Andre Hei Wang Law

4017 5600

COEN 366 – FL-X

Mininet Assignment 1 + Wireshark Assignment 2

1. Mininet Assignment 1

Question 1: Give a brief explanation about each topology mentioned above. (In your own words)

Single: A single topology is a basic topology with N hosts connected to one switch.

Reversed: This topology is similar to the single topology, since all N hosts are connected to only one switch. However, the difference lies in the reverse order of connections between the N hosts and the switches.

Linear: A linear topology have N switches and N hosts connected in a linear fashion forming a sequence with each switch connected to the one in the previous line.

Tree: A tree topology, as its name suggest, is a multilevel hierarchical tree where there is a starting node that branches off with each new generation child resulting in a new level, this a N level topology. In addition, this topology has two hosts per switch (two child per node).

Question 2: Using the CLI, generate the above topologies with the following nodes:

Single: 10 Hosts

```
vboxuser@ubu22to:~/Desktop$ sudo mn --topo single,10
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1) (h5, s1) (h6, s1) (h7, s1) (h8, s1) (h9, s1) (h10, s1)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> nodes
available nodes are:
c0 h1 h10 h2 h3 h4 h5 h6 h7 h8 h9 s1
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10
h10 -> h1 h2 h3 h4 h5 h6 h7 h8 h9
*** Results: 0% dropped (90/90 received)
mininet> exit
*** Stopping 1 controllers
c0
*** Stopping 10 links
..........
*** Stopping 1 switches
s1
*** Stopping 10 hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10
*** Done
completed in 16.720 seconds
vboxuser@ubu22to:~/Desktop$
```

Reversed: 10 Hosts

```
vboxuser@ubu22to:~/Desktop$ sudo mn --topo reversed,10
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1) (h5, s1) (h6, s1) (h7, s1) (h8, s1) (h9, s1) (h10, s1)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> nodes
available nodes are:
c0 h1 h10 h2 h3 h4 h5 h6 h7 h8 h9 s1
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10
h10 -> h1 h2 h3 h4 h5 h6 h7 h8 h9
*** Results: 0% dropped (90/90 received)
mininet> exit
*** Stopping 1 controllers
c0
*** Stopping 10 links
..........
*** Stopping 1 switches
s1
*** Stopping 10 hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10
*** Done
completed in 15.446 seconds
vboxuser@ubu22to:~/Desktop$
```

Linear: 10 Switches

```
vboxuser@ubu22to:~/Desktop$ sudo mn --topo linear,10
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7 s8 s9 s10
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (h4, s4) (h5, s5) (h6, s6) (h7, s7) (h8, s8) (h9, s9) (h10,
s10) (s2, s1) (s3, s2) (s4, s3) (s5, s4) (s6, s5) (s7, s6) (s8, s7) (s9, s8) (s10, s9)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10
*** Starting controller
c0
*** Starting 10 switches
s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 ...
*** Starting CLI:
mininet> nodes
available nodes are:
c0 h1 h10 h2 h3 h4 h5 h6 h7 h8 h9 s1 s10 s2 s3 s4 s5 s6 s7 s8 s9
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10
h10 -> h1 h2 h3 h4 h5 h6 h7 h8 h9
*** Results: 0% dropped (90/90 received)
mininet> exit
*** Stopping 1 controllers
c0
*** Stopping 19 links
...................
*** Stopping 10 switches
s1 s2 s3 s4 s5 s6 s7 s8 s9 s10
*** Stopping 10 hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10
*** Done
completed in 12.087 seconds
vboxuser@ubu22to:~/Desktop$
```

Tree: 3 Switches and 4 Hosts

```
vboxuser@ubu22to:~/Desktop$ sudo mn --topo tree,depth=2,fanout=2
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3
*** Adding links:
(s1, s2) (s1, s3) (s2, h1) (s2, h2) (s3, h3) (s3, h4)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
mininet> nodes
available nodes are:
c0 h1 h2 h3 h4 s1 s2 s3
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
mininet> exit
*** Stopping 1 controllers
c0
*** Stopping 6 links
......
*** Stopping 3 switches
s1 s2 s3
*** Stopping 4 hosts
h1 h2 h3 h4
*** Done
completed in 8.968 seconds
vboxuser@ubu22to:~/Desktop$
```

Question 3: Write a python script that generates the following topology

1. In every topology there should be controller. Make sure to add a controller c0 in your code.

```
 9          # add controller
10          c0 = net.addController('c0')
```

2. You must use the Mid-Level APIs to create the given topology.

```
 6          # mininet network object
 7          net = Mininet(controller=Controller, switch=OVSKernelSwitch)
 8
 9          # add controller
10          c0 = net.addController('c0')
11
12          # add the 3 hosts
13          h1 = net.addHost('h1')
14          h2 = net.addHost('h2')
15          h3 = net.addHost('h3')
16
17          # add the 3 switches
18          s1 = net.addSwitch('s1')
19          s2 = net.addSwitch('s2')
20          s3 = net.addSwitch('s3')
21
22          # links logic:
23          # s1-s2-s3 all connected together in triangle connection
24          # s1-h1, s2-h2, s3-h3 connections
25          net.addLink(s1, s2)
26          net.addLink(s2, s3)
27          net.addLink(s3, s1)
28          net.addLink(h1, s1)
29          net.addLink(h2, s2)
30          net.addLink(h3, s3)
```

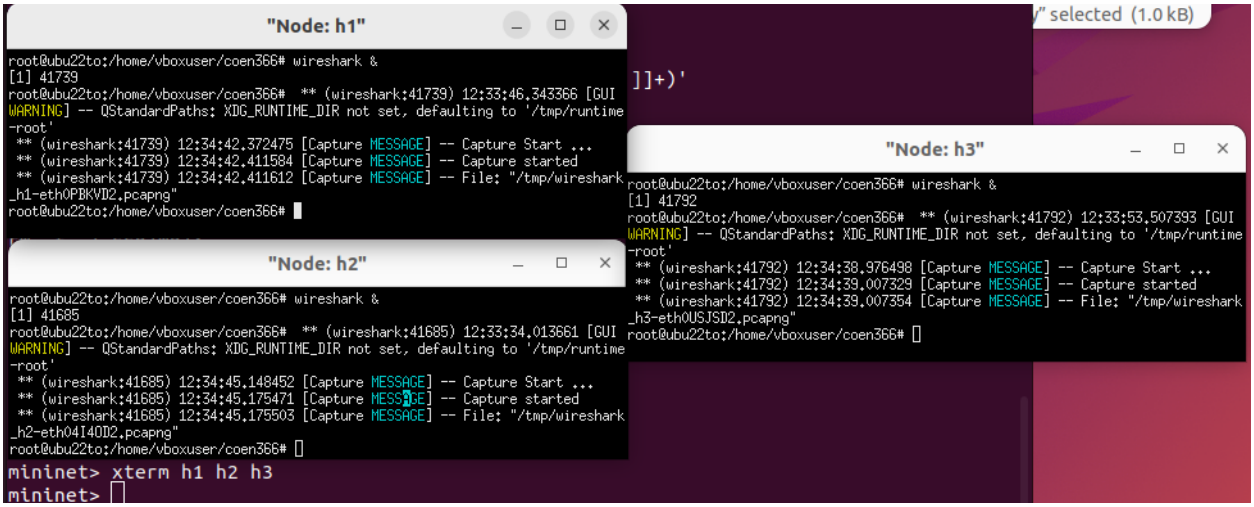
3. Use the CLI object to open the CLI of Mininet

