininet 10485760 Oct 22 18:53 out.file
mininet@cse150:~/dir1$
```

c. [2 pts] Explain this data transfer in your own words.

The file was downloaded in directory 2. The first command in dir1 was to setup the directory to receive the contents of the file and store it in out.file. The second command in dir2 was to start the transfer of data to dir1 out.file. The ls -l command shows the transfer was successful as the out.file as the same memory size as the original file in dir2.

d. [3 pts] What transport layer protocol does *netcat* use to transfer the file? Why is this transport protocol an apt choice? Verify in your Wireshark capture with appropriate screenshot, markup and explanation.

Netcat uses TCP to transfer the file. TCP is an apt choice because it is far more reliable than UDP. It is connection-oriented and basically guarantees all the data to transfer correctly. It is useful in transferring large file sizes.

e. [4 pts] Choose one of the packets transferred and drill down to find the source and destination IP addresses. What are the addresses? Do the addresses make sense? Explain and show your Wireshark capture to verify.

The source and destination IP addresses are both 127.0.0.1 on the loopback

interface. This makes sense as the files are being transferred locally.

mininet@cse150: ~

```
mininet@cse150:~$ date
Sat Oct 22 07:06:16 PM PDT 2022
```

*any

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	TCP	76	38286 → 9999
2	0.000017384	127.0.0.1	127.0.0.1	TCP	76	9999 → 38286
3	0.000030095	127.0.0.1	127.0.0.1	TCP	68	38286 → 9999
4	0.000091500	127.0.0.1	127.0.0.1	TCP	16452	38286 → 9999
5	0.000097582	127.0.0.1	127.0.0.1	TCP	68	9999 → 38286
6	0.000110892	127.0.0.1	127.0.0.1	TCP	16452	38286 → 9999
7	0.000134548	127.0.0.1	127.0.0.1	TCP	32809	38286 → 9999
8	0.000529705	127.0.0.1	127.0.0.1	TCP	68	9999 → 38286
9	0.005451979	127.0.0.1	127.0.0.1	TCP	68	[TCP Window ...]
10	0.005460514	127.0.0.1	127.0.0.1	TCP	32809	38286 → 9999
11	0.005466253	127.0.0.1	127.0.0.1	TCP	32809	38286 → 9999
12	0.005486917	127.0.0.1	127.0.0.1	TCP	68	9999 → 38286
13	0.005523645	127.0.0.1	127.0.0.1	TCP	68	[TCP Window ...]

Frame 5: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface any, id 0

Linux cooked capture v1

Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

Transmission Control Protocol, Src Port: 9999, Dst Port: 38286, Seq: 1, Ack: 16385, Len: 0

Source Port: 9999

Destination Port: 38286

[Stream index: 0]

[Conversation completeness: Complete, WITH_DATA (31)]

[TCP Segment Len: 0]

Sequence Number: 1 (relative sequence number)

Sequence Number (raw): 3652708747

[Next Sequence Number: 1 (relative sequence number)]

Acknowledgment Number: 16385 (relative ack number)

Acknowledgment number (raw): 3435114291

1000000000 = Header Length: 32 bytes (8)

- f. [5 pts] During the *netcat* transfer, the client transmits some TCP segments with a len=0. What is the purpose of these packets and why are they needed? If you do not see any of these ACKs, please contact your TA in lab.

The purpose of these TCP segments is to let the server know that the data has been transmitted successfully. No data is necessarily being transmitted in this packet.

3. [3 pts] Compare *netcat* with the *cp* command (for example by reading the *man* documentation) and respond:
- a. [1.5 pts] What is the main difference between *netcat* and *cp*?

