Exercise Session 4: The Domain Name System

COM-208: Computer Networks

When a process running in the application layer of your computer wants to communicate with another, remote process, it must form the remote process's name; for that, it needs to know the IP address of the network interface behind which the remote process is running.

For example, when you type a URL in your web browser, the latter must form the process name of the web server that stores that URL; for that, it needs to know the IP address of the network interface behind which the web server is running.

To form the process name of the target web server, the web browser extracts the Domain Name System (DNS) name from the URL and asks a DNS client for the corresponding IP address.

In today's exercise session you will to get a sense of how the DNS essentially works.

Lab: The DNS protocol

The `dig` utility relies on the DNS protocol to provide information related to DNS names and IP addresses. It is similar to the `host` utility (that you used in Exercise Session 1), but provides more detailed information.

Start a Wireshak capture and type "dns" in the filter to catch DNS messages. Run "`dig adelaide.edu.au`". Stop the capture and answer the following questions:

- Based on the captured packets, what transport-layer protocol does DNS use? what port number is associated with the DNS protocol?
- Why do you think that DNS uses this transport-layer protocol? Summarize the advantages and disadvantages of this choice.

Lab: DNS resource records, questions and answers

DNS servers store information in the form of DNS **resource records** (RRs), of different types. DNS clients and servers generate DNS **queries** (or “questions” or “requests”), while DNS servers provide DNS **answers** (or “responses”) that contain RRs. A DNS message may carry multiple questions and/or answers.

- What kind of information do the following RR types provide: A, CNAME, PTR, MX, NS, and SOA? You can find the answer on Wikipedia and/or RFC1033 (or you can just google it, and you will see what that is).

Now you know the kind of information different RR types provide. Use these information to answer the next parts of this lab exercise.

DNS Lookup

- What is the IP address of `epfl.ch`? Which RR type stores the information needed to answer this question?
- What is the DNS name associated with IP address obtained in previous question? Which RR type stores the information needed to answer this question?

  To do reverse lookup, use the ‘`-x`’ option; you can view more details about the option using “`man dig`”.

Authoritative and local DNS servers

Each lower-level domain, e.g., epfl.ch, has a set of **authoritative DNS servers**, which store all the latest information that the DNS system has about this domain.

When a DNS server provides a DNS answer that concerns a domain for which the server is authoritative, we say that the answer itself is **authoritative**.

- Which are the authoritative DNS servers for epfl.ch? What RR type stores the information needed to answer this question?

Your computer (like any Internet end-system in the world) knows the IP address(es) of one or more **local DNS servers**. When a DNS client process running in the application layer of your computer (e.g., `dig`) needs information from the DNS system, it sends a DNS question to one of these local DNS servers.

- Look carefully at the answers provided by `dig` so far. Can you identify in them the IP address of the local DNS server used by your computer? Are you using one of the authoritative DNS servers for epfl.ch as your local DNS server?

A DNS client can send a DNS message to any DNS server in the world; it is not obligated to contact only the local DNS servers. If you run: "`dig @*IP address* ...`" then `dig` will send its DNS question to the DNS server that has the specified `*IP address*`.

- Ask the DNS server with IP address `8.8.8.8` for the "mail servers" that serve the epfl.ch domain. Did you get an authoritative answer? Hint: look at the `HEADER` flags.
- What do you need to do to get an authoritative answer to your question?

DNS caching and time-to-live (TTL)

DNS clients and servers – at all levels of the DNS hierarchy – **cache** the RRs they receive. To prevent inconsistency between authoritative and cached RRs, each RR is associated with a **time to live** (TTL), which indicates until when the RR is expected to be valid, hence until when it should be cached.

Imagine that the EPFL sysadmins need to urgently change the names of the mail servers that serve epfl.ch. Hence, they login to the authoritative DNS servers for epfl.ch and change the RR that specifies the mail-server names, before the RR's TTL has expired.

- What will happen now if a DNS client asks `8.8.8.8` for the mail servers that serve epfl.ch? How long will it take until `8.8.8.8` can answer this question correctly?
- What could the EPFL sysadmins do to make the change as quickly as possible without causing any inconsistency in the DNS system?

A quick recap: DNS hierarchy and queries

Whenever you execute the '`dig`' command, the application queries a DNS client; the DNS client then asks a local DNS server.

However, the query process does not always end there.

If the local DNS server does not know the answer, it asks another DNS server, according to a DNS hierarchy that consists of three kinds of DNS servers:

- A root server knows the IP address of at least one (typically several) top-level-domain (TLD) servers for each TLD.
- A TLD server knows the IP address of at least one (typically several) authoritative servers for each domain that falls under its TLD.
- An authoritative server knows the IP address of every DNS name that falls under its domain.

There are two ways in which the local DNS resolves the DNS query: **recursively** and **iteratively**. The two differ how a DNS server react when it receives the query but does not know the answer

- If the query is resolved recursively: the DNS server directly asks another DNS server that may know the answer; so, a root server asks a TLD server, and a TLD server asks an authoritative server.
- If a query is resolved iteratively: the DNS server returns the IP address of the other DNS server that may know the answer; so, a root server returns the IP address of a TLD server, and a TLD server returns the IP address of an authoritative server.