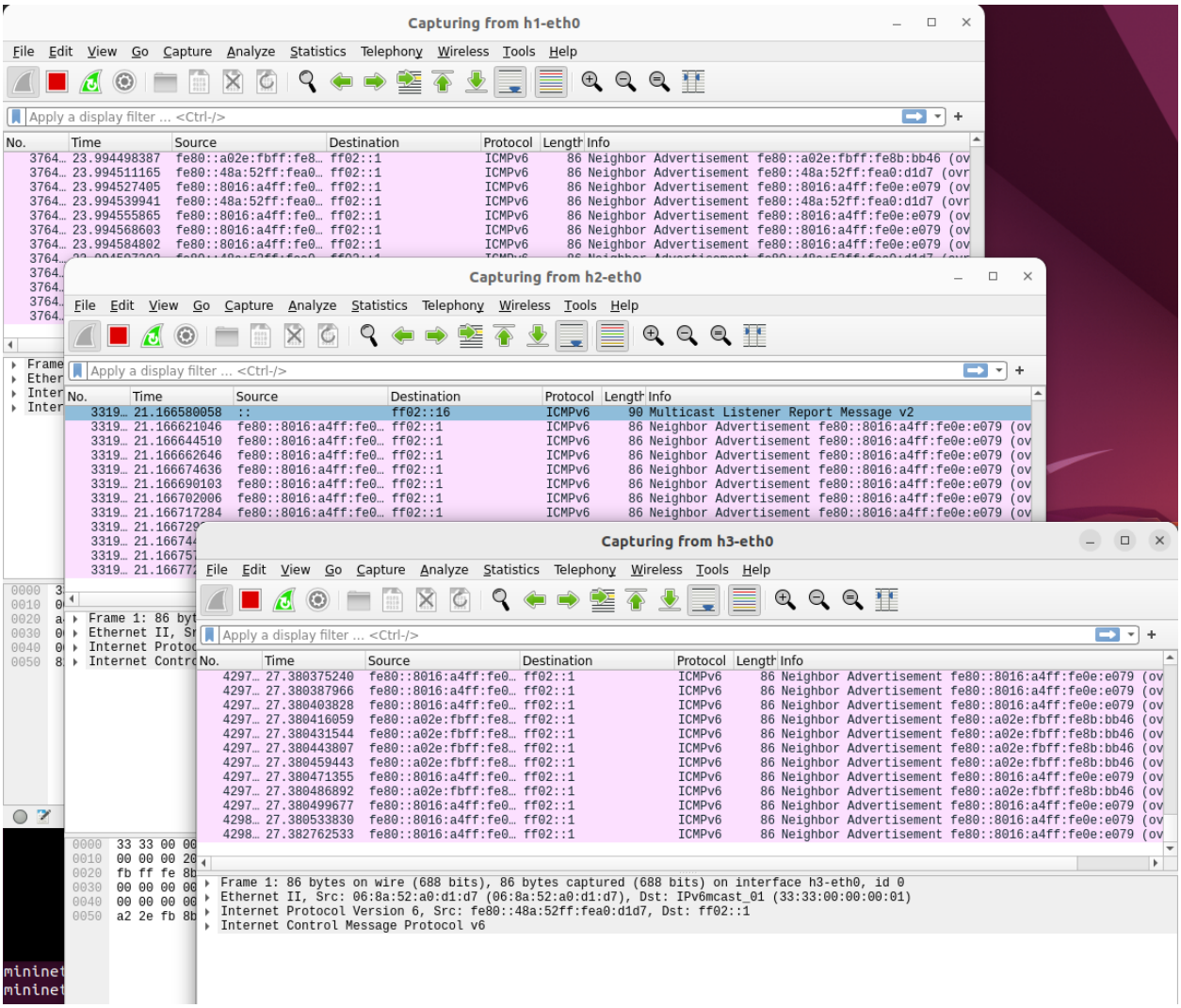
```
42          # open cli mininet
43          CLI(net)
```

4. Run the nodes and show the result. (Provide screenshot)

```
vboxuser@ubu22to:~/coen366$ sudo python3 topo.py
mininet> nodes
available nodes are:
c0 h1 h2 h3 s1 s2 s3
mininet>
```

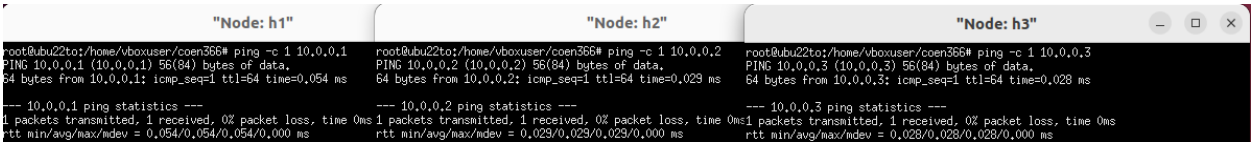
5. Run Wireshark on h1, h2 and on h3.



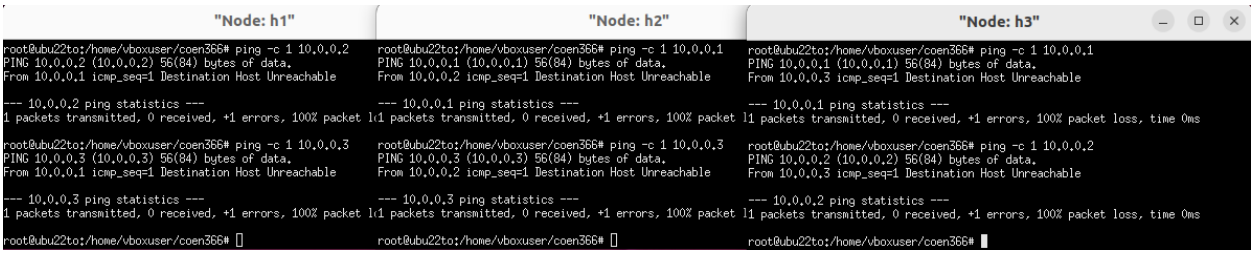


6. Ping h1 from h3 (one packet), how long did it take to ping? Ping h3 from h1. How long does it take? Is there a difference? Capture the incoming packet in Wireshark and provide a screenshot of it. These will help the TAs validate that your submission is working correctly on your machine if for some reason your submitted code does not work in the grading environment.

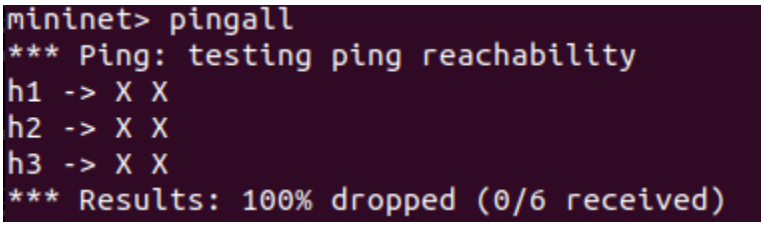
I was not able to have a valid ping connection between h1, h2 and h3. I was only able to capture the ping of itself such as h1 to h1, h2 to h2 and h3 to h3 in which their time is 0.054ms, 0.029ms and 0.028ms respectively.



7. Repeat Step 6 for h1-h2 and h2-h3.



8. Perform a Pingall and copy the output, verifying that all hosts are pingable.



From running “pingall” command, I know the issues lies in the code where it has a network connectivity problem. The pings are failing due to the host unable to reach each and other. I have tried setting IP addresses to each host such as 10.0.0.1 for h1, but it still won’t work.

9. Submit your .py file along with your assignment document.

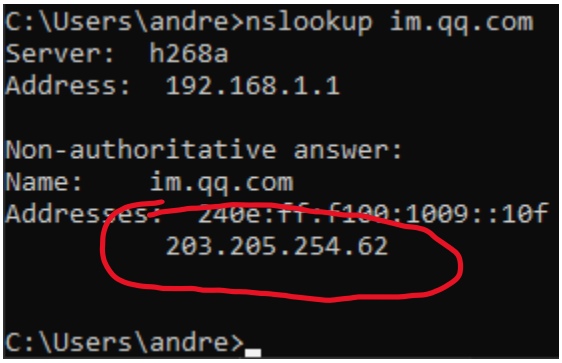
The “topo.py” code is copy and paste below as well as attached to the final submission document.

