

# Internet Domain Name System

Lab 6

*50.005 Computer System Engineering*

---

**Due: 15 Apr 08:30 AM (Week 12)**

[Overview and Learning Objectives](#)

[Submission](#)

[Part 1: Exploring DNS using dig](#)

[Part 2: DNS Hierarchy](#)

[Part 3: DNS Caching](#)

[Part 4: Tracing DNS using Wireshark](#)

# Overview and Learning Objectives

In NS Module 4, we learnt about the role of the Domain Name System (DNS) in Internet naming and addressing. In this lab exercise, we will go deeper into DNS by using specialised network tools to perform and analyse DNS queries.

At the end of this lab exercise, you should be able to:

- Use `dig` to perform DNS queries (e.g., to look up an IP address)
- Read and interpret DNS records of different types
- Understand how a DNS query is resolved using hierarchy and recursion
- Observe and understand the effect of caching on DNS lookup times
- Use Wireshark to trace and read DNS packets sent to and from a machine

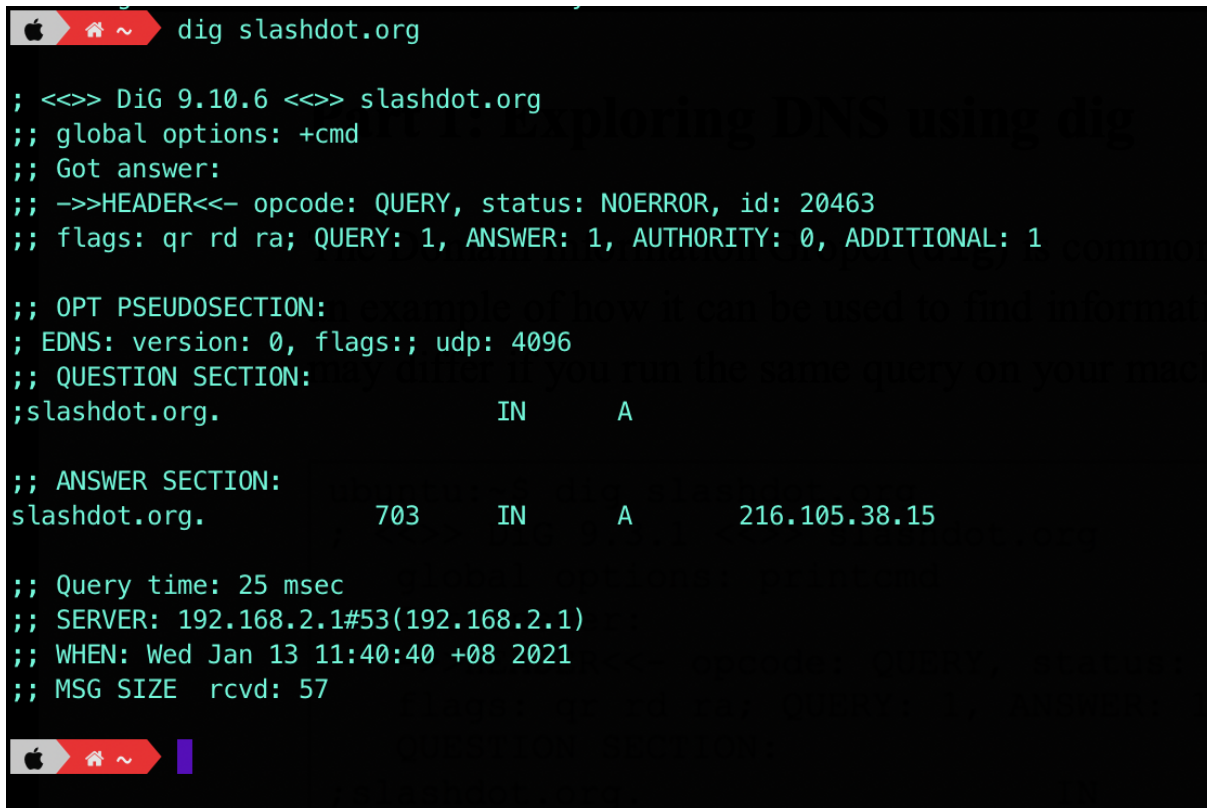
If you are using Ubuntu, `dig` should already be available on your system. To install Wireshark, run `sudo apt-get install wireshark` from the command line.

