CSC358H5: Principles of Computer Networking — Winter 2025

Wireshark Lab 1: Domain Name System (DNS)

Due Date: Sunday, Feb 23, 11:59 PM, on Quercus

In this lab, you will look into the details of one of the most important protocols in the internet, which provides network functionality of "name resolution". WL1 is contributes 1% towards your course grade and will be autograded. Please submit your answers to the questions in the handout to the corresponding quiz implemented on Quercus. It's noteworthy that this Labs are modeled after similar labs in "Computer Networking: A Top-Down Approach," by Jim Kurose and Keith Ross.

1 What is DNS?

The Domain Name System (DNS) protocol is a fundamental part of the internet that translates human-readable domain names (like google.com) into IP addresses (like 142.251.32.67) that computers use to locate and communicate with each other. This allows you to simply type human readable URL like https://www.google.com on your browser instead of 142.251.32.67.

The following are the relevant terms when discussing DNS:

- **DNS Query**: A request with the hostname to some DNS Server to retrieve the corresponding IP address. There are two types of DNS queries recursive and iterative.
- **DNS Response**: A response to the query containing the IP address if found, otherwise contains some error message indicating the hostname was not found.
- **DNS Server**: Receives DNS queries and generates the DNS response, for instance the Google Public DNS is an example of a DNS Server provided by Google.

2 Name Servers

It would be great if there was single centralized server that stored a mapping from every domain to every IP address that everyone could query, but unfortunately, there is no server big enough to store the IP address of every domain on the Internet and fast enough to handle the volume of DNS requests generated by the entire world. Instead, DNS uses a collection of many name servers, which are servers dedicated to replying to DNS requests.

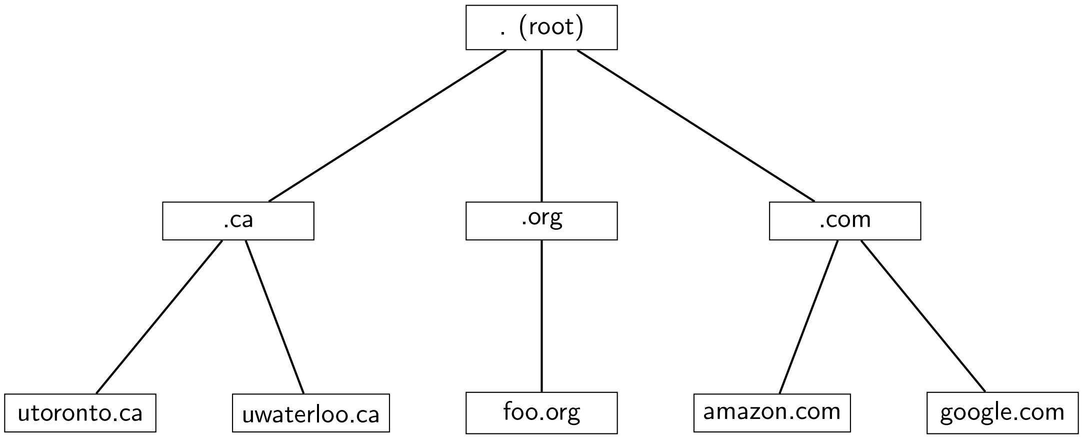
Each name server is responsible for a specific zone of domains, so that no single server needs to store every domain on the Internet. For example, a name server responsible for the `.com` zone only needs to answer queries for domains that end in .com.com. This name server doesn't need to store any DNS information related to `wikipedia.org`. Likewise, a name server responsible for the `utoronro.ca` zone doesn't need to store any DNS information related to `uwaterloo.ca`.

Even though it has a special purpose (responding to DNS requests), a name server is just like any other server you can contact on the Internet–each one has a human-readable domain name (*e.g.*, `a.edu-servers.net`) and a computer-readable IP address (*e.g.*, `192.5.6.30`). Be careful not to confuse the domain name with the zone. For example, this name server has `.net` in its domain, but it responds to DNS requests for `.edu` domains.

3 Name Server Hierarchy and DNS Lookup (Conceptual)

When you query a name server, instead of always returning the IP address of the domain you queried, the name server can also direct you to another name server for the answer. This allows name servers with large zones such as `.ca` to redirect your query to other name servers with smaller zones such as `utoronto.ca`. Now, the name server for the `.ca` zone doesn't need to store any information about `ece.utoronto.ca`, `stat.utoronto.ca`, etc. Instead, the `.ca` name server stores information about the `utoronto.ca` name server and redirects requests for `ece.utoronto.ca`, `stat.utoronto.ca`, etc. to a `utoronto.ca` name server.

