



Erle Robotics: Introduction to Networking in Linux

Table of Contents

1. [Introduction](#)
2. [Introduction to Networking](#)
 - i. [What is *Networking*?](#)
 - ii. [Linux networking](#)
 - i. [TCP/IP and packets](#)
3. [Basics: First steps in networking](#)
 - i. [Basic commands](#)
 - ii. [IP address](#)
 - i. [Basic concepts](#)
 - ii. [IPv4 vs IPv6](#)
 - iii. [Assigning an IP address](#)
 - iv. [IP alias](#)
 - iii. [Ethernet](#)
 - i. [ARP protocol](#)
4. [Linux Network Devices](#)
 - i. [Linux Interfaces](#)
 - ii. [Accessing Serial Devices](#)
5. [DNS](#)
 - i. [What is *DNS*?](#)
 - ii. [How *DNS* works](#)
6. [Host](#)
 - i. [Host basics](#)
7. [Network Configuration](#)
 - i. [Network configuration commands](#)
 - ii. [Network configuration files](#)
8. [Internet/Intranet applications](#)
 - i. [Internet basics](#)
 - ii. [Servers](#)
 - i. [Server types](#)
 - iii. [Web](#)
 - i. [Proxy servers](#)
 - iv. [Mail](#)
 - v. [FTP](#)
 - vi. [Authentication services](#)
9. [Remote execution of applications](#)
 - i. [Rsh, rlogin and telnet](#)
 - ii. [The X Window System](#)
 - iii. [SSH suite](#)
 - iv. [Extra concepts](#)
10. [Security](#)
 - i. [Secure services and security tips](#)
 - ii. [Firewall](#)
 - i. [IP Filtering](#)
 - ii. [Iptables](#)
 - i. [Introduction to Iptables](#)
 - ii. [Orders and parameters](#)
11. [Practice](#)
 - i. [General Networking](#)
 - ii. [Security](#)

Erle Robotics: Introduction to Networking in Linux

book **passing**

Book

This book presents an introduction to **networking** in Linux, using [Erle-brain](#). **Erle is a small-size Linux computer for making drones.**

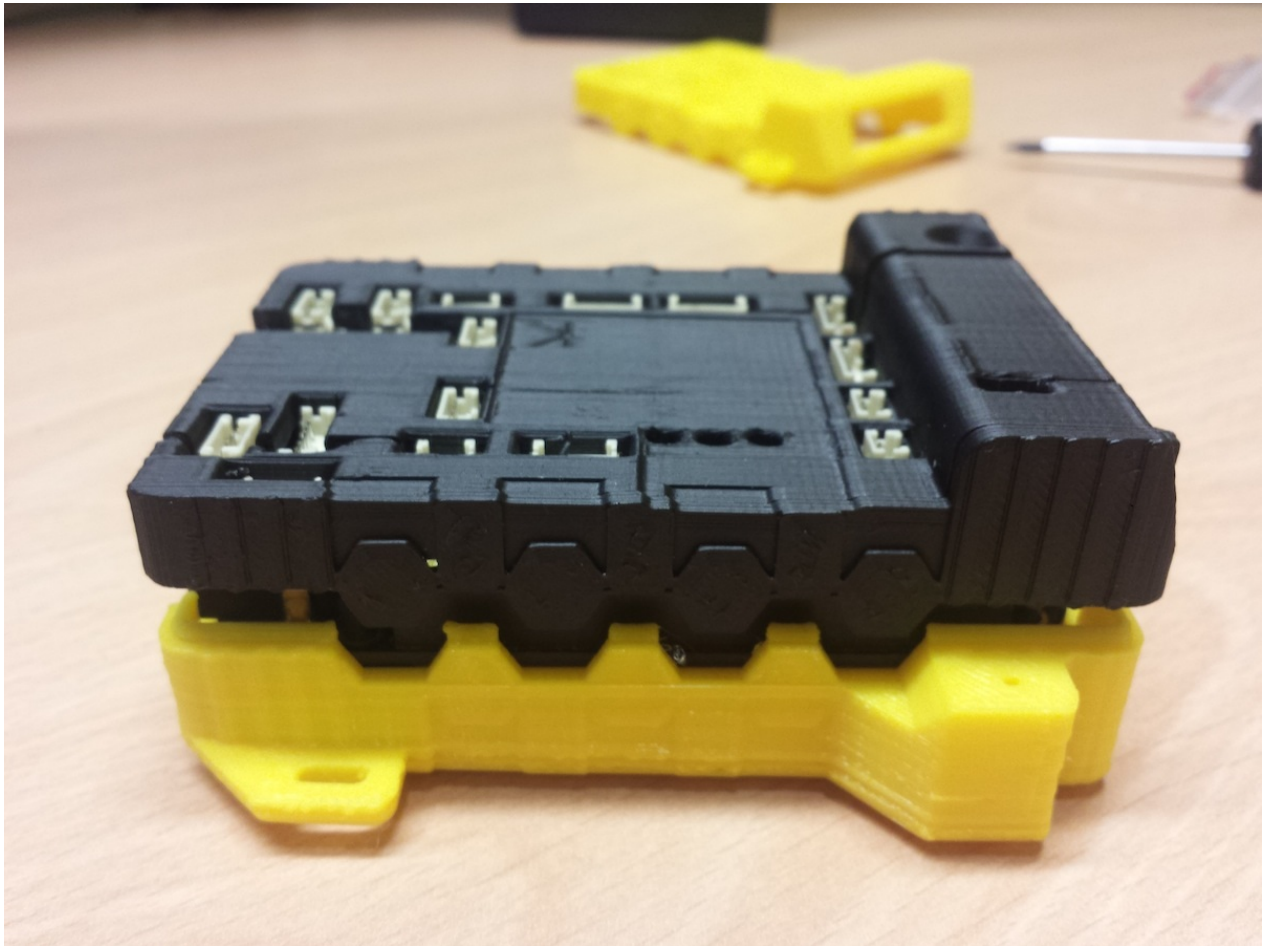
Through this tutorial you will learn the basics about networking, concepts such as host or DNS, the usual commands in Linux networking and their functions...



About

For years we've been working in the robotics field, particularly with drones. We have passed through different Universities and research centers and in all these places we actually found that most of the **drones are black boxes** (check out our [60s pitch](#)). Not meant to be used for learning, research. The software they use is in most of the cases unknown, closed source or *not documented*. Given these conditions, how are we going to educate the next generations on this technologies? How do you get started programming drones if you don't have \$1000+ budget? Which platform allows me to get started with drones without risking a hand?

We are coming up with an answer to all these questions, our technology at [Erle Robotics](#) and our drones brain: [Erle-brain](#).



Inspired by the BeagleBone development board, we have designed a small computer with about 36+ sensors, plenty of I/O and processing power for real-time analysis. Erle is the enabling technology for the next generation of aerial and terrestrial robots that will be used in cities solving tasks such as surveillance, enviromental monitoring or even providing aid at catastrophes.

Our small-size Linux computer is bringing robotics to the people and businesses.

License

Unless specified, this content is licensed under the Creative Commons Attribution-NonComercial-Share Alike 3.0 Unported License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/3.0/> or send a letter to *Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA*.



All derivative works are to be attributed to *Erle Robotics S.L.*. For a list of authors refer to erle_gitbook/graphs/contributors.

For any questions, concerns, or issues submit them to support [at] erlerobot.com.

Introduction to Networking

Here you can find a presentation of what you should learn through this book, and the definition of the most basic concepts of network.

What is *Networking*?

A *network* consists of multiple machines (computers) that are connected together and share each other all kinds of information. This connection between the network can be developed through waves and signals or wires, depending on which is most convenient for work and the type of information that needs to be shared.