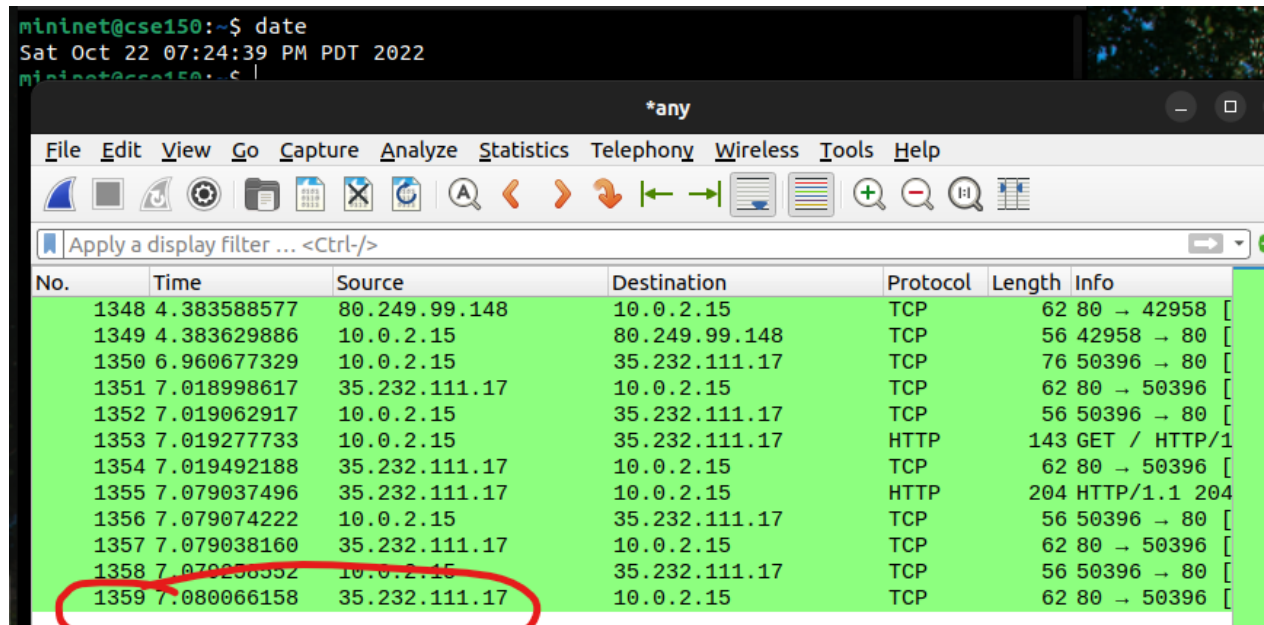
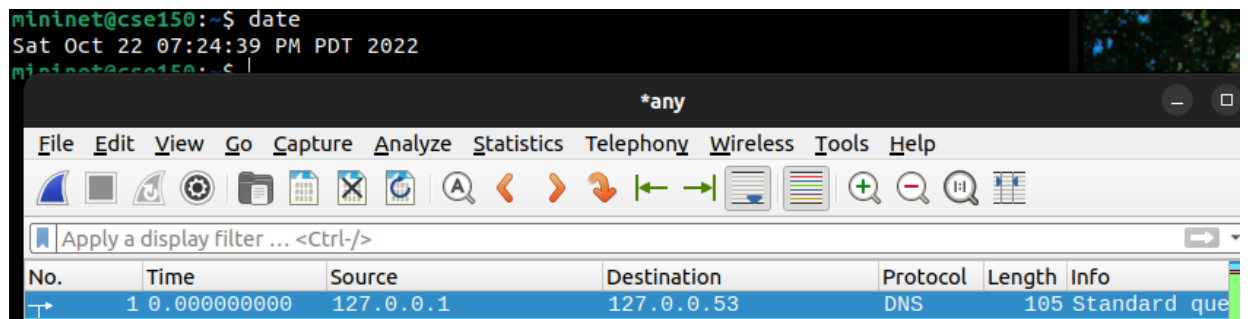
The netcat command is mainly used for anything involving TCP, UDP, or Unix-domain sockets. The cp command is used to copy local files and directories.

- b. [1.5 pts] Which command is more appropriate for copying files within a local device? Why?