## Submission

- The total marks for this Lab are 25 (Part 1: 10, Part 2: 5, Part 3: 10).
- Complete the activities and answer the questions in the handout **denoted in blue**. As usual, export and edit this document.
- Export as pdf and **ZIP** it (not RAR, or any other compression algorithm).
- **Upload** to @cse-submitbot telegram bot using the command `/submitlab6`
- **CHECK** your submission by using the command `/checksubmission`

# Part 1: Exploring DNS using dig

The Domain Information Groper (dig) is commonly used for performing DNS lookups. Here is an example of how it can be used to find information about the host slashdot.org. The results may differ if you run the same query on your machine.



```

dig slashdot.org

; <<>> DiG 9.10.6 <<>> slashdot.org
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 20463
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:: udp: 4096
;; QUESTION SECTION:
;slashdot.org.                IN      A

;; ANSWER SECTION:
slashdot.org.                703     IN      A      216.105.38.15

;; Query time: 25 msec
;; SERVER: 192.168.2.1#53(192.168.2.1)
;; WHEN: Wed Jan 13 11:40:40 +08 2021
;; MSG SIZE rcvd: 57

```

When the command `dig slashdot.org` is run, dig performs a DNS lookup and displays information about the request and the response it receives. At the bottom of the printout, we can see that the query was sent to the DNS server running on 192.168.2.1, and that the query took 25 ms to complete. Most of the information that we are interested in can be found in the **ANSWER SECTION**.

The answer section for this query contains a DNS record:

|                    |               |              |             |               |
|--------------------|---------------|--------------|-------------|---------------|
| slashdot.org       | 703           | IN           | A           | 216.105.38.15 |
| <i>Server Name</i> | <i>Expiry</i> | <i>Class</i> | <i>Type</i> | <i>Data</i>   |

We can see that the result is of type A, an address record. It tells us that the IP address for the domain name slashdot.org is 216.105.38.15. The expiry time field

indicates that this record is valid for 703 seconds. The value of the class field is usually **IN** (Internet) for all records.

If you'd like to know who's the authoritative NS for the queried domain, you can add the **trace** option: `dig slashdot.org +trace`

```
slashdot.org.      900      IN       A       216.105.38.15
slashdot.org.      86400    IN       NS      ns3.dnsmadeeasy.com.
slashdot.org.      86400    IN       NS      ns4.dnsmadeeasy.com.
slashdot.org.      86400    IN       NS      ns0.dnsmadeeasy.com.
slashdot.org.      86400    IN       NS      ns1.dnsmadeeasy.com.
slashdot.org.      86400    IN       NS      ns2.dnsmadeeasy.com.
;; Received 162 bytes from 208.80.125.2#53(ns3.dnsmadeeasy.com) in 22 ms
```

The records of type NS indicate the names of the DNS servers storing records for a particular domain. Here, we can see that the hosts `ns.3.dnsmadeeasy.com.`, etc. are responsible for providing authoritative responses to names in the `slashdot.org` domain.