In the *network* multiple machines (host) are connected to the communication sub-net that allows the dialog between them. They can communicate in two basic ways:

- Through channels point to point (PPP)
- Through broadcast channels

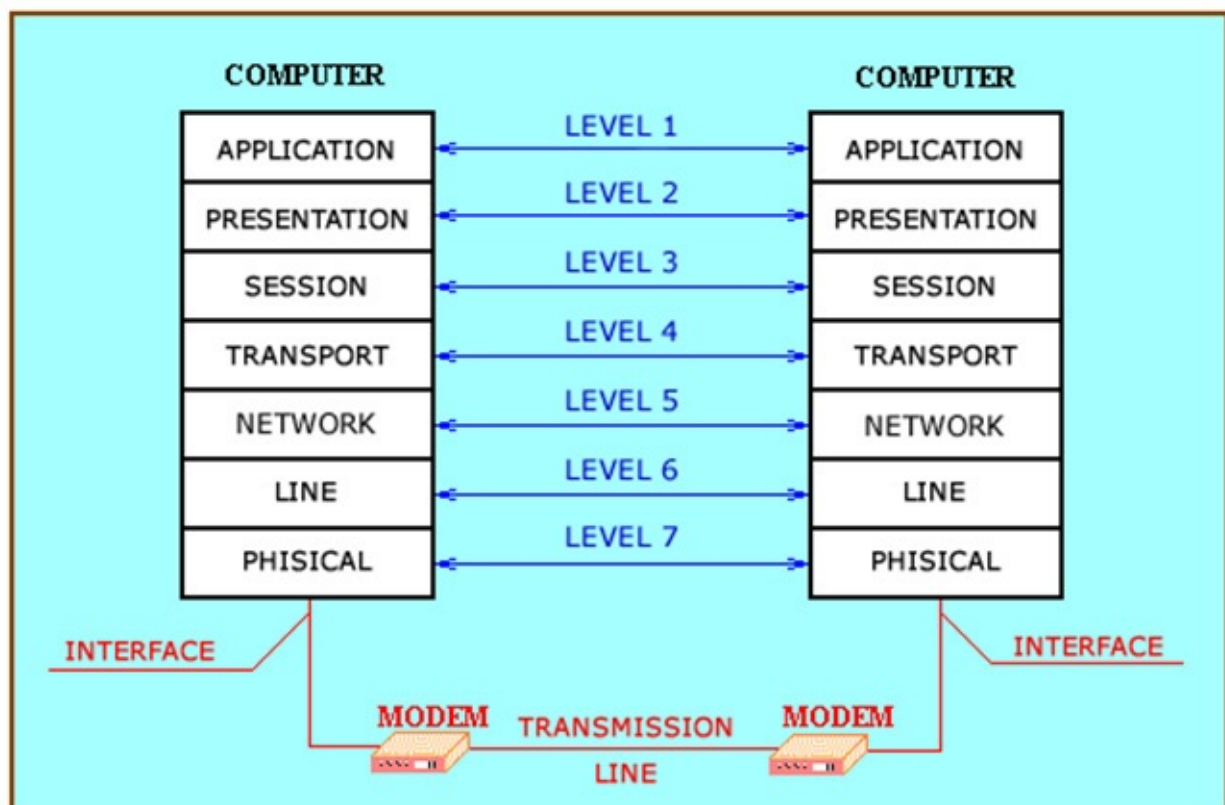
For communicating machines that aren't able to communicate by themselves, **routers** (intermediate machines) are used.

Moreover, the **protocols** are a set of rules (interfaces, algorithms, formats messages...) known by the entities exchanging data through the communications network.

The protocols used by the machines are organised in different **layers or levels**, in such a way that:

- Each layer offers services to a higher level
- Each layer is supported by services offered by a lower level

Each level in a machine "talks with" his twin in another. The rules governing this "conversation" form the protocol of that level.



Here you can find some example of protocols and his corresponding level:

Layer name	Protocols
Application layer	HTTP, DNS, SMTP, POP, ...
Transport layer	TCP, UDP
Network layer	IP, IPv6

Network access layer	PPP, PPPoE, Ethernet
----------------------	----------------------

When we talk about **Network Architecture**, we are talking about the set of levels and protocols of a computers network.

Linux networking

To go into the next chapters, you should have a basic knowledge of Linux/Unix. If not, we recommend you to read first a [Uix introduction tutorial](#).

As you know, Linux itself is just a kernel. Linux can be configured as a networked workstation, a DNS server, a DHCP server, a web server, a mail server, a file and print server, database server, a firewall, a gateway router and many more. Linux is capable to be all that. In this section, you will learn the odds of **Linux as a network operating system or server system**, to be all that mention above.

TCP and packets

Modern networking applications require a sophisticated approach to carrying data from one machine to another. The solution that Unix systems, and subsequently many non-Unix systems, have adopted is known as TCP/IP.

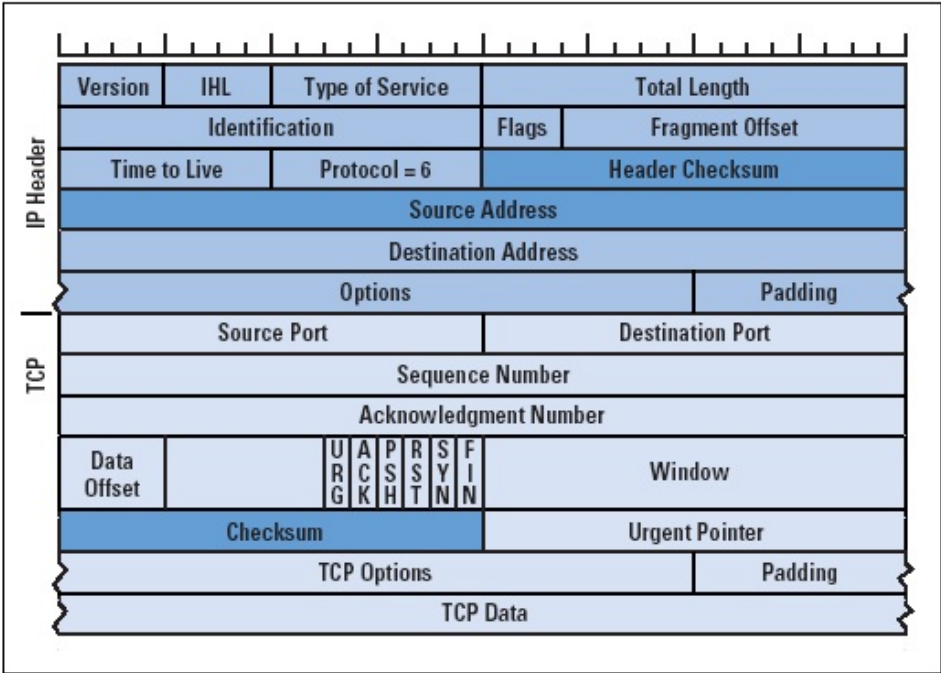
The **Transport Control Protocol (TCP)** and the **Internet Protocol (IP)** are the two most popular ways of communicating on the Internet. A lot of applications, such as your browser and E-mail program, are built on top of this protocol suite. (Remember: A protocol is system of digital rules for data exchange within or between computers).

When talking about TCP/IP networks you will hear the term datagram, which technically has a special meaning but is often used interchangeably with packet. Very simply put, IP provides a solution for sending packets of information from one machine to another, while TCP ensures that the packets are arranged in streams, so that packets from different applications don't get mixed up, and that the packets are sent and received in the correct order.