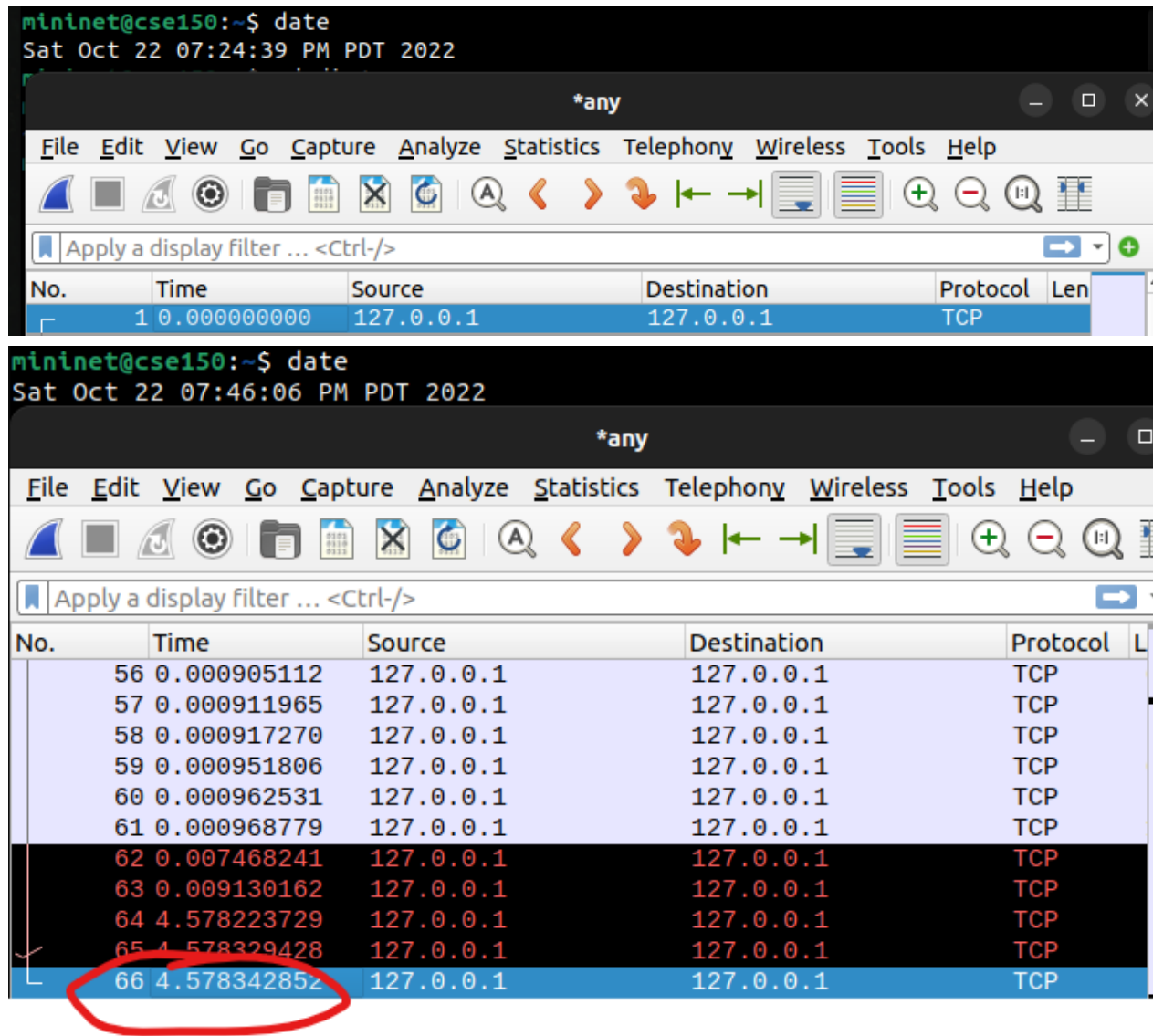
The cp command would be more appropriate for copying files within a local device as it is much more simpler and wouldn't normally require a socket connection since the transfer is not happening over the internet.

4. [5 pts] Compare in Wireshark the following download times:

- Download time of 10MB.zip using wget (from the time the transfer was initiated until it the connection was closed) [Question 2.A.1]



- Transfer time of 10MB.zip using netcat [Question 2.2.a]



What were the total times for wget and for netcat? Explain the difference. Support your answer with screenshots and highlighting the first and the last packet for each command.

Wget was about 7 seconds and netcat took about 4 seconds. Netcat was much faster because the file didn't need to be downloaded over the internet. It was just transferred locally.

