CS-202 Exercises on web & DNS (L10 - L11)

Before You start

In today's exercise session, you will get a sense of what happens when you type a URL in your web browser. More specifically, you will learn about the application layer protocols involved: DNS and HTTP.

When you type in a URL, your web browser connects to the web server that stores the base file of the webpage you are trying to access. The two (web browser and web server) use the HTTP protocol to exchange messages. The web browser sends an HTTP GET request and the web server replies with (one or more) HTTP response message(s) that contains the requested file.

But before your machine can connect to the web server, it needs to know the IP address of that server. For this, it uses the Domain Name System (DNS).

To find the IP address of the target web server, the web browser extracts the DNS name from the URL and asks the DNS client for the corresponding IP address. The DNS client, running on your machine, then queries the local DNS server for the domain's IP address.

The DNS hierarchy

Every network has a local DNS server that your DNS client can send queries to. However, the local DNS does not always have an answer. In such cases, the local DNS asks other DNS servers, 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 server.
    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

Paper and pencil exercises: DNS and HTTP

In these exercises assume that DNS and web-browser caches are initially empty. 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?

Exercise 1: DNS and HTTP message exchange [Basic]

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 2a, 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	reply: 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	reply: 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	reply: 6.6.6.6 is IP of www.unil.ch
8	3.3.3.3	5.5.5.5	DNS	reply: 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.

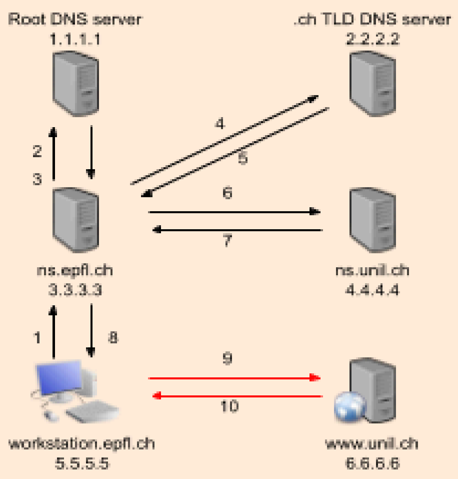


Figure 1: The messages transmitted in Table 2

- Immediately after retrieving this URL, you type in http://www.unil.ch/logo.png. In Table 2b, list all the DNS and HTTP messages that get transmitted as a result of your action.

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}

Exercise 2: Adding referenced files [Basic]

Now, suppose instead that http://www.unil.ch/index.html is an HTML page that references two image files: http://www.unil.ch/logo.png and http://www.unil.ch/banner.jpg.

- You open your web browser and type in http://www.unil.ch/index.html. In Table 3, list all the DNS and HTTP messages that would be transmitted as a result of your action. Assume the same DNS servers are involved as listed in Table 1 and all of these perform iterative requests.

Message	Source	Destination	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	reply: 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	reply: 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	reply: 6.6.6.6 is IP of www.unil.ch
8	3.3.3.3	5.5.5.5	DNS	reply: 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}
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}
13	5.5.5.5	6.6.6.6	HTTP	HTTP GET /banner.jpg
14	6.6.6.6	5.5.5.5	HTTP	HTTP OK {banner.jpg}

Table 3: Transmitted DNS and HTTP messages.

Exercise 3: Adding a security twist [Advanced]

Three users, Alice, Bob, and Persa, are logged into their computers, all located inside ETHZ's network.

ETHZ has a 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.