“topo.py” Code

```
from mininet.net import Mininet
from mininet.node import Controller, OVSKernelSwitch
from mininet.cli import CLI

def topo():
      # mininet network object
      net = Mininet(controller=Controller, switch=OVSKernelSwitch)

      # add controller, hosts and switches
      c0 = net.addController('c0')
      h1 = net.addHost('h1')
      h2 = net.addHost('h2')
      h3 = net.addHost('h3')
      s1 = net.addSwitch('s1')
      s2 = net.addSwitch('s2')
      s3 = net.addSwitch('s3')

      # links logic:
      # s1-s2-s3 all connected together in triangle connection
      # s1-h1, s2-h2, s3-h3 connections
      net.addLink(s1, s2)
      net.addLink(s1, s2)
      net.addLink(s2, s3)
      net.addLink(h1, s1)
      net.addLink(h2, s2)
      net.addLink(h3, s3)

      # begin network
      net.build()
      c0.start()
      s1.start([c0])
      s2.start([c0])
      s3.start([c0])
      h1.cmd('ifconfig h1-eth0 10.0.0.1/24')
      h2.cmd('ifconfig h2-eth0 10.0.0.2/24')
      h3.cmd('ifconfig h3-eth0 10.0.0.3/24')

      # open cli mininet
      CLI(net)

      # stop network
      net.stop()

if __name__ == '__main__':
      topo()
```

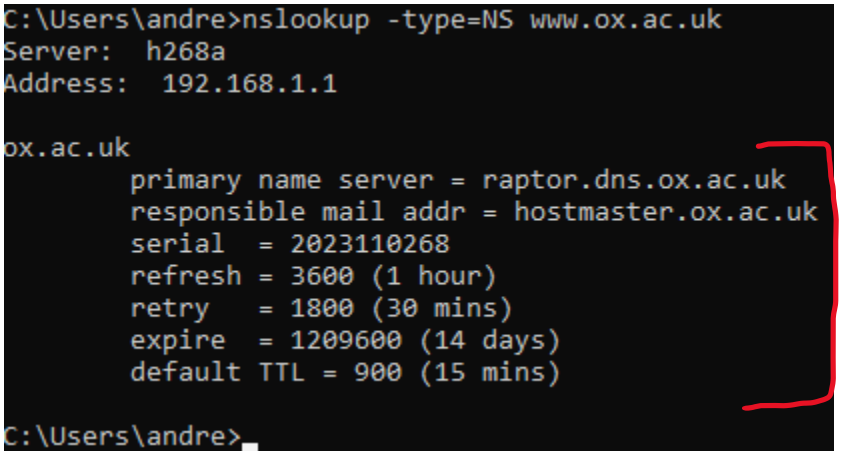
2. **Wireshark Assignment 2**

1. Run nslookup to obtain the IP address of a Web server in Asia. What is the IP address of that server?



I chose https://im.qq.com/ as the website of choice which has an IP address of 203.205.254.62.

2. Run nslookup to determine the authoritative DNS servers for a university in Europe.

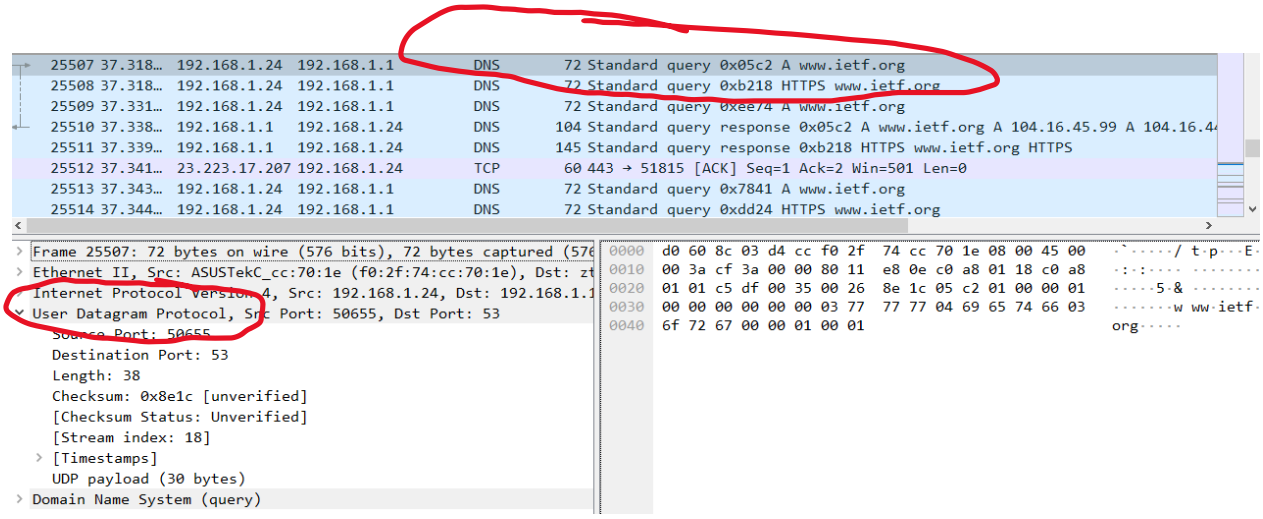


3. Run nslookup so that one of the DNS servers obtained in Question 2 is queried for the mail servers for Yahoo! mail. What is its IP address?

```
C:\Users\andre>nslookup www.ox.ac.uk mail.yahoo.com
DNS request timed out.
    timeout was 2 seconds.
Server:  UnKnown
Address:  69.147.92.11

DNS request timed out.
    timeout was 2 seconds.
DNS request timed out.
    timeout was 2 seconds.
DNS request timed out.
    timeout was 2 seconds.
DNS request timed out.
    timeout was 2 seconds.
*** Request to UnKnown timed-out

C:\Users\andre>_
```

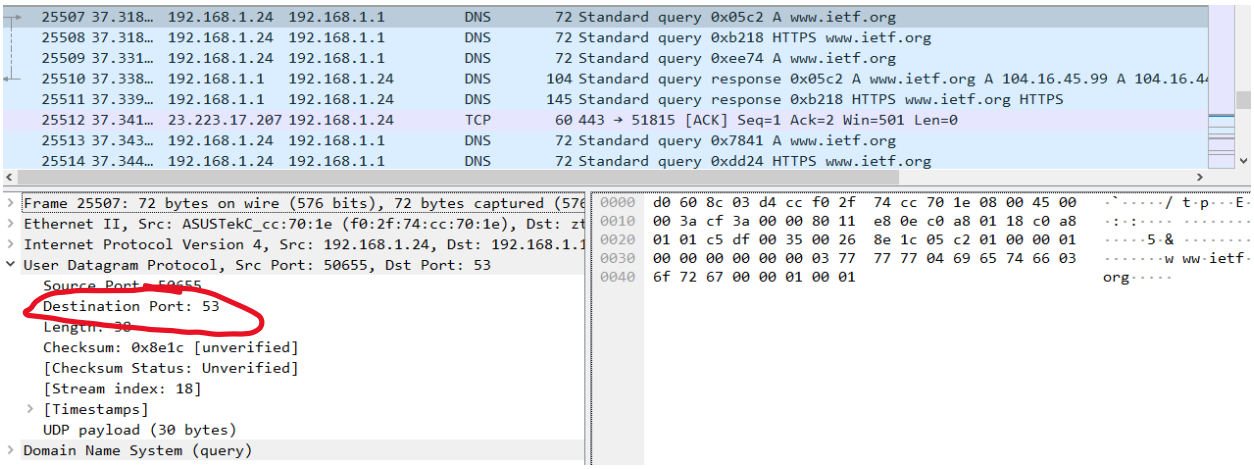
The IP address is 69.147.92.11.