We can query a specific server for information about a host by using the `@` option. For example, to perform a lookup using the DNS server `ns3.dnsmadeeasy.com.`, we can run the command `dig @ns3.dnsmadeeasy.com. slashdot.org`.

```
dig @ns3.dnsmadeeasy.com. slashdot.org.

; <<>> DiG 9.10.6 <<>> @ns3.dnsmadeeasy.com. slashdot.org.
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 18649
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 5, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1280
;; QUESTION SECTION:
;slashdot.org.                IN      A

;; ANSWER SECTION:
slashdot.org.      900      IN      A       216.105.38.15

;; AUTHORITY SECTION:
slashdot.org.      86400    IN      NS      ns3.dnsmadeeasy.com.
slashdot.org.      86400    IN      NS      ns1.dnsmadeeasy.com.
slashdot.org.      86400    IN      NS      ns0.dnsmadeeasy.com.
slashdot.org.      86400    IN      NS      ns4.dnsmadeeasy.com.
slashdot.org.      86400    IN      NS      ns2.dnsmadeeasy.com.

;; Query time: 20 msec
;; SERVER: 208.80.125.2#53(208.80.125.2)
;; WHEN: Mon Jan 18 00:10:10 +08 2021
;; MSG SIZE rcvd: 162
```

There are three flags under the header: qr, aa, and rd.

- This means that the message is a query (qr), and dig is requesting a recursive lookup (rd stands for ‘*recursion desired*’) and the server is the authoritative name server (aa stands for ‘*authoritative answer*’).
- Not all servers perform recursive lookups due to the heavier load involved, and so you don’t see any ra flags here (ra stands for ‘*recursion available*’).

dig only prints the final result of a recursive search, but you can mimic the individual steps involved by making a query with the +norecurs option enabled. For example, to send a non- recursive query to one of the root servers:

```

dig @a.ROOT-SERVERS.NET www.slashdot.org +norecurs

; <<>> DiG 9.10.6 <<>> @a.ROOT-SERVERS.NET www.slashdot.org +norecurs
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 2821
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 6, ADDITIONAL: 13

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1472
;; QUESTION SECTION:
;www.slashdot.org.                IN      A

;; AUTHORITY SECTION:
org.                172800  IN      NS      a0.org.afilias-nst.info.
org.                172800  IN      NS      a2.org.afilias-nst.info.
org.                172800  IN      NS      b0.org.afilias-nst.org.
org.                172800  IN      NS      b2.org.afilias-nst.org.
org.                172800  IN      NS      c0.org.afilias-nst.info.
org.                172800  IN      NS      d0.org.afilias-nst.org.

;; ADDITIONAL SECTION:
a0.org.afilias-nst.info. 172800 IN      A      199.19.56.1
a2.org.afilias-nst.info. 172800 IN      A      199.249.112.1
b0.org.afilias-nst.org. 172800 IN      A      199.19.54.1
b2.org.afilias-nst.org. 172800 IN      A      199.249.120.1
c0.org.afilias-nst.info. 172800 IN      A      199.19.53.1
d0.org.afilias-nst.org. 172800 IN      A      199.19.57.1
a0.org.afilias-nst.info. 172800 IN      AAAA   2001:500:e::1
a2.org.afilias-nst.info. 172800 IN      AAAA   2001:500:40::1
b0.org.afilias-nst.org. 172800 IN      AAAA   2001:500:c::1
b2.org.afilias-nst.org. 172800 IN      AAAA   2001:500:48::1
c0.org.afilias-nst.info. 172800 IN      AAAA   2001:500:b::1
d0.org.afilias-nst.org. 172800 IN      AAAA   2001:500:f::1

;; Query time: 120 msec
;; SERVER: 198.41.0.4#53(198.41.0.4)
;; WHEN: Mon Jan 18 00:16:48 +08 2021
;; MSG SIZE rcvd: 447

```

As you can see, the server does not know the answer (there’s 0 ANSWER) and instead provides information about the servers most likely to be able to provide an

authoritative answer for the question. In this case, the best that the root server knows is the identities of the servers for the org. top-level domain.

Now answer the questions below.

