

Practice Lab 1. TCP/IP Configuration on Windows & Linux

1. Introduction

To perform this lab, you need access to two different environments: Windows and Linux.

The objective is to review the basic procedures for installing and configuring TCP/IP protocols on Windows and Linux. This includes using useful tools to troubleshoot these protocols, configure network software, verify operations, and adjust TCP/IP parameters.

2. TCP/IP Configuration on Windows

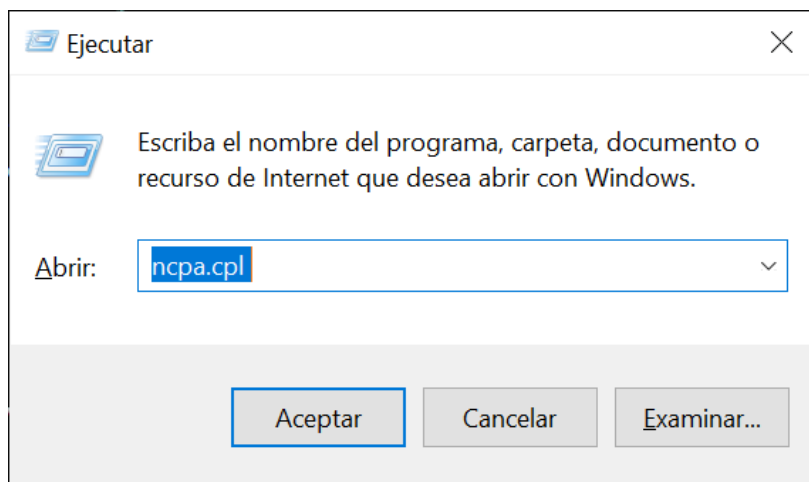
Start your practice computer in the Windows partition, logging in under your usual username and password.

To use TCP/IP protocols from a Windows machine connected to a local area network, a network adapter (NIC) must be installed. On our computers, this adapter is usually already installed and configured.

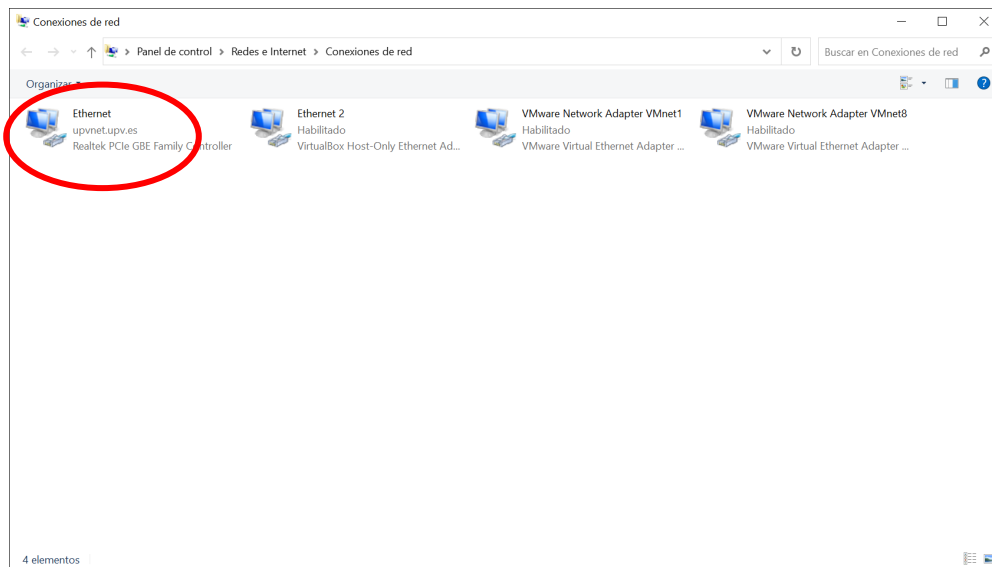
Depending on whether you have an Ethernet or WiFi network card, the configuration display may vary slightly. Let's review both cases:

