V.2.3 – 26th · Feb · 2024

Practicals on Computer Networks and Distributed Systems

Network Interface Configuration (WIP)

Internet end systems, or Internet hosts, connect to Internet by means of a Network Interface Cards (NIC for short). The PCs in our laboratories use the GNU/Linux operating system (Usually the Debian or Ubuntu distributions). In this practical we set out to study the commands that allow our PCs to configure, manage and monitor the NIC that is used for Internet communication. You can check the commands and procedures explained hereon by using your own Linux laptop or one of the PCs in Lab B6.

Physical network interfaces

At the beginning of this course on Computer Networks we study that networking is a systems discipline. Their complexity is tackled by conceptually organizing networks into four conceptual layers. Each layer offers services that can be used by calling the layer's interface. You simply call the interface in order to make the service work. 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 (Apple Berkeley Unix OS) 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 configuration listing capabilities and will disregard its configuration setting capabilities altogether.

Example 1. In the following illustration (You need execute no command yet, this is an illustration only) of issuing the ifconfig command, we assume the **$** symbol as the *shell prompt*: In this specific case, the command execution doesn't require the command's full pathname. That's because the shell PATH environment variable contains the file path where the command is stored in. The **en0** device name passed as first command line argument is a typical Ethernet device name used by the OS-X system from Apple Computer:

[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
```

Exercise 1. Obtaining a listing of <u>all</u> of the NICs installed in your system. The *list hardware* command (**lshw**) provides a listing of all of the hardware installed in your machine. Here, we are interested in network devices only. These are identified with a defining string containing the word 'network' on the output produced by the **lshw** command:

a. Login as administrator

b. Print out the file path to the lshw command in your system

```
$ whereis lshw
 lshw: /usr/bin/lshw /home/administrator/lshw.txt
 /usr/share/man/man1/lshw.1.gz
```

c. Submit the lshw command:

```
$ /usr/bin/lshw
```

The output contains all of your machine's configuration, including the network devices.

d. Identify the NICs in the resulting listing (Search the standard output for the *network* substring). For example, we found the following two network devices in the system mentioned above:

```
*-network
description: Ethernet interface
product: PCI Express Gigabit Ethernet Controller
vendor: Realtek Semiconductor Co., Ltd.
physical id: 0
logical name: enp1s0
…

*-network
description: Ethernet interface
product: Ethernet Connection (7) I219-V
vendor: Intel Corporation
physical id: 1f.6
logical name: eno1
…
```

Observe that the found device names are enp1s0 from the first network device and eno1 from the second one. Those are the device names that you will pass as first argument to ifconfig whenever you want to restrict the configuration to a concrete device, and not want them all. In the two preceding cases, both are Ethernet devices.

Exercise 2. Execute ifconfig to obtain the configuration to the NICs registered in your Linux. The ifconfig command by default produces a listing of all of the network devices configured at boot time. Your system may have other network devices connected, though not configured yet for doing Internet communication.

a. Check the path to the ifconfig executable in your system:

```
$ whereis ifconfig
```

Results obtained at one of Lab B6 PCs:

```
ifconfig: /usr/sbin/ifconfig /usr/share/man/man8/ifconfig.8.gz
```

b. Execute the ifconfig command to obtain the configuration of your network interfaces:

```
$ /usr/sbin/ifconfig
```

Results obtained at one of Lab B6 PCs:

```
eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.89  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::e2d5:5eff:fed8:86a1  prefixlen 64  scopeid 0x20<link>
        ether e0:d5:5e:d8:86:a1  txqueuelen 1000  (Ethernet)
        RX packets 232556  bytes 27020790 (25.7 MiB)
        RX errors 0  dropped 74  overruns 0  frame 0
        TX packets 495878  bytes 611377209 (583.0 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 16  memory 0xa2400000-a2420000

