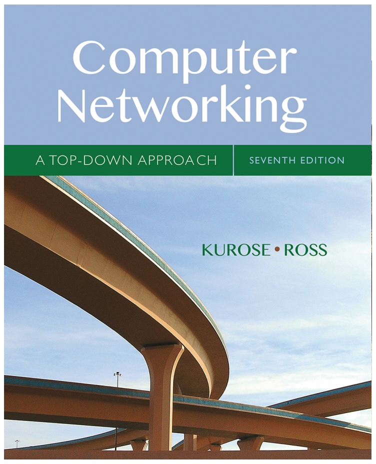
CS 447: Network and Data Communication
Wireshark Lab #01: DNS



Note:
- Make sure you produce your answers and any packet prints in PDF. Moodle will only accept PDF files.
- Provide a screenshot with each answer wherever applicable or possible as proof of your work.

As described in Section 2.4 of the text[1], the Domain Name System (DNS) translates hostnames to IP addresses, fulfilling a critical role in the Internet infrastructure. In this lab, we'll take a closer look at the client side of DNS. Recall that the client's role in the DNS is relatively simple – a client sends a *query* to its local DNS server and receives a *response* back. As shown in Figures 2.19 and 2.20 in the textbook, much can go on "under the covers," invisible to a DNS client, as the hierarchical DNS servers communicate with each other to either recursively or iteratively resolve the client's DNS query. From the DNS client's standpoint, however, the protocol is quite simple – a query is formulated to the local DNS server and a response is received from that server.

Before beginning this lab, you'll probably want to review DNS by reading Section 2.4 of the text. In particular, you may want to review the material on **local DNS servers**, **DNS caching**, **DNS records and messages**, and the **TYPE field** in the DNS record. Another good source of information is the RFC https://www.ietf.org/rfc/rfc1035.txt.

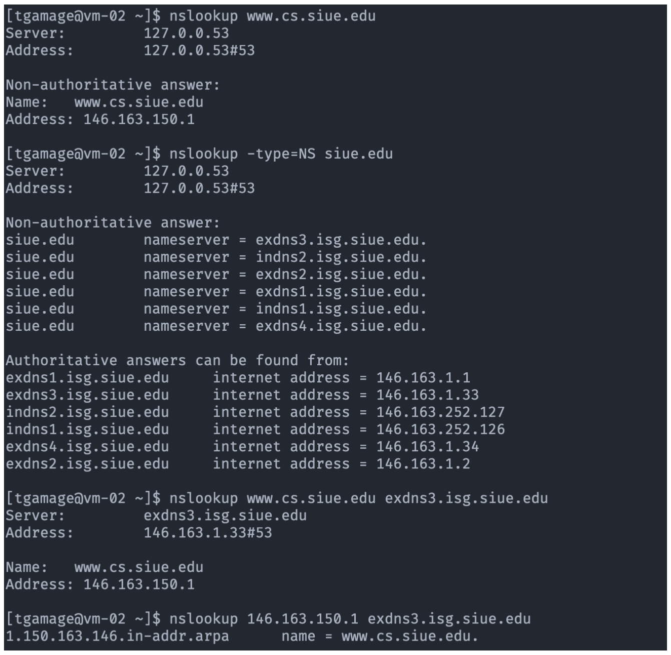
• nslookup

In this lab, we'll make extensive use of the `nslookup` tool, which is available in pretty much all major operating system platforms. Simply open a terminal (Linux and iOS) or command prompt (Windows) and type `nslookup`.

In its most basic operation, `nslookup` allows the host running `nslookup` to query any specified DNS server for a DNS record. The queried DNS server can be a root DNS server, a top-level-domain (TLD) DNS server, an authoritative DNS server, or an intermediate DNS server (see the textbook for definitions of these terms). For example, `nslookup` can be used to retrieve a "`Type=A`" DNS record that maps a hostname (e.g., `www.siue.edu`) to its IP address. To accomplish this task, `nslookup` sends a DNS query to the specified DNS server (or the default local DNS server for the host on which `nslookup` is run, if no specific DNS server is specified), receives a DNS response from that DNS server, and displays the result.

The below screenshot shows the results of several independent `nslookup` commands executed on a Linux Terminal. In this example, the client host is the CS home server.