A. Ethernet Adapter Configuration visualization:

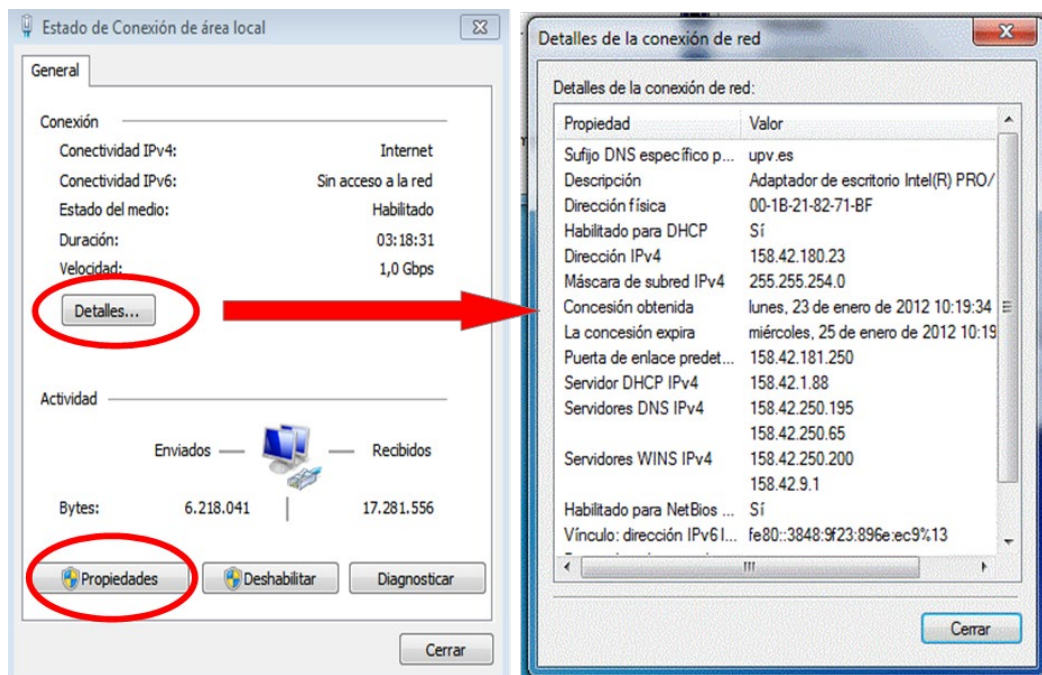
1. Press Windows + R to open the Run dialog box. Type **ncpa.cpl** and press Enter.



- This will display the available network adapters on the system. **Double-click the Ethernet link.**



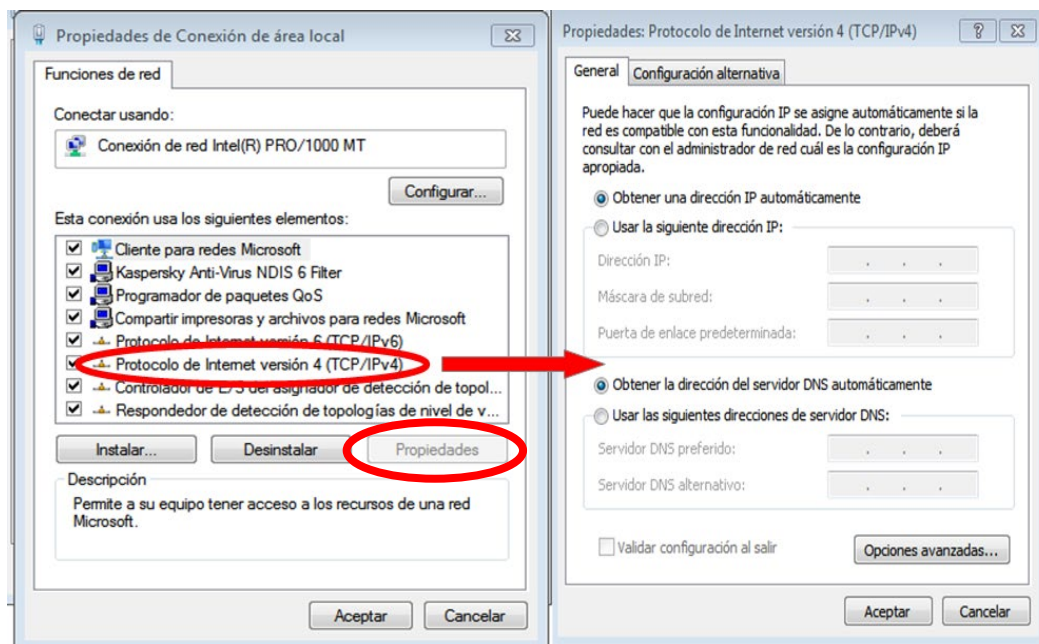
- The Ethernet adapter **status** window will open. Clicking **Details** will show key network connection properties, including the physical adapter address, associated IP address, subnet mask, and more.