For the rest of the problems, assume: (1) all the DNS communication happens over UDP, and (2) all the DNS and web-browser caches are initially empty.

Also, assume that the web browsers and web servers communicate over *persistent* TCP connections, i.e., they use the same TCP connection to exchange multiple HTTP messages (so that they do not have to pay the TCP connection-setup cost for every new HTTP message they exchange).

In some problems, you will be asked to fill in a table, stating all the messages that were transmitted or received as a result of some action. For each message, briefly describe the goal, e.g., is this message an HTTP GET request for a particular URL? is it a DNS request for the IP address of a particular DNS name?

DNS and HTTP messages

You are working on an EPFL computer called `workstation.epfl.ch`. Your local DNS server is `ns.epfl.ch`. This DNS server knows the IP address of root server `a.root-servers.net`, which knows the IP address of .ch TLD server `a.nic.ch`, which knows the IP address of epfl.ch authoritative server `ns.epfl.ch` and unil.ch authoritative server `ns.unil.ch`. All these DNS servers perform *iterative* requests. Table 1 shows information about all the servers involved in this problem.

Server	DNS name	IP address
Root DNS server	`a.root-servers.net`	1.1.1.1
`.ch` TLD DNS server	`a.nic.ch`	2.2.2.2
EPFL DNS server	`ns.epfl.ch`	3.3.3.3
UNIL DNS server	`ns.unil.ch`	4.4.4.4
EPFL workstation	`workstation.epfl.ch`	5.5.5.5
UNIL web server	`www.unil.ch`	6.6.6.6

Table 1: Server DNS names and IP addresses.

- You open your web browser and type in `http://www.unil.ch/index.html`. This URL's base file does not reference any other URLs. In Table 2, list all the DNS and HTTP messages that get transmitted as a result of your action.

Packet	Source	Destination	Application protocol	Message
1	5.5.5.5	3.3.3.3	DNS	query: A for www.unil.ch
2				
3				
4				
5				
6				
7				
8				
9				
10				

Table 2: Transmitted DNS and HTTP messages.

- Immediately after retrieving this URL, you type in `http://www.unil.ch/logo.png`. In Table 3, list all the DNS and HTTP messages that get transmitted as a result of your action.

Packet	Source	Destination	Application protocol	Message
11				
12				

Table 3: Transmitted DNS and HTTP messages.

Adding a security twist

Three users, Alice, Bob, and Persa, are logged into their computers, respectively called `alice.ethz.ch`, `bob.ethz.ch`, and `persa.ethz.ch`, all located inside ETHZ's network.

ETHZ has web server `www.ethz.ch` and local DNS server `ns.ethz.ch`, which is also the authoritative server for the `ethz.ch` domain.

EPFL has web server `www.epfl.ch` and local DNS server `ns.epfl.ch`, which is also the authoritative server for the `epfl.ch` domain.

All DNS servers perform *recursive* requests.

Figure 2 illustrates the setup for this problem.

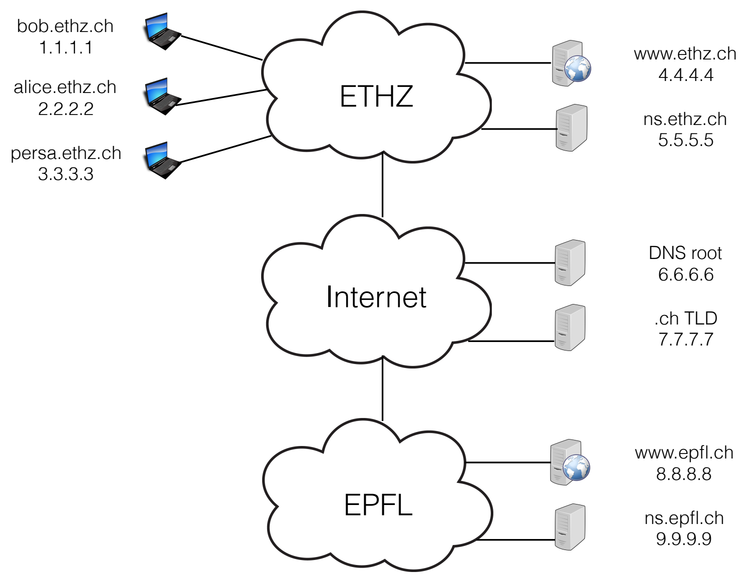

Figure 2: Question Setup

- Alice types in her web browser `http://www.epfl.ch/index.html`. This URL's base file references two other URLs, `http://www.epfl.ch/image.jpg` and `http://www.ethz.ch/file.html` (which does not reference any other URL).

  In Table 4, list all the application-layer messages and connection-setup packets that are transmitted as a result of this action.

Packet	Source IP	Dest. IP	Transport protocol	Application protocol	Purpose
ex.1	`1.0.0.1`	`1.0.0.2`	TCP	HTTP	HTTP reply with `image.jpg`

Table 4: Transmitted messages and connection-setup packets.

- After Alice has retrieved `http://www.epfl.ch/index.html`, Bob wants to access the same URL.

  Persa is a malicious user who guesses exactly when Bob tries to access `http://www.epfl.ch/index.html`. She wants to trick Bob and make him access a web server running on her own computer, thinking that he is accessing the EPFL web server.

How can Persa do that by sending DNS traffic to Bob?