Exercise Session 4: The Domain Name System - Solutions

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?

UDP.

The DNS protocol uses port number '53'. You can verify this by looking at the Wireshark traces. DNS query messages have port number 53 as their destination port, and DNS response messages have port number 53 as their source port.

- Why do you think that DNS uses this transport-layer protocol? Summarize the advantages and disadvantages of this choice.

The DNS query and DNS response messages are short and they can fit in one IP packet. By using UDP, we avoid the connection setup overhead of TCP, which involves a handshake that requires an extra round-trip time (RTT).

The disadvantage is that UDP does not provide reliable delivery of messages, thus DNS clients must keep track of the requests that were sent and wait for the replies. If the client receives no response within a particular amount of time, the client must either retransmit the original request, or time out and return an error.

For DNS, decreasing request latency to improve performance is considered an acceptable trade-off for the added complexity of having to deal with packet loss at the application layer.

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).

> **A** = address record: stores the IP address for a hostname.
>
> **CNAME** = canonical name record: a machine may have several names (aliases) associated with it; the CNAME record points from the aliases to the "main" one. The resolver would then query for the A record of the canonical name (but in practice the A record is usually returned together with the CNAME record to make the query faster). Alternatively, the domain's administrator could just create an A record for each alias, but then would have to make sure they are all modified consistently whenever the IP address is changed.
>
> **PTR** = pointer record; stores the canonical name for an IP address.
>
> **MX** = mail exchange record: stores the hostname of the e-mail server for the domain. These are servers that accept messages via SMTP.
>
> **NS** = indicates the hostname of the nameservers that are responsible (authoritative) for the domain.
>
> **SOA** = start-of-authority record: stores various administrative information about a domain: the name of the primary authoritative nameserver, the e-mail address of the administrator (note that the @ sign is replaced with a dot), the serial number of the configuration file, how often the secondary authoritative nameservers should synchronize with the primary etc.
>
> More types of DNS records, and corresponding details can be found on this Wikipedia Link.

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?

```
user@host:~$ dig epfl.ch

; <<>> DiG 9.16.1-Ubuntu <<>> epfl.ch
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 49
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL
    ↪ : 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;epfl.ch.                IN    A

;; ANSWER SECTION:
epfl.ch.         86400    IN    A    128.178.222.83

;; Query time: 0 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: mar oct 11 15:48:06 CEST 2022
;; MSG SIZE  rcvd: 52
```

The IP address of epfl.ch is `128.178.222.83`. To get this answer we make a type A query.

We can validate the result with Wireshark: use the filter "DNS" to show only DNS messages, then click on a query packet and show `Domain Name System (query)` → `Queries` → `epfl.ch: Type A, class IN`.

- 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`".

```
user@host:~$ dig -x 128.178.222.83

; <<>> DiG 9.16.1-Ubuntu <<>> -x 128.178.222.83
```

```
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4689
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL
    ↪ : 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;83.222.178.128.in-addr.arpa.    IN    PTR

;; ANSWER SECTION:
83.222.178.128.in-addr.arpa. 86400 IN    PTR    app-os-exopge.
    ↪ epfl.ch.

;; Query time: 0 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: mar oct 11 16:07:38 CEST 2022
;; MSG SIZE  rcvd: 91
```

The DNS name associated with `128.178.222.83` is `app-os-exopge.epfl.ch`. We made a type PTR query for the name `83.222.178.128.in-addr.arpa`.

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?

```
user@host:~$ dig epfl.ch NS

; <<>> DiG 9.16.1-Ubuntu <<>> epfl.ch NS
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 3016
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL
    ↪ : 1
```

```
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;epfl.ch.                IN    NS

;; ANSWER SECTION:
epfl.ch.         86400    IN    NS    stisun2.epfl.ch.
epfl.ch.         86400    IN    NS    stisun1.epfl.ch.

;; Query time: 0 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: mar oct 11 16:26:12 CEST 2022
;; MSG SIZE  rcvd: 80
```

The authoritative DNS servers for EPFL are `stisun1.epfl.ch` and `stisun2.epfl.ch`, and the type of resource records is `NS` as show in the answer section of the `dig` result above.

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?