Besides the network adapter, TCP/IP protocols must also be installed to use Internet applications. These protocols are already installed on our computers. An administrator can verify this by performing the following steps:


Note: As you are not an administrator user, you do not have the needed rights.

- From the Ethernet **Adapter Status window**, click **Properties** to display the elements using this network adapter. Among them, you'll find Internet Protocol Version 4 (TCP/IPv4), which represents the TCP/IP stack. Selecting it and clicking **Properties** will display operational parameters.



Most TCP/IP parameters, including the IP address, are typically not manually configured, but instead obtained automatically at system startup through the DHCP (Dynamic Host Configuration Protocol). This protocol is frequently used and will be studied in a later practice during this semester. Besides the IP address, the DHCP server provides additional information necessary for TCP/IP protocol operations (DNS server IP address, router IP address, etc.).

B. Wi-Fi Adapter Configuration visualization:

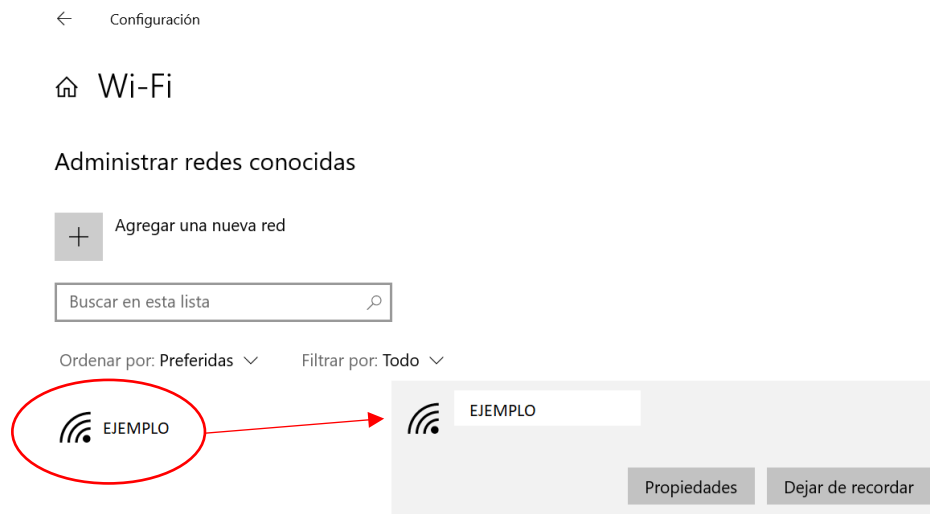
- Click **Start**, then **Settings** (or the  icon) and select **Network & Internet**.
- Under **Wi-Fi**, you will find information about the active wireless network connected.
- In the **Hardware Properties**, you can view key properties of the wireless network connection, such as the physical adapter address and associated IP address.



