V.1.6.1 28th · Feb · 2019

# Practical Exercises in Computer Networks and Distributed Systems

## Network Interfaces and Protocol Stacks (WIP)



Every computer connects to the Internet by means of a network adapter card, normally one such card suffices. The PCs in our laboratories use the Linux operating system in one of its several flavors (Debian, Ubuntu, Fedora, etc). In this laboratory practice we set out to understand the basic commands that allow our system to communicate in Internet. You can check the commands and processes explained hereon by using your own laptop or one of our lab PCs.

## The network interface

At the beginning of this course on Computer Networks we study that networking is a systems discipline that defines networks by using a number of concepts organized into a hierarchy of layers. Each layer offers one or several services that can be used by calling the layer interface. You simply call the interface in order to make the service work in some useful manner. When some application in our computer needs to access the network, it does so by calling the *network interface.*

The operating system offers us a series of commands to properly bootstrap the network adapter, configure it, use it, monitor it, etc. Before we can issue a command to the NIC[1] we will have to find out its name, that is, the name given to it by the operating system in the bootstrap process, for more than one NIC may be present in the system. The kernel associates a name and a device driver with each NIC and the system administrator allocates one or more IP addresses to it according to the network design specifications. If you bring your laptop with you, it will retrieve its IP address wirelessly by using a broadcast-based protocol named DHCP, thereby granting you the basic conditions for network connectivity. Virtually, no intervention on your part is required as much as the system boot process is concerned. We will study this protocol in a few lessons; for the time being, we will proceed to explain the most basic network configuration command: ifconfig.

The OS-X and Linux command ifconfig reports the network parameters of one or more NICs, also, upon system boot up it will configure the different NICs according to the contents of a series of text files written by the administrator, for now, we aim to just check ifconfig reporting capabilities.

### Exercise 1. ifconfig

Login to your system and execute the following ifconfig command which will let you know its IP address and other significant network parameters. The example system is OS-X, in Linux you will have to substitute en0 (The network device name) for eth0 or maybe enp1s4, eno1, etc. If you provide no specific device name, ifconfig will list out all the device names known by the system (In most of the command lines we assume **$** as the *shell prompt*):

[1] The acronym NIC means Network Interface Card, which is comprised of an electronic circuitry along with some software installed on it

```
$ ifconfig en0
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu
1500
     options=2b<RXCSUM,TXCSUM,VLAN_HWTAGGING,TSO4>
     ether c8:2a:14:55:1d:93
     inet6 fe80::ca2a:14ff:fe55:1d93%en0 prefixlen 64 scopeid 0x4
     inet 192.168.99.99 netmask 0xffffff00 broadcast 192.168.99.255
     media: autoselect (1000baseT <full-duplex,flow-control>)
     status: active
```

ifconfig stands for "interface configuration", i.e. "network device interface configuration", you will find that command in all the main stream operating systems, including Windows[2]. Let's review the parameter list reported by the command highlighting the most important ones. In the present case, the kernel booted up and in the process it detected an administratively established configuration for our 1000BASE-T device, the point-to-point communication medium is a set of copper wires connected to the transmitting electronics. The medium is managed in such a way that it allows simultaneous bidirectional communication (full-duplex) and the computer and the network device to which it is connected support flow-control, that is, the regulated reception of information with no possibility of overflowing its limited-size input buffer. Other significant parameters are related to the internet identification of our computer system, that is, its **IP address (Internetwork Protocol)**, its **network mask** or net mask that indicates which number of upper-weight bits of the IP address constitute the network number (IP addresses are hierarchical and contain three parts: the network number, the subnetwork number and the node number).

The **broadcast address** is a special network-address used for communicating with all the computers belonging to a specific network, in the case of our lab, the network will be of the **Ethernet** type and the broadcast address of this network is used by certain applications to offer and discover network-based services, for example, DHCPv4 is a broadcast-based protocol that serves the purpose of assigning IP addresses and other network-related parameters to end nodes in an unattended, automatic manner. The **ether** field represents the hardware address of our Ethernet NIC, this number is assigned by the NIC's manufacturer and is normally not changed, it allows our NIC to be identified univocally within our network, these addresses in the Ethernet technology realm are named hardware or MAC (Medium Access Control) addresses. These addresses are not to be confused with the IP addresses mentioned before: the MAC addresses guarantee that traffic directed to our system is properly delivered and received by the destination system, the IP addresses identify our system in an **internetwork** (IP, Internetwork Protocol), that is in **Internet**. As you already guessed, there is a mechanism that maps IP addresses to MAC addresses within any local network, this mechanism includes a protocol that we will study soon, its name is **ARP** (Address Resolution Protocol).

### Exercise 2. dmesg

Depending on machine-specific details, another command that comes in handy if you need to find out the network devices configured on a boot is dmesg, normally you will have to issue it via sudo, that is, by requesting to raise your user privilege level to that of root user

```
$ sudo dmesg | grep ether
```

[2] In Windows the command to configure your NIC is ipconfig

```
BCM5701Enet: Ethernet address 3c:07:54:52:ae:2a
AirPort_Brcm4331: Ethernet address 60:c5:47:23:9e:e5
AppleBCM5701Ethernet: a 1 extract_link_settings - EEE logic error
Ethernet [AppleBCM5701Ethernet]: Link up on en0, 10-Megabit, Half-duplex, No flow-
control, Debug [794d,0101,0de1,0300,0000,0000]
Ethernet [AppleBCM5701Ethernet]: Link down on en0
Ethernet [AppleBCM5701Ethernet]: Link up on en0, 1-Gigabit, Full-duplex, Symmetric
flow-control, Debug [796d,2301,0de1,0300,c5e1,3800]
AppleBCM5701Ethernet: 0 1 BCM5701Enet::replaceOrCopyPacket worked after N tries
```

When this machine booted, it eventually configured an ethernet NIC named AppleBMC5701Ethernet, the system device name is en0 and it follows the Gigabit standard in full-duplex mode and with flow-control.