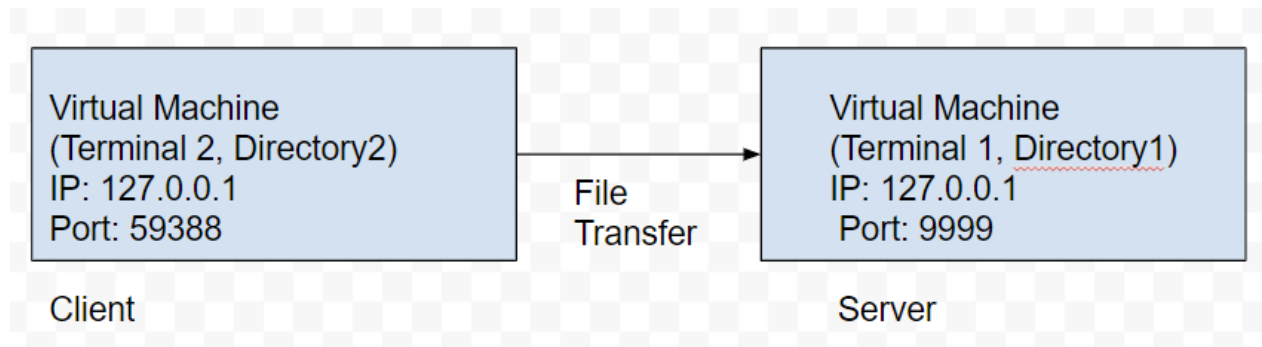
B. [15 pts] Networking meets art – Let's draw what's going on!

1. Netcat transfer: Draw a simple Client-Server architecture overview of the file transfer using Netcat. Show the following elements and the interaction involved in the transfer: a.

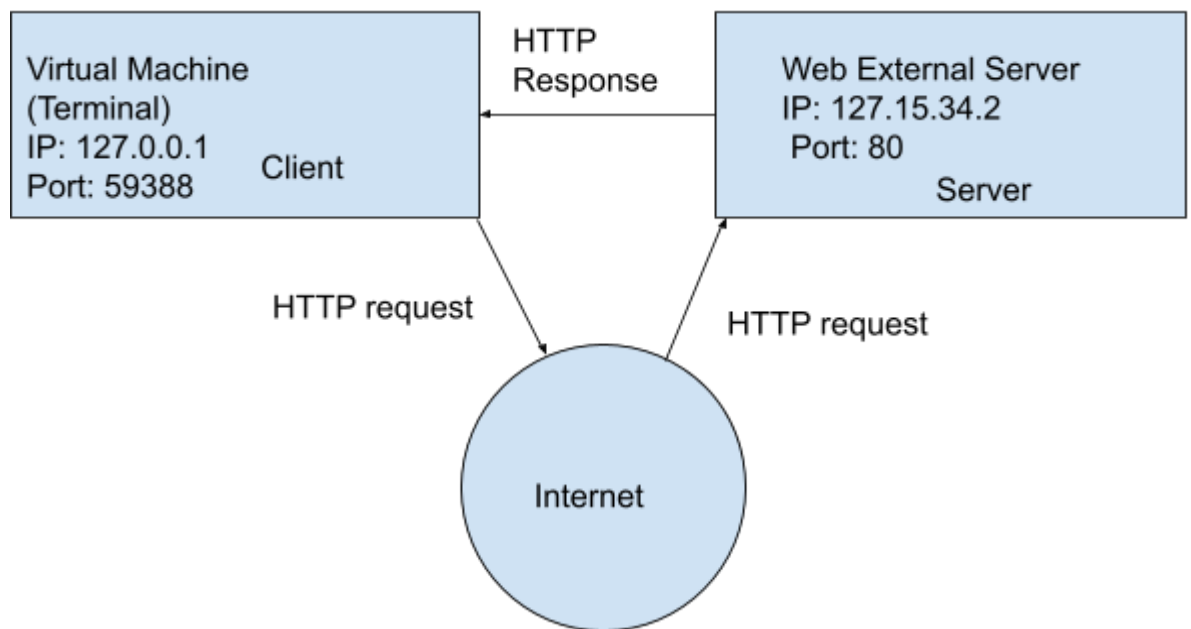
The hardware and software components (clients, servers, VM, etc).

b. All IP addresses and ports. They should be labeled in the drawing. You can refer to the Wireshark capture to find out the source/destination IP addresses. Remember

that the client and server were set up inside your own VM, so start your drawing with a box representing the VM and label it "Virtual Machine".



2. Web transfer: Suppose now the client requests a web page from an web external server with IP address 127.15.34.2.
- Add this server to your drawing, including the additional intermediate elements (like internet connection). Show the port number that the server would be listening on.
 - Indicate in your drawing the communication between the client and server.



You can update your drawing above to show the new communication (don't need a second drawing).

Note: refer to your class notes for sample drawings containing a simple network, with clients and server. Additionally, hand made drawings are not acceptable.

Extra Credit [5 pts]:

An ACK is said to be “piggybacked” when it is sent alongside a TCP segment containing a payload. Screenshot an example of piggybacking from either the `wget` or `netcat` download, highlighting the relevant TCP segment information. Provide an argument as to why this constitutes piggybacking.

Submission:

You will submit 1 file for this assignment directly on Canvas: a PDF with all of your solutions to the questions.

Naming convention: name the PDF file [YourCruzID].pdf.