Here is a description of what a packet actually looks like. The start of each packet says where it's going, where it came from, the type of the packet, and other administrative details. This part is called the 'packet header'. The rest of the packet, containing the actual data being transmitted, is usually called the 'packet body'.

- Any IP packet begins with an 'IP header': at least 20 bytes long.
- If the protocol fields says this is a TCP packet, then a TCP header will immediately follow this IP header: the TCP header is also at least 20 bytes long.

Figure 1: TCP/IP Header Fields Altered by NATs (Outgoing Packet)



If you want to know more about TPC/IP protocols and the packets, you can visit this [link](#).

Basics: First steps in networking

In this chapter you can find the very basics of networking in Linux:

- basic comands
- basic concepts

Basic commands

This is a resume containing some of the most used comands when networking.

`ifconfig` command

This command lets you configure network interface parameters or assingn an address to a network.

`netstat` command

With this command you can list network connections, view routing table and gain information about network interface. Definitely, `netstat` show network status.

Also, `netstat -r` command is widely used to show the routing tables.

`route` command

Linux `route` command manipulates the IP routing table. `route` is a utility used to manually manipulate the network routing.

`ping` command

Send ICMP ECHO_REQUEST packets to network hosts is what Linux `ping` command does.

`telnet` command

`telnet` command allows you to communicate with another host using the Telnet protocol. Probably telnet is one of those terms that most users are familiar with. If `telnet` is invoked without the host argument, it enters command mode, indicated by its prompt (``telnet>`"). In this mode, it accepts and executes the commands.

`ftp` command

This program allows users to transfer files to and from a remote network site using the Internet standard File Transfer Protocol.

`arp` command

The arp utility displays and modifies the Internet-to-Ethernet address translation tables used by the address resolution protocol.

`lsusb` command

Is a utility for displaying information about USB buses in the system and the devices connected to them.

To make use of all the features of this program, you need to have a Linux kernel which supports the `/proc/bus/usb` interface (e.g., Linux kernel 2.3.15 or newer).

`lsmod` command

`lsmod` shows information about all loaded modules. The format is name, size, use count, list of referring modules. The information displayed is identical to that available from `/proc/modules`.

`ip` command

IP is the transport layer protocol used by the Internet protocol family. Options may be set at the IP level when using higher-level protocols that are based on IP (such as TCP and UDP). It may also be accessed through a "raw socket" when developing new protocols, or special-purpose applica-tions. With this command you can display the IP address or the Ip

route.

traceroute command

Displays the route packets take to network host. `traceroute` utilizes the IP protocol 'time to live' field and attempts to elicit an ICMP TIME_EXCEEDED response from each gateway along the path to some host to determine the route one packet follows.

whois command

Specific domain name information can be queried using the `whois` command. This utility looks up records in the databases maintained by several Network Information Centers (NICs) and then displays internet domain name and network number directory service

sudo command

This command allows a permitted user to execute a command as the superuser or another user, as specified in the sudoers file.

If the invoking user is root or if the target user is the same as the invoking user, no password is required. Otherwise, sudo requires that users authenticate themselves with a password by default (NOTE: in the default configuration this is the user's password, not the root password). Once a user has been authenticated, a time stamp is updated and the user may then use sudo without a password for a short period of time (5 minutes unless overridden in sudoers).

Command	Meaning
ifconfig	configure network interface
netstat	show network status
route	manually manipulate the network routing
ping	send ICMP ECHO_REQUEST packets to network hosts
telnet	communicate with another host using the Telnet protocol
ftp	transfer files to and from a remote network site
arp	address resolution display and control
lsusb	displays info. about usb connections
lsmod	information about all loaded modules
ip -- show	display Ip address/route
traceroute	display packet's route
whois	domain name information
sudo	execute a command as another user

Here you can find some examples of the execution of these commands:

```

root@erlerobot:~# ifconfig
lo          Link encap:Local Loopback
            inet addr:127.0.0.1  Mask:255.0.0.0
            inet6 addr: ::1/128 Scope:Host
            UP LOOPBACK RUNNING  MTU:65536  Metric:1
            RX packets:46 errors:0 dropped:0 overruns:0 frame:0
            TX packets:46 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:0
            RX bytes:4160 (4.1 KB)  TX bytes:4160 (4.1 KB)

usb0       Link encap:Ethernet  HWaddr 4e:97:62:fb:16:20
            inet addr:11.0.0.1  Bcast:11.0.0.255  Mask:255.255.255.0
            inet6 addr: fe80::4c97:62ff:fe9b:1620/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
            RX packets:343 errors:0 dropped:0 overruns:0 frame:0
            TX packets:169 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:56354 (56.3 KB)  TX bytes:12135 (12.1 KB)

wlan2     Link encap:Ethernet  HWaddr 80:1f:02:95:cf:86
            inet addr:192.168.1.36  Bcast:192.168.1.255  Mask:255.255.255.0
            inet6 addr: fe80::821f:2ff:fe95:cf86/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
            RX packets:790 errors:0 dropped:3 overruns:0 frame:0
            TX packets:81 errors:0 dropped:1 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:89689 (89.6 KB)  TX bytes:11052 (11.0 KB)

root@erlerobot:~# route
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
default 11.0.0.2 0.0.0.0 UG 100 0 0 usb0
11.0.0.0 * 255.255.255.0 U 0 0 0 usb0
192.168.1.0 * 255.255.255.0 U 0 0 0 wlan2

root@erlerobot:~# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
From 11.0.0.1 icmp_seq=1 Destination Host Unreachable
From 11.0.0.1 icmp_seq=2 Destination Host Unreachable
From 11.0.0.1 icmp_seq=3 Destination Host Unreachable
From 11.0.0.1 icmp_seq=4 Destination Host Unreachable
^C
--- 8.8.8.8 ping statistics ---
4 packets transmitted, 0 received, +4 errors, 100% packet loss, time 3000ms
pipe 4
root@erlerobot:~#

```

```

root@erlerobot:~# netstat -nr
Kernel IP routing table
Destination      Gateway          Genmask          Flags   MSS Window  irtt Iface
0.0.0.0          11.0.0.2        0.0.0.0          UG      0 0      0 usb0
11.0.0.0         0.0.0.0         255.255.255.0    U       0 0      0 usb0
192.168.1.0      0.0.0.0         255.255.255.0    U       0 0      0 wlan2
root@erlerobot:~#
root@erlerobot:~# lsusb
Bus 001 Device 002: ID 7392:7811 Edimax Technology Co., Ltd EW-7811Un 802.11n Wireless Adapter
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
root@erlerobot:~# lsmod
Module                Size  Used by
bnep                   8941  2
rfcomm                 26161  0
bluetooth             155193  10 bnep,rfcomm
rfkill                 16483  2 bluetooth
8192cu                 423094  0
g_ether                23528  0
libcomposite           14601  1 g_ether
root@erlerobot:~# ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN qlen 1000
    link/ether 00:18:38:f7:38:21 brd ff:ff:ff:ff:ff:ff
3: usb0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 4e:97:62:fb:16:20 brd ff:ff:ff:ff:ff:ff
    inet 11.0.0.1/24 brd 11.0.0.255 scope global usb0
    inet6 fe80::4c97:62ff:fe9b:1620/64 scope link
        valid_lft forever preferred_lft forever
4: wlan2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen 1000
    link/ether 80:1f:02:95:cf:86 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.36/24 brd 192.168.1.255 scope global wlan2
    inet6 fe80::821f:2ff:fe95:cf86/64 scope link
        valid_lft forever preferred_lft forever
root@erlerobot:~# 

```

IP address

In this section you will find the elemental concepts related to the IP protocol and IP address.