enp1s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::523e:aaff:fe0f:d3c6  prefixlen 64  scopeid 0x20<link>
        ether 50:3e:aa:0f:d3:c6  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 332  bytes 34563 (33.7 KiB)
        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 1000  (Local Loopback)
        RX packets 664  bytes 65549 (64.0 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 664  bytes 65549 (64.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

ifconfig stands for "interface configuration". In this context, the word interface is to be understood as physical network interface. You will find that command in all of the main stream operating systems, including Windows[2]. In example 1, above, the ifconfig command prints out a list of flags and parameter values currently set in the interface. After the OS-X system kernel booted up it detected a stable configuration for the interface, which includes a number of parameters such as the IP address that is to be applied to the device, among others that we'll review soon. The kernel polls the device for its identity (A label that is to be used in future executions of ifconfig). Also, a string describing the device's technology is printed. In this case that technology is IEEE[3] **1000BASE-T** (Observe in the ifconfig output produced by example 1, above). That technology is an Ethernet standard for point-to-point communication where the medium is a set of copper wires connected to the transmission 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 (A LAN switch, for example) support flow-control, that is, the regulated reception of information with no possibility of overflowing the device's limited-size input buffer.

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, or higher structured cabling). If either the transmission medium itself or its physical surroundings are such that that speed cannot be reliably attained, the NIC will fall down to a 100Mbps speed in a mode known as Fast Ethernet. If necessary, that is, if the surrounding electrical noise power is too high for the data signals to be successfully decoded by the receiver, then it will fall back to 10Mbps, the speed used in the original 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.

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 3. Alternate procedure for listing all of the NICs installed in your machine. In Linux, we can search the file system under /sys/class/net for determining the network devices installed in our system. This procedure for finding

[2] In Windows one command to configure your NIC is **ipconfig**
[3] IEEE: Institute of Electrical and Electronics Engineers

out the list of network interfaces is particularly convenient whenever you have lost network connectivity, for whatever reason, and, consequently you cannot download the lshw program from the network, which will eventually let you find out the logical names to your network interfaces. These logical names are necessary for your administering the automatic network configuration upon Linux boot up.

Results obtained at one of Lab B6 PCs (Obtain your own listing at your PC in Lab B6):

```
$ ls /sys/class/net
eno1  enp1s0  lo
```

This machine has three NICs, *i.e.,* three Network Adapters.

Exercise 4. How many NICs are installed in the Lab B6 host where you are working at this time?

Exercise 5. Is lo (Or maybe lo0, lo1, etc.) a NIC also?

Exercise 6. List the IP addresses applied to all of the network interfaces

Exercise 7. According to your observations from the network device listing, can a host own more than one IP address?

Exercise 8. Checklist of Basic Linux commands, essential for the CN course

The following commands usually offer a lot of help when working in Linux. You may practice with each, depending on your previous experience in Linux/Networking. Interpreting the results will enable us to better understand.

```
$ 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 (Notice that the NIC device names of the specific system you are working at will not necessarily be the same ones 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 that ifconfig command, only.

```
$ su
```

```
# ifconfig <Network device in your machine> up
```

The latter command activates the interface passed on to ifconfig on the first command line argument, if it was 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, recall that if your shell cannot find ifconfig command in its PATH variable, you can 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, again seraches for the location of the ifconfig command. The system responds with **`/sbin/ifconfig`**. You can use that answer to compose the command's full path name.

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

Here, we proceed to executing the command by submitting its full pathname:

```
$ /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 that file:

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

Print the current directory:

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

Change directory to folder home:

```
$ cd /home
```

Change to your home directory (Execute the **cd** command without any command line parameters):

```
$ cd

$ pwd

/home/administrator
```

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
```

Redirect the standard output of the command ifconfig eno1 into a file for use in generating documentation:

```
$ ifconfig eno1 > mydoc
```

Now, you can edit that file with an a command line visual editor like vi or vim:

```
$ vi mydoc
```

Editing with vi or vim is out of the scope of this course. Other editors in Linux can be used, for example gedit or nano.

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 Linux entails using the following commands. Switch to super-user first (Commands for managing software in Linux need super-user privileges. The shell's prompt will become #)

`$ su` *(Typical of Debian; in Ubuntu sudo su must be used)*

Type the password. Notice no echo will be sent back to the terminal by the su program.

Update your distribution's package repositories:

```
# apt update
```

Upgrade your distribution's software packages:

```
# apt 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):

```
# apt install openssh-server
```

Now, install the network *useful* **netcat** utility, which we use in this course for simple networking exercises:

```
# apt install netcat
```

Exercise 9. Ifconfig options

a. Review the main command-line options to the ifconfig command in your system and explain one of them that you choose:

b. In exercise 2.b, observe the configuration to device **eno1** at the illustrating example. What's the meaning of the flags string <`UP,BROADCAST,RUNNING,MULTICAST`> ? You may obtain help from the man page to the ifconfig command in Linux.

Exercise 10. Setting a stable network configuration at boot time by using the dhcp protocol.

The Linux NetworkManager program is disabled in all of Lab B6 hosts, consequently, the network configuration must be set by entering the network parameters in the /etc/network/interfaces text file. Its syntax is extensively documented in Debian's docs web site and on the man interfaces page (You should simply type `$ man interfaces` on a terminal).

In the following sections to exercise 10, we aim to make several network configurations which illustrate some of the essential possibilities that the Linux stack offers.

a. Become super-user

```
$ su
```

b. Disable all of the network devices in the PC you are working at by deleting all of the lines in /etc/network/interfaces file except those that contain the configuration to the virtual network device **lo**.

c. Reboot the system.

d. Check that you have no IP connectivity by issuing the following ping test to host 193.146.96.2 which is out of the Lab B6 network:

```
$ ping 193.146.96.2
```

e. Check the network devices currently connected to your PC by listing the files at Linux system directory /sys/class/net

```
$ ls -l /sys/class/net
```

f. Now, you know the names to the network devices (*Typically* eno1 and enp1s0). Choose eno1 (Or the device that is included in your mother board) and insert the two lines that will configure it upon boot; for example, for configuring eno1. Edit /etc/network/interfaces file, type these two lines and save the file:

```
auto eno1
iface eno1 inet dhcp
```

g. Reboot and, then check that you have connectivity to Internet.

```
$ ping 193.146.96.2
```

In this configuration, network device eno1 has obtained an IP configuration by requesting it and downloading it from a server in Lab B6 network. The protocol that allows downloading IP configurations is dhcp (Dynamic Host Configuration Protocol). The dhcp server in Lab B6 is the WRT Router.

Exercise 11. Setting a stable network configuration at boot time by using a *manual* interface configuration

a. Become super-user:

```
$ su
```

b. Edit /etc/network/interfaces. In the configuration to device eno1 replace the string **dhcp** for **manual**. Save the file.

c. Reboot:

```
# shutdown -r now
```

d. After your system reboots, set the following configuration for your eno1 device (Include your unique last IP component in mandatory <last> field, for example 100):

```
$ su

# PATH=$PATH:/usr/sbin

# ifconfig eno1 192.168.1.<last> netmask 255.255.255.0
```

e. Test the default gateway:

```
$ ping 192.168.1.1
```

f. Test connectivity to Internet:

```
$ ping 193.146.96.2
```

g. Was all successful? Follow the instructions provided on the board to configure for full Internet access

Exercise 12. Setting a stable network configuration at boot time by using a *static* interface configuration

a) Skim the man page to the interfaces file in Linux ($ man interfaces) and seek the information about building a static configuration for device eno1. The difference with the previous case is that now you will enter the IP configuration directly in the lines that follow the iface statement:

```
auto eno1
iface eno1 inet static
# Complete this line
# Complete this line
# …
```

b) Check for local and external IP connectivity and solve the problem that might arise.

Reference information. Example of /etc/network/interfaces:

```
auto lo
iface lo inet loopback

auto eno1
iface eno1 inet dhcp

auto enp1s0
iface enp1s0 inet manual
```