[1] References to sections are for the 7th edition of our text, *Computer Networks, A Top-down Approach, 7th ed., J.F. Kurose and K.W. Ross, Addison-Wesley/Pearson, 2016.* Check the authors' website http://gaia.cs.umass.edu/kurose_ross for lots of interesting open material there.



```
[tgamage@vm-02 ~]$ nslookup www.cs.siue.edu
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   www.cs.siue.edu
Address: 146.163.150.1

[tgamage@vm-02 ~]$ nslookup -type=NS siue.edu
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
siue.edu        nameserver = exdns3.isg.siue.edu.
siue.edu        nameserver = indns2.isg.siue.edu.
siue.edu        nameserver = exdns2.isg.siue.edu.
siue.edu        nameserver = exdns1.isg.siue.edu.
siue.edu        nameserver = indns1.isg.siue.edu.
siue.edu        nameserver = exdns4.isg.siue.edu.

Authoritative answers can be found from:
exdns1.isg.siue.edu     internet address = 146.163.1.1
exdns3.isg.siue.edu     internet address = 146.163.1.33
indns2.isg.siue.edu     internet address = 146.163.252.127
indns1.isg.siue.edu     internet address = 146.163.252.126
exdns4.isg.siue.edu     internet address = 146.163.1.34
exdns2.isg.siue.edu     internet address = 146.163.1.2

[tgamage@vm-02 ~]$ nslookup www.cs.siue.edu exdns3.isg.siue.edu
Server:         exdns3.isg.siue.edu
Address:        146.163.1.33#53

Name:   www.cs.siue.edu
Address: 146.163.150.1

[tgamage@vm-02 ~]$ nslookup 146.163.150.1 exdns3.isg.siue.edu
1.150.163.146.in-addr.arpa      name = www.cs.siue.edu.
```

Consider the first command:

```
nslookup www.cs.siue.edu
```

Here, we are asking the question *"What is the IP address of www.cs.siue.edu"?* from our local DNS server. We do get an answer to our question, but notice it says it's a non-authoritative answer, which in essence means that the result was fetched from some DNS cache somewhere. The local DNS server in this case is the same machine from which the `nslookup` command was run; the first line of the response identifies the responding server; the second line identifies the socket address of the responding DNS server.

Let's now consider the second command:

```
nslookup -type=NS siue.edu
```

Now we are asking the question *"Who is/are the authoritative DNS server(s) for the siue.edu domain"?* Feel free to experiment and query nameservers of other domains you are curious about. Also, there are several other TYPE options that you can pass to `nslookup` that might be worth checking out. In fact, if a TYPE is not defined, the default behavior is a "TYPE=A", which answers the query using the local DNS server (like in the first command).

In the third command, we are using an authoritative server we've identified in the second command to get an "Authoritative answer" for our first question.

```
nslookup www.cs.siue.edu exdns3.isg.siue.edu
```

In other words, we are asking the question *"Hey! exdns3.isg.siue.edu. What is the IP address of www.cs.siue.edu"?* Notice, unlike in the first command, this time we are getting a direct answer from an authoritative DNS server (and not a cached answer).

`nslookup` also can be used to perform reverse DNS queries (i.e., IP address → DNS). This is the behavior you are seeing on the fourth command:

```
nslookup 146.163.150.1
```

In other words, we are asking the question *"What is the hostname of the device with IP address 146.163.150.1"?*

`nslookup` has a number of additional uses and options beyond what's listed here that you might want to explore. Refer to the "*man page*" (manual) either by typing `man nslookup` on your Linux/iOS terminal or visiting this website: https://linux.die.net/man/1/nslookup. Here's also another website with screenshots of ten popular `nslookup` uses: https://www.cloudns.net/blog/10-most-used-nslookup-commands/. The "*man page"* (manual), either type `man nslookup` on your Linux/iOS terminal or visit this web site.

Now that we have provided an overview of `nslookup`, it is time for you to test drive it yourself. Do the following and write down the results. Provide screenshots in your report as proof:

1. Run `nslookup` to obtain the IP address of a university web server in Canada.
   a. What is the IP address of that server?
   b. What is the IP address of the DNS server that answered your query?
2. Run `nslookup` to determine the authoritative DNS server(s) for a university in South Africa.
3. Run `nslookup` so that one of the DNS servers obtained in Question 2 is queried for the mail servers for Google Mail (Gmail). What is its IP address?

• The DNS cache on your computer

From the description of iterative and recursive DNS query resolution (Figures 2.19 and 2.20) in our textbook, you might think that the local DNS server must be contacted *every* time an application needs to translate from a hostname to an IP address. That's not always true in practice!