**Question 1 [1p]:** Using dig, find the IP address for thyme.lcs.mit.edu. What is the IP address?

**Your answer: 18.26.0.122**

**Question 2 [1p]:** The dig answer for the previous question includes a record of type CNAME. What does CNAME mean?

**Your answer: CNAME stands for Canonical Name. A CNAME record is used to create an alias that maps an alias domain name (thyme.lcs.mit.edu) to another true/canonical domain name (mercury.lcs.mit.edu). When an A record lookup for thyme occurs, the resolver will detect the CNAME record and restart the check for mercury and return 18.26.0.122 automatically, since recursive behaviour is enabled by default for dig.**

**Question 3 [1p]:** What is the expiration time for the CNAME record?

**Your answer: 1767 seconds.**

**Question 4:** Run the following commands to find out what your computer receives when it looks up 'ai' and 'ai.' in the mit.edu domain. What are the two resulting IP addresses?

- `dig +domain=mit.edu ai`

**Your answer [1p]: No IP address.**

- `dig +domain=mit.edu ai.`

**Your answer [1p]: 209.59.119.34 (in some environments, adding a "@8.8.8.8" might be necessary to get the correct IP address).**

**Question 5 [1p]:** Why are the results for both queries different? Look up the manual for `dig` to find out what the `+domain` parameter does. Based on the output of the two commands, what is the difference between the DNS searches being performed for `'ai'` and `'ai.'`?

**Your answer:** The results for both queries are different since the intended actual domain names to be queried are different. The `+domain` parameter sets the search list to contain the single domain specified as the value of said parameter, as if it is specified in a domain directive in `/etc/resolv.conf` and enables search processing as if the `+search` option was given. Hence, it searches for that specific single domain. For the first query, the server's name is `ai.mit.edu`, while for the second query, the server's name is the absolute path `ai.`, which is the Internet country-code top-level domain for Anguilla. In particular, the IP address `209.59.119.34` goes to Anguilla's Offshore Information Services Ltd.'s website.

## Part 2: DNS Hierarchy

In the previous section, you ran `dig` without changing the default options. This causes `dig` to perform a recursive lookup if the DNS server being queried supports it. In this part, you will trace the intermediate steps involved in performing a recursive query by beginning at a root server and manually going through the DNS hierarchy to resolve a host name. You can obtain a list of all the root servers by running the command `dig . NS`.

**Question 6 [1p]:** Use `dig` to query one of the DNS root servers for the IP address of `lirone.csail.mit.edu` without using recursion. What is the command that you use to do this?

**Your answer:**

- `dig @a.root-servers.net lirone.csail.mit.edu +norecurse`

**Question 7 [3p]:** Go through the DNS hierarchy from the root until you have found the IP address of `lirone.csail.mit.edu`. You should disable recursion and follow the referrals manually. Which commands did you use, and what addresses did you find? You can provide screenshots for each step.