Propiedades

SSID:
 Protocolo:
 Tipo de seguridad:
 Banda de red:
 Canal de red:
 Velocidad de vínculo (recepción/transmisión):
 Dirección IPv6 local de vínculo:
 Dirección IPv4:
 Servidores DNS IPv4:
 Fabricante:
 Descripción:
 Versión del controlador:
 Dirección física (MAC):

4. To view related elements for this Wi-Fi adapter, click **Manage Known Networks** and select your Wi-Fi network (e.g., *Ejemplo*). The Properties window will display network adapter elements.



5. Among them, the Internet Protocol version 4 stands out, which represents the TCP/IP stack. After selecting the **Edit** button in **IP Configuration**, if we do not have an automatic configuration set by default, we can access a new window that shows the IP operating parameters.



As with the Ethernet adapter, DHCP is commonly used to obtain the necessary information for TCP/IP protocols. (DNS server IP address, *router* IP address, etc.).

2.1 The ipconfig Command

Once TCP/IP protocols are installed, the **ipconfig** command, executed from the **Command Prompt**, provides information about network configuration for each installed network adapter.

Running **ipconfig /?** displays information about the command and its options, including the ability to obtain details about:

- **IPv4 and IPv6 Addresses:** IP addresses static assigned or dynamically assigned using DHCP.
- **Subnet Mask:** Identifies the network and host portions of the IPv4 address. The UPV network is globally assigned the IPv4 address block 158.42.0.0/16, which has been broken down into a series of subnets. The subnet mask (255.255.254.0) indicates that, in the case of the laboratory subnet, the 23 most significant bits of each IPv4 address (bits set to 1 in the mask) should be considered the network identifier, and the last 9 bits (set to 0 in the mask) are used as the host identifier.
- **Default Gateway:** The IP address of the router connecting the subnet to other networks, the UPV network and others like Internet.

Running **ipconfig /all** provides additional information such as:

- **Physical Address:** Corresponds to the network adapter (Ethernet in our lab).
- **DNS Servers:** IP address of the server that resolves domain names.
- **DHCP Server:** IP address of the machine assigning the IP address.
- **Lease Obtained/Expires:** Dates indicating the lease duration for the IP address. Applicable only in the case of IP information obtained by DHCP.

The commands **ipconfig /release** and **ipconfig /renew** allow you to release and renew the DHCP-obtained IPv4 address.

Exercise 1

From the **Command Prompt** (press **Windows + R**, type **cmd**, and press Enter), execute **ipconfig /all** and identify the following information for the Ethernet 4 adapter:

- Physical address
- IPv4 address
- Subnet mask
- Router (default gateway) IP address
- DNS servers
- DHCP server

Based on the information obtained:

- What is the IP address of the network your computer is connected to?
- Are the DNS and DHCP servers in the same subnet as your computer? How did you determine this?

Exercise 2

Check the contents of the DNS cache. Note in the table the values of one of the records that appear that is of type 1: [Run ipconfig /displaydns](#)

| | |
|------------------------------------|---|
| Register Type | 1 |
| Register Name | safebrowsing.googleapis.com |
| Register Value (a <host> register) | 142.250.200.74 |

2.2 The ping command

By using the **ping command** (executed from the **Command Prompt**), an estimate of the round-trip time (RTT) of a packet can be obtained, from the station where the command is executed to the specified destination station. The operation of the ping command is based on the use of ICMP echo messages, which will be studied in a later practice.

Example: Execute the command **ping www.rediris.es** and observe its behaviour.

The ping command accepts several options. For now, the only one of interest is shown below:

ping [-n count] destination , -n count: Number of echo requests to send.

This command will be analysed in detail in the practice dedicated to studying the ICMP protocol. Until then, we will only use it to determine whether a specific destination can be reached through the network and to identify its IP address.

2.3 The tracert Command

The **tracert** command (executed from the **Command Prompt**) allows you to determine the path (sequence of routers) that a packet must traverse to travel from the source station to the destination station. It is based on the use of ICMP messages and will, therefore, also be studied in the practice dedicated to this protocol.