Basic concepts

One of the first thing you should learn about networking is **IP protocol** and **IP addresses**.

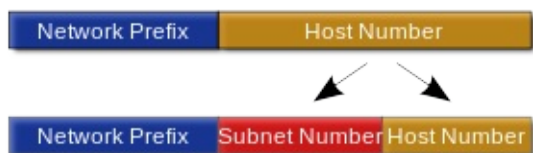
IP protocol is one of the protocols that make a networking possible. Remember that a protocol is a rules that govern communication between two systems. IP is a protocol at *Network Level* that follows a pattern:

- *not oriented to connection*: whenever there is data to send, a message is send to the destination without establishment phases.
- *datagram-based*: each message to a destination routed separately. The unit of data that is send by IP is called IP datagram.
- *untrusted*: messages may be lost, get duplicates, arrive out of order.

When talking about IP protocol, you should know the next subject which is IP address. IP address is a unique address assigned to one network computer. This is important because a network computer must know which computer it is communicating with such as in client talking to a server situation. Therefore an IP address is assigned to each physic machine connected to the Internet.

IP addresses have 32 bytes and are divides into two parts:

- network identifier/network prefix: set of bytes that identify the network the machine is connected to.
- machine identifier/host prefix: unique set of bytes for each machine inside a network.



Example:

IP address: 212.128.4.4

Network identifier	Machine identifier
212.128.4	4

IP address is divided into five classes. Only the three first classes are assigned to machines (class A, class B and class C). In each class the range and the size of the identifiers is different:

The diagram shows the 32-bit structure of IP address classes. Class A uses the first 8 bits for the network and 24 bits for the host. Class B uses the first 16 bits for the network and 16 bits for the host. Class C uses the first 24 bits for the network and 8 bits for the host. Class D is for multicast and Class E is reserved.

Class	Network	Host	Range of host addresses
A	8	24	1.0.0.0 to 255.255.255.255
B	16	16	128.0.0.0 to 191.255.255.255
C	24	8	192.0.0.0 to 223.255.255.255
D	28	4	224.0.0.0 to 239.255.255.255
E	32	0	240.0.0.0 to 255.255.255.255

Next is **subnet mask**. Every time we assign an IP address to a network computer, we must also assign a subnet mask. Ip address comes with different subnet mask depending on it's class. In each IP classes, the subnet mask defines network segment of that system. It says how much of the address is used for the network is defined by the subnet mask. Definitely, the subnet mask is used to specify the networks and the sub-networks used.

Default subnet mask:

A-255.0.0.0

B-255.255.0.0

C-255.255.255.0

Related to networking and net-masks is the concept of the **gateway**: It's an IP address that defines the junction between the local network and the external network. In other words, a gateway is a host that is connected to two or more physical networks simultaneously and is configured to switch packets between them. While the gateway IP address is a part of the local network such as the public Internet. The gateway IP address is normally configured in the routing table for the local system, as defined by the route or `netstat -r` command.

For knowing the IP address of your computer, type:

```
ifconfig
```

```
root@erlerobot:/# ifconfig
lo          Link encap:Local Loopback
            inet addr:127.0.0.1  Mask:255.0.0.0
            inet6 addr: ::1/128  Scope:Host
            UP LOOPBACK RUNNING  MTU:65536  Metric:1
            RX packets:36 errors:0 dropped:0 overruns:0 frame:0
            TX packets:36 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:0
            RX bytes:3074 (3.0 KB)  TX bytes:3074 (3.0 KB)

usb0        Link encap:Ethernet  HWaddr 9a:84:d7:d4:82:47
            inet addr:11.0.0.1  Bcast:11.0.0.255  Mask:255.255.255.0
            inet6 addr: fe80::9884:d7ff:fed4:8247/64  Scope:Link
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
            RX packets:234 errors:0 dropped:0 overruns:0 frame:0
            TX packets:140 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:40190 (40.1 KB)  TX bytes:10750 (10.7 KB)

wlan2       Link encap:Ethernet  HWaddr 80:1f:02:95:cf:86
            inet addr:192.168.1.36  Bcast:192.168.1.255  Mask:255.255.255.0
            inet6 addr: fe80::821f:2ff:fe95:cf86/64  Scope:Link
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
            RX packets:508 errors:0 dropped:3 overruns:0 frame:0
            TX packets:65 errors:0 dropped:1 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:58935 (58.9 KB)  TX bytes:10167 (10.1 KB)

root@erlerobot:/#
```

In `inet addr` you can find the IP address.

As we have seen in basic commands, there is also a command that does this for you.

```
ip addr show
```

IPv4 vs IPv6

As we have seen, at first glance, an IP address is like a phone number or a street address. When you connect to the Internet, your device (computer, cell phone, tablet) is assigned with an IP address, as well as each site you visit has an IP address.

The addressing system that we have used since the Internet was born is called IPv4, and the new addressing scheme is called IPv6.

It is important to define the difference between these two addressing systems.

IPv4 address

To understand that the IPv4 address space is limited to 4.3 billion addresses, we can decompose an IPv4 address. An IPv4 address is a 32-bit number consisting of four octets (8-bit numbers) in a decimal notation, separated by periods. The bit can be either a 1 and a 0 (two possibilities), therefore the decimal notation would be 2^8 raised to the 4th power of various options (256 of them, to be exact). Since we start counting from 0, the possible values of a byte in an IP address range from 0 to 255.

Examples of IPv4 addresses: 192.168.0.1, 66.228.118.51, 173.194.33.16

If an IPv4 address is made up of four sections with 256 possibilities in each section, to find the total number of IPv4 addresses, you should just multiply $256 \times 256 \times 256 \times 256$ to find results 4,294,967,296 addresses. To put it another way, we have 32 bits then 2 to the 32nd power will give the same result obtained.

IPv6 address

IPv6 was designed to support the growth of Internet in the future generations. As mentioned earlier, each device that connects to the Internet using an IP address that needs to be unique. Other existing or developing solutions consider the "share" the same IP address between different devices, which leads to complicated networks become fragile and at the same time are difficult to analyze to correct problems.

The minimum allocation unit is IPv6, CIDR notation using a / 64. This implies that each device connected to the Internet using IPv6 has available 2 to the 64th power (18,446,744,073,709,551,616) IP addresses.

IPv6 addresses are 128 bits based. Using the same math above, we have 2 to the 128th power to find the total of total IPv6 address, same as mentioned above. Since space in IPv6 is much more extensive than the IPv4 would be very difficult to define space with decimal notation ... would have 2 raised to the 32nd power in each section.

To allow the use of such a large number of IPv6 addresses easier, IPv6 is composed of eight 16-bit sections, separated by a colon (:). Since each section is 16 bits, there are 2 raised to the 16 variations (which are 65,536 different possibilities). Using decimal numbers 0 through 65,535, would have represented a rather long address, and to facilitate it is that IPv6 addresses are expressed in hexadecimal notation (16 different characters 0-9 and a-f).

Example of an IPv6 address: 2607: f0d0: 4545: 3: 200: f8ff: FE21: 67CF

which remains a long term but is more manageable to do with decimal alternatives.