**Your answer:**

**Commands:**

- `dig @a.root-servers.net lirone.csail.mit.edu +norecurse`



```

~ dig @a.root-servers.net lirone.csail.mit.edu +norecurse
; <<> DiG 9.16.1-Ubuntu <<> @a.root-servers.net lirone.csail.mit.edu +norecurse
; (2 servers found)
; global options: +cmd
; Got answer:
; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 54192
; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 13, ADDITIONAL: 27

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags::, udp: 1472
;; QUESTION SECTION:
;lirone.csail.mit.edu.      IN      A

;; AUTHORITY SECTION:
edu.      172800 IN      NS      a.edu-servers.net.
edu.      172800 IN      NS      b.edu-servers.net.
edu.      172800 IN      NS      c.edu-servers.net.
edu.      172800 IN      NS      d.edu-servers.net.
edu.      172800 IN      NS      e.edu-servers.net.
edu.      172800 IN      NS      f.edu-servers.net.
edu.      172800 IN      NS      g.edu-servers.net.
edu.      172800 IN      NS      h.edu-servers.net.
edu.      172800 IN      NS      i.edu-servers.net.
edu.      172800 IN      NS      j.edu-servers.net.
edu.      172800 IN      NS      k.edu-servers.net.
edu.      172800 IN      NS      l.edu-servers.net.
edu.      172800 IN      NS      m.edu-servers.net.

;; ADDITIONAL SECTION:
a.edu-servers.net. 172800 IN      A      192.5.6.30
b.edu-servers.net. 172800 IN      A      192.33.14.30
c.edu-servers.net. 172800 IN      A      192.26.92.30
d.edu-servers.net. 172800 IN      A      192.31.80.30
e.edu-servers.net. 172800 IN      A      192.12.94.30
f.edu-servers.net. 172800 IN      A      192.35.51.30
g.edu-servers.net. 172800 IN      A      192.42.93.30
h.edu-servers.net. 172800 IN      A      192.54.112.30
i.edu-servers.net. 172800 IN      A      192.43.172.30
j.edu-servers.net. 172800 IN      A      192.48.79.30
k.edu-servers.net. 172800 IN      A      192.52.178.30
l.edu-servers.net. 172800 IN      A      192.41.162.30
m.edu-servers.net. 172800 IN      A      192.55.83.30
a.edu-servers.net. 172800 IN      AAAA   2001:503:a83e::2:30
b.edu-servers.net. 172800 IN      AAAA   2001:503:231d::2:30
c.edu-servers.net. 172800 IN      AAAA   2001:503:83eb::30
d.edu-servers.net. 172800 IN      AAAA   2001:500:856e::30
e.edu-servers.net. 172800 IN      AAAA   2001:502:1ca1::30
f.edu-servers.net. 172800 IN      AAAA   2001:503:d414::30
g.edu-servers.net. 172800 IN      AAAA   2001:503:eea3::30
h.edu-servers.net. 172800 IN      AAAA   2001:502:8cc::30
i.edu-servers.net. 172800 IN      AAAA   2001:503:39c1::30
j.edu-servers.net. 172800 IN      AAAA   2001:502:7094::30
k.edu-servers.net. 172800 IN      AAAA   2001:503:d2d::30
l.edu-servers.net. 172800 IN      AAAA   2001:500:d937::30
m.edu-servers.net. 172800 IN      AAAA   2001:501:b1f9::30

;; Query time: 78 msec
;; SERVER: 198.41.0.4#53(198.41.0.4)
;; WHEN: Sat Apr 10 18:23:30 +08 2021
;; MSG SIZE rcvd: 844

```

- `dig @a.edu-servers.net lirone.csail.mit.edu +norecurse`

```

~ dig @a.edu-servers.net lirone.csail.mit.edu +norecurse

; <<>> DiG 9.16.1-Ubuntu <<>> @a.edu-servers.net lirone.csail.mit.edu +norecurse
; (2 servers found)
; global options: +cmd
; Got answer:
; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 52374
; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 8, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags::; udp: 4096
;; QUESTION SECTION:
;lirone.csail.mit.edu.          IN      A

;; AUTHORITY SECTION:
mit.edu.          172800  IN      NS      usw2.akam.net.
mit.edu.          172800  IN      NS      asia1.akam.net.
mit.edu.          172800  IN      NS      asia2.akam.net.
mit.edu.          172800  IN      NS      use2.akam.net.
mit.edu.          172800  IN      NS      ns1-37.akam.net.
mit.edu.          172800  IN      NS      ns1-173.akam.net.
mit.edu.          172800  IN      NS      eur5.akam.net.
mit.edu.          172800  IN      NS      use5.akam.net.

;; Query time: 217 msec
;; SERVER: 192.5.6.30#53(192.5.6.30)
;; WHEN: Sat Apr 10 18:24:20 +08 2021
;; MSG SIZE rcvd: 216