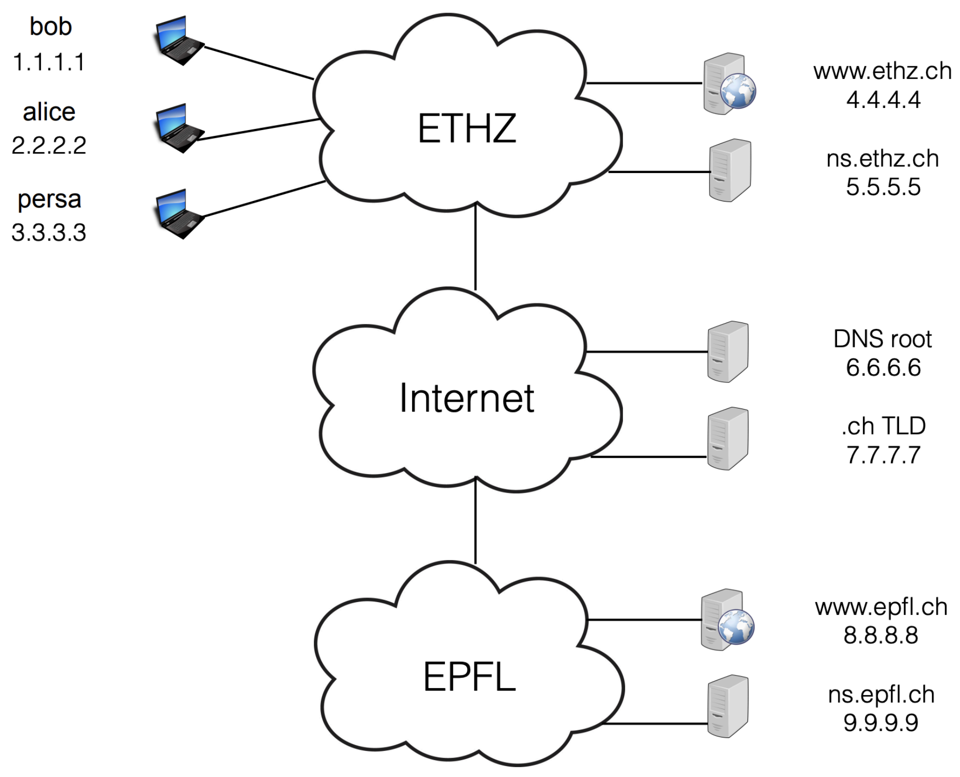


Figure 2: Question Setup

- 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 HTTP and DNS packets that are transmitted as a result of this action

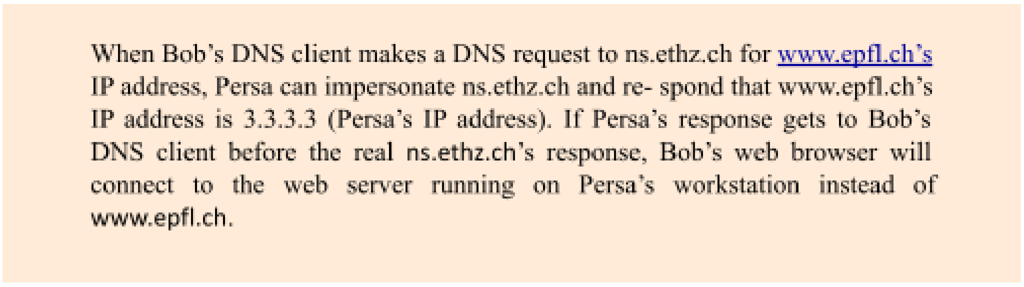
Packet	Source IP	Dest. IP	Application protocol	Purpose
1	2.2.2.2	5.5.5.5	DNS	query for www.epfl.ch
2	5.5.5.5	6.6.6.6	DNS	query for www.epfl.ch IP
3	6.6.6.6	7.7.7.7	DNS	query for www.epfl.ch
4	7.7.7.7	9.9.9.9	DNS	query for www.epfl.ch
5	9.9.9.9	7.7.7.7	DNS	replay: 8.8.8.8 is IP of www.epfl.ch
6	7.7.7.7	6.6.6.6	DNS	replay: 8.8.8.8 is IP of www.epfl.ch
7	6.6.6.6	5.5.5.5	DNS	replay: 8.8.8.8 is IP of www.epfl.ch
8	5.5.5.5	2.2.2.2	DNS	replay: 8.8.8.8 is IP of www.epfl.ch
11	2.2.2.2	8.8.8.8	HTTP	HTTP GET /index.html
12	8.8.8.8	2.2.2.2	HTTP	HTTP OK {index.html}
13	2.2.2.2	8.8.8.8	HTTP	HTTP GET image.jpg
14	8.8.8.8	2.2.2.2	HTTP	HTTP OK {image.jpg}
15	2.2.2.2	5.5.5.5	DNS	query for www.ethz.ch
16	5.5.5.5	2.2.2.2	DNS	reply: 4.4.4.4 is IP of www.ethz.ch
19	2.2.2.2	4.4.4.4	HTTP	HTTP GET /file.html
20	4.4.4.4	2.2.2.2	HTTP	HTTP OK {file.html}