### Exercise 3. Listing all the NICs installed

Observe below the output obtained at a Linux system that has three NICs, *i.e.,* three Network Adapters and respond to these questions:

1. How many NICs are installed in the host (Internet host is a computer system that has the Internet protocols installed)?
2. Is lo0 a NIC also?
3. List the IP addresses applied to the interfaces
4. According to your observations, can a host have more than one IP address?

```
$ ifconfig
enp1s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.2.99  netmask 255.255.255.0  broadcast 192.168.2.255
        inet6 fe80::e2d5:5eff:fe27:2f72  prefixlen 64  scopeid 0x20<link>
        ether e0:d5:5e:27:2f:72  txqueuelen 1000  (Ethernet)
        RX packets 12664002  bytes 8520448423 (7.9 GiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8751984  bytes 8692363357 (8.0 GiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp3s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.99  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::8616:f9ff:fe03:c504  prefixlen 64  scopeid 0x20<link>
        ether 84:16:f9:03:c5:04  txqueuelen 1000  (Ethernet)
        RX packets 3189873  bytes 249203072 (237.6 MiB)
        RX errors 0  dropped 105601  overruns 0  frame 0
        TX packets 4888951  bytes 6790010291 (6.3 GiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp4s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.100.100  netmask 255.255.255.0  broadcast 192.168.100.255
        inet6 fe80::93eb:1f8b:5d1b:a225  prefixlen 64  scopeid 0x20<link>
```

```
        ether 18:d6:c7:02:8b:59  txqueuelen 1000  (Ethernet)
        RX packets 4621272  bytes 405429294 (386.6 MiB)
        RX errors 0  dropped 19240  overruns 0  frame 0
        TX packets 17830  bytes 1474441 (1.4 MiB)
        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
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1  (Local Loopback)
        RX packets 5005022  bytes 10502237579 (9.7 GiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 5005022  bytes 10502237579 (9.7 GiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Not all operating systems produce the same information about each NIC installed. Observe the output obtained at an OS-X system:

```
$ ifconfig en0
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
 options=2b<RXCSUM,TXCSUM,VLAN_HWTAGGING,TSO4>
 ether 3c:07:54:52:ae:2a
 inet6 fe80::3e07:54ff:fe52:ae2a%en0 prefixlen 64 scopeid 0x4
 inet 192.168.2.106 netmask 0xffffff00 broadcast 192.168.2.255
 media: autoselect (1000baseT <full-duplex,flow-control>)
 status: active
```

1000BASE-T means that our NIC has successfully negotiated a fully functional Gigabit Ethernet over copper link (1000 Mbps, baseband signaling over cat 5e or cat6 structured cabling). If either the transmission medium itself or its physical surroundings are such that that speed can not be reliably attained, the NIC will fall down to a 100Mbps speed in a mode known as Fast Ethernet. As you can see, Ethernet technologies play an important role in today's campus networking and is the reference technology employed in the deployment of the networks of the University of Leon.

## Exercise 4. Basic Linux commands

The following commands usually offer a lot of help when working in Linux. Play with each and try to interpret the results.

**$ man ifconfig**

The man command uses to offer very valuable and reliable information, also, you can find a lot of information about ifconfig in internet. The ifconfig command is an example of a command that can be found in a number of systems, however, that does not mean they accept exactly the same option. Some options may or may not be available depending on the specific Linux/UNIX system considered. Using ifconfig for setting up the network interface entails root privileges, therefore, either you switch user to root via the su or sudo commands (Recall that the NIC device names of the specific system you are working at will not be the same that appear in the examples below):

**$ su** or **$ sudo su**

Alternatively,

You can submit the ifconfig command as a sudo parameter, in which case, the super-user change will apply to the execution of the ifconfig command:

```
$ sudo ifconfig enp0s4 up
```

The latter command activates the interface enp0s4 if it were not activated at the time, or it will do nothing in case it was already activated. It's an example of ifconfig execution that requires super-user privileges since it attempts to modify a network interface configuration.

Lastly, if your shell can not find ifconfig, issue a whereis command by passing it the name of the command which file system location you wish to find out. The example right below asks the location of ifconfig and the system responds with **/sbin/ifconfig**. You can use the answer to compose the command's full path name.

```
$ whereis ifconfig
/sbin/ifconfig
```

Here, we proceed to execute the command:

```
$ /sbin/ifconfig
```

If you suspect that you'll be often using commands from the /sbin folder, maybe you should append it to your PATH environment variable in your $HOME/.profile file; add the following line to the file:

```
$ PATH=$PATH:/sbin
```

The following commands should be helpful in these practices, also:

Print the current directory:

```
$ pwd
/home/student
```

Change directory to folder practice-1 which resides in my current directory:

```
$ cd practice-1
```

Listing the files in the current directory

```
$ ls -l
-rwxr-xr-x  1 networks networks       8640 Oct 17 20:22 sizes
-rw-r--r--  1 networks networks        281 Oct 10 12:59 sizes.c
-rw-r--r--  1 networks networks       2378 Jan 11 14:58 sockoptions.c
drwxr-xr-x  4 root     root           4096 Apr 13  2018 sphinx
drwxr-xr-x  6 root     root           4096 Mar 24  2018 tutorial
```

Capture the standard output of the command ifconfig eno1 into a file

```
$ ifconfig enp4s0 > mydoc
```

Now, you can edit that file with an editor (gedit, for example) or a command line visual editor like vi:

```
$ vi mydoc
```

Add some text for separating the different sections of the documentation you're generating to ease editing later:

```
$ echo "------- Exercise 23 -------" >> mydoc
```

If you wish to *append* more text to file mydoc, you can do it with the shell append operator (Represented by the digraph >>); for example, I'm going to append the list of UDP sockets active in my system now to the foregoing file. The netstat command generates the listing and grep will filter the lines containing the string "udp"; the >> operator will append the foregoing output to the contents of file mydoc:

```
$ netstat –av | grep udp >> mydoc
```

Now, we check the file contents is what we expected:

```
$ cat mydoc
enp4s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.100.100  netmask 255.255.255.0  broadcast 192.168.100.255
        inet6 fe80::93eb:1f8b:5d1b:a225  prefixlen 64  scopeid 0x20<link>
        ether 18:d6:c7:02:8b:59  txqueuelen 1000  (Ethernet)
        RX packets 27376  bytes 2038864 (1.9 MiB)
        RX errors 0  dropped 1137  overruns 0  frame 0
        TX packets 356  bytes 28882 (28.2 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

------- Exercise 23 -------

udp        0      0 0.0.0.0:35429           0.0.0.0:*
udp        0      0 0.0.0.0:ipp             0.0.0.0:*
udp        0      0 0.0.0.0:mdns            0.0.0.0:*
udp        0      0 0.0.0.0:1900            0.0.0.0:*
udp        0      0 protocol:ntp            0.0.0.0:*
udp        0      0 protocol:ntp            0.0.0.0:*
udp        0      0 protocol:ntp            0.0.0.0:*
udp        0      0 localhost:ntp           0.0.0.0:*
udp        0      0 0.0.0.0:ntp             0.0.0.0:*
```

Installing new software under Debian or Ubuntu Linux entails using the following commands:

Update your distribution's package repositories:

```
$ sudo apt-get update
```

Update your distribution's packages:

```
$ sudo apt-get upgrade
```

Install a software product; in this case we are installing the ssh (Secure Shell server) by including the package name (Typically, these names can be found by searching the Internet):

```
$ sudo apt-get install ssh
```

### Exercise 5. Ifconfig options

Review the main command-line options of the ifconfig command in your system.

### Exercise 6. Ifconfig for setting NIC parameters

1. Can the ifconfig command be used to configure the NIC or just to report its network parameters? Compose the ifconfig command line that sets the network configuration of your system's main Ethernet interface (Typically en0 or eth0), the interface parameters to be used in this example case are the following:

   IP address 192.168.1.240; netmask 255.255.255.0, etc.

2. Once you submit the command, if it does not report any error, reboot your system and observe the boot process to identify the moment at which the system scans and configures each of the NICs present in it.

3. Finally, log on as administrator and check the interface configuration by using ifconfig. Did the new configuration survive the reboot? If not, what administrative steps would you take to have the system automatically set the interface configuration that the administrator wishes?

4. Search the internet for the specific steps that *your Linux distribution* entails for making ifconfig configuration changes permanent. (The distributions at use in our lab are Debian and Ubuntu).

**Exercise 7. Ifconfig for setting the NIC MTU (Maximum Transfer Unit)**

1. Execute the ifconfig command and take note of the MTU of your main network interface.

2. What does MTU mean? Explain your answer.

3. Can the MTU be changed? If, so, change your MTU to 1000, then reboot your system and see whether the change survived the reboot.

So far, we have reviewed the ifconfig command which is used to setup the network interfaces of the system, now, we are to move to explaining the use of several utilities that help us establish whether our system has internet connectivity with other hosts in the Internet.

## The ping command

The UNIX ping command is used to see whether an Internet host has IP connectivity to another Internet host. The ping name stands for *Packet Internet Gopher* and is an old-time utility in Internet hosts; it functions by using the ICMP (Internet Control Management Protocol) protocol which is an essential IP control-plane protocol. Depending on the options included by the user, ping can send and receive different ICMP protocol messages. The simplest form of the ping command sends the ICMP ECHO message to the specified destination IP address and the destination (receiving) IP module[3] will react by sending back the same ECHO message. The sending ICMP can measure the Rtt (Round Trip Time) upon receipt of the ICMP ECHO back message. Below is an example of ping executed in OS-X: (If you want to stop the ping command, compose the key combination ctrl-C)

```
$ ping www.telefonica.net
```

[3] Module refers to the software organization that implements the TCP/IP protocols in the Linux operating system, including ICMP

```
PING www.telefonica.net (213.4.130.95): 56 data bytes
64 bytes from 213.4.130.95: icmp_seq=0 ttl=120 time=43.752 ms
64 bytes from 213.4.130.95: icmp_seq=1 ttl=120 time=41.791 ms
64 bytes from 213.4.130.95: icmp_seq=2 ttl=120 time=43.660 ms
64 bytes from 213.4.130.95: icmp_seq=3 ttl=120 time=43.293 ms
64 bytes from 213.4.130.95: icmp_seq=4 ttl=120 time=43.070 ms
64 bytes from 213.4.130.95: icmp_seq=5 ttl=120 time=42.248 ms
64 bytes from 213.4.130.95: icmp_seq=6 ttl=120 time=43.548 ms
64 bytes from 213.4.130.95: icmp_seq=7 ttl=120 time=43.364 ms
64 bytes from 213.4.130.95: icmp_seq=8 ttl=120 time=43.014 ms
64 bytes from 213.4.130.95: icmp_seq=9 ttl=120 time=42.474 ms
64 bytes from 213.4.130.95: icmp_seq=10 ttl=120 time=43.837 ms
^C
--- www.telefonica.net ping statistics ---
11 packets transmitted, 11 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 41.791/43.096/43.837/0.635 ms
```

Depending on the specific platform, the output may differ but essentially ping tells whether there exists IP connectivity between two Internet hosts. As we will review, the Internet architecture has a single, central protocol in the network layer, its name is IP (Internetwork Protocol), there is no other network protocol in this architecture. From an implementation standpoint, the IP module belonging to the TCP/IP stack of a typical Linux/Unix/Window must contain other two modules in turn: the ARP and the ICMP. As we mentioned above, the ARP module maps internetwork addresses (IP addresses) to level-2 MAC addresses (The hardware address of an Ethernet NIC, for example), the ICMP protocol belongs to the control plane, that is, it is not involved in effective data transfers but cares for those transfers by coordinating the network equipment in ways that we will take up later. Together, the three protocol modules: IP, ARP and ICMP constitute what is normally known as the *IP module.* The technical specifications corresponding to the protocols that govern the Internet are called RFCs (Request For Comments) by the IETF (Internet Engineering Task Force), for instance, the RFC that documents the ICMP protocol is RFC 792. It's common that some RFCs affect other future RFCs, sometimes to the point of being superseded by them. At the beginning of an RFC, a list of affected RFCs appears alongside its past history. To finish this brief foray into the realm of Internet protocols we must recall that ICMP messages are encapsulated in IP datagrams; we will delve into these important concepts in chapter 1 and further.

### Exercise 8. ICMP protocol messages

1. Which ICMP code defines an ECHO REQUEST? And, which ICMP code defines an ECHO REPLY?

2. Search the Internet for RFC 792; these documents are just a simple Internet search away and are available in several formats for improved viewing, searching, etc. Skim that RFC 792 and search for the message type and code that defines an ICMP Echo and an ICMP Echo Reply.

3. Speculate about different scenarios where the response to an ICMP echo request is not received? In that case, what is the behavior of the originating node? See the following example and try to reproduce it in your computer, maybe by trying to contact other hosts that might have not been powered up yet (Compose IP addresses that that belong to our lab's network whose network number is 192.168.1.0 and whose netmask is 255.255.255.0, for example: 192.168.1.120, 192.168.1.121, etc.).