This is shown by dig in the output for any query that does not use a user-specified nameserver:

```
user@host:~$ dig epfl.ch

; <<>> DiG 9.16.1-Ubuntu <<>> epfl.ch
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 49
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL
    ↪ : 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
```

```
;epfl.ch.                IN    A

;; ANSWER SECTION:
epfl.ch.         86400    IN    A    128.178.222.83

;; Query time: 0 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: mar oct 11 15:48:06 CEST 2022
;; MSG SIZE  rcvd: 52
```

In the above `dig` output, the IP address of local nameserver is `127.0.0.53`, and the port used is '53', which is the default port for DNS

The answer can also be found in the file `/etc/resolv.conf`:
`nameserver 127.0.0.53`

Note though that this address is a local (loopback) address! That means that the DNS server reported by dig is inside your computer! Why? This allows a faster DNS resolution. Your machine have a service called `systemd-resolved` which caches in the local machine DNS responses and sends DNS requests over the network only when the record is not available in the local cache. In addition to caching, `systemd-resolved` supports other more advanced DNS features, like DNS over TLS and DNSSEC for more secure and private DNS resolutions.

Nevertheless, whenever a request can not be served by the local cache, `systemd-resolved` has to ask a remote DNS server, to provide an answer. To find the DNS server that `systemd-resolved` will use, you can run `systemd-resolve -status` and inspect the Global/DNS Servers field:

```
user@host:~$\$$ systemd-resolve --status

Global
       LLMNR setting: no
MulticastDNS setting: no
  DNSOverTLS setting: no
      DNSSEC setting: no
    DNSSEC supported: no
          DNSSEC NTA: 10.in-addr.arpa
                      16.172.in-addr.arpa
                      168.192.in-addr.arpa
                      17.172.in-addr.arpa
                      18.172.in-addr.arpa
                      19.172.in-addr.arpa
                      20.172.in-addr.arpa
                      21.172.in-addr.arpa
                      22.172.in-addr.arpa
```

```
                    23.172.in-addr.arpa
                    24.172.in-addr.arpa
                    25.172.in-addr.arpa
                    26.172.in-addr.arpa
                    27.172.in-addr.arpa
                    28.172.in-addr.arpa
                    29.172.in-addr.arpa
                    30.172.in-addr.arpa
                    31.172.in-addr.arpa
                    corp
                    d.f.ip6.arpa
                    home
                    internal
                    intranet
                    lan
                    local
                    private
                    test

Link 2 (ens160)
      Current Scopes: DNS
DefaultRoute setting: yes
       LLMNR setting: yes
MulticastDNS setting: no
  DNSOverTLS setting: no
      DNSSEC setting: no
    DNSSEC supported: no
  Current DNS Server: 128.178.15.227
         DNS Servers: 128.178.15.227
                      128.178.15.228
          DNS Domain: ~.
                      intranet.epfl.ch
```

In this case, `systemd-resolved` relies on `128.178.15.227` and `128.178.15.228` which correspond to the authoritative DNS servers (`stisun1.epfl.ch` and `stisun2.epfl.ch`).

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.

```
user@host:~$\$$ dig @8.8.8.8 epfl.ch MX

; <<>> DiG 9.16.1-Ubuntu <<>> @8.8.8.8 epfl.ch MX
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 37617
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL
    ↪ : 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;epfl.ch.                IN    MX

;; ANSWER SECTION:
epfl.ch.         21512    IN    MX    50 mx3.epfl.ch.
epfl.ch.         21512    IN    MX    50 mx2.epfl.ch.
epfl.ch.         21512    IN    MX    50 mx1.epfl.ch.

;; Query time: 7 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: mar oct 11 17:21:34 CEST 2022
;; MSG SIZE  rcvd: 96
```

We see that Google's server cannot give us an authoritative answer, since the flags do not contain `aa`. This is because it does not have authority for the EPFL domain.

- What do you need to do to get an authoritative answer to your question?