Comparison between IPv4 and IPv6



	IPv4	IPv6
Standard since	1974	1998
Developed by	IETF	IETF
Length in bits	32	128
Amount of addresses	$2^{32} = 4,294,967,296$	$2^{128} = 340,282,366,920,938,463,463,374,607,431,768,211,456$
Address format	Dotted decimal 192.168.100.1	Hexadecimal Notation: 2001:0DB8:0234:AB00: 0123:4567:8901:ABCD
Dynamic addressing	DHCP	SLAAC / DHCPv6
IPSec	Optional	Mandatory
Header length	Variable	Fixed
Minimal packet size	576 bytes (fragmented)	1280 bytes
Header checksum	Yes	No
Header options	Yes	No (extensions)
Flow	No	Packet flow label

IPv4 and IPv6 are very similar in terms of functionality (but not in terms of mechanisms)

IPv6:Security::nl

Assigning an IP address

Computers may be assigned a static IP address or assigned one dynamically. Typically a server will require a static IP while a workstation will use DHCP (dynamic IP assignment). The Linux server requires a static IP so that those who wish to use its resources can find the system. It is more easily found if the IP address does not change and is static. This is not important for the Linux client workstation and thus it is easier to use an automated Dynamic Host Configuration Protocol (DHCP) for IP address assignment.

The syntax of the command used to assign a IP address is:

```
ifconfig interface [atype] options | address ...
```

Static IP address assignment example:

```
ifconfig eth0 192.168.1.5 netmask 255.255.255.0 up
```

The `ifconfig` command does NOT store this information permanently. Upon reboot this information is lost.

After setting up a Linux IP address with `ifconfig` command, use Linux `route` command to set up static routes to specific hosts or networks. Linux route command is a tool where you can setup where to send which data. Basically you can use route command to show and manipulate routing table. This is an example of how to use `route` command in Linux. Issue route command without any option will print current routing table.

To add a new IP route via eth0 (first Ethernet card), issue route add command in this format:

```
route add -net <ip> netmask <ip> dev eth0
```

```
root@erlerobot:~# route add -net 192.168.1.0 netmask 255.255.255.0 dev eth0
```

You can also add a gateway of the network:

```
route add -net <ip> gw <ip> netmask <ip> dev eth0
```

```
root@erlerobot:~# route add -net 192.168.1.0 gw 192.168.1.1 netmask 255.255.255.0 dev eth0
```

Now, check the routing table again:

```
root@erlerobot:~# route
Kernel IP routing table
Destination    Gateway         Genmask         Flags Metric Ref    Use Iface
default        11.0.0.2       0.0.0.0         UG    100    0      0  usb0
11.0.0.0       *              255.255.255.0   U      0      0      0  usb0
192.168.1.0    192.168.1.1    255.255.255.0   UG      0      0      0  eth0
192.168.1.0    *              255.255.255.0   U      0      0      0  eth0
192.168.1.0    *              255.255.255.0   U      0      0      0  wlan2
192.168.1.0    *              255.255.255.0   U      0      0      0  eth0
root@erlerobot:~#
```

We can see all configurations we made is in the result above.

IP alias

New kernels support a feature that can completely replace the dummy interface and serve other useful functions. **IP Alias allows you to configure multiple IP addresses onto a physical device.** In the simplest case, you could replicate the function of the dummy interface by configuring the host address (or the host ID portion of an IP address), as an alias onto the [loopback interface](#), represented as *lo* and completely avoid using the dummy interface. In more complex uses, you could configure your host to look like many different hosts, each with its own IP address. This configuration is sometimes called *Virtual Hosting*.

To configure an alias for an interface, you must first ensure that your kernel has been compiled with support for IP Alias (check that you have a `/proc/net/ip_alias` file; if not, you will have to recompile your kernel). Configuration of an IP alias is virtually identical to configuring a real network device; you use a special name to indicate it's an alias that you want. For example:

```
ifconfig lo:0 172.16.1.1
```

This command would produce an alias for the loopback interface with the address 172.16.1.1. IP aliases are referred to by appending `:n` to the actual network device, in which `n` is an integer. In our example, the network device we are creating the alias on is `lo`, and we are creating an alias numbered zero for it. This way, a single physical device may support a number of aliases. Each alias may be treated as though it is a separate device, and as far as the kernel IP software is concerned, it will be; however, it will be sharing its hardware with another interface.

Ethernet

The most common type of LAN **hardware** is known as Ethernet. In its simplest form, it consists of a single cable with hosts attached to it through connectors, taps, or transceivers. Simple Ethernets are relatively inexpensive to install, which together with a net transfer rate of 10, 100, or even 1,000 Megabits per second, accounts for much of its popularity.

Ethernets come in three flavors: thick, thin, and twisted pair.

Ethernet works like a bus system, where a host may send packets (or frames) of up to 1,500 bytes to another host on the same Ethernet. A host is addressed by a six-byte address hardcoded into the firmware of its Ethernet network interface card (NIC). These addresses are usually written as a sequence of two-digit hex numbers separated by colons, as in aa:bb:cc:dd:ee:ff.

A frame sent by one station is seen by all attached stations, but only the destination host actually picks it up and processes it. If two stations try to send at the same time, a collision occurs. Collisions on an Ethernet are detected very quickly by the electronics of the interface cards and are resolved by the two stations aborting the send, each waiting a random interval and re-attempting the transmission.

ARP protocol

The ARP protocol allows to find out the Ethernet address of a machine knowing its IP address.

The `arp` utility displays and modifies the Internet-to-Ethernet address translation tables used by the address resolution protocol (arp(4)). With no flags, the program displays the current ARP entry for hostname. The host may be specified by name or by number, using Internet dot notation.

One of the most useful commands is:

```
arp -a
```

With this the program displays or deletes all of the current ARP entries.

```
root@erlerobot:~# arp -a
? (11.0.0.2) at <incomplete> on usb0
root@erlerobot:~#
```

Linux Network Devices

This chapter introduce the use and access to hardware drivers in Linux.

Linux Interfaces

The Linux kernel supports a number of hardware drivers for various types of equipment. This section gives a short overview of the driver families available and the interface names they use. There is a number of standard names for interfaces in Linux, which are listed here. Most drivers support more than one interface, in which case the interfaces are numbered, as in `eth0` and `eth1` :

Lo

This is the local loopback interface. It is used for testing purposes, as well as a couple of network applications. It works like a closed circuit in that any datagram written to it will immediately be returned to the host's networking layer. In other words, the loopback network device is an interface network always represents the virtual device itself regardless of the IP address that you may have assigned. Your IP is 127.0.0.1. There's always one loopback device present in the kernel, and there's little sense in having more.

eth0, eth1, &

These are the Ethernet card interfaces. They are used for most Ethernet cards, including many of the parallel port Ethernet cards.

tr0, tr1, &

These are the Token Ring card interfaces. They are used for most Token Ring cards, including non-IBM manufactured cards.

sl0, sl1, &

These are the SLIP interfaces. SLIP interfaces are associated with serial lines in the order in which they are allocated for SLIP.

ppp0, ppp1, &

These are the PPP interfaces. Just like SLIP interfaces, a PPP interface is associated with a serial line once it is converted to PPP mode.

plip0, plip1, &

These are the PLIP interfaces. PLIP transports IP datagrams over parallel lines. The interfaces are allocated by the PLIP driver at system boot time and are mapped onto parallel ports. In the 2.0.x kernels there is a direct relationship between the device name and the I/O port of the parallel port, but in later kernels the device names are allocated sequentially, just as for SLIP and PPP devices.

ax0, ax1, &

These are the AX.25 interfaces. AX.25 is the primary protocol used by amateur radio operators. AX.25 interfaces are allocated and mapped in a similar fashion to SLIP devices. There are many other types of interfaces available for other network drivers. We've listed only the most common ones.

Accessing Serial Devices

Like all devices in a Unix system, serial ports are accessed through device special files, located in the `/dev` directory. Type this to list the content of dev directory:

```
cd /dev
ls
```

There are two varieties of device files related to serial drivers, and there is one device file of each type for each port. The device will behave slightly differently, depending on which of its device files we open. We'll cover the differences because it will help you understand some of the configurations and advice that you might see relating to serial devices, but in practice you need to use only one of these.