Example: Execute the command **tracert www.rediris.es** and observe its behaviour. Once it completes, execute **ping www.rediris.es** and try to explain the TTL field value in the responses. [TTL starts at 63, and as it does 7 hops \(as seen with tracert\), TTL becomes 63 - 7 = 55](#)

2.4 The netstat command

The **netstat** command (executed from the **Command Prompt**) provides various information about the status and statistics of network protocols. Data can be obtained about key events for Ethernet, IP, ICMP, UDP, and TCP. Execute **netstat -h** to view the different usage options. We will use several of these options below, though you can experiment with the rest.

By using the command **netstat -r**, we can obtain information about the routing table (this produces the same output as the route print command).

Remember that when a datagram needs to be routed, the forwarding process follows the steps studied in the theory classes. Specifically, the mechanism is as follows:

1. For each line in the routing table, a logical AND is performed between the destination IP address of the datagram and the network mask. IP compares the result with the destination network and marks all routes where a match occurs.
2. From the list of matching routes, IP selects the route with the most bits in the mask. This is the most specific route, known as the "longest matching" route.

3. If there are multiple longest matching routes, the one with the lowest metric is used. If several have the same metric, any one of them can be used

~~NOTA: Creo que solo se puede hacer en los ordenadores de la UPV porque no me aparece ninguna network destination similar a la IP de los sitios.~~

Exercise 3:

View the routing table (IPv4 section) of the computer you are working on.

Find the IP addresses of the following destinations (refer to the previous exercises) and analyze which route from the table would be selected for each:

- a) A packet destined to **zoltar.redes.upv.es**
- b) A packet destined to **www.upv.es**
- c) A packet destined to **www.usc.edu**

Exercise 4:

The **netstat -e** command provides statistics on the number of bytes and frames sent and received by the network adapter. It details the number of unicast (single destination) frames, non-unicast (multiple destinations and broadcasts) frames, erroneous packets, and discarded packets

Run **netstat -e** and observe the results for the following categories

- Unicast packets.
- Non-unicast packets.
- Discarded packets.
- Errors.

Run **netstat -es** and compare its output with **netstat -e**. Note any differences in format or the level of detail provided in the statistics.

Exercise 5:

The **netstat -sp IP** command provides statistics about IP traffic. Execute this command and analyse the output to find the following information:

- Packets received.
- Header errors received.
- Address errors received
- Datagrams forwarded.
- Unknown protocols received.
- Datagrams successfully fragmented.

Exercise 6:

Similarly, the **netstat -sp TCP** command provides statistics on TCP traffic (statistics for ICMP and UDP protocols can also be requested). Execute this command and identify the following:

- Active opens
- Passive opens
- Failed connection attempts
- Current connections

What do the first two values ("Active opens" and "Passive opens") refer to?

The **netstat** command without arguments provides information about the active connections on our machine. When used with the **-a** option, in addition to the previous information, it also displays the list of TCP and UDP ports where an application is listening (ready to accept TCP connections or UDP datagrams).

2.5 The arp Command

This command is also very useful for network configuration and troubleshooting. A practice session dedicated to the ARP protocol will be held later this semester to analyse it in detail.

Example: Execute the command **arp -a** to view the IP addresses of the machines your PC has interacted with and their associated physical addresses.

3. TCP/IP Configuration in Linux

*For this section you must restart your practice computer, selecting the Linux partition, and using the **redlocal** user. The teacher will provide you with the password.*

In Linux, we can find equivalent utilities to those studied in the previous section. To check connectivity with a specific machine, we have the **traceroute** and **ping** commands, which serve the same purpose as **tracert** and **ping** in Windows 10, respectively. Unlike Windows, in Linux, **ping** continues to send packets until execution is aborted (*Ctrl + C*) unless the **-c <number>** parameter is specified to limit the number of attempts.

Newer versions of the operating system use a single tool, **ip**, to access local network configuration, offering superior functionalities compared to the **ipconfig**, **arp**, and **netstat** commands studied for Windows. Interestingly, in earlier versions, the **arp** and **netstat** commands are available, as well as **ifconfig**, which is similar to **ipconfig**.