4. Locate the DNS query and response messages. Are then sent over UDP or TCP?



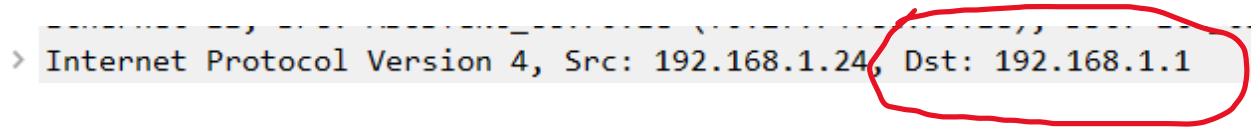
They are sent over User Datagram Protocol (UDP).

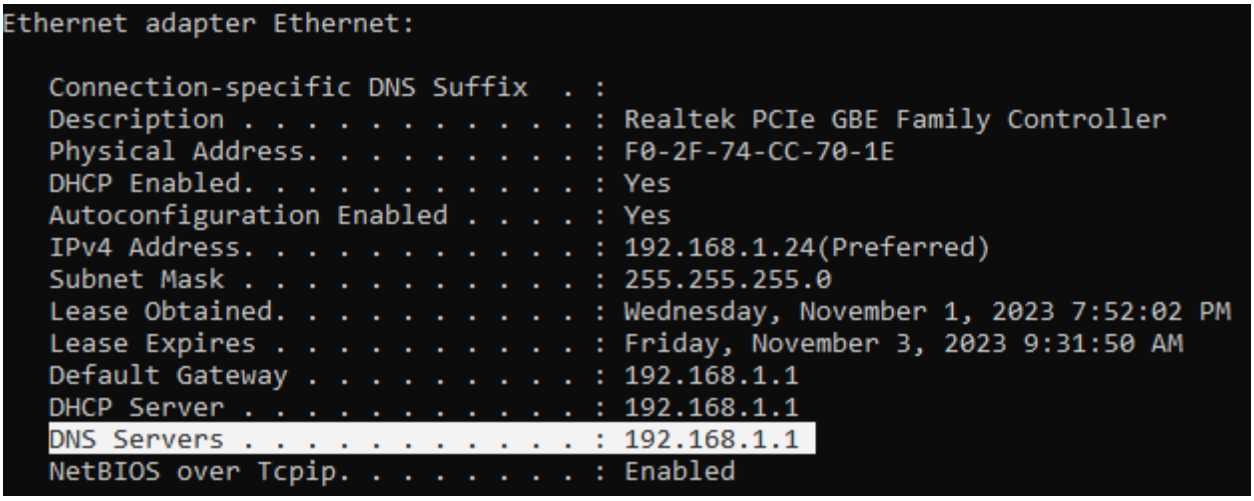
5. What is the destination port for the DNS query message? What is the source port of DNS response message?



The destination port of the query message and the source port of the response message are 53.

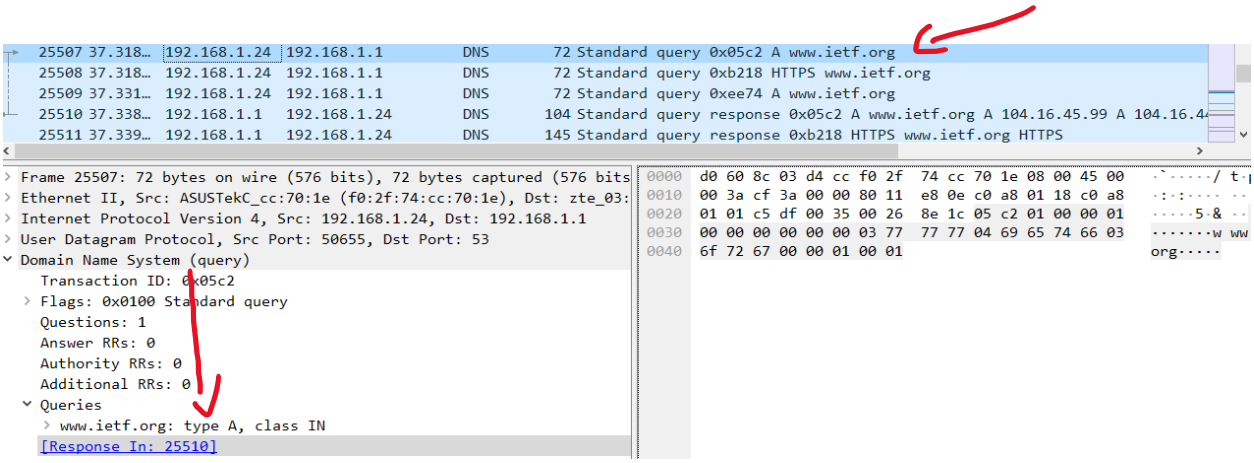
6. To what IP address is the DNS query message sent? Use ipconfig to determine the IP address of your local DNS server. Are these two IP addresses the same?





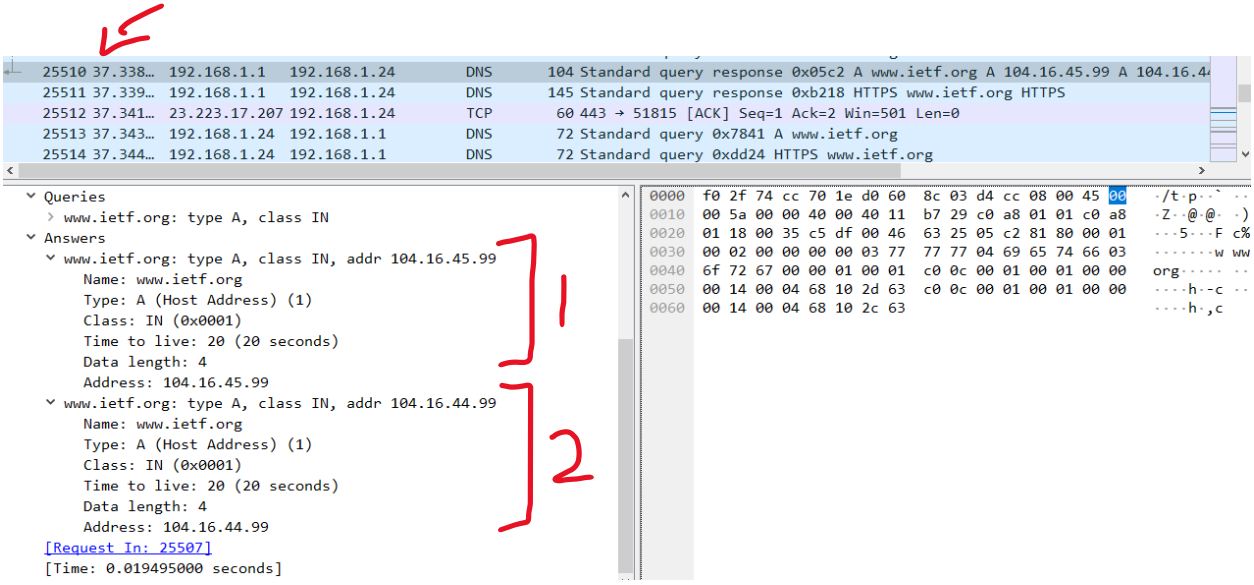
Yes, both IP addresses are 192.168.1.1.

7. Examine the DNS query message. What "Type" of DNS query is it? Does the query message contain any "answers"?



It is of type A, however it doesn't contain any answers.