Table 4: Transmitted 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 re- spond 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.

Mini-lab: IP addresses and process names

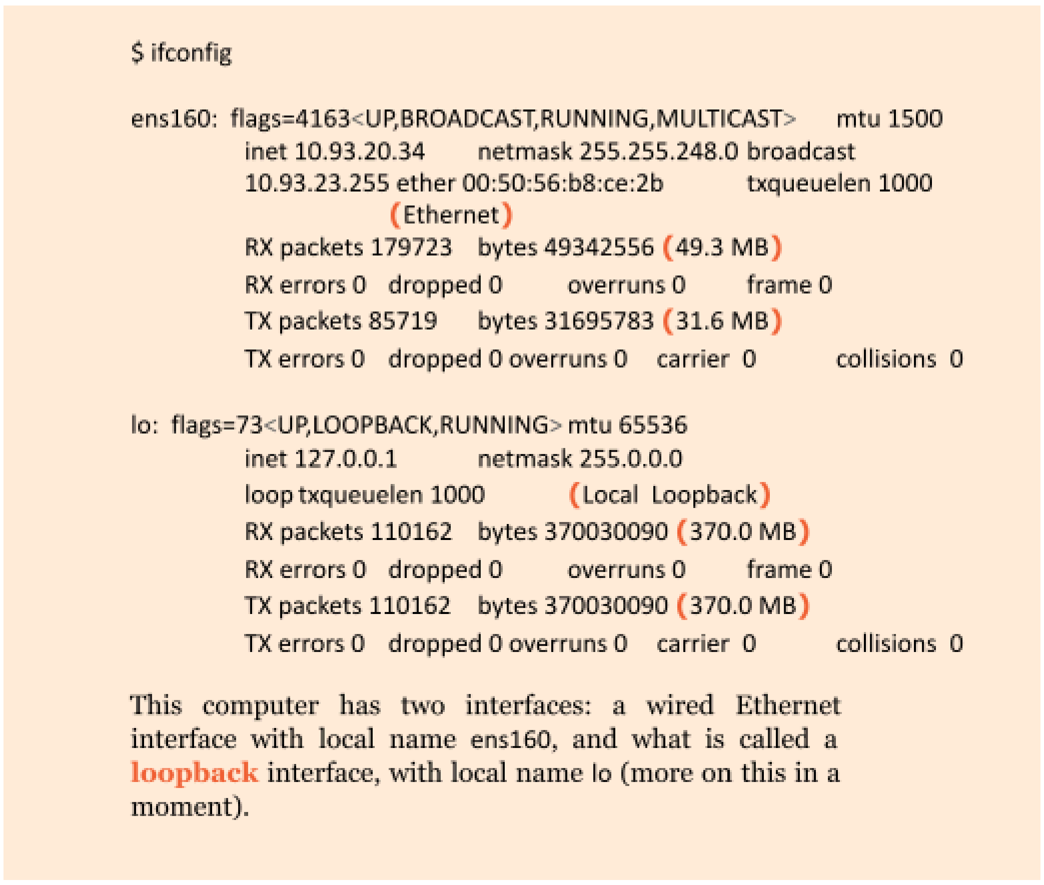
Exercise 4: Find your own IP address using the ifconfig utility [Basic]

Every computer in the world has at least one **network interface**. Whenever an entity outside the computer wants to communicate with the computer, it needs to name one of its network interfaces. Different entities use different names to refer to a computer's network interface: the network layer uses IP addresses, the link layer uses MAC addresses, the computer's operating system (OS) uses local interface names.