DNS arranges all the name servers in a tree hierarchy based on their zones: The root server at the top level of



the tree has all domains in its zone (this zone is usually written as `.`). Name servers at lower levels of the tree have smaller, more specific zones.

DNS queries always start at the root. The root will direct your query to one of its children name servers. Then you make a query to the child name server, and that name server redirects you to one of its children. The process repeats until you make a query to a name server that knows the answer, which will return the IP address corresponding to your domain. Once we get the answer, we can store it in our cache so that we don't have to ask again if we need this record again later.

4 DNS Query with dig

Domain Information Groper (dig) is a command-line tool for DNS querying that is available in most Unix operating systems. Let's walk through a real DNS query for the IP address of `ece.utoronto.ca`. You can try this at home with the dig utility – remember to set the `+norecurse` flag so you can unravel the recursion yourself.

Every DNS query begins with the root server. The names and IP addresses of the root servers are usually hard-coded into resolvers, in the form of a root hints file. Here you can check out a root hints file if you're curious.

The first root server has domain `a.root-servers.net` and IP address `198.41.0.4`. We can use dig to send a DNS request to this address, asking for the IP address of `ece.utoronto.ca`.

Linux Command: Sending a DNS request to root server `a.root-servers.net`

```
dig +norecurse ece.utoronto.ca @198.41.0.4
```

Output

```
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 15507
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 4, ADDITIONAL: 9

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
```

```
;ece.utoronto.ca.               IN      A

;; AUTHORITY SECTION:
ca.                     172800  IN      NS      d.ca-servers.ca.
ca.                     172800  IN      NS      any.ca-servers.ca.
ca.                     172800  IN      NS      c.ca-servers.ca.
ca.                     172800  IN      NS      j.ca-servers.ca.

;; ADDITIONAL SECTION:
d.ca-servers.ca.        172800  IN      A       45.142.220.101
d.ca-servers.ca.        172800  IN      AAAA    2a0e:dbc0::101
any.ca-servers.ca.      172800  IN      A       199.4.144.2
any.ca-servers.ca.      172800  IN      AAAA    2001:500:a7::2
c.ca-servers.ca.        172800  IN      A       185.159.196.2
c.ca-servers.ca.        172800  IN      AAAA    2620:10a:8053::2
j.ca-servers.ca.        172800  IN      A       198.182.167.1
j.ca-servers.ca.        172800  IN      AAAA    2001:500:83::1
...
```

In the first section of the answer, we can see the header information, including the ID field (15507), the return flags (NOERROR), and the number of records returned in each section.

The **question section** contains 1 record (you can verify by seeing QUERY: 1 in the header). It has key `ece.utoronto.ca`, type A, and a blank value. This represents the domain we queried for (the value is blank because we don't know the corresponding IP address).

The **answer section** is blank (ANSWER: 0 in the header), because the root server didn't provide a direct answer to our query.

The **authority section** contains 4 records. The first one has key .ca, type NS, and value `d.ca-servers.ca`. This is the root server giving us the zone and the domain name of the next name servers we could contact. Each record in this section corresponds to a potential name server we could ask next. Usually, the server with the earliest name (alphabetically or numerically) is the primary server (here, `d.ca-servers.ca`), and the rest are mirrors. Also, note that it's okay that there are multiple records with the same name (.ca). This just tells us there are multiple name servers that can all answer queries for this zone.

The **additional section** contains 9 records. The first one has key `d.ca-servers.ca`, type A, and value `45.142.220.101`. This is the root server giving us the IP address of the next name server by mapping a domain from the authority section to an IP address.[1]

Together, the authority section and additional section combined give us the zone, domain name, and IP address of the next name server. This information is spread across two sections to maintain the key-value structure of the DNS message. Thus, just like before, we query `d.ca-servers.ca`, whose location we know because of the records in the additional section, for the IP address of `ece.utoronto.ca`.

Linux Command: Sending a DNS request to `d.ca-servers.ca`

```
dig +norecurse ece.utoronto.ca @45.142.220.101
```

[1]For completeness, note that the value 172800 in the answer denotes the TTL (time-to-live) for each record, here set as 172800 seconds (*i.e.*, 48 hours). Furthermore, the IN is the Internet class and can basically be ignored. Sometimes you will see records of type AAAA, which correspond to IPv6 addresses. The usual A type records correspond to IPv4 addresses.