```

- `dig @asia1.akam.net lirone.csail.mit.edu +norecurse`

```

~ dig @asia1.akam.net lirone.csail.mit.edu +norecurse

; <<>> DiG 9.16.1-Ubuntu <<>> @asia1.akam.net lirone.csail.mit.edu +norecurse
; (1 server found)
; global options: +cmd
; Got answer:
; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 33122
; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 4, ADDITIONAL: 7

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags::; udp: 4096
;; QUESTION SECTION:
;lirone.csail.mit.edu.          IN      A

;; AUTHORITY SECTION:
csail.mit.edu.      1800    IN      NS      auth-ns3.csail.mit.edu.
csail.mit.edu.      1800    IN      NS      auth-ns2.csail.mit.edu.
csail.mit.edu.      1800    IN      NS      auth-ns0.csail.mit.edu.
csail.mit.edu.      1800    IN      NS      auth-ns1.csail.mit.edu.

;; ADDITIONAL SECTION:
auth-ns3.csail.mit.edu. 1800    IN      A      18.220.24.142
auth-ns0.csail.mit.edu. 1800    IN      AAAA   2603:400a:0:7d2::801e:27b
auth-ns3.csail.mit.edu. 1800    IN      AAAA   2600:1f16:ceb:1302::a
auth-ns2.csail.mit.edu. 1800    IN      A      128.52.32.80
auth-ns0.csail.mit.edu. 1800    IN      A      128.30.2.123
auth-ns1.csail.mit.edu. 1800    IN      A      128.31.0.18

;; Query time: 45 msec
;; SERVER: 95.100.175.64#53(95.100.175.64)
;; WHEN: Sat Apr 10 18:24:53 +08 2021
;; MSG SIZE rcvd: 261

```

- `dig @auth-ns0.csail.mit.edu lirone.csail.mit.edu +norecurse`

```
~ dig @auth-ns0.csail.mit.edu lirone.csail.mit.edu +norecurse

;<<> DiG 9.16.1-Ubuntu <<> @auth-ns0.csail.mit.edu lirone.csail.mit.edu +norecurse
;; (2 servers found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 5849
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
;; EDNS: version: 0, flags:; udp: 1232
;; COOKIE: d1e4a20d96d866130100000060717d11e5e1d6dd5ff985f8 (good)
;; QUESTION SECTION:
;; lirone.csail.mit.edu.          IN      A

;; ANSWER SECTION:
lirone.csail.mit.edu.  1800    IN      A      128.52.129.186

;; Query time: 289 msec
;; SERVER: 128.30.2.123#53(128.30.2.123)
;; WHEN: Sat Apr 10 18:25:22 +08 2021
;; MSG SIZE rcvd: 93
```

**Final actual IP address of `lirone.csail.mit.edu`: 128.52.129.186**

## Part 3: DNS Caching

**Question 8:** Without using recursion, query your default (local) DNS server for information about `www.dmoz.org` and answer the following questions.

- **[1p]** What is the command that you used?
- **[1p]** Did your default server have the answer in its cache? How did you know?
- **[1p]** How long did the query take?

*If the information was cached, find another host name that was not cached and complete all the questions in this section using that host.*

*You can set your default DNS server as your router (your ISP might do that too). Why can your router support DNS (layer 5, application layer protocol) when “routers” are supposed to support up to Network layer only?*

**Your answer:** `dig www.dmoz.org +norecurse` gave no answer section and therefore, it was not cached. The query took 11 milliseconds. Status is NOERROR.

Today’s routers are fancier and more advanced than ever before. Routers are specified on layer 3 since they transport layer 3 information. However, this does not mean that they cannot understand higher level information if they have extra features baked in during manufacturing.

**Question 9 [1p]:** Query your default DNS server for information about the host in the previous question, using the recursion option this time. How long did the query take?

**Your answer:** 1040 milliseconds.

**Question 10 [1p]:** Query your default DNS server for information about the same host without using recursion. How long did the query take? Has the cache served its purpose? Explain why.