The **ifconfig** utility lists a computer's network interfaces and displays or updates their configuration.

Type ifconfig in a terminal command line and answer the following questions:

- How many active network interfaces does your computer have? (Hint: look at 'status' under your interfaces, usually named something like eth0, en0, or wlan0.)
- What is the IP address of each active interface? (Hint: Look for something like inet or inet6)



```
$ ifconfig

ens160:  flags=4163<UP,BROADCAST,RUNNING,MULTICAST>      mtu 1500
           inet 10.93.20.34       netmask 255.255.248.0 broadcast
           10.93.23.255 ether 00:50:56:b8:ce:2b          txqueuelen 1000
                    (Ethernet)
           RX packets 179723    bytes 49342556 (49.3 MB)
           RX errors 0   dropped 0        overruns 0        frame 0
           TX packets 85719     bytes 31695783 (31.6 MB)
           TX errors 0   dropped 0 overruns 0    carrier  0         collisions  0

lo:  flags=73<UP,LOOPBACK,RUNNING> mtu 65536
           inet 127.0.0.1         netmask 255.0.0.0
           loop txqueuelen 1000           (Local  Loopback)
           RX packets 110162   bytes 370030090 (370.0 MB)
           RX errors 0   dropped 0        overruns 0        frame 0
           TX packets 110162   bytes 370030090 (370.0 MB)
           TX errors 0   dropped 0 overruns 0    carrier  0         collisions  0
```

This computer has two interfaces: a wired Ethernet interface with local name ens160, and what is called a **loopback** interface, with local name lo (more on this in a moment).

To find the IP address we look at the inet value of each interface (or inet6 if inet is not available).

In this computer, the IP addresses are 10.93.20.34 for ens160 and 127.0.0.1 for the loopback interface.

Note: your interface might have inet6 but no inet field. This is normal, as your computer might be using the recent version of the IP addresses which is IPv6 (we have not talked about IPv6 yet but they will be covered during the course).

- Can you guess why it has more than one?

If a computer is connected to the network through multiple network links, then it has one network interface for each link. E.g., the INF3 computer is connected through a wired Ethernet link, as well as a wireless Ethernet link.