Output

```
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 2642
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 2, ADDITIONAL: 5

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1452
; COOKIE: b3c2d4a5264289620cc6074467a963f5061e24c19d4bead6 (good)
;; QUESTION SECTION:
;ece.utoronto.ca.               IN      A

;; AUTHORITY SECTION:
utoronto.ca.            86400   IN      NS      ns2.d-zone.ca.
utoronto.ca.            86400   IN      NS      ns1.d-zone.ca.

;; ADDITIONAL SECTION:
ns1.d-zone.ca.          86400   IN      A       162.219.54.2
ns2.d-zone.ca.          86400   IN      A       162.219.55.2
ns1.d-zone.ca.          86400   IN      AAAA    2620:10a:80eb::2
ns2.d-zone.ca.          86400   IN      AAAA    2620:10a:80ec::2
...
```

This query also had an empty answer section, with NS records in the authority section and A records in the additional section which give us the domains and IP addresses of name servers responsible for the `utoronto.ca` zone.

Linux Command: Sending a DNS request to `ns1.d-zone.ca`

```
dig +norecurse ece.utoronto.ca @162.219.54.2
```

Output

```
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 53706
;; flags: qr; QUERY: 1, ANSWER: 0, AUTHORITY: 3, ADDITIONAL: 4

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;ece.utoronto.ca.               IN      A

;; AUTHORITY SECTION:
ece.utoronto.ca.        86400   IN      NS      ns2.utoronto.ca.
ece.utoronto.ca.        86400   IN      NS      ns7.utoronto.ca.
ece.utoronto.ca.        86400   IN      NS      dns1.ece.utoronto.ca.

;; ADDITIONAL SECTION:
dns1.ece.utoronto.ca.   86400   IN      A       128.100.10.237
ns2.utoronto.ca.        86400   IN      A       128.100.72.168
ns7.utoronto.ca.        86400   IN      A       162.243.71.42
...
```

This query also had an empty answer section, with NS records in the authority section and A records in the additional section which give us the domains and IP addresses of name servers responsible for the `ece.utoronto.ca` zone. Finally, the last query gives us the IP address corresponding to `ece.utoronto.ca` in the form of a single A type record in the answer section.

Linux Command: Sending a DNS request to `dns1.ece.utoronto.ca`

```
dig +norecurse ece.utoronto.ca @128.100.10.237
```

Output

```
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 47362
;; flags: qr aa; QUERY: 1, ANSWER: 1, AUTHORITY: 5, ADDITIONAL: 3

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 1c248789587d28c90100000067a967eb36e8e3bc0e830b3c (good)
; EDE: 18 (Prohibited)
;; QUESTION SECTION:
;ece.utoronto.ca.               IN      A

;; ANSWER SECTION:
ece.utoronto.ca.        86400   IN      A       128.100.82.35

;; AUTHORITY SECTION:
ece.utoronto.ca.        86400   IN      NS      zakspeed.eecg.utoronto.ca.
ece.utoronto.ca.        86400   IN      NS      ns2.utoronto.ca.
ece.utoronto.ca.        86400   IN      NS      tesla.comm.utoronto.ca.
ece.utoronto.ca.        86400   IN      NS      dns1.ece.utoronto.ca.
ece.utoronto.ca.        86400   IN      NS      dns2.ece.utoronto.ca.

;; ADDITIONAL SECTION:
dns1.ece.utoronto.ca.   86400   IN      A       128.100.10.237
dns2.ece.utoronto.ca.   86400   IN      A       128.100.10.238
...
```

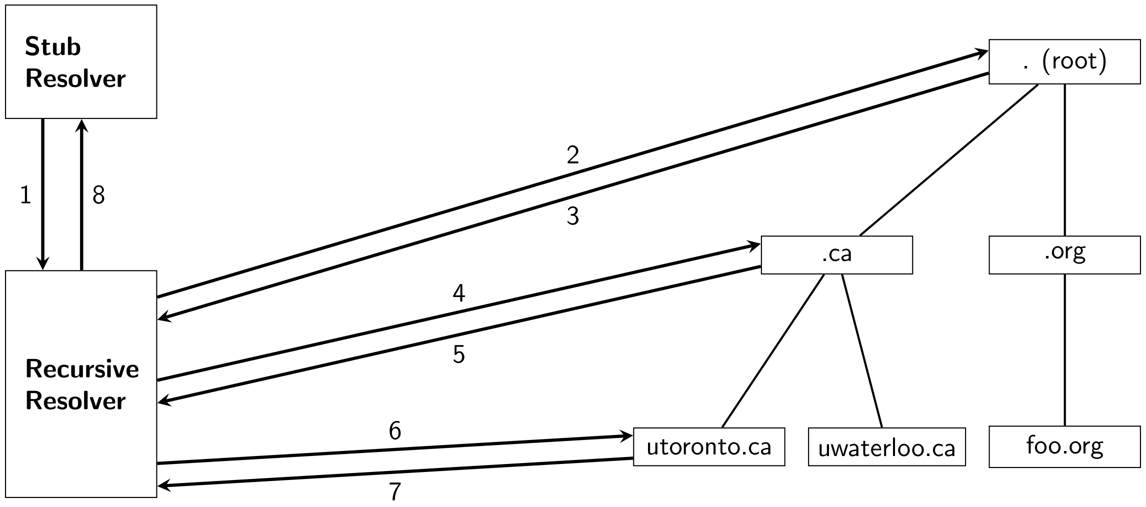
In practice, because the recursive resolver caches as many answers as possible, most queries can skip the first few steps and used cached records instead of asking root servers and high-level name servers like `.ca` every time. Caching helps speed up DNS, because fewer packets need to be sent across the network to translate a domain name to an IP address. Caching also helps reduce request load on the highest-level name servers.