8. Examine the DNS response message. How many "answers" are provided? What do each of these answers contain?

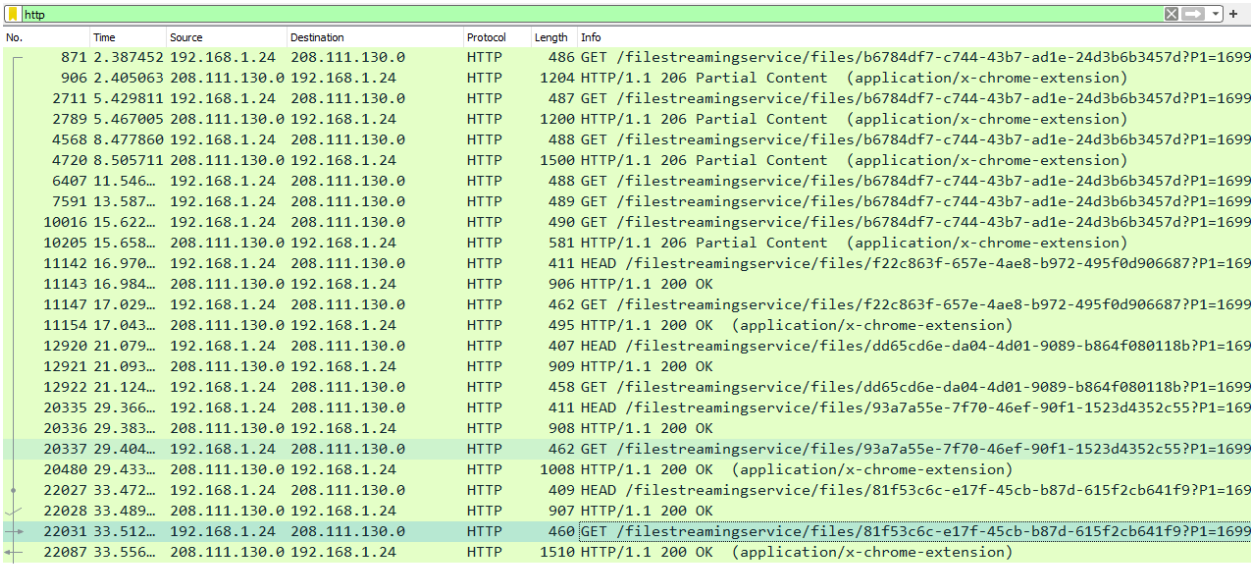


The response No.25510 has two answers with the difference being their IP addresses being 104.16.45.99 and 104.16.44.99 respectively. They contain name, class, time, data len and address.

9. Consider the subsequent TCP SYN packet sent by your host. Does the destination IP address of the SYN packet correspond to any of the IP addresses provided in the DNS response message?

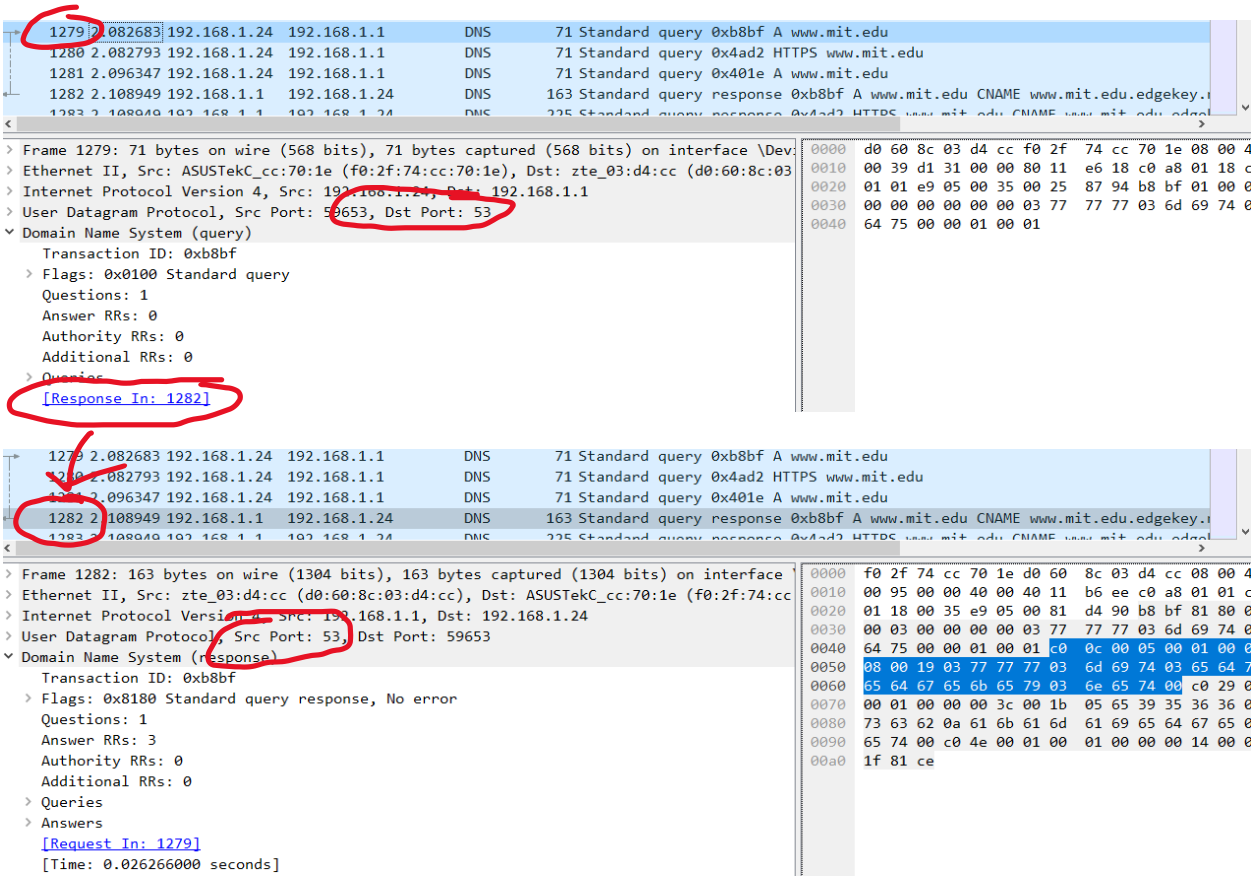
The destination IP address of the SYN packet corresponds to the first IP address provided in the DNS response message being 104.16.45.99.

10. This web page contains images. Before retrieving each image, does your host issue new DNS queries?



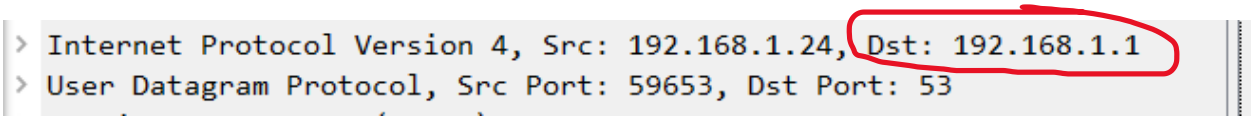
No, all images are loaded directly from http://www.ieft.org. This can be observed due to the fact that no HTTP GET image are found when filtering for “http”.

11. What is the destination port for the DNS query message? What is the source port of DNS response message?



The destination port of query message (No. 1279) is 53, while the source port of the response message (No. 1282) is also 53.