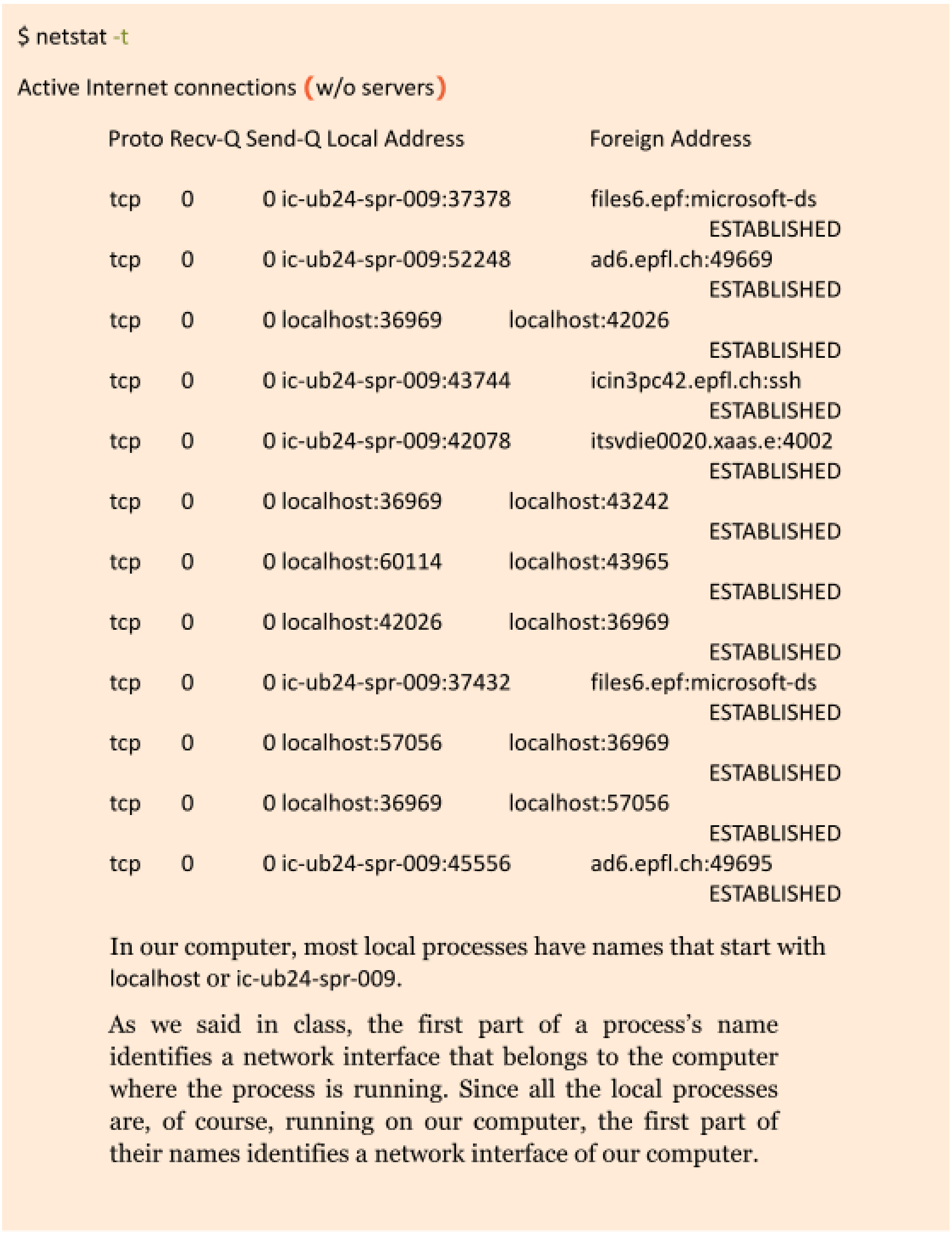
The loopback interface is what we call a **virtual** network interface. This means that it is not associated with an actual physical link. It is typically used for testing and debugging, and when a process running locally on the computer wants to communicate with another process also running locally on the computer. In the latter case, there is no need to communicate through a "normal" network interface, associated with an actual physical link, since both processes are running on the same computer.

If you are curious and want to learn more about a command, you can use your environment's man pages (e.g. if you type man ifconfig in the command line, it will display everything you could possibly want to know about the ifconfig command), or you can turn to online resources (e.g. Wikipedia).

Exercise 5: See local and remote addresses using the netstat utility [Basic]

The **netstat** utility displays the contents of various network-related data structures that are stored in your computer. E.g., if you type netstat -t in the command line, that will display the list of "communication sessions" that are active between your computer and remote computers.

- The “Local Address” column lists processes that are running in the application layer of your computer. Notice that the names of all (or most of) these processes share a common prefix. Why is that? What does this prefix correspond to?