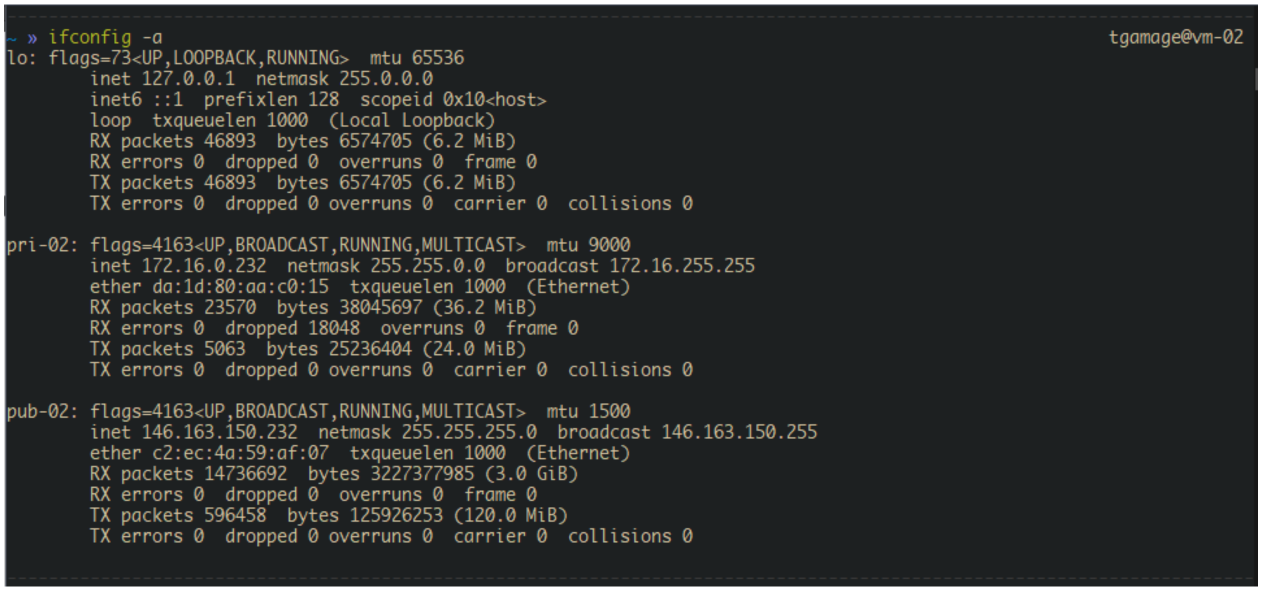
Most hosts (e.g., your personal computer) keep a *cache* of recently retrieved DNS records (sometimes called a DNS *resolver cache*), just like many Web browsers keep a cache of objects recently retrieved by HTTP. When DNS services need to be invoked by a host, that host will first check if the DNS record needed is resident in this host's DNS cache; if the record is found, the host will not even bother to contact the local DNS server and will instead use this cached DNS record. A DNS record in a resolver cache will eventually timeout and be removed from the resolver cache, just as records cached in a local DNS server (see Figures 2.19, 2.20) will timeout.

• ipconfig/ifconfig

`ipconfig` (for Windows) and `ifconfig` (for Linux/Unix) are among the most useful little utilities in your host, especially for debugging network issues. On Windows, `ipconfig` can be used to show your current TCP/IP information, including your address, DNS server addresses, adapter type and so on. Most Linux distributions do not cache DNS entries in this manner, unless a dedicated DNS reslover, such as `nsd`, `dnsmasq`, etc, is installed. Linux machines depend on the entries in the *etc/resolve.conf* to resolve DNS queries. However, `ifconfig` on Linux can be used for various other useful functions such as enabling promiscuous mode, setting net masks, tunnelling, etc. See the man page for `ifconfig` by typing `man ifconfig` on a Linux terminal for more information.

The following screenshot was obtained using a on a Linux terminal by typing:

```
ifconfig -a
```



```
~ » ifconfig -a                                                                    tgamage@vm-02
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 46893  bytes 6574705 (6.2 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 46893  bytes 6574705 (6.2 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

pri-02: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 9000
        inet 172.16.0.232  netmask 255.255.0.0  broadcast 172.16.255.255
        ether da:1d:80:aa:c0:15  txqueuelen 1000  (Ethernet)
        RX packets 23570  bytes 38045697 (36.2 MiB)
        RX errors 0  dropped 18048  overruns 0  frame 0
        TX packets 5063  bytes 25236404 (24.0 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

pub-02: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 146.163.150.232  netmask 255.255.255.0  broadcast 146.163.150.255
        ether c2:ec:4a:59:af:07  txqueuelen 1000  (Ethernet)
        RX packets 14736692  bytes 3227377985 (3.0 GiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 596458  bytes 125926253 (120.0 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

The Windows equivalent*, ipconfig,* is also very useful for managing the DNS information stored in your host. In Section 2.5 we learned that a host can cache DNS records it recently obtained. To see these cached records, after the prompt C:\> provide the following command:

```
ipconfig /displaydns
```

Each entry shows the remaining Time to Live (TTL) in seconds. You can also explicitly clear the records in your DNS cache. There's no harm in doing so – it just makes your computer invoke the distributed DNS service next time it needs to use the DNS name resolution service, since it will find no records in the cache.

- On Linux → `systemd-resolve –flush-caches`
- On Windows → `ipconfig /flushdns`
- On iOS → `sudo dscacheutil -flushcache;sudo killall -HUP mDNSResponder`

• Tracing DNS with Wireshark

Now that we are familiar with *nslookup* and *ipconfig*, we're ready to get down to some serious business. Let's first capture the DNS packets that are generated by ordinary Web-surfing activity.

- Clear the DNS cache in your host, as described above[2].
- Open your Web browser and clear your browser cache.

[2] On Linux, you may need to install `nscd` (using your package manager) and restart `nscd` service by typing `sudo nscd restart or sudo nscd –i hosts.` See *man page* here: https://linux.die.net/man/8/nscd

- Open Wireshark and enter `ip.addr == <your_IP_address>` into the display filter, where `<your_IP_address>` is the IPv4 address of your computer[3]. With this filter, Wireshark will only display packets that either originate from, or are destined to, your host.
- Start packet capture in Wireshark.
- With your browser, visit the Web page: https://www.iana.org/
- Stop packet capture.

Whenever possible, when answering a question below, you should hand in a screenshot showing proof of your work. Annotate your screenshots to explain your answer. You can also print packets and annotate them as well. To print a packet, use *File->Print*, choose *Selected packet only*, choose *Packet summary line,* and select the minimum amount of packet detail that you need to answer the question. Please don't turn in pages and pages of irrelevant packet prints.

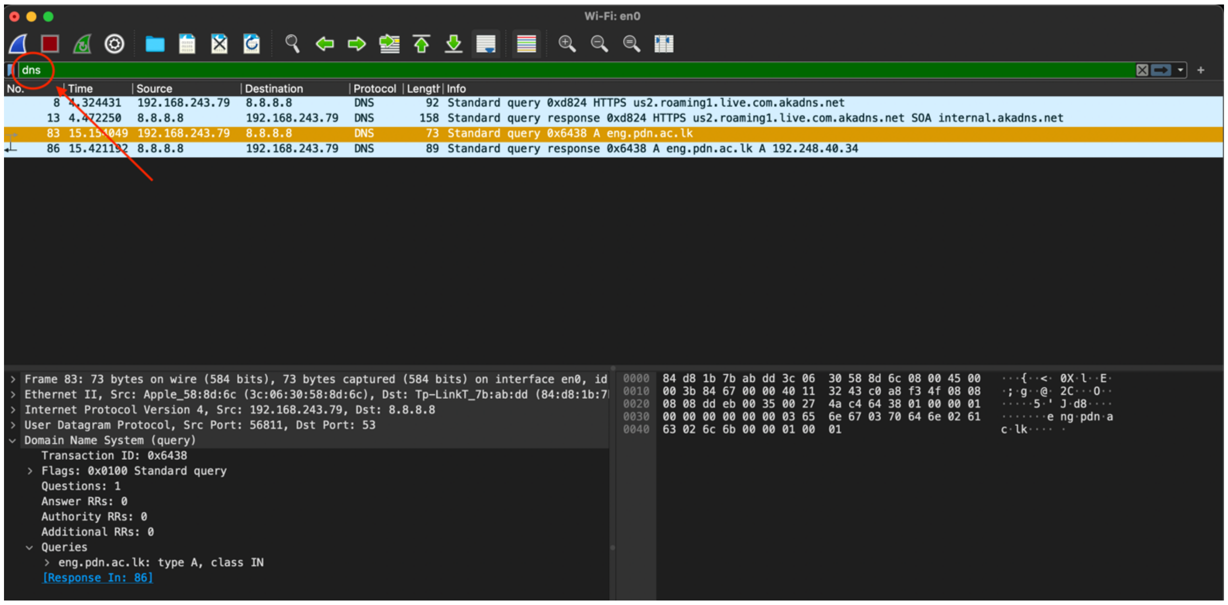
4. Locate the DNS query and response messages. Are they sent over UDP or TCP?
5. What is the destination port for the DNS query message? What is the source port of DNS response message?
6. To what IP address is the DNS query message sent? Use ipconfig to determine the IP address of your local DNS server. Are these two IP addresses the same?
7. Examine the DNS query message. What "TYPE" of DNS query is it? Does the query message contain any "answers"?
8. Examine the DNS response message. How many "answers" are provided? What do each of these answers contain?
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?
10. This web page contains images. Before retrieving each image, does your host issue new DNS queries?

Now let's play with *nslookup*[4].

- Start packet capture.
- Do an *nslookup* on eng.pdn.ac.lk
- Stop packet capture and answer the questions.

You should get a trace that looks something similar to the screenshot shown below. Note the use of a "*dns"* display filter to declutter captured packets and only show DNS packets. Also note how there are two sets of DNS query/responses and only the second one really corresponds to the command we issued using *nslookup*. If you happen to make similar observations, you can safely ignore the first set as it's specific to *nslookup* and are not normally generated by standard Internet applications. You should instead focus on the latter (or last) query and response messages.

[3] If you're not sure how to find the IP address of your computer, you can search the Web for articles for your operating system. Windows 10 info is here; Mac info is here; Linux info is here