```
$ ping 192.168.99.102
PING 192.168.99.102 (192.168.99.102): 56 data bytes
Request timeout for icmp_seq 0
Request timeout for icmp_seq 1
```

```
Request timeout for icmp_seq 2
Request timeout for icmp_seq 3
ping: sendto: No route to host
Request timeout for icmp_seq 4
ping: sendto: Host is down
Request timeout for icmp_seq 5
ping: sendto: Host is down
```

4. The procedure to have a host not respond to ICMP echo requests consists of setting kernel parameter to the right integer value (1) as is illustrated below. Use ping to test connectivity with a host that has disabled the echo responses and check whether other Internet services such as the Web keep functioning.

```
# List the kernel parameters that contain echo_ignore in their name
$ sysctl -a | grep echo_ignore
net.ipv4.icmp_echo_ignore_all = 0
net.ipv4.icmp_echo_ignore_broadcasts = 1

# Disable ICMP echo responses at this host
$ sudo sysctl –w net.ipv4.icmp_echo_ignore_all=1

# Enable ICMP echo responses at this host
$ sudo sysctl –w net.ipv4.icmp_echo_ignore_all=0
```

**Exercise 9. Other tasks with the ping command**

1. Find your system's current IP and send ping to it (Execute ifconfig). Do you you receive echo response? If that is true, observe the RTTs (Round-trip times), check that those RTTs are substantially smaller than those received when pinging www.telefonica.net, for instance or www.princeton.edu or www.cisco.com.

2. In the preceding exercise you contacted your own IP in order to establish whether connectivity from your computer to itself were possible, that is a very common case in which a client application wants to connect with a server application running on the same system. What would happen if effectively the network to which your system is connected were down and yet you needed to test your client/server connection? That testing is possible even in the case your network is down, to that end, every IP system has a special IP address (**127.0.0.1 or host name localhost**). See the following example and execute the same test it in your system:

```
$ ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.066 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.139 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.181 ms
^C
--- 127.0.0.1 ping statistics ---
3 packets transmitted, 3 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 0.066/0.129/0.181/0.048 ms

$ ping localhost
PING localhost (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.050 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.179 ms
```

```
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.179 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.179 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.174 ms
^C
--- localhost ping statistics ---
5 packets transmitted, 5 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 0.050/0.152/0.179/0.051 ms
```

**Exercise 10. Other options of the ping command**

Explore other ping options (Execute man ping) and flags in your specific operating system –these options and flags are not the same across the sundry of mainstream operating systems of today. Tell us which ones seem useful.

## Checking the state of your system's TCP/IP stack: netstat

**netstat** is a versatile and powerful command that has a lot of options for checking the state of the TCP/IP stack of your system; the stack is made up of all the network software modules that take responsibility for the TCP/IP protocols and services in your system, the word stack represents the fact that these software modules are organized in a hierarchical manner that might remind us of a stack of modules.

Your system's networking software is made up not only of your NIC interfaces, obviously other essential pieces are built upon pure software as is the case with the IP module (Recall this module comprises modules for IP/ARP/ICMP). If we look up the hierarchy from the physical level of your Gigabit Ethernet (GBE) adapter we find the signaling electronics that belong to the integrated circuits that make up your GBE adapter, then, at level 2 we find two sublayers: the MAC and LLC, the former providing the correct *shared medium* access and the latter builds the frames with the IEEE 802 standardized structure. Then upwards in the hierarchy we find the IP module made in pure software and last by now, at level 4 we find two transport protocol modules, the powerhouse TCP and the more humble UDP and, last in this illustration, at level 5 (session) we find the famous Sockets that ultimately connect applications in a an end-to-end fashion over the internet: yes, the internet connects clients to servers and the clients are implemented by application threads running on an average operating system like Windows and the server, likewise, runs on threads of an average operating system like Unix. If we want to take such a overall-hierarchy look we use the netstat command., see the following glimpse at the output of the netstat command in a BSD-Unix-based operating system (Apple's MAc OS-X):

**Exercise 11. Watching the connections active in my host at this time**

Start your web browser and compose the following URL (Universal Resource Locator, a standardized form of pointer to web resources): http://examples.oreilly.com/9780596007218/. When the web page shows up, click on the zip file's link displayed, the associated file contains the Java code corresponding to a nice book on Java Network programming from O'Reilly, now, while the file is being downloaded type the following command on your terminal, this command will let you see the end-to-end TCP connection that binds your system to O'Reilly's, you can see the result from my own system here:

```
paloalto:~ Chema$ netstat  -t | grep oreilly
tcp4       0      0  192.168.99.99.53921    ftp.oreilly.com.http   ESTABLISHED
tcp4       0      0  192.168.99.99.53920    ftp.oreilly.com.http   ESTABLISHED
```

If you want to see the full listing of TCP connections to/from your system and their associated state, suppress the stdout filtering grep from the command line. In the case above, the web browser established two different connections from my system to O'Reilly, one on TCP port 53921 and the other 53920, the technical details that justify the two TCP connections will be studied in our brief exposure to the application level in chapter 9. Briefly explained, the output from the command manifest the two TCP connections, their current state (ESTABLISHED)at the time the command is executed, the originating IP address, which in both cases is 192.168.99.99 and the destination IP address which in this case is represented by its DNS (Domain Name System) equivalent "ftp.oreilly.com". The originating TCP ports in 192.168.99.99 are 53921 and 53920 respectively, as to the destination end of both connections, you can see that the port is the same (http) which number is 80, you can check that http as a port name corresponds to TCP port number 80 by grepping against your local port names database (/etc/services):

**Exercise 12. Seeing the contents of the services file in your Linux system**

Check whether your /etc/services file contains the name for TCP port number 80:

```
paloalto:~ Chema$ egrep "80/" /etc/services
http            80/udp     www www-http # World Wide Web HTTP
http            80/tcp     www www-http # World Wide Web HTTP
```

TCP port number 80 corresponds to the http service, i.e., access to the World Wide Web with its protocol, http (Hyper Text Transfer Protocol).

**Exercise 13. The options of the netstat command**

Consult the options available to the netstat command in your operating system. Check whether you have available some graphical front end to netstat, for example, in MAC OS-X, under applications/utilities you have such a GUI-based utility named "Network Utility":


Figure 1. A network monitoring utility from the OS-X system

GUI-based utilities like this come in handy when one does not want to memorize more and more commands and options,

this is the result of executing it:

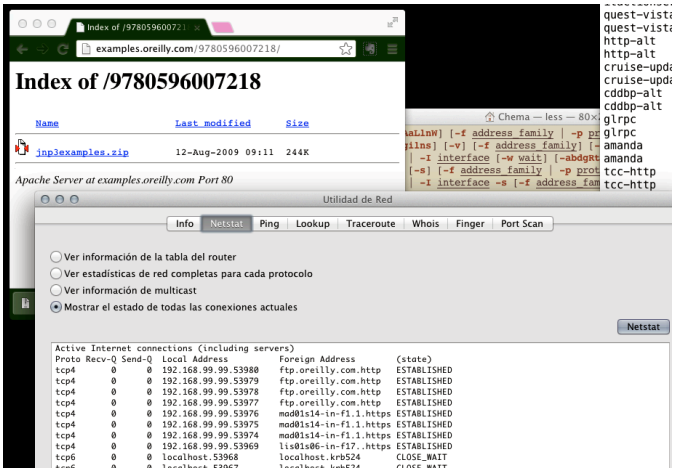


Figure 2. Output from netstat on the GUI network utility of the OS-X system

I am sure you have observed that there are other network utilities available, we will introduce some of them in this and upcoming experiments.

## Trace the path to an end system in Internet: the traceroute command

The ICMP protocol messages can be used for purposes other than IP connectivity, for example, to trace the network pathway from an originating system to a destination system. The use of ICMP for this purpose is somewhat tricky and involves the use of a field of the IP packet named TTL or *Time To Live.*

When an IP router receives a new IP packet, it decrements its TTL field (TTL = TTL - 1) so it will eventually reach the value 0. IP prescribes that an IP packet's TTL reaching 0 should be discarded altogether (The packet is dropped). TTL was conceived as an upper bound to the number of routers an IP packet should cross on its trip to its destination, thus, in case of a network design or network management error caused a network loop, packets would have a guarantee not to loop in there forever. A looping packet would waste a variety of network and compute resources, and consequently it has to be avoided at all costs.

The traceroute utility aims to estimate the sequence of routers that comprise the network pathway to a destination in Internet, at the specific time the test is being carried out (Network pathways can change at any time, due to the changing nature of Internet congestion and availability of links. The traceroute utility, at a host, sends a test IP packet containing a null-payload UDP message from UDP port 33434 (By default) with TTL=1. This IP packet, upon being received by the first IP router on the pathway to a destination chosen by the user, will get TTL=0 and the router will send back an ICMP *Time Exceeded Message (TEM).* When the traceroute host receives this message, it knows about the existence of the first router: the received IP packet will tell us its IP address, etc. Now, traceroute will send (To the same destination indicated by the user) a new test IP packet containing TTL=2, which will elicit an ICMP *TEM* from the second router on the pathway. This process of sending a test packet and expecting an ICMP *TEM* continues until it is the destination host which sends the ICMP TEM, at which time, traceroute will finish.

```
Brainstorm:~ chema$ traceroute www.princeton.edu
traceroute to www.princeton.edu (128.112.132.86), 64 hops max, 52 byte packets
 1 192.168.2.1 (192.168.2.1) 1.362 ms 1.030 ms 0.713 ms
 2 192.168.1.1 (192.168.1.1) 1.626 ms 1.726 ms 1.441 ms
 3 192.168.153.1 (192.168.153.1) 36.751 ms 36.398 ms 35.612 ms
 4 241.red-81-46-53.staticip.rima-tde.net (81.46.53.241) 35.946 ms 37.531 ms
37.617 ms
 5 so2-0-0-0-grtmadpe3.red.telefonica-wholesale.net (84.16.6.201) 42.791 ms 44.928
ms 43.290 ms
 6 xe2-1-0-0-grtpartv2.red.telefonica-wholesale.net (84.16.15.170) 66.932 ms
    xe7-1-2-0-grtparix1.red.telefonica-wholesale.net (213.140.36.134) 121.620 ms
63.980 ms
 7 xe-6-0-6-0-grtwaseq2.red.telefonica-wholesale.net (94.142.116.233) 155.683 ms
    xe7-1-2-0-grtwaseq2.red.telefonica-wholesale.net (94.142.126.81) 146.309 ms
    xe6-1-4-0-grtwaseq2.red.telefonica-wholesale.net (94.142.116.217) 146.060 ms
 8 xe5-0-1-0-grtwaseq5.red.telefonica-wholesale.net (213.140.36.61) 146.071 ms
```