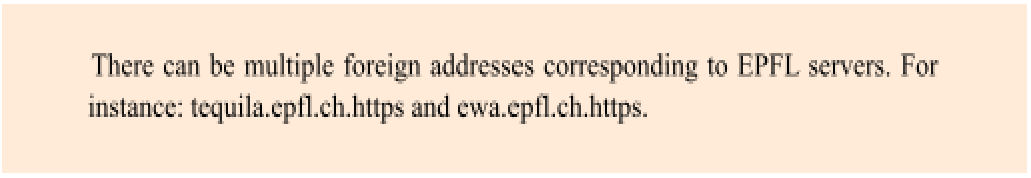


```
$ netstat -t

Active Internet connections (w/o servers)

      Proto Recv-Q Send-Q Local Address              Foreign Address

      tcp      0         0 ic-ub24-spr-009:37378          files6.epf:microsoft-ds
                                                                    ESTABLISHED
      tcp      0         0 ic-ub24-spr-009:52248          ad6.epfl.ch:49669
                                                                    ESTABLISHED
      tcp      0         0 localhost:36969         localhost:42026
                                                                    ESTABLISHED
      tcp      0         0 ic-ub24-spr-009:43744          icin3pc42.epfl.ch:ssh
                                                                    ESTABLISHED
      tcp      0         0 ic-ub24-spr-009:42078          itsvdie0020.xaas.e:4002
                                                                    ESTABLISHED
      tcp      0         0 localhost:36969         localhost:43242
                                                                    ESTABLISHED
      tcp      0         0 localhost:60114         localhost:43965
                                                                    ESTABLISHED
      tcp      0         0 localhost:42026         localhost:36969
                                                                    ESTABLISHED
      tcp      0         0 ic-ub24-spr-009:37432          files6.epf:microsoft-ds
                                                                    ESTABLISHED
      tcp      0         0 localhost:57056         localhost:36969
                                                                    ESTABLISHED
      tcp      0         0 localhost:36969         localhost:57056
                                                                    ESTABLISHED
      tcp      0         0 ic-ub24-spr-009:45556          ad6.epfl.ch:49695
                                                                    ESTABLISHED
```

In our computer, most local processes have names that start with localhost or ic-ub24-spr-009.

As we said in class, the first part of a process’s name identifies a network interface that belongs to the computer where the process is running. Since all the local processes are, of course, running on our computer, the first part of their names identifies a network interface of our computer.

- The “Foreign Address” column lists all the processes that are running in the application layer of a remote computer that your computer is communicating with. Login to Moodle in your browser and then run the netstat utility. Can you tell which foreign address(es) correspond to EPFL server(s)?



There can be multiple foreign addresses corresponding to EPFL servers. For instance: tequila.epfl.ch.https and ewa.epfl.ch.https.

Mini-lab: Playing with DNS

The dig utility

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.

Run “dig adelaide.edu.au” and answer the following questions:

- What is the primary IP address (A record) associated with adelaide.edu.au according to the output?
- What DNS server provided the response to your query, and how long did the query take to complete?

The primary IP address (A record) is 129.127.149.1, and the response was provided by a local DNS server at 192.168.1.1. The response time may range from a few milliseconds to several seconds, depending on whether the record was cached. If cached, the response is typically delivered almost instantly.
Note: The A record shown in the output may vary over time, and your DNS server may also differ.

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 **responses** (or “answers”) that contain RRs. A DNS message may carry multiple queries and/or responses.

- 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 informa- tion 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 this information to answer the next parts of this lab exercise.

Exercise 6: DNS Lookup [Basic]

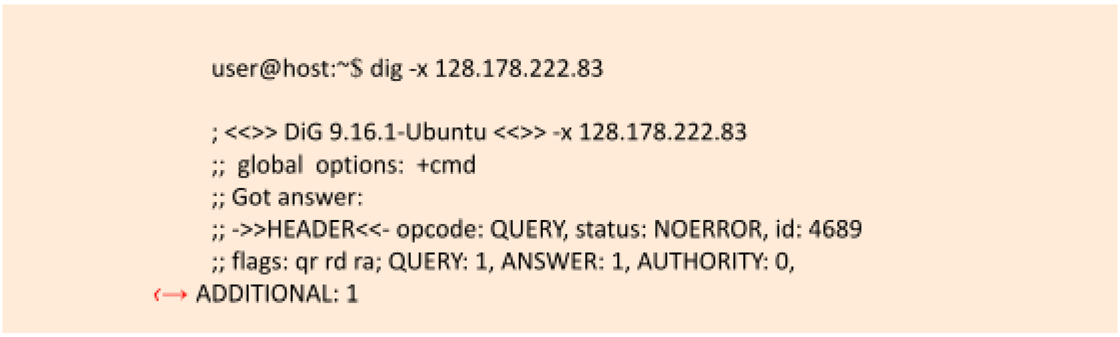
- What is the IP address of epfl.ch? Which RR type stores the information needed to answer this question?

```
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.

- What is the DNS name associated with the IP address obtained in the 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.

Exercise 7: Authoritative and local DNS servers [Basic]

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 query 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
```