The most important of the two classes of serial device has a major number of 4, and its device special files are named **ttyS0** , **ttyS1** , etc. The second variety has a major number of 5, and was designed for use when dialing out (calling out) through a port; its device special files are called **cua0** , **cua1** , etc. In the Unix world, counting generally starts at zero, while laypeople tend to start at one. This creates a small amount of confusion for people because COM1: is represented by `/dev/ttyS0` , COM2: by `/dev/ttyS1` , etc.

The cua , or "callout," devices were created to solve the problem of avoiding conflicts on serial devices for modems that have to support both incoming and outgoing connections. Unfortunately, they've created their own problems and are now likely to be discontinued.

When dealing with devices `stty` command is very useful. The `stty` utility sets or reports on terminal characteristics for the device that is its standard input. If no options or operands are specified, it reports the settings of a subset of characteristics as well as additional ones if they differ from their default values. Otherwise it modifies the terminal state according to the specified arguments. Some combinations of arguments are mutually exclusive on some terminal types.

`stty` command can be used to display the terminal configuration parameters of a tty device. To display all of the active settings on a tty device, use:

```
stty -a -F /dev/ttyS1
```

```
loop1      ram3      tty19      tty45  ubi_ctrl  zero
root@erlerobot:/dev# stty -a -F /dev/tty19
speed 38400 baud; rows 30; columns 80; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>;
eol2 = <undef>; swch = <undef>; start = ^Q; stop = ^S; susp = ^Z; rprnt = ^R;
werase = ^W; lnext = ^V; flush = ^O; min = 1; time = 0;
-parenb -parodd cs8 hupcl -cstopb cread -clocal -crtscts
-ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl ixon -ixoff
-iuclc -ixany -imaxbel iutf8
opost -olcuc -ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0 bs0 vt0 ff0
isig icanon iexten echo echoe echok -echonl -noflsh -xcase -tostop -echopr
echoctl echoke
root@erlerobot:/dev#
```

DNS

In this chapter you will understand what a DNS is and how it works.

What is *DNS*?

Domain Name System or DNS is a hierarchical naming system for computers, services, or any resource connected to the Internet or a private network. This system associates various information with domain names assigned to each of the participants. Its most important function is to translate (resolve) intelligible names for people in binary identifiers associated with the devices connected to the network, all this in order to locating and addressing these devices worldwide.

The DNS server uses a distributed, hierarchical database that stores information associated with domain names in Internet networks. Although as the DNS database is able to associate different types of information on each name, the most common uses are mapping domain names to IP addresses and the location of the mail servers in each domain.

Hierarchy of domains

- Root domain or domain `.`:
 - managed by ICANN (Internet Corporation for Assigned Names and Numbers).
 - The servers are root nameservers.
- First level domains (TLDs, Top Level Domains):

- Generic tradicional domains:

com, edu, gov, mil, org, net, int

- Generic modern domains:

aero, biz, coop, info, museum, name, pro, jobs,
mobi, tel, travel, cat, asia

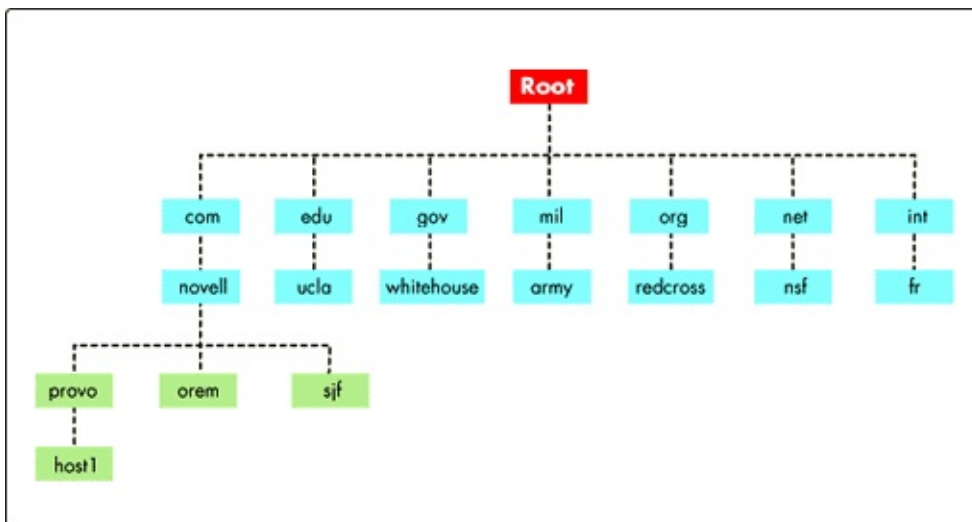
- Domain for the DNS infrastructure:

arpa

- Domains by ISO country code:

uk, mx, ar, de, es, jp. . .

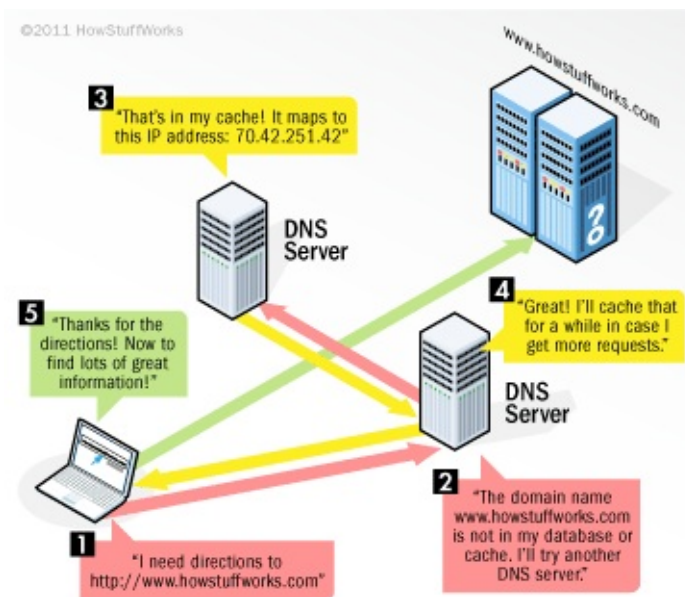
- Second, third ... level domains.



How DNS works

DNS servers accept requests from programs and other name servers to convert domain names into IP addresses. When a request comes in, the DNS server can do one of four things with it: It can answer the request with an IP address because it already knows the IP address for the requested domain. It can contact another DNS server and try to find the IP address for the name requested. It may have to do this multiple times. It can say, "I don't know the IP address for the domain you requested, but here's the IP address for a DNS server that knows more than I do." It can return an error message because the requested domain name is invalid or does not exist.

How a DNS works in an image:



Note: In networking, when talking about *cach  * we refer to data temporarily stored in anticipation of using them again and thus save computation time, search or download from the network.

As we have seen, DNS organizes hostnames in a domain hierarchy. A domain is a collection of sites that are related in some sense because they form a proper network (e.g., all machines on a campus, or all hosts on BITNET), because they all belong to a certain organization (e.g., the U.S. government), or because they're simply geographically close. For instance, universities are commonly grouped in the `edu` domain, with each university or college using a separate subdomain, below which their hosts are subsumed.

Groucho Marx University have the `groucho.edu` domain, while the LAN of the Mathematics department is assigned `maths.groucho.edu`. Hosts on the departmental network would have this domain name tacked onto their hostname, so `erdos` would be known as `erdos.maths.groucho.edu`. This is called the fully qualified domain name (FQDN), which uniquely identifies this host worldwide.