5 Stub Resolvers and Recursive Resolvers

Originally, these iterative DNS lookups were done by the end host (*e.g.*, your computer), contacting each name server. Nowadays, your local computer usually delegates the task of DNS lookups to a DNS Recursive Resolver, which queries the name servers for you. When performing a lookup, the DNS Stub Resolver on your computer sends a query to the recursive resolver, lets the resolver do all the work, and receives the response back from the resolver.

Some well-known resolvers on the Internet include 1.1.1.1 (run by Cloudflare) and 8.8.8.8 (run by Google). They often have memorable IP addresses so we don't have to refer to them by name. Otherwise, we'd have to do a DNS lookup to find their IP address, and the whole point of these servers is to do DNS lookups for us. In addition to tech companies, Internet Service Providers (ISPs) also operate resolvers, as part of the Internet service they sell to customers. The router in your home can also act as a resolver. DNS queries can be faster if you use a router that's physically close to you.

One major benefit of a resolver is better caching. The resolver processes queries from lots of different end hosts (not just your own computer), so it builds a much larger cache. If you ask the resolver about `ece.utoronto.ca`, and somebody before you asked the resolver that same question recently, the resolver can give you the answer without contacting any additional name servers. Note that even though recursive resolvers store larger caches, the stub resolver can still maintain its own separate cache. Some queries can get answered by the stub resolver's cache, without even asking the recursive resolver.



6 DNS Query with nslookup

`nslookup` is a command-line tool that is available in most Microsoft, Apple iOS, or Linux operating systems. It essentially sends a DNS query to a specified DNS server and receives a DNS response.
To test out `nslookup`, run the command on a hostname, for instance:

Linux Command

```
nslookup ece.utoronto.ca
```

Output

```
Server:         10.24.1.1
Address:        10.24.1.1#53

Non-authoritative answer:
Name:   ece.utoronto.ca
Address: 128.100.82.35
```

`nslookup` essentially asks for the IP address for the host `ece.utoronto.ca`. The response from this command provides multiple pieces of information:

1. The query is being processed by the local DNS server at IP address 10.24.1.1
2. The server is running on port 53, the standard DNS port.
3. **Non-authoritative answer:**
    - The DNS response is non-authoritative, meaning the information is not directly retrieved from the authoritative DNS server for the domain, there have been some intermediate queries.

- The domain `ece.utoronto.ca` is associated with IP address `128.100.82.35`

To retrieve an authoritative answer the -type=NS flag can be added to the nslookup command. The NS type refers to **name servers** for the domain `ece.utoronto.ca`. Name servers handle DNS queries and returns a DNS response within their local area.

Linux Command

```
nslookup -type=NS ece.utoronto.ca
```

Output

```
Server:         10.24.1.1
Address:        10.24.1.1#53

Non-authoritative answer:
ece.utoronto.ca nameserver = dns2.ece.utoronto.ca.
ece.utoronto.ca nameserver = dns1.ece.utoronto.ca.
ece.utoronto.ca nameserver = ns2.utoronto.ca.
ece.utoronto.ca nameserver = zakspeed.eecg.utoronto.ca.
ece.utoronto.ca nameserver = tesla.comm.utoronto.ca.

Authoritative answers can be found from:
ns2.utoronto.ca internet address = 128.100.72.168
dns1.ece.utoronto.ca    internet address = 128.100.10.237
```

The **Authoritative answers** section provides the nameserver and the IP addresses of these nameservers:

- `ns2.utoronto.ca` with IP address `128.100.72.168`
- `dns1.ece.utoronto.ca` with IP address `128.100.10.237`

Additionally nslookup can retrieve the hostname associated with an IP address. In the example below, IP address 128.100.82.35 is resolved to the host name with that address gaia.cs.umass.edu

Linux Command

```
nslookup 128.100.82.35
```

Output

```
35.82.100.128.in-addr.arpa      name = www.ece.utoronto.ca.

Authoritative answers can be found from:
100.128.in-addr.arpa    nameserver = ns1.d-zone.ca.
100.128.in-addr.arpa    nameserver = ns2.d-zone.ca.
ns2.d-zone.ca   internet address = 162.219.55.2
ns1.d-zone.ca   internet address = 162.219.54.2
ns2.d-zone.ca   has AAAA address 2620:10a:80ec::2
ns1.d-zone.ca   has AAAA address 2620:10a:80eb::2
```

7 DNS Cache

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 will timeout.