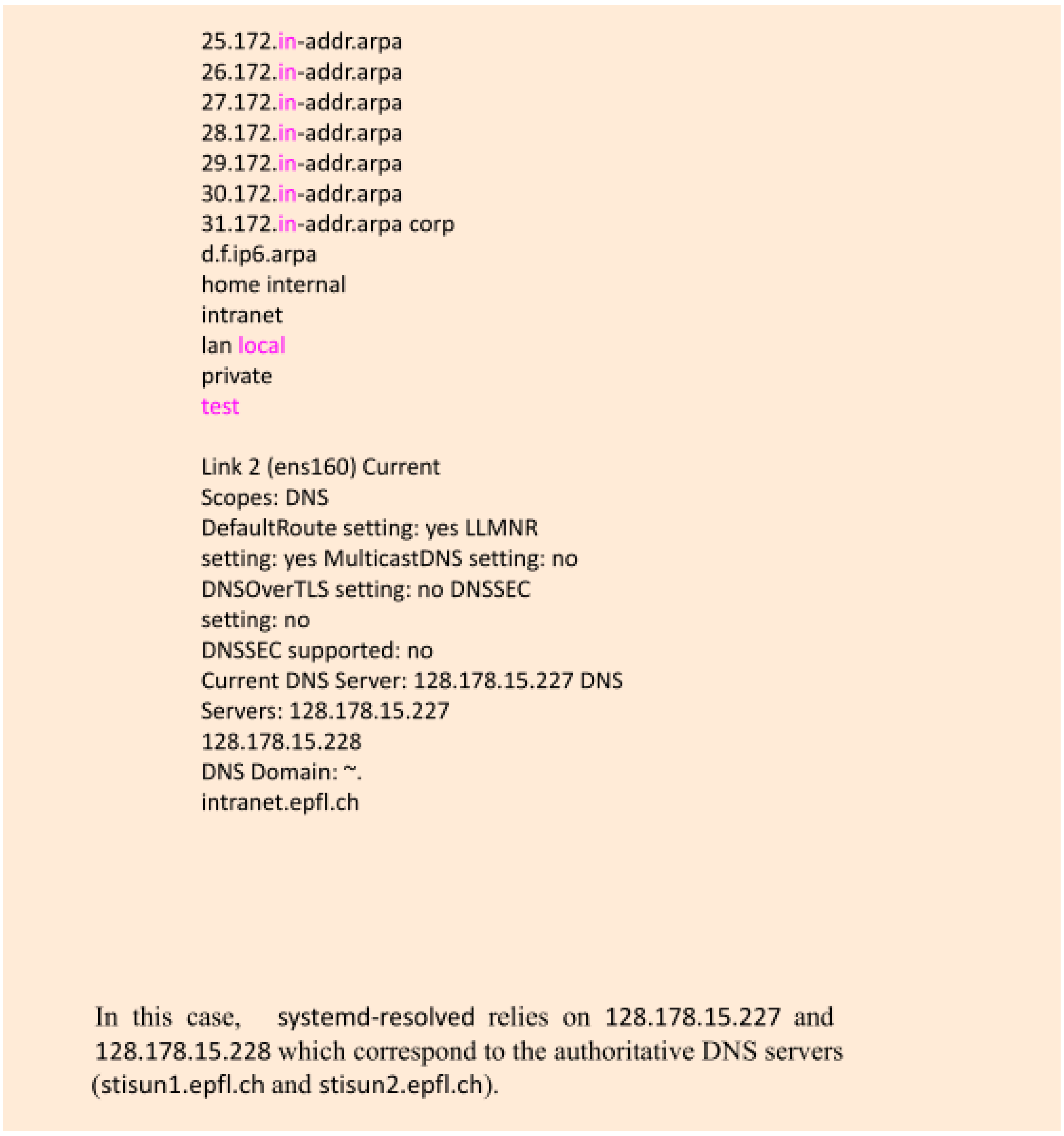
In the above dig output, the IP address of local nameserver is 127.0.0.53, and the port used is '53', which is the default port for DNS

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 has 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).