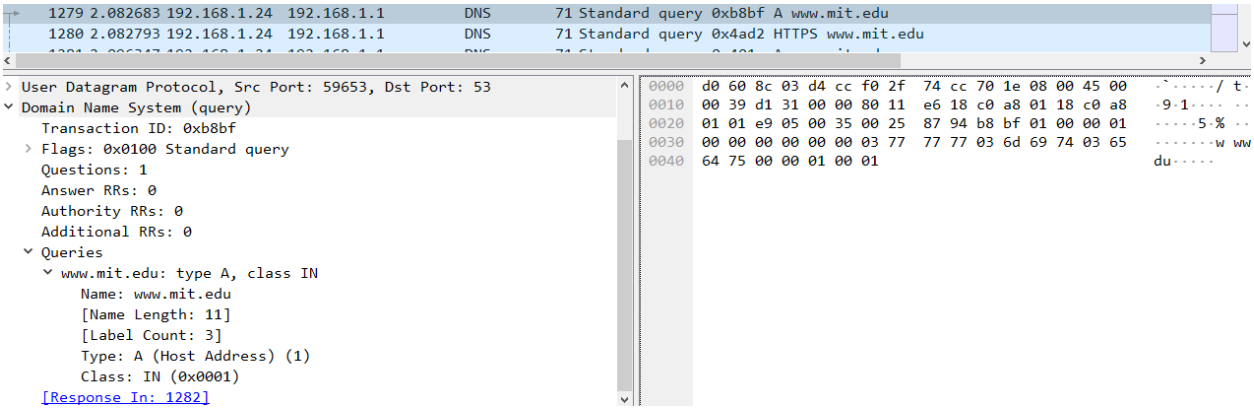
12. To what IP address is the DNS query message sent? Is this the IP address of your default local DNS server?



```
Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : Realtek PCIe GBE Family Controller
   Physical Address. . . . . . . . . : F0-2F-74-CC-70-1E
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes
   IPv4 Address. . . . . . . . . . . : 192.168.1.24(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Lease Obtained. . . . . . . . . . : Wednesday, November 1, 2023 7:52:02 PM
   Lease Expires . . . . . . . . . . : Friday, November 3, 2023 9:31:50 AM
   Default Gateway . . . . . . . . . : 192.168.1.1
   DHCP Server . . . . . . . . . . . : 192.168.1.1
   DNS Servers . . . . . . . . . . . : 192.168.1.1
   NetBIOS over Tcpip. . . . . . . . : Enabled
```

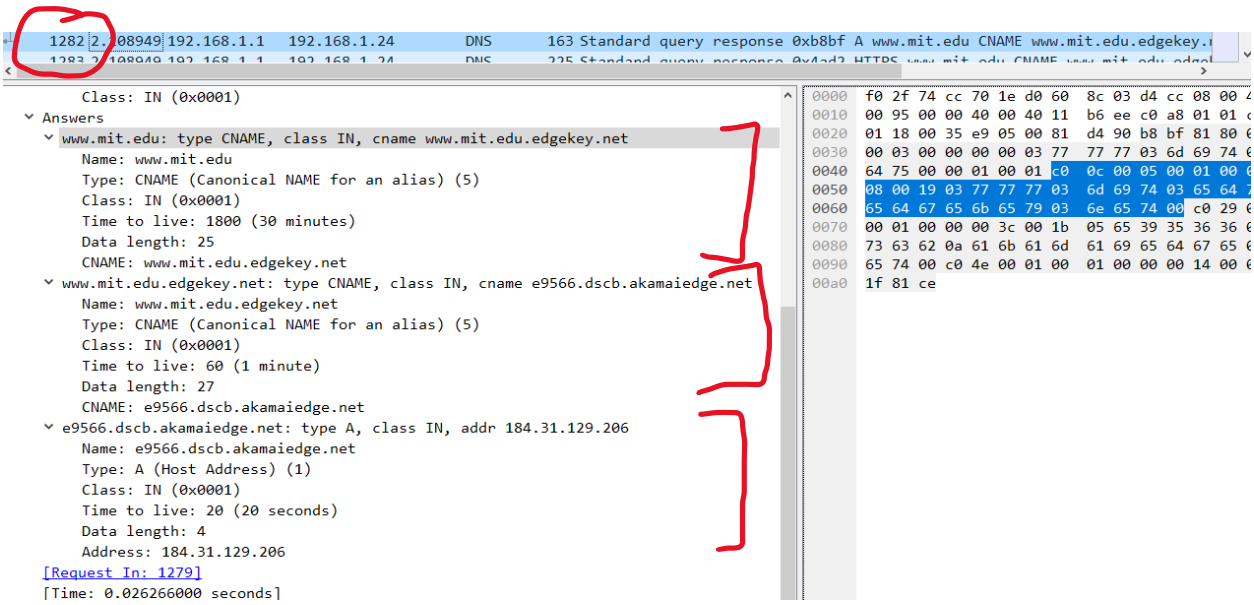
The IP address sent is to my default local DNS server of 192.168.1.1.

13. Examine the DNS query message. What "Type" of DNS query is it? Does the query message contain any "answers"?



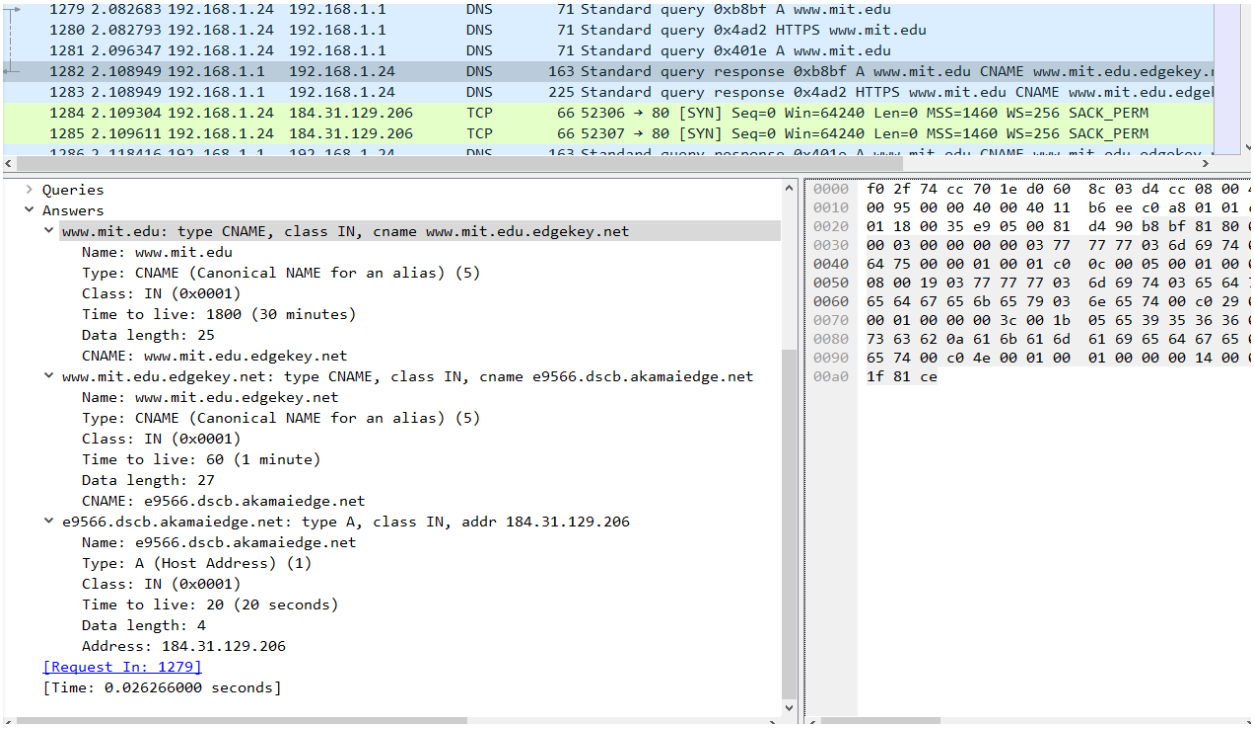
It is of type A, however it doesn't contain any answers.

14. Examine the DNS response message. How many "answers" are provided? What do each of these answers contain?



We have three answers. The first two are type CNAME while one is of type A. The first two contain name, class, time, data len and cname. The last one contains name, class, time, data len and address.