```
No.   Time       Source          Destination     Protocol Length Info
    8 4.324431   192.168.243.79  8.8.8.8         DNS      92 Standard query 0xd824 HTTPS us2.roaming1.live.com.akadns.net
   13 4.472250   8.8.8.8         192.168.243.79  DNS     158 Standard query response 0xd824 HTTPS us2.roaming1.live.com.akadns.net SOA internal.akadns.net
   83 15.154049  192.168.243.79  8.8.8.8         DNS      73 Standard query 0x6438 A eng.pdn.ac.lk
   86 15.421192  8.8.8.8         192.168.243.79  DNS      89 Standard query response 0x6438 A eng.pdn.ac.lk A 192.248.40.34
```

```
> Frame 83: 73 bytes on wire (584 bits), 73 bytes captured (584 bits) on interface en0, id
> Ethernet II, Src: Apple_58:8d:6c (3c:06:30:58:8d:6c), Dst: Tp-LinkT_7b:ab:dd (84:d8:1b:7
> Internet Protocol Version 4, Src: 192.168.243.79, Dst: 8.8.8.8
> User Datagram Protocol, Src Port: 56811, Dst Port: 53
v Domain Name System (query)
    Transaction ID: 0x6438
  > Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  v Queries
    > eng.pdn.ac.lk: type A, class IN
    [Response In: 86]
```

```
0000  84 d8 1b 7b ab dd 3c 06  30 58 8d 6c 08 00 45 00   ···{··<· 0X·l··E·
0010  00 3b 84 67 00 00 40 11  32 43 c0 a8 f3 4f 08 08   ·;·g··@· 2C···O··
0020  08 08 dd eb 00 35 00 27  4a c4 64 38 01 00 00 01   ·····5·' J·d8····
0030  00 00 00 00 00 00 03 65  6e 67 03 70 64 6e 02 61   ·······e ng·pdn·a
0040  63 02 6c 6b 00 00 01 00  01                        c·lk···· ·
```

11. What is the destination port for the DNS query message? What is the source port of DNS response message?
12. To what IP address is the DNS query message sent? Is this the IP address of your default local DNS server?
13. Examine the DNS query message. What "TYPE" of DNS query is it? Does the query message contain any "answers"?
14. Examine the DNS response message. How many "answers" are provided? What do each of these answers contain?
15. Provide a screenshot.

Now repeat the previous experiment, but instead issue the command:

```
nslookup -type=MX umass.edu
```

Answer the following questions:

16. To what IP address is the DNS query message sent? Is this the IP address of your default local DNS server?
17. Examine the DNS query message. What "TYPE" of DNS query is it? Does the query message contain any "answers"?
18. Now perform a similar `nslookup` for the `mit.edu` domain but this time for nameservers. 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 nameservers?

Now repeat the previous experiment, but this time query the IP address of `eng.pdn.ac.lk` from one of the authoritative nameservers you discovered in Q18. For example:

```
nslookup eng.pdn.ac.lk usw2.akam.net
```

Answer the following questions:

19. 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?
20. What is the transaction ID of the DNS query message? Does the corresponding DNS response message also has the same transaction ID?
21. Examine the DNS response message. Did you get any “answers”?
    a. If you did, how many answers did you get?
22. If you didn’t receive any “answers” in your DNS response message, examine the DNS response message flags.
    a. What is the reply code (RCODE) corresponding to? What does it mean *(do some research).*
    b. Is your query a recursive or iterative?
    c. What can you do to get an “answer”? What is the corresponding `nslookup` command syntax?