3.1 The ip command

The **ip** command, which can be executed from a **command terminal**, allows configuring and retrieving information about various aspects of network configuration. Running without parameters provides a description of its capabilities:

```
redlocal@redes:~$ ip
Usage: ip [ OPTIONS ] OBJECT { COMMAND | help }
       ip [ -force ] -batch filename
where  OBJECT := { link | address | addrlabel | route | rule | neigh | ntable |
                  tunnel | tuntap | maddress | mroute | mrule | monitor | xfrm |
                  netns | l2tp | fou | macsec | tcp_metrics | token | netconf | ila |
                  vrf | sr }
       OPTIONS := { -V[ersion] | -s[tatistics] | -d[etails] | -r[esolve] |
                   -h[uman-readable] | -i[ec] |
                   -f[amily] { inet | inet6 | ipx | dnet | mpls | bridge | link } |
                   -4 | -6 | -I | -D | -B | -O |
                   -l[oops] { maximum-addr-flush-attempts } | -br[ief] |
                   -o[neline] | -t[imestamp] | -ts[hort] | -b[atch] [filename] |
                   -rc[vbuf] [size] | -n[etns] name | -a[ll] | -c[olor]}
```

As can be seen, **ip** offers far more possibilities than **ipconfig** in Windows 10. It allows working across several layers of the protocol stack:

- **ip link:** Accesses network adapters.
- **ip address:** Refers to the adapter's IPv4 and IPv6 addresses.
- **ip route:** Accesses the forwarding table.
- **ip neighbour:** Accesses neighbor cache tables (ARP protocol in IPv4 and ND in IPv6).
- **ip ntables:** Manages these tables.

There are additional functionalities that will be discussed in later practice sessions.

3.1.1 `ip address` command

To obtain information about the system's network adapters and their associated IP addresses, the appropriate command is **`ip address list`** or **`ip address show`**. It is also valid to use just **`ip address`**, as **`show`** is the default option. Additionally, parameters can be abbreviated as long as there is no ambiguity; for instance, **`ip address`** can be written as **`ip addr`** or even **`ip a`**.

```
redlocal@redes:~$ ip address show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen
1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp3s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
default qlen 1000
    link/ether 08:00:27:84:e5:1b brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic enp3s3
        valid_lft 85360sec preferred_lft 85360sec
    inet6 fe80::a00:27ff:fe84:e51b/64 scope link
        valid_lft forever preferred_lft forever
```

As observed, the machine used for testing has two network interfaces. The **`lo`** interface corresponds to a virtual network adapter that implements access to the local loopback, meaning it receives IP datagrams sent to addresses of the type `127.x.x.x`, and returns them as if they had been received from the network. On the other hand, the **`enp3s0`** interface corresponds to the computer's network adapter, which enables connection to the local network and, through it, to the Internet.

For each interface, **`ip`** displays the associated IP addresses, both IPv4 and IPv6, their network prefix, and range. In the case of IPv4 addresses, all are global, while the assigned IPv6 address belongs to a more restricted scope (local addresses).

Information related to the adapter at the data link level is also displayed: its physical address (which will be examined in subsequent practices), the network MTU (essential for the next fragmentation practice), and whether the interface is active (UP) or not.

Exercise 7:

Use the **`ip`** command to determine how many network adapters exist on your computer, as well as the IP address and network mask of each one. Analyse whether they support broadcasts and multicasts, and identify the MTU of the network accessible through each adapter. Additionally, verify whether the addresses are permanent or need to be periodically renewed, and whether they are global or local.