15. Provide a screenshot.



16. To what IP address is the DNS query message sent? Is this the IP address of your default local DNS server?

```
C:\Users\andre>nslookup -type=NS mit.edu
Server:  h268a
Address:  192.168.1.1

Non-authoritative answer:
mit.edu nameserver = eur5.akam.net
mit.edu nameserver = asia2.akam.net
mit.edu nameserver = ns1-37.akam.net
mit.edu nameserver = ns1-173.akam.net
mit.edu nameserver = asia1.akam.net
mit.edu nameserver = use5.akam.net
mit.edu nameserver = use2.akam.net
mit.edu nameserver = usw2.akam.net

eur5.akam.net   internet address = 23.74.25.64
use2.akam.net   internet address = 96.7.49.64
use5.akam.net   internet address = 2.16.40.64
asia1.akam.net  internet address = 95.100.175.64
asia2.akam.net  internet address = 95.101.36.64
use5.akam.net   AAAA IPv6 address = 2600:1403:a::40
```

```
    23 1.743085 192.168.1.24  192.168.1.1      DNS      67 Standard query 0x0002 NS mit.edu
    24 1.770691 192.168.1.1   192.168.1.24     DNS     342 Standard query response 0x0002 NS mit.edu NS eur5.akam.net NS asia2.akar
    26 2.788957 192.168.1.24  184.150.157.58   TCP      66 52598 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
    27 2.819107 184.150.157…  192.168.1.24     TCP      66 443 → 52598 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 SACK_PERM WS
    28 2.819135 192.168.1.24  184.150.157.58   TCP      54 52598 → 443 [ACK] Seq=1 Ack=1 Win=131328 Len=0
```

```
> Frame 23: 67 bytes on wire (536 bits), 67 bytes captured (536 bits) on interface \Devic
> Ethernet II, Src: ASUSTekC_cc:70:1e (f0:2f:74:cc:70:1e), Dst: zte_03:d4:cc (d0:60:8c:03
> Internet Protocol Version 4, Src: 192.168.1.24, Dst: 192.168.1.1
> User Datagram Protocol, Src Port: 49920, Dst Port: 53
v Domain Name System (query)
    Transaction ID: 0x0002
  > Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  > Queries
    [Response In: 24]
```

```
0000  d0 60 8c 03 d4 cc f0 2f  74 cc 70 1e 08 00
0010  00 35 d2 06 00 00 80 11  e5 47 c0 a8 01 18
0020  01 01 c3 00 00 35 00 21  e0 4d 00 02 01 00
0030  00 00 00 00 00 00 03 6d  69 74 03 65 64 75
0040  02 00 01
```

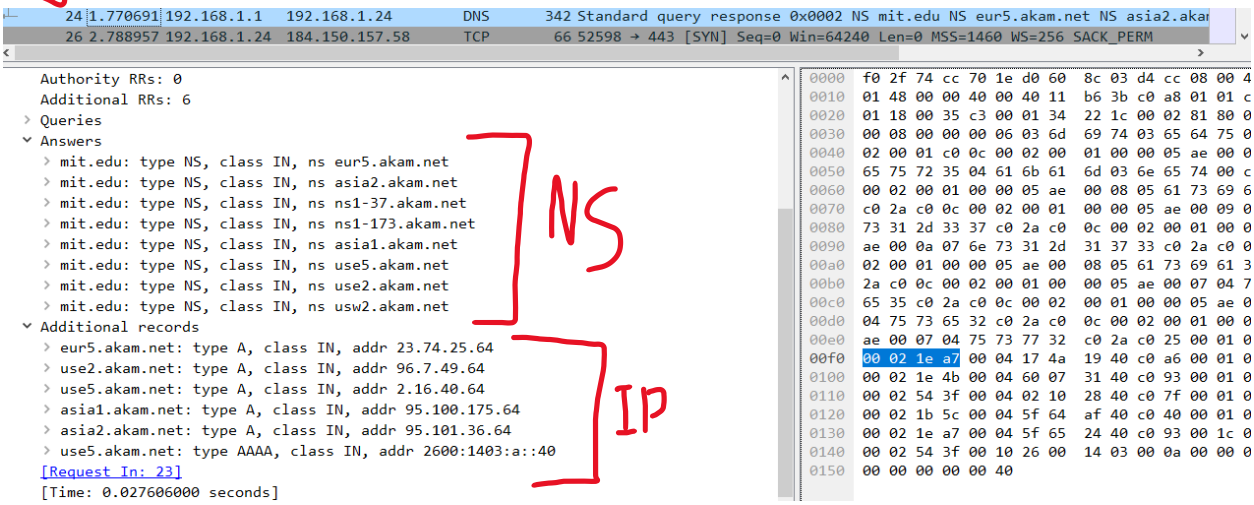
The DNS query message was sent to my default local DNS server with IP address 192.168.1.1.

17. Examine the DNS query message. What "Type" of DNS query is it? Does the query message contain any "answers"?

```
v Queries
  v mit.edu: type NS, class IN
      Name: mit.edu
      [Name Length: 7]
      [Label Count: 2]
      Type: NS (authoritative Name Server) (2)
      Class: IN (0x0001)
    [Response In: 24]
```

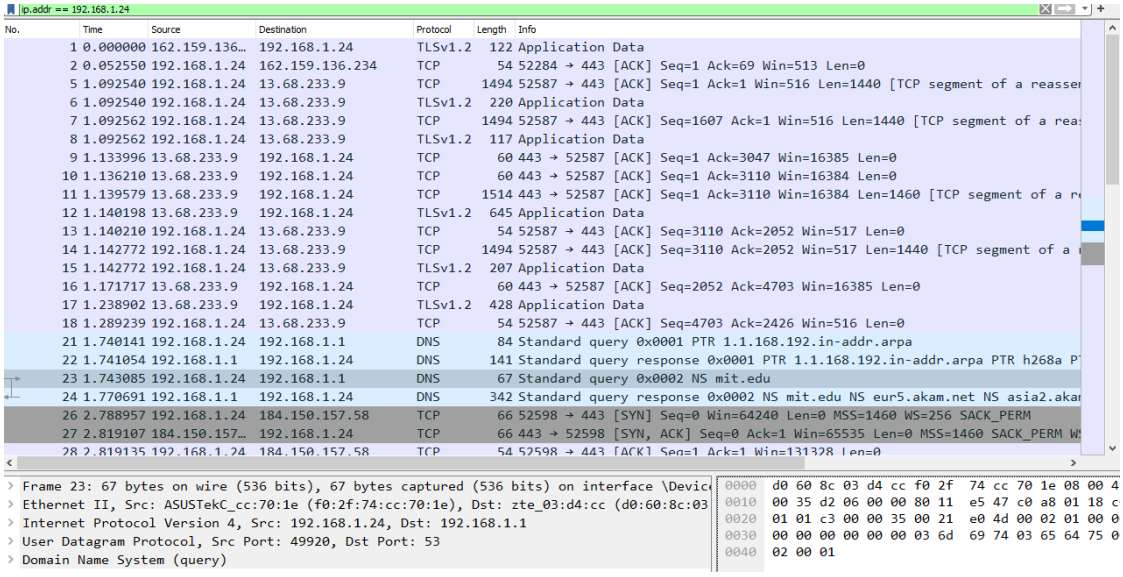
It is of type NS and it has no answers.

18. Examine the DNS response message. What MIT nameservers does the response message provide? Does this response message also provide the IP addresses of the MIT namesers?

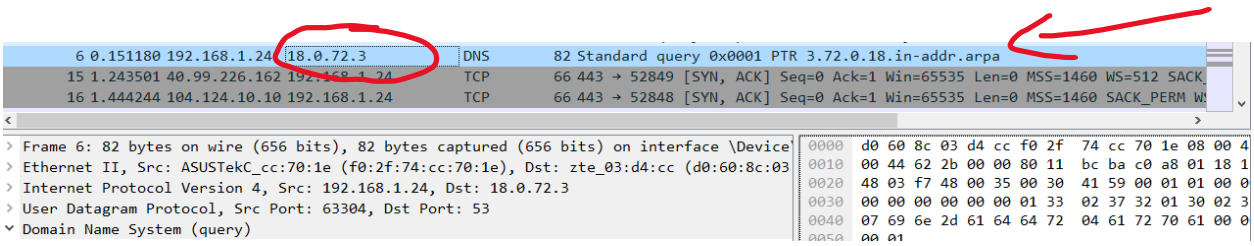


The name servers are: eur5.akam.net asia2.akam.net ns1-37.akam.net ns1-173.akam.net asia1.akam.net use5.akam.net use2.akam.net usw2.akam.net. Their corresponding IP addresses are on “Additional records” sections.