```
Brainstorm:~ chema$ traceroute paloalto.unileon.es
traceroute to paloalto.unileon.es (193.146.101.46), 64 hops max, 52 byte packets
 1 192.168.2.1 (192.168.2.1) 3.036 ms 0.950 ms 0.689 ms
 2 192.168.1.1 (192.168.1.1) 1.426 ms 1.491 ms 1.324 ms
 3 192.168.153.1 (192.168.153.1) 36.583 ms 37.640 ms 37.073 ms
 4 130.red-81-46-37.staticip.rima-tde.net (81.46.37.130) 38.480 ms 36.203 ms
37.621 ms
 5 rediris-2.espanix.net (193.149.1.154) 49.484 ms 48.600 ms 47.699 ms
 6 ciemat.ae2.uva.rt1.cyl.red.rediris.es (130.206.245.10) 49.008 ms 48.970 ms
49.420 ms
 7 unileon-router.red.rediris.es (130.206.201.46) 56.238 ms 56.826 ms 56.040 ms
...
```

**Exercise 14. Establishing the network path to a host: the traceroute command**

Repeat the traceroute to paloalto.unileon.es from the network location where you are, let the trace complete, contrast the results to the screen dump above.

**Exercise 15. Migrating your wired network connection to a WIFI AP**

Move your network access point to the wireless access point and repeat the traceroute of exercise 1 and try to make sense of the results, how they change when you simply move from a wired concentrator to a Wi-FI AP, all within the campus network of the University.

**Exercise 16.1 Traceroute to a network location outside of Red Iris**

Do a traceroute to a network location that does not belong to Red Iris, for instance www.hp.com, can you tell the DNS name or the IP address of the first routing interface that does not belong to Unileon?

Some of the intervening network nodes will block the outbound ICMP traffic for security reasons, that is the reason why you will see some routers represented by asterisks (* * *).

**Exercise 16.2. The functioning of traceroute is based on the ICMP and UDP protocols**

Run Wireshark or tcpdump and develop the protocol stack resulting from a traceroute utility sending a test IP packet.

This question solved right below so you can follow a model in solving the ensuing two questions. Notice that the protocols in the protocol stack (Figure 3, right) are ordered into the same top-down order used in the architecture (4 through 1), however, the graphical representation displayed by Wireshark (Figure 3, left) is ordered in exactly the reverse order (The protocol appearing on the top is the physical Ethernet (Frame). Take this fact into account when deducing protocol stacks from Wireshark.

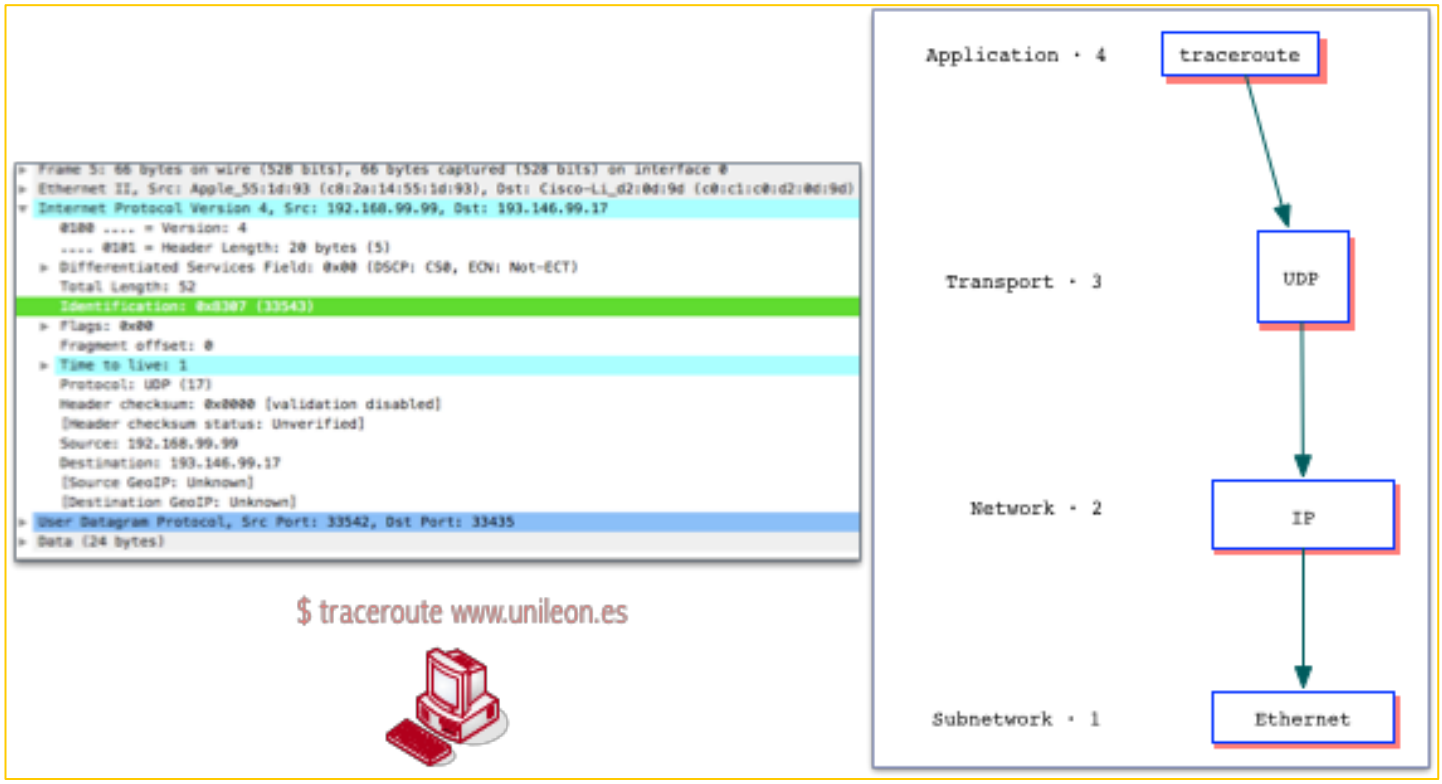



Figure 3. The protocol stack resulting from a traceroute probe packet (Right). The trace was obtained with Wireshark (Left)

**Exercise 17. Traceroute sending further UDP packets with increasing TTL**

Shortly after the sending traceroute receives each ICMP TEM, it sends a new test IP packet with a TTL value one higher than the one just sent (Actually, several UDP test packets can be quickly sent in succession, without waiting to receive any response). Carefully check whether or not you faithfully observe the foregoing behavior in the Wireshark trace. Finally, compare it to other systems' traceroute implementations for if you observe some differences (Windows traceroute counterpart utility is named tracert).

**Exercise 18. Traceroute protocol stack**

Develop the protocol stack resulting corresponding to any of the ICMP TEM responses. Each of these is sent from any IP router laying amidst the pathway to an Internet destination of your choosing (For example, www.uclouvain.be) after receiving a test packet from traceroute which TTL=1.

## Virtual terminal and file transfer utilities

One the oldest command utilities in the development of Internet is telnet which allows us to connect remotely to another system, log in and then, after the prompt symbol appears on the screen you will be able to submit commands to the remote system and transparently receive its responses as though you were directly connected with a cable, serial connection.

Today, telnet is not used very much because of its lack of security, all the character strings that make up the commands typed locally and sent over the internet to the host system are sent in the clear, that is, with no protection against eavesdropping. When we want to sent those strings with guarantees of privacy we encrypt them by using a symmetric-key that will allow the receiver to decrypt them and getting the original text sent. telnet is considered so insecure that network administrators will close the telnet TCP port by default thereby filtering any traffic to or from that port. Nevertheless, the functionality offered by telnet must be important: to be able to submit commands to a remote machine, then, what is the solution? There is a wide spectrum of solutions to this problem which involve the use of different forms of cryptography, one of the most famous consists in using a secure virtual terminal program named ssh or Secure Shell. There are open software versions of ssh for Linux, Unix and Windows.

**Exercise 19. Ssh to paloalto**

Does paloalto.unileon.es have its ssh port open? Which port number is it? Consult this in the /etc/services text file and search for ssh.

**Exercise 20. Observing ssh connections**

Type the ssh command in your terminal and connect with the host name or IP address written on the Lab's white board (Usually it will be 192.168.1.254). If it is the first time that you connect with that host with ssh, your local ssh client will output a fingerprint of the remote host for you to check that it effectively corresponds to 192.168.1.254, this helps avoid the man-in-the-middle attack in case you do know the remote host fingerprint, see the following example:

```
Brainstorm:~ chema$ ssh chema@paloalto.unileon.es
The authenticity of host 'paloalto.unileon.es (193.146.101.46)'
can't be established.
RSA key fingerprint is
d2:23:8d:cb:af:65:7a:91:d3:b7:1c:4a:74:0d:c1:9c.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'paloalto.unileon.es,193.146.101.46'
(RSA) to the list of known hosts.
Password:
Password:
```

```
Last login: Thu Feb 28 11:18:42 2013 from 139.red-81-35-
94.dynamicip.rima-tde.net

paloalto:~ Chema$ who
Chema console Feb 20 13:22
Chema ttys000 Feb 27 11:02
Chema ttys001 Feb 27 11:02
Chema ttys002 Feb 28 11:22 (139.red-81-35-94.dynamicip.rima-tde.net)
```

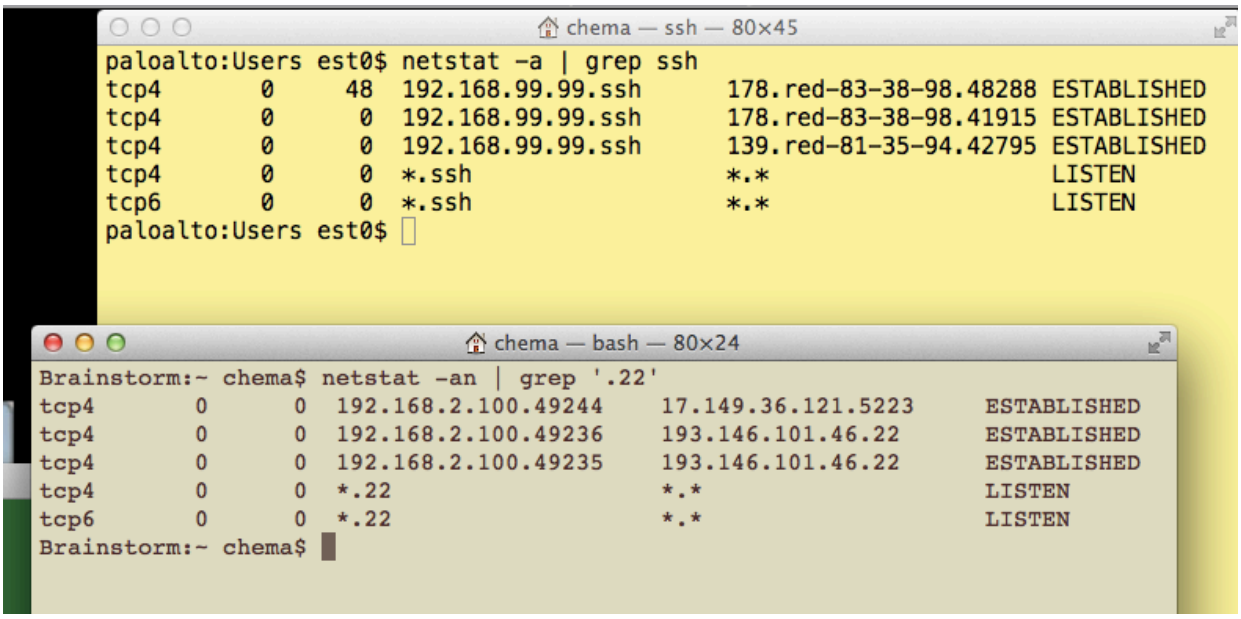
Observe that when we have effectively logged in, the command prompt is not "Brainstorm" anymore but "paloalto". I typed the who command, you can see its output reporting four sessions, the first on the host graphical console, the other three on network consoles (ttys*) the last one corresponding to my session (ttys002) which manifests the public IP address that was granted by my ISP (Internet Service Provider).

### Exercise 21. man ssh

Scan the ssh man page in your system, get an overall idea of the information usually available in Unix/Linux man pages.

### Exercise 22. Observing ssh connections locally and remotely

Establish a remote session with user est? of 192.168.1.254[4], once you have logged in successfully, execute a netstat command that allow you to see your ssh connection from the host standpoint, then, request a new local terminal and type the same netstat command so that you contrast the local/remote information displayed by both commands. Here is an example where the local system name is Brainstorm and the remote is paloalto.unileon.es:



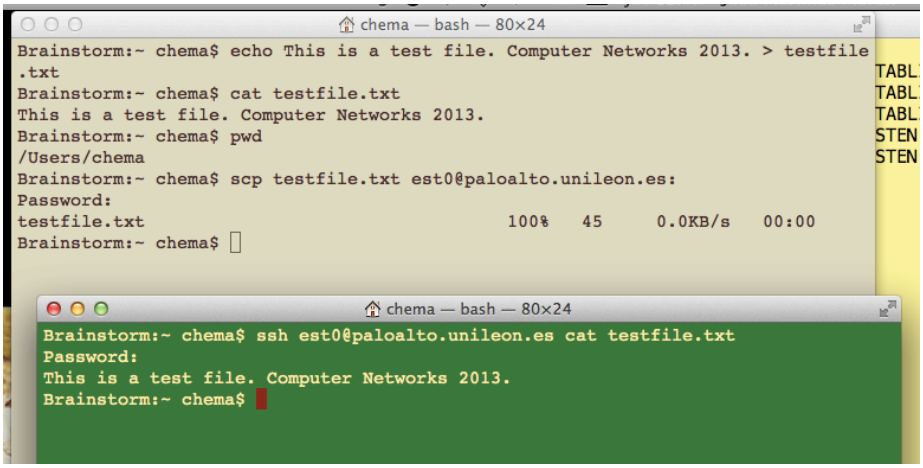

The output above reports some connections in state "ESTABLISHED" and others in state "LISTEN", the former represent the effective connections between an ssh client and a ssh server, the latter represents a connection point that, if requested, can result in real connections, we call this a "welcoming socket" in Sockets parlance, we will study the Sockets API soon in a

[4] Users est0 – est9 have been created at 192.168.1.254. Use the habitual lab password to log in.

specific introductory laboratory session.

Another important utility in the initial development of Internet was the *file transfer utilities.* At the time there was ftp or File Transport Protocol which is still in use today, though it has the same lack of security problems that we mentioned above regarding telnet. Both utilities, telnet and ftp have the same origin in the US DoD (United States Department of Defense), for that reason, both utilities are known as DARPA services (Defense Advanced Research Projects Agency). Other similar utilities originated in a different set of software project led by the University of California, Berkeley, the names of those utilities were rlogin and rcp, these utilities are known as Berkeley services. Having the same security problems already mentioned, another secure file transfer utility was developed: scp, actually, ssh and scp both use the same underlying secure-transport protocol, though for different purposes.

19. Let's transfer a file named testfile.txt from our current directory to the home directory of user student at host 192.168.1.254. Study the following example, then try to upload a file of your choosing to user student of host 192.168.1.254 and check that the file was successfully uploaded (A small file suffices).