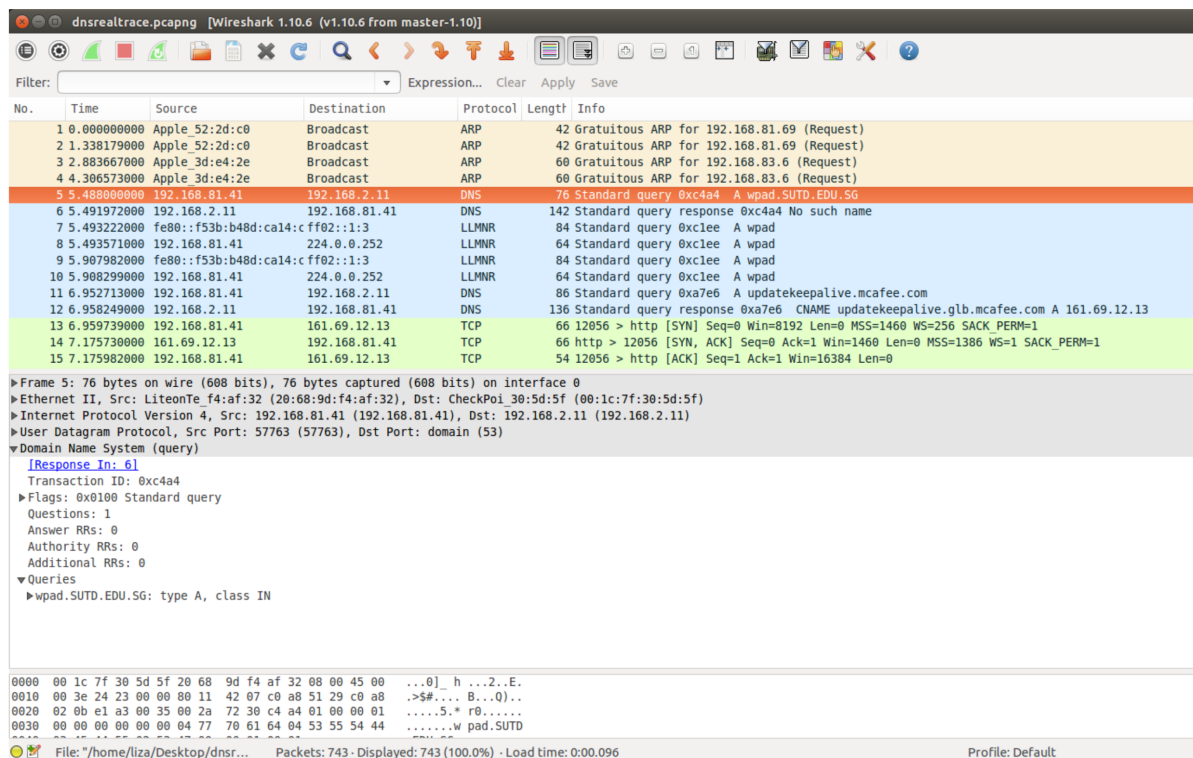
**Your answer:** 7 milliseconds. Since our local DNS server has enabled client-side caching, the information about [www.dmoz.org](http://www.dmoz.org), including its IP address, has been cached at our default local DNS server (even if we locally

**flush the DNS cache on our host computer/device). As such, the cache has served its purpose of serving DNS information and thus websites faster.**

## Part 4: Tracing DNS using Wireshark

Wireshark is a powerful tool used to capture packets sent over a network and analyse the content of the packets retrieved. The file [dnsrealtrace.pcapng](#) contains a trace of the packets sent and received when a web page is downloaded from a web server over the SUTD network. In the process of downloading the web page, DNS is used to find the IP address of the server.

Open the *dnsrealtrace.pcapng* in Wireshark and answer the following questions. You can refer to Wireshark tutorial [here](#) before proceeding.



**Question 11 [1p]:** Locate the DNS query and response messages. Are they sent over UDP or TCP?

**Your answer: UDP.**

**Question 12 [2p]:** What is the destination port for the DNS query message? What is the source port of the DNS response message?