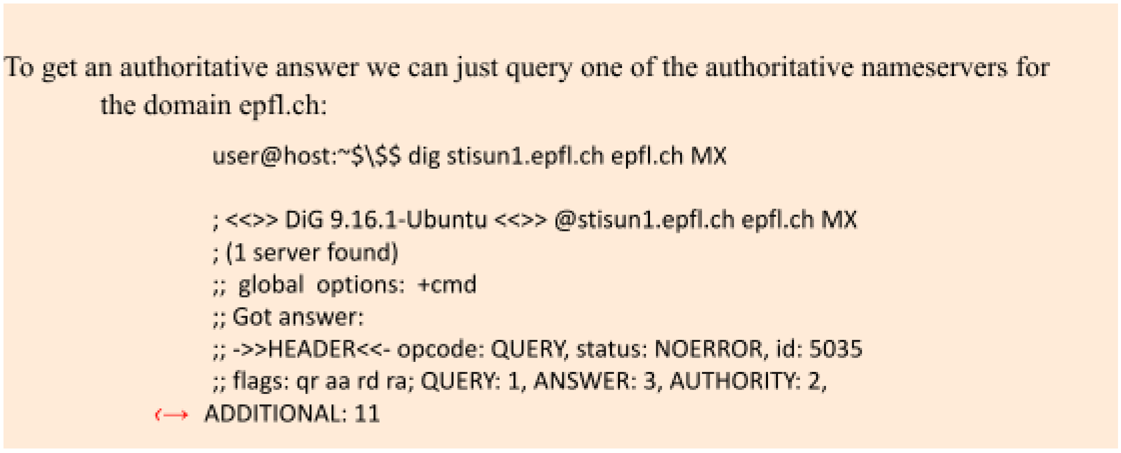
A DNS client can send a DNS message to any DNS server in the world; it is not obligated to contact only the local DNS servers. If you run: “dig @<IP address> ...” then dig will send its DNS query to the DNS server that has the specified <IP address>.

- 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
```

Exercise 8: DNS caching and time-to-live (TTL) [Advanced]

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.

Mini-lab: Cookies **[Optional]**

Cookies enable a web server to **link subsequent HTTP requests** to the same web browser: if you send 10 HTTP GET requests, for 10 different resources, to the same web server, the web server can use cookies to figure out that these 10 requests came from the same web browser, even if you did not explicitly provide any identification information (e.g., you did not login).

Before you start, figure out how to control cookie settings in your browser. In Firefox:

- To view or delete the cookies that have been stored on your computer: ≡ → Settings → Privacy & Security → Cookies and Site Data → Manage Data or Clear Data...
- To view all the cookies stored due to visiting the web page: Go to Storage from the top panel, and then select Cookies.

Let us see cookies in action:

- Allow your browser to exchange cookies. Delete existing cookies. Open Moodle. Did the EPFL web server send you any cookies? And are they all from the same domain?

Yes, there are four cookies from two domains: two cookies from .epfl.ch and two from domain moodle.epfl.ch.

You can read more about EPFL's cookies here.

- Login to your moodle account. Restart your web browser and re-open Moodle. Does it ask you to login again? Explain your browser's behavior.

No, we are automatically logged in. This happens because, along with the current HTTP request, our web browser sent a TequilaPHP cookie for moodle.epfl.ch, which contains our login information. This way the server "remembers" our credentials and directly shows the courses we are registered in.

You can read more about EPFL's Tequila authentication service here.
(it has been deprecated now)

- Delete existing cookies. Restart your web browser and re-open Moodle. Does it ask you to login again? Explain your browser's behavior.

Yes, it asks us to login again: since we deleted the cookies, the TequilaPHP cookie is longer there. So it is as if we visited the website for the first time.