Records in a DNS cache can also be cleared manually. There's no harm in doing so – it will just mean that the computer will need to 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. The DNS cache can be cleared with the following command.

- **On Mac:** `sudo killall -HUP mDNSResponder`
- **On Windows:** `ipconfig /flushdns`
- **On Linux:** `sudo systemctl restart systemd-resolved`

8 Submission

Submit your answer to the following questions to "Wireshark Lab 1" Quiz on Quercus.

8.1 Creating DNS Query with nslookup

Q1 Run `nslookup` to obtain the IP address of the UTM MCS web server. What is the IP address of `mcs.utm.utoronto.ca`?

Q2 Use the `nslookup` command to determine the name of the authoritative name server for the `cs.toronto.edu` domain. What is that name? (If there is more than one authoritative server, what is the name of the first authoritative server returned by `nslookup`?)

8.2 Tracing DNS with Wireshark – Part 1

Download `dns-wireshark-trace1-1.pcapng` from Quercus, available under the section `Files/Wireshark Labs/Lab 1`, and answer the following questions. Note that the packet trace file was captured while browsing ordinary websites on Jim Kurose's computer.

Q3 Locate the first DNS query message resolving the name `gaia.cs.umass.edu`. What is the packet number in the trace for the DNS query message? Is this query message sent over UDP or TCP? Note that TCP and UDP are transport layer protocols.

Q4 What is the **destination port** for the DNS query message? What is the **source port** of the DNS response message? Port number is the addressing scheme used by L4 (*i.e.*, Transport layer), which we'll discuss later in the course.

Q5 To what IP address is the DNS query message sent?

Q6 Examine the DNS query message. How many "questions" does this DNS message contain? How many answers does it contain?

Q7 Now locate the corresponding DNS response to the initial DNS query. What is the packet number in the trace for the DNS response message? Is this response message received via UDP or TCP?

Q8 Examine the DNS response message to the initial query message. How many "questions" does this DNS message contain? How many "answers" does it contain?

Q9 The web page for the base file `http://gaia.cs.umass.edu/kurose_ross/` references the image object `http://gaia.cs.umass.edu/kurose_ross/header_graphic_book_8E_2.jpg`, which, like the base webpage, is on `gaia.cs.umass.edu`.

9.a What is the packet number in the trace for the initial HTTP `GET` request for the base file `http://gaia.cs.umass.edu/kurose_ross/`?

9.b What is the packet number in the trace for the HTTP `GET` request for the image object `http://gaia.cs.umass.edu/kurose_ross/header_graphic_book_8E_2.jpg`?

9.c What is the packet number in the trace for the DNS query made to resolve `gaia.cs.umass.edu` so that this second HTTP request can be sent to the `gaia.cs.umass.edu` IP address?

8.3 Tracing DNS with Wireshark – Part 2

Download the `dns-wireshark-trace2-1.pcapng` file from Quercus, available under the section `Files/Wireshark Labs/Lab 1`. Note that the packet trace file was captured while performing `nslookup` on `www.cs.umass.edu`. Look at the first type A query (which is packet number 19, and indicated by the "A" in the Info column for that packet) and answer the following questions:

Q10 What is the destination port for the DNS query message? What is the source port of the DNS response message?

Q11 To what IP address is the DNS query message sent? Is this the IP address of your default local DNS server?

Q12 Examine the DNS query message. What "Type" of DNS query is it? Does the query message contain any "answers"?

Q13 Examine the DNS response message to the query message. How many "questions" does this DNS response message contain? How many "answers"?

8.4 Tracing DNS with Wireshark – Part 3

Download the `dns-wireshark-trace3-1.pcapng` file from Quercus, available under the section `Files/Wireshark Labs/Lab 1`. Answer the following questions:

Q14 Examine the DNS query message. How many questions does the query have? Does the query message contain any "answers"?

Q15 Examine the DNS response message. How many answers does the response have? What information is contained in the answers? How many additional resource records are returned? What additional information is included in these additional resource records?