**Your answer:**

- **Destination port of DNS query: 53**

- **Source port of DNS response: 53**

**They should be the same port numbers.**

**Question 13 [2p]:** What is the IP address to which the DNS query message was sent? Use `ifconfig` to determine the IP address of your local DNS server. Are these two addresses the same?

**Your answer: 192.168.2.11. No, they are not the same (my local DNS server's IP addresses are 192.168.2.100 and 192.168.2.101).**

**Question 14 [2p]:** Examine the second DNS query message. What type of DNS query is it? Does the query message contain any answers?

**Your answer: The second DNS query message is a standard recursive Type A DNS query to `updatekeepalive.mcafee.com`, which contains no answers.**

**Question 15 [2p]:** Examine the second DNS response message. How many answers are provided? What does each of these answers contain?

**Your answer: There are 2 answers:**

- **One CNAME (Canonical Name) record of class IN, pointing to `updatekeepalive.glb.mcafee.com`. The time to live is 209 seconds and the UDP data length is 22 bytes.**
- **One A (Host Address) record of class IN, pointing to the corresponding IP address of `161.69.12.13`. The time to live is 3 seconds and the UDP data length is 4 bytes.**

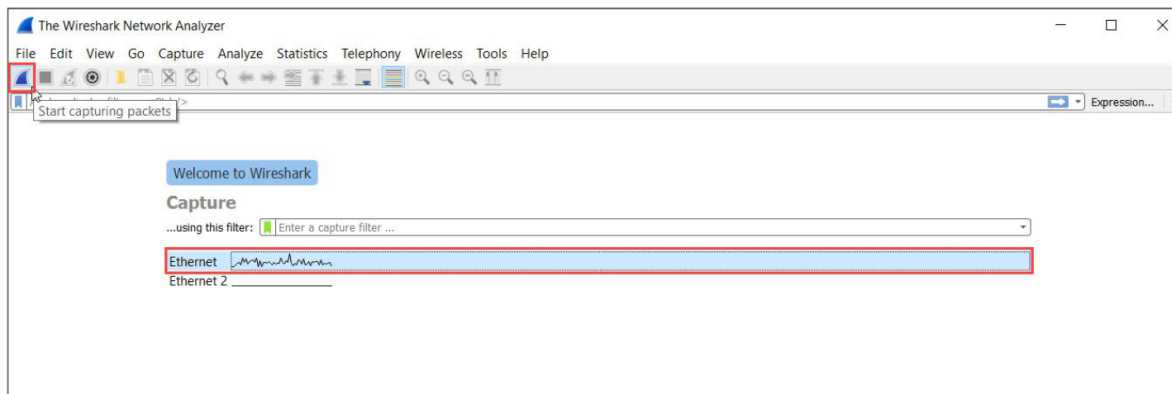
**Question 16 [1p]:** Locate a TCP SYN packet sent by your host subsequent to the above DNS response. This packet opens a TCP connection between your host and the web server. Does the destination IP address of the SYN packet correspond to any of the IP addresses provided in the DNS response message?

**Your answer: Yes, the destination address of the TCP SYN packet is `161.69.12.13`, which is the IP address of `updatekeepalive.glb.mcafee.com`.**

## Optional Activity:

Capturing packets for packet analysis: Steps:

1. Once Wireshark is installed, launch the program to begin.
2. Once the program is launched, select the network interface to capture and click on the *shark fin* icon at the top left of the application right under the menu bar to begin capturing packets.



3. To explore the interface, mention the interface (e.g., eth0, wlan) in the capture option.
4. There are display filters to analyse the packets.
  - Protocols: TCP, UDP, ARP, SMTP, etc.
  - Protocol Fields: port, src.addr, length, etc. (e.g., ip.src == 192.168.1.1)
5. For more detailed instructions on Wireshark, refer to <https://www.wireshark.org/>