19. Provide a screenshot.

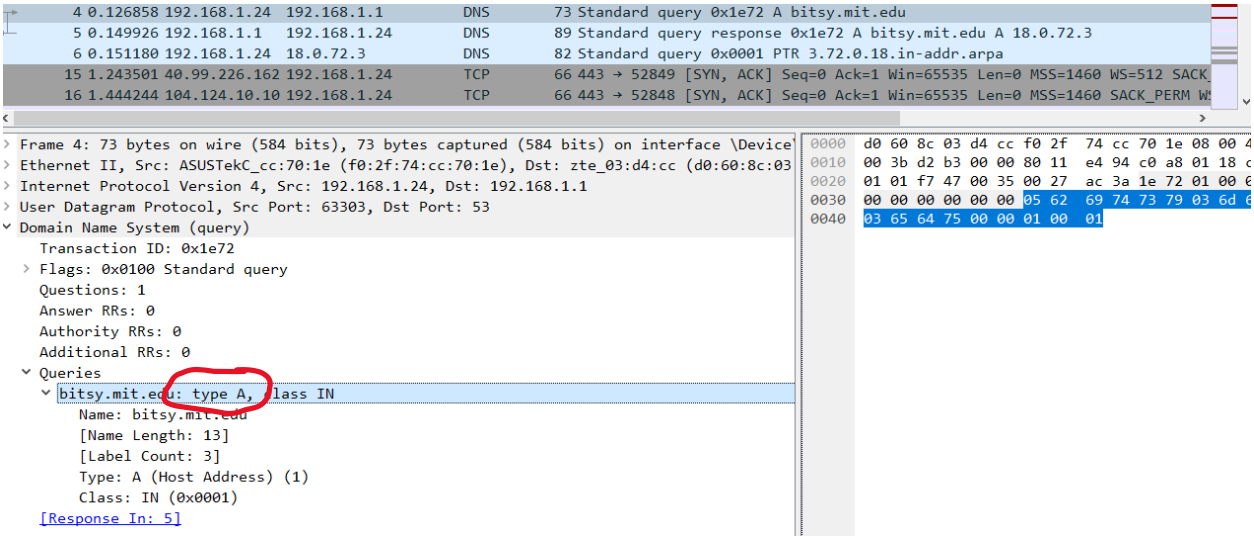


20. To what IP address is the DNS query message sent? Is this the IP address of your default local DNS server? If not, what does the IP address correspond to?



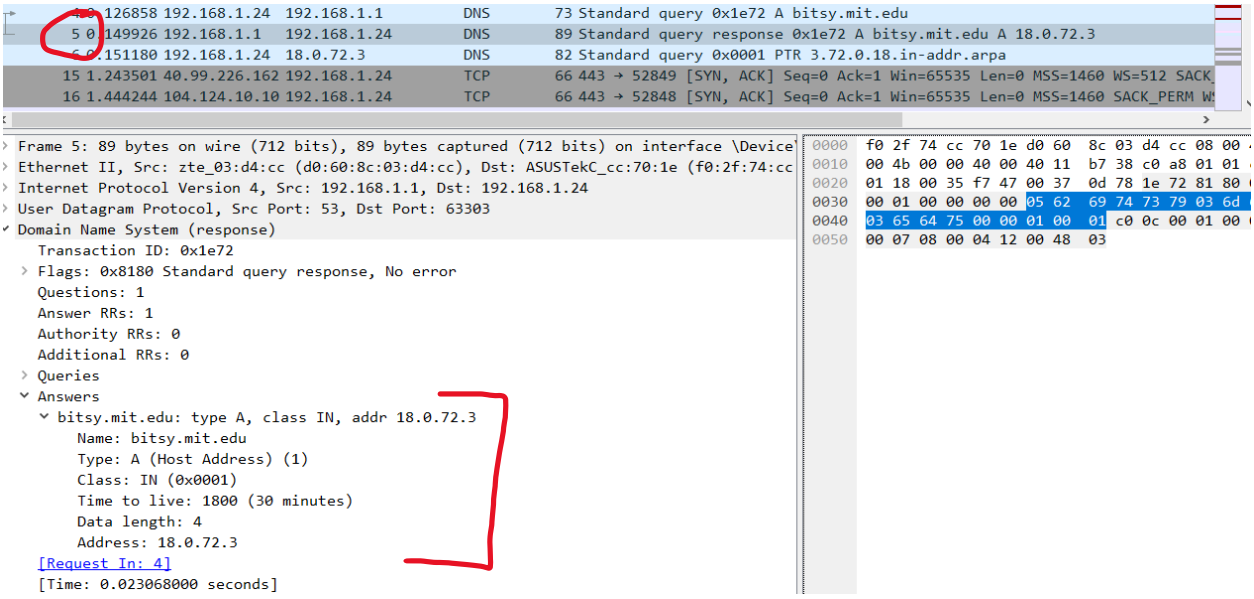
It was sent to 18.0.72.3 which is not my default DNS server. The IP corresponds to bitsy.mit.edu.

21. Examine the DNS query message. What "Type" of DNS query is it? Does the query message contain any "answers"?



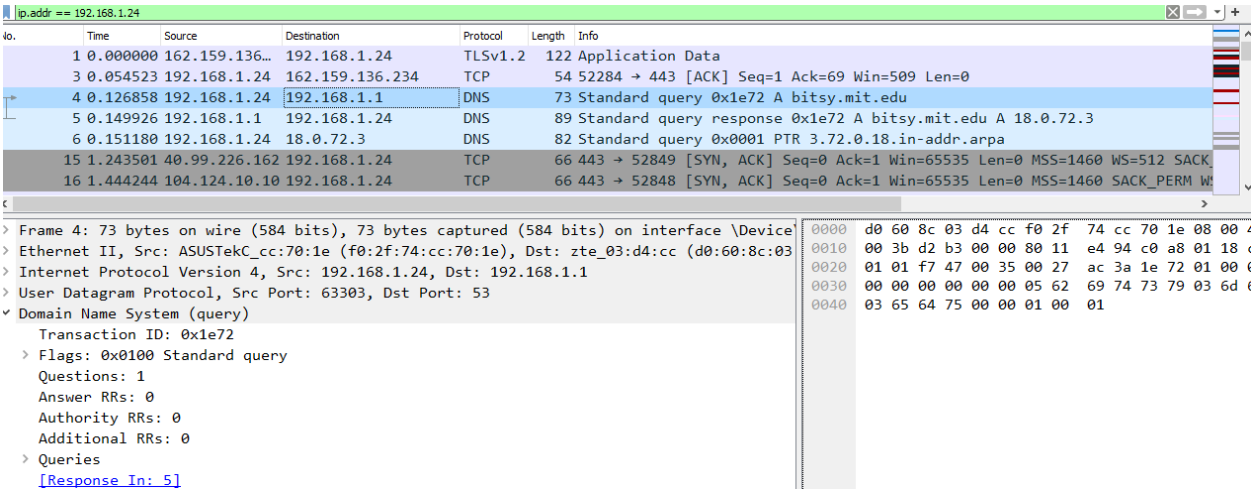
It is of type A and it contains no answers.

22. Examine the DNS response message. How many "answers" are provided? What does each of these answers contain?



The DNS response message has one answer provided. It contains the name, type, class, time to live, data length and address.

23. Provide a screenshot.



3. Concepts Learned from this Lab

With the Mininet assignment 1, students were able to work on the virtual machine Ubuntu and tests four topologies. They are the single, reversed, linear and tree topology. By working on coding a python topology code, we were also able to understand more about hosts, switches and controller which all ties together into a network. With the Wireshark assignment 2, students learned about the domain name system (DNS). By capturing packages on Wireshark, we could read queries and responses, sources and destinations IP addresses, DNS message types, DNS answers, etc. Overall, these two provided students with hands-on experience in topology and network traffic analysis of DNS.