Organizing the namespace in a hierarchy of domain names nicely solves the problem of name uniqueness; with DNS, a hostname has to be unique only within its domain to give it a name different from all other hosts worldwide. Furthermore, fully qualified names are easy to remember. Taken by themselves, these are already very good reasons to split up a large domain into several subdomains.

DNS does even more for you than this. It also allows you to delegate authority over a subdomain to its administrators.

For example, the maintainers at the Groucho Computing Center might create a subdomain for each department; we already encountered the math and physics subdomains above. When they find the network at the Physics department too large and chaotic to manage from outside (after all, physicists are known to be an unruly bunch of people), they may simply pass control of the `physics.groucho.edu` domain to the administrators of this network. These administrators are free to use whatever hostnames they like and assign them IP addresses from their network in whatever fashion they desire, without outside interference.

To this end, the namespace is split up into zones, each rooted at a domain. Note the subtle difference between a *zone* and a *domain*:

The domain groucho.edu encompasses all hosts at Groucho Marx University, while the zone groucho.edu includes only the hosts that are managed by the Computing Center directly; those at the Mathematics department, for example. The hosts at the Physics department belong to a different zone, namely physics.groucho.edu.

Host

In this section you find the basic concepts of Host.

Host basics

A network host is a computer or other device connected to a computer network. A network host may offer information resources, services, and applications to users or other nodes on the network. A network host is a network node that is assigned a network layer host address.

The `/etc/hostname` file contains the computer name adopted when the operating system starts. If you use the command `hostname` you can see the name of your host computer.

```
root@erlerobot:~# hostname
erlerobot
root@erlerobot:~#
```

The names of the machines can be with complete or qualification relating to the local domain.

The `/etc/host.conf` tells the older Linux standard library resolver functions which services to use, and in what order. In other words, `host.conf` indicates the order of the sources will use the address of operating system to obtain DNS resolutions requiring computer applications. There are two options:

- Look for them inside: `/etc/hosts`
- Search them outside: `/etc/resolv.conf`

Options in `host.conf` must appear on separate lines. Fields may be separated by white space (spaces or tabs). A hash sign (`#`) introduces a comment that extends to the next newline. For example typing the option `order` determines the order in which the resolving services are tried

The `/etc/hosts` is a simple mechanism of name resolution. Contains one record per line, consisting of a IP address, hostname and optionally a list of aliases for the hostname. The fields are separated by tabs or spaces and field with the `@` IP should begin in the first column. The command `host` is a simple utility for performing DNS lookups. It is normally used to convert names to IP addresses and vice versa. When no arguments or options are given, `host` prints a short summary of its command line arguments and options.

The file `/etc/resolv.conf` contains the IP addresses of machines that can provide DNS services to our host. The statement points to DNS nameserver have the function that the host can use them to make their decisions.

Moreover, just as with IP addresses, sometimes You can also use a symbolic name for network numbers. For this purpose, the file `/etc/hosts` has a companion called `/etc/networks`, that associates names network with corresponding numbers and vice versa.

Network Configuration

Here you can find some ways of changing the configuration of your network.

Network configuration commands

ip

IP is the transport layer protocol used by the Internet protocol family. Options may be set at the IP level when using higher-level protocols that are based on IP (such as TCP and UDP).

```
ip addr show
```

Displays the IP address

```
ip route show
```

Displays the IP route

```
root@erlerobot:~# ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN qlen 1000
    link/ether 00:18:30:f7:38:21 brd ff:ff:ff:ff:ff:ff
3: usb0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state
    link/ether ba:9c:4f:2c:a5:42 brd ff:ff:ff:ff:ff:ff
    inet 11.0.0.1/24 brd 11.0.0.255 scope global usb0
    inet6 fe80::b89c:4fff:fe2c:a542/64 scope link
        valid_lft forever preferred_lft forever
4: wlan2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen
    link/ether 80:1f:02:95:cf:86 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.36/24 brd 192.168.1.255 scope global wlan2
    inet6 fe80::821f:2ff:fe95:cf86/64 scope link
        valid_lft forever preferred_lft forever
root@erlerobot:~#
root@erlerobot:~#
root@erlerobot:~# ip route show
default via 11.0.0.2 dev usb0 metric 100
11.0.0.0/24 dev usb0 proto kernel scope link src 11.0.0.1
192.168.1.0/24 dev wlan2 proto kernel scope link src 192.168.1.36
root@erlerobot:~#
```

ifconfig

There are many more parameters to `ifconfig` than we have described so far. Its normal invocation is this:

```
ifconfig interface [address [parameters]]
```

interface is the interface name, and address is the IP address to be assigned to the interface. This may be either an IP address in dotted quad notation or a name that `ifconfig` will look up in `/etc/hosts`.

```
ifconfig
```

As we have seen this command displays the configuration information for all network interfaces of the system.

```

root@erlerobot:/proc/net# ifconfig
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:36 errors:0 dropped:0 overruns:0 frame:0
          TX packets:36 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:3074 (3.0 KB)  TX bytes:3074 (3.0 KB)

usb0      Link encap:Ethernet  HWaddr 9a:84:d7:d4:82:47
          inet addr:11.0.0.1  Bcast:11.0.0.255  Mask:255.255.255.0
          inet6 addr: fe80::9884:d7ff:fed4:8247/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:108 errors:0 dropped:0 overruns:0 frame:0
          TX packets:140 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:20982 (20.9 KB)  TX bytes:10750 (10.7 KB)

wlan2     Link encap:Ethernet  HWaddr 80:1f:02:95:cf:86
          inet addr:192.168.1.36  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::821f:2ff:fe95:cf86/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:188 errors:0 dropped:3 overruns:0 frame:0
          TX packets:45 errors:0 dropped:1 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:24881 (24.8 KB)  TX bytes:8727 (8.7 KB)

```

```
ifconfig eth0 down
```

Disables the network interface eth0.

```
ifconfig eth0
```

Check the status of the eth0 interface.

```

root@erlerobot:/proc# ifconfig wlan2
wlan2     Link encap:Ethernet  HWaddr 80:1f:02:95:cf:86
          inet addr:192.168.1.36  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::821f:2ff:fe95:cf86/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:280 errors:0 dropped:3 overruns:0 frame:0
          TX packets:49 errors:0 dropped:1 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:33679 (33.6 KB)  TX bytes:9015 (9.0 KB)

```

```
ifconfig interfase dirección-ip
```

Interfacing basic configuration. Assign IP address and activates it. The other parameters take values assigned by default. For example, the subnet mask takes the value corresponding to the type of network the Ip address belong to. Like that we have 255.255.0.0 for a class B address.

```
ifconfig eth0 100.200.26.1 netmask 255.255.255.0 broadcast 100.200.26.255
```

Sets the network interface from scratch.

In this [link](#) you can find other option when working with ifconfig.

route

The syntax is the following one:

```
route [-n] command [-net|-host] destination
```

In the syntax above, destination is the destination host or network, gateway is the next-hop intermediary via which packets should be routed. Routes to a particular host may be distinguished from those to a network by interpreting the Internet address specified as the destination argument. The optional modifiers -net and -host force the destination to be interpreted as a network or a host, respectively.

The `route` command itself displays the IP routing table of the system. Also allows to add or remove an entry in the routing table. This command allows you to set static routes when network routing.

```
root@erlerobot:~# route
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
default 11.0.0.2 0.0.0.0 UG 100 0 0 usb0
11.0.0.0 * 255.255.255.0 U 0 0 0 usb0
192.168.1.0 * 255.255.255.0 U 0 0 0 wlan2
```

```
option -net
```

Specifies that the specified target is a network.

```
option -host
```

Specifies that the specified target is a computer.

```
route -n
```

Prints the routing table full core if run without arguments (the -n option makes use dotted quad instead of hostnames).

```
route add -net 192.168.1.0 gw 192.168.1.2
```

Add a route to a network through a gateway.

```
route add - host 192.168.1.250 dev eth0
```

Adds a route to an particular machine via a local interfacing.

```
route add domain
```

Adds a route to the routing table using the names defined in the file `/etc /networks`. Thus prevents write-net indicator because route knows that this is network

```
route add default gw host
```

Routing through the default gateway.

netstat

Depending on the selected option `netstat` command shows the networks interfaces, the PID associated to each interface...

```
root@erlerobot:/# netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags               Type                   State                  I-Node  Path
unix  2      [ ]               DGRAM                  4153                   /var/run/wpa_supplicant/wlan2
unix  8      [ ]               DGRAM                  3722                   /dev/log
unix  3      [ ]               STREAM                 CONNECTED              4732
unix  3      [ ]               STREAM                 CONNECTED              3882
unix  3      [ ]               STREAM                 CONNECTED              4922                   /var/run/dbus/system_bus_socket
unix  3      [ ]               STREAM                 CONNECTED              4725
unix  3      [ ]               STREAM                 CONNECTED              3171                   @/com/ubuntu/upstart
unix  3      [ ]               STREAM                 CONNECTED              4909                   /var/run/dbus/system_bus_socket
unix  3      [ ]               STREAM                 CONNECTED              3691
unix  3      [ ]               STREAM                 CONNECTED              3782                   /var/run/dbus/system_bus_socket
unix  3      [ ]               STREAM                 CONNECTED              4731
unix  3      [ ]               STREAM                 CONNECTED              4944
unix  2      [ ]               DGRAM                  4128
unix  2      [ ]               DGRAM                  4157
```

```
netstat -c
```

Renews the information continually till you push `^c` (ctrl + c).

```
netstat -i
```

Displays a list with all the network interfaces.

```
netstat -p
```

Shows a list with all the PID.

```
netstat -r
```

Displays the information of the routing table.

```
netstat -t
```

Shows the active connections of the TCP ports Muestra las conexiones activas a puertos TCP.

```
netstat -u
```

Displays active connections to UDP ports. If included "a" will also display the ports that are waiting for a connection (listening).

```
netstat -nr
```

Displays routing information. The -n shows network addresses as numbers and the -r shows the routing tables.

ping

The sintaxis is the following one:

```
ping hostname
```

The ping command sends an ICMP echo request protocol to the specified hostname and shows the time to receive confirmation echo. The default sends messages indefinitely until the order is canceled by `^C`.

```
ping -c n hostname
```

n messages will be sent. Al finalizar los mensajes especificados se muestra la estadística de los resultados. After sending the specified messages, statistical results are shown.

```
root@erlerobot:~# ping -c 6 11.0.0.2
PING 11.0.0.2 (11.0.0.2) 56(84) bytes of data.
64 bytes from 11.0.0.2: icmp_req=1 ttl=64 time=1.00 ms
64 bytes from 11.0.0.2: icmp_req=2 ttl=64 time=0.554 ms
64 bytes from 11.0.0.2: icmp_req=3 ttl=64 time=0.531 ms
64 bytes from 11.0.0.2: icmp_req=4 ttl=64 time=0.552 ms
64 bytes from 11.0.0.2: icmp_req=5 ttl=64 time=0.654 ms
64 bytes from 11.0.0.2: icmp_req=6 ttl=64 time=0.689 ms

--- 11.0.0.2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5007ms
rtt min/avg/max/mdev = 0.531/0.664/1.005/0.163 ms
root@erlerobot:~#
```

traceroute

The Internet is a large and complex aggregation of network hardware, connected together by gateways. Tracking the route one's packets follow (or finding the miscreant gateway that's discarding your packets) can be difficult. traceroute utilizes the IP protocol 'time to live' field and attempts to elicit an ICMP TIME_EXCEEDED response from each gateway along the path to some host.

For example, you can check the route the packets follow to a network host using this command:

```
/usr/sbin/traceroute www.eunet.be
```

Network configuration files

In Linux we are able to configure the network by editing the configuration files. We can find these files in the etc directory and edit them using, for example vi text editor. (If they don't exist, you can create them and they will update their content automatically).

```
root@erlerobot:/etc# ls
ConsoleKit          fonts               lsb-base            rc4.d
X11                 foomatic           lsb-base-logging.sh rc5.d
adduser.conf        fstab              lsb-release         rc6.d
adjtime             fstab.d            magic               rcS.d
alternatives        gai.conf           magic.mime          resolv.conf
apache2             gdb                mailcap             resolvconf
apparmor            ghostscript        mailcap.order       rmt
apparmor.d          group              mercurial           ros
appport             group-             mime.types          rpc
apt                 gshadow            mke2fs.conf         rsyslog.conf
avahi                gshadow-           modprobe.d          rsyslog.d
avrdude.conf        gtk-2.0            modules             samba
bash.bashrc         host.conf          modules-load.d      sane.d
bash_completion.d   hostapd            mtd                 securetty
bindresvport.blacklist hostname           mtabs               security
blkid.conf          hosts              mysql               sensors.d
blkid.tab           hosts.allow        nanorc              sensors3.conf
bluetooth           hosts.deny          network             services
ca-certificates     ifplugd            networks            sgml
ca-certificates.conf init               newt                shadow
calendar            init.d             nginx               shadow-
colord.conf          initramfs-tools    nsswitch.conf       shells
console-setup       inputrc            os-release          skel
cron.d              insserv            pam.conf            snap
cron.daily          insserv.conf       pam.d              ssh
cron.hourly         iproute2           passwd              ssl
cron.monthly        issue              papersize            subversion
cron.weekly          issue.dpkg-dist    passwd-             sudoers
crontab             issue.net          perl                sudoers.d
cups                kbd                php5                 sysctl.conf
dbus-1              kernel             pnm2ppa.conf        sysctl.d
debconf.conf        ld.so.cache        polkit-1             systemd
debian_version     ld.so.conf         ppp                  terminfo
default             ld.so.conf.d       profile              timezone
deluser.conf        legal              protocols            ucf.conf
depmod.d            libl-3             pulse                udev
dhcp                libpaper.d         python               ufw
dhcp3               locale.alias       python2.7            update-motd.d
dkms                localtime          rc.local             vim
dnsmasq.conf        logcheck           rc0.d                vtrgb
dnsmasq.conf.orig   login.defs         rc1.d                wgetrc
dnsmasq.d            logrotate.conf     rc2.d                wpa_supplicant
dpkg                logrotate.d        rc3.d                xsl
emacs               environment
environment
```

/etc/network/interfaces

Contains information required to configure the interfaces of host network at boot time. It also allows you to set static routes to other networks.

What you find if you go to networks directory and open interfaces, by typing:

```
vi interfaces
```

Is text editable file like this:


```

1 # interfaces(5) file used by ifup(8) and ifdown(8)
2
3 # loopback network interface
4 auto lo
5 iface lo inet loopback
6
7 # primary network interface
8 #auto eth0
9 #iface eth0 inet dhcp
10 #hwaddress ether DE:AD:BE:EF:CA:FE
11
12 # wireless network interface
13 #auto wlan0
14 #iface wlan0 inet dhcp
15 # wpa-ssid "my_wifi_name"
16 # wpa-key "my_wifi_name"
17
18 auto wlan2
19 iface wlan2 inet dhcp
20 wpa-driver wext
21 wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
22
23 ## wlan3 for access point
"interfaces" [readonly] 37L, 697C

```

1,1

Top

/etc/hostname

Here is specified the name of