The **`ip address`** command allows a system administrator to add or remove network addresses on an adapter. Additionally, multiple IP addresses can be assigned to the same interface. For example, running: **`sudo ip address add 192.168.1.153/255.255.255.0 dev enp3s0`** assigns a second address to the `enp3s0` interface.

```
redlocal@redes:~$ sudo ip address add 192.168.1.153/255.255.255.0 dev enp3s0
[sudo] contraseña para redlocal:
redlocal@redes:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen
1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
```

```
valid_lft forever preferred_lft forever
2: enp3s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
default qlen 1000
    link/ether 08:00:27:84:e5:1b brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic enp3s0
        valid_lft 70735sec preferred_lft 70735sec
    inet 192.168.1.153/24 scope global enp3s0
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe84:e51b/64 scope link
        valid_lft forever preferred_lft forever
```

As usual, the first time `sudo` is used, the password must be re-entered. It is also possible to specify a broadcast address for the same interface with: **ip address add 192.168.1.153/24 broadcast + dev enp3s0**. The `+` symbol indicates that the broadcast address should be derived by replacing each host bit of the provided IP address with **1**. Similarly, replacing **add** with **del** allows removing this assignment.

This functionality enables, for example, testing a client program locally on a fixed IP server by assigning that IP to the **lo** adapter:

```
redlocal@redes:~$ sudo ip address add 158.41.1.1/255.255.0.0 dev lo
redlocal@redes:~$ ping 158.41.1.1
PING 158.41.1.1 (158.41.1.1) 56(84) bytes of data.
64 bytes from 158.41.1.1: icmp_seq=1 ttl=64 time=0.020 ms
64 bytes from 158.41.1.1: icmp_seq=2 ttl=64 time=0.032 ms
64 bytes from 158.41.1.1: icmp_seq=3 ttl=64 time=0.030 ms
^C
--- 158.41.1.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2053ms
rtt min/avg/max/mdev = 0.020/0.027/0.032/0.006 ms
redlocal@redes:~$ traceroute 158.41.1.1
traceroute to 158.41.1.1 (158.41.1.1), 30 hops max, 60 byte packets
 1  158.41.1.1 (158.41.1.1)  0.024 ms  0.009 ms  0.008 ms
```

Exercise 8:

Access the main UPV website (www.upv.es) using a browser like Firefox. Then, use the **ip** command to assign the corresponding IP address (158.42.4.23/16) to your **lo** adapter. You can install your own "web server" to see the client's request by running: **sudo nc -l 80** in another terminal. Repeat the web access (make sure to use `http://`) and check the terminal's response. Revert the change by removing this IP address from your adapter and verify the web access again.

3.1.2 *ip link* command

It is possible to obtain a list of the existing network interfaces on a computer by executing **ip link**. However, this command does not provide information about configuration related to protocols above the link layer.

```
redlocal@redes:~$ ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group
default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp3s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode
DEFAULT group default qlen 1000
    link/ether 08:00:27:84:e5:1b brd ff:ff:ff:ff:ff:ff
```

Usage statistics for the interfaces can also be displayed using the `-s` parameter. It shows both the number of bytes/packets correctly transmitted and received, as well as the number of packets that encountered errors.

Exercise 9:

Use the command `ip -s link show` to determine how many network adapters exist, and the number of bytes transmitted and received through each of them.

Verify that the figures increase on the appropriate adapter after executing: `ping -c 2 localhost` and `ping -c 2 www.rediris.es`

This command allows administrators (in our case, by prefixing with `sudo`) to manually add and remove network interfaces, although this functionality goes beyond the scope of this exercise. However, it can be useful to modify certain parameters. For example, it is possible to deactivate an interface with: `sudo ip link set <interfaz> down`, thus preventing its usage. To restore its functionality, use: `ip link set <interfaz> up`.

Exercise 10:

Deactivate one of your computer's interfaces and check the behavior of the previous *pings*. Then, repeat the test after reactivating the first interface and deactivating another. Observe the results.

Another interesting option with `ip link set` is the ability to manipulate almost any link parameter. For example:

- `ip link set dev <adapter> arp (on|off)` enables or disables ARP responses, which will be discussed in later exercises.
- `ip link set dev <adapter> multicast (on|off)` allows or blocks multicast traffic through the interface.
- `ip link set dev <adapter> promisc (on|off)` switches the adapter to promiscuous mode, meaning it will receive all traffic, even if it is not addressed to it. This is especially useful for network monitoring, for example, using Wireshark.
- `ip link set dev <adapter> address <address>` allows manually changing the adapter's physical address.
- `ip link set dev <adapter> mtu <size>` modifies the maximum size of the data field in the data link unit (frame).