To get an authoritative answer we can just query one of the authoritative nameservers for the domain epfl.ch:

```
user@host:~$\$$ dig stisun1.epfl.ch epfl.ch MX

; <<>> DiG 9.16.1-Ubuntu <<>> @stisun1.epfl.ch epfl.ch MX
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 5035
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 2,
    ↪ ADDITIONAL: 11
```

```
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 17dc11d99416782cac3fc9bc63458bfcf0430d1a9137ab86 (good)
;; QUESTION SECTION:
;epfl.ch.                IN    MX

;; ANSWER SECTION:
epfl.ch.         86400    IN    MX    50 smtp5.epfl.ch.
epfl.ch.         86400    IN    MX    50 smtp4.epfl.ch.
epfl.ch.         86400    IN    MX    50 smtp0.epfl.ch.

;; AUTHORITY SECTION:
epfl.ch.         86400    IN    NS    stisun2.epfl.ch.
epfl.ch.         86400    IN    NS    stisun1.epfl.ch.

;; ADDITIONAL SECTION:
smtp0.epfl.ch.           86400    IN    A    128.178.224.218
smtp4.epfl.ch.           86400    IN    A    128.178.224.219
smtp5.epfl.ch.           86400    IN    A    128.178.224.8
stisun1.epfl.ch.         86400    IN    A    128.178.15.8
stisun2.epfl.ch.         86400    IN    A    128.178.15.7
smtp0.epfl.ch.           86400    IN    AAAA    2001:620:618:1e0
    ↪ :1:80b2:e058:1
smtp4.epfl.ch.           86400    IN    AAAA    2001:620:618:1e0
    ↪ :1:80b2:e059:1
smtp5.epfl.ch.           86400    IN    AAAA    2001:620:618:1e0
    ↪ :1:80b2:e034:1
stisun1.epfl.ch.         86400    IN    AAAA    2001:620:618:10f
    ↪ :1:80b2:f08:1
stisun2.epfl.ch.         86400    IN    AAAA    2001:620:618:10f
    ↪ :1:80b2:f07:1

;; Query time: 7 msec
;; SERVER: 128.178.15.8#53(128.178.15.8)
;; WHEN: mar oct 11 17:30:04 CEST 2022
;; MSG SIZE  rcvd: 394
```

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?

> Since we query the DNS server of Google (`8.8.8.8`), the answer is not authoritative, thus not guaranteed to be correct. This is because Google may already have the old record in its cache; until the record expires (remaining TTL value), it keeps serving the old value without knowing that the value is incorrect.

- What could the EPFL sysadmins do to make the change as quickly as possible without causing any inconsistency in the DNS system?

> From the query results (`dig @stisun1.epfl.ch epfl.ch MX`), we can see that TTL value EPFL RRs are set to 86400 seconds, that is, 24 hours or 1 day. In the worst case scenario, where a DNS server would cache the EPFL RR just before the configuration change, it will take 24 hours for that cache to invalidate.
>
> In order for EPFL sysadmins to make change as quickly as possible, they should inspect the TTL of the MX records, in this case 24 hours. We change it to a small value (e.g. 1 second or even zero). Then we wait for 24 hours until the new record propagates to all servers that may have cached the old one. Now we can change the content of the MX record and set the TTL back to the old value.
>
> The downside is that for 24 hours EPFL's DNS server will get much higher DNS traffic due to those queries (since the caches of all the other resolvers on the Internet expire quickly). To avoid that, we can make the TTL change in 2 phases: first we change it to a few minutes, and after 24 hours we change it to 1 or 0 seconds. In this case we will get a very high volume of traffic only for a few minutes.

Summarizing the DNS hierarchy

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 servers (of all levels) talk with each other to get the answer. It goes in the following order:
  1. The local DNS server talks with a root server.
  2. The root server talks with a TLD server.
  3. And the TLD server talks with an authoritative server.
- If a query is resolved **iteratively**: the local DNS server handles asking the other DNS servers to get the final answer. The order goes like this:
  1. The local DNS server asks the root server for the IP of *a* TLD sever.
  2. The local DNS server asks the TLD server for the IP of *an* authoritative server.
  3. The local DNS server asks the authoritative server for the final answer.

Before you start

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

Also, 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	Purpose
1	5.5.5.5	3.3.3.3	DNS	query for www.unil.ch
2	3.3.3.3	1.1.1.1	DNS	query for www.unil.ch
3	1.1.1.1	3.3.3.3	DNS	replay: NS for .ch is a.nic.ch (2.2.2.2)
4	3.3.3.3	2.2.2.2	DNS	query for www.unil.ch
5	2.2.2.2	3.3.3.3	DNS	replay: NS for unil.ch is ns.unil.ch (4.4.4.4)
6	3.3.3.3	4.4.4.4	DNS	query for www.unil.ch
7	4.4.4.4	3.3.3.3	DNS	replay: 6.6.6.6 is IP of www.unil.ch
8	3.3.3.3	5.5.5.5	DNS	replay: 6.6.6.6 is IP of www.unil.ch
9	5.5.5.5	6.6.6.6	HTTP	HTTP GET /index.html
10	6.6.6.6	5.5.5.5	HTTP	HTTP OK {index.html}

Table 2: Transmitted DNS and HTTP messages.

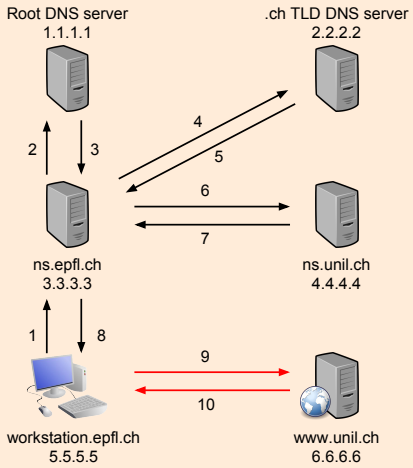

Figure 1: The messages transmitted in Table 2

- 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.