On the terminal with green background I requested the remote execution of the command "cat test file.txt" as a check that the file was actually uploaded, I did not establish a full session as in the above examples, I requested the remote execution of a single command. This capability of ssh comes in handy in many situations.

**Exercise 23. scp for uploading and downloading files**

Execute a secure file download by using scp of any file that you know that exists on the student user account in 192.168.1.254, you will have to use the correct syntax by specifying the remote file and the the local resulting file name; please, skim the scp manual page if necessary.

## Check of Java Compiler and run-time

In this laboratory session we will want to check that your Java infrastructure is operational, to that end, we are going to practice with a Java Network API named NetworkInterface which allows us to determine the network interfaces installed in our computer and obtain some of its properties.

In the initial sections of this laboratory experiment we presented a series of commands related to your system network configuration, specifically ifconfig, but, ifconfig is a finished, operational command that offers us a lot of functionality, nevertheless in some situations one needs to have programmatic access to the network interfaces from Java. The following NifEnumerator.java obtains a list of the network interfaces installed in your system and prints them out on the console.

**Exercise 24. Installation and check of C and Java compilers**

Download the source code of the java example program illustrating the NetworkInterface API, use the following URL:

http://paloalto.unileon.es/cn/labs/NIfEnumerator.java

Observe the package name used if any and compile and run the program. Contrast the results obtained to those reported by the ifconfig program that we explained above.

**Exercise 25.** Look up the Java Docs for the Java NetworkInterface API, scan the methods and the most outstanding fields. Then access the Oracle Java Tutorial and scan the section devoted to NetworkInterface in it.

**Exercise 26.** Study the NifEnumerator.java source code, particularly those sections that might result least familiar to you, make sure you recall the Java Collections used.

## Check of C Compiler

In order to check your C compiler installation (GNU C compiler), carry out the following exercise, making sure that you understand each of its parts, since the involved facilities will often prove handy in other, upcoming CN Labs:

**Exercise 27.** Installing and checking the Gnu C compiler (gcc)

a. Install gcc, login to your Linux PC and issue the following command to start the installation:

```
$ sudo apt-get install gcc
(Provide your Linux user's password and respond Y to the ensuing prompts)
```

b. Download a short, test C program from our CN web server (paloalto), you may use the wget program from the command line or copy-paste the URL into your Firefox address box and, then, pressing the return key:

```
$ wget http://paloalto.unileon.es/cn/labs/check-gcc.c

$ ls –l check*
-rw-r--r--  1 Student  admin   171 22 feb 22:55 check-gcc.c

$ gcc –o check-gcc check-gcc.c
(If the compiler issued no errors, execute the program and observe the
message printed out on your terminal window)
```

```
$ ./check-gcc
Hello world, you compiled and ran this program successfully!
```

On next practice, we will write a simple networking program in C using one of the Sockets API.