3.1.3 *ip route* command

The `ip route` command provides access to the system's forwarding tables. Linux uses several tables to manage routes. In general, routes are stored in Table 254, also known as **main**, and the system only uses this table when calculating routes. Table 255, called **local**, stores routes to local and broadcast addresses. This table structure is particularly useful when applying different forwarding policies.

The simplest way to use this command is by running `ip route show`:

```
redlocal@redes:~$ ip route show
default via 10.0.2.2 dev enp3s0 proto dhcp src 10.0.2.15 metric 100
10.0.2.0/24 dev enp3s0 proto kernel scope link src 10.0.2.15
10.0.2.2 dev enp3s0 proto dhcp scope link src 10.0.2.15 metric 100
```

If no table is specified, the **main** table is displayed by default. The interpretation of this forwarding table is straightforward. The default route (default) consists of transmitting the packet to 10.0.2.2 (the default router) using the network adapter **enp3s0**. This configuration has been obtained through DHCP, the automatic IP parameter configuration protocol, which will be studied in later exercises. Finally, the source IP for IP datagrams sent by default will be 10.0.2.15 (our IP address), and the metric associated with this link is 100.

This metric serves as an additional parameter for route selection. As explained in theoretical classes, if multiple entries in the forwarding table are applicable, the route with the longest network prefix is chosen. If the prefix lengths are equal, this metric, representing the link's usage cost, is used to select the optimal (least-cost) route. The second line applies to direct deliveries, i.e., forwarding IP datagrams whose destination is within the same IP network. In this case, the information has been included in the table by the operating system *kernel*.

It is possible to access the IPv6 forwarding table using the `-6` parameter:

```
redlocal@redes:~$ ip -6 route show
fe80::/64 dev enp3s0 proto kernel metric 256 pref medium
```

Exercise 11:

Use the command: **`ip route show table all`** to display all the forwarding rules in your system and interpret the results. Also, test the main and local tables by replacing *all* in the previous command.

For users with appropriate permissions, running **`sudo ip route add default via 192.168.1.150`** will modify the system's default route. You can also add specific routes to this table: **`ip route add 172.16.32.0/24 via 192.168.1.150 dev enp3s0`** or even particular routes for individual addresses: **`ip route add 172.16.32.32 via 192.168.1.150 dev enp3s0`**. As in the previous case, using **`del`** cancels the effect of **`add`**.

In this section, we will use the possibility of modifying the tables to observe the effect of the most important entries on packet routing. In particular, we will analyze the default entry, which allows us to reach the rest of the Internet.

Exercise 12:

- 1) Display your computer's routing table using: **`ip route show`**
- 2) Find and note the address of your network's exit router. We will refer to it as **`router_IP_address`**.
- 3) Delete the default network address entry: **`sudo ip route del default`** and display the routing table.
- 4) Try to access a destination outside your IP network using: **`ping -c 2 www.rediris.es`** and explain what happens.
- 5) Try the ping again using the IP address of `www.rediris.es` (130.206.13.20) instead of the server name: **`ping -c 2 130.206.13.20`**
- 6) Try to access a destination within the same IP network as your computer. For example, `zoltar.redes.upv.es` using **`ping -c 2 zoltar.redes.upv.es`**. Then test the result by directly using the IP address of `zoltar.redes.upv.es` (158.42.180.62). **`ping -c 2 158.42.180.62`** Observe the difference.
- 7) Restore the routing table entry you deleted: **`sudo ip route add default via your_router_IP_address`**.
- 8) Check the status of the routing table. It should now be the same as before you deleted the route. Also, verify that pings to computers outside your network now work. Explain why the ping to `zoltar.redes.upv.es` now functions correctly.