Since the DNS client have cached the IP address of the web server (6.6.6.6) , then no DNS lookup will be needed at this stage.

Packet	Source	Destination	Application protocol	Purpose
11	5.5.5.5	6.6.6.6	HTTP	HTTP GET /logo.png
12	6.6.6.6	5.5.5.5	HTTP	HTTP OK {logo.png}

Table 3: Transmitted DNS and HTTP messages.

Adding a security twist

Three users, Alice, Bob, and Persa, are logged into their computers, 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.

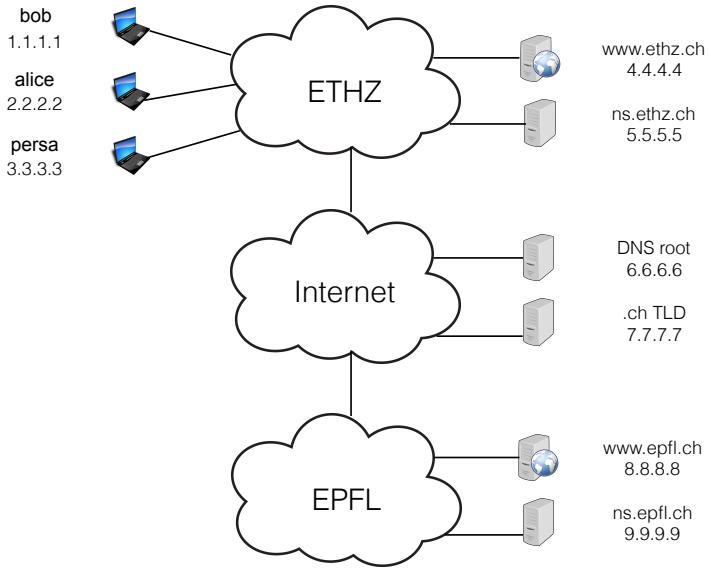

Figure 2: Question Setup

- 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
1	2.2.2.2	5.5.5.5	UDP	DNS	query for www.epfl.ch
2	5.5.5.5	6.6.6.6	UDP	DNS	query for www.epfl.ch IP
3	6.6.6.6	7.7.7.7	UDP	DNS	query for www.epfl.ch
4	7.7.7.7	9.9.9.9	UDP	DNS	query for www.epfl.ch
5	9.9.9.9	7.7.7.7	UDP	DNS	replay: 8.8.8.8 is IP of www.epfl.ch
6	7.7.7.7	6.6.6.6	UDP	DNS	replay: 8.8.8.8 is IP of www.epfl.ch
7	6.6.6.6	5.5.5.5	UDP	DNS	replay: 8.8.8.8 is IP of www.epfl.ch
8	5.5.5.5	2.2.2.2	UDP	DNS	replay: 8.8.8.8 is IP of www.epfl.ch
9	2.2.2.2	8.8.8.8	TCP		Connection request
10	8.8.8.8	2.2.2.2	TCP		Connection response
11	2.2.2.2	8.8.8.8	TCP	HTTP	HTTP GET /index.html
12	8.8.8.8	2.2.2.2	TCP	HTTP	HTTP OK {index.html}
13	2.2.2.2	8.8.8.8	TCP	HTTP	HTTP GET image.jpg
14	8.8.8.8	2.2.2.2	TCP	HTTP	HTTP OK {image.jpg}
15	2.2.2.2	5.5.5.5	UDP	DNS	query for www.ethz.ch
16	5.5.5.5	2.2.2.2	UDP	DNS	reply: 4.4.4.4 is IP of www.ethz.ch
17	2.2.2.2	4.4.4.4	TCP		Connection request
18	4.4.4.4	2.2.2.2	TCP		Connection response
19	2.2.2.2	4.4.4.4	TCP	HTTP	HTTP GET /file.html
20	4.4.4.4	2.2.2.2	TCP	HTTP	HTTP OK {file.html}

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?

  When Bob's DNS client makes a DNS request to `ns.ethz.ch` for `www.epfl.ch`'s IP address, Persa can impersonate `ns.ethz.ch` and respond that `www.epfl.ch`'s IP address is `3.3.3.3` (Persa's IP address). If

Persa's response gets to Bob's DNS client before the real `ns.ethz.ch`'s response, Bob's web browser will connect to the web server running on Persa's workstation instead of `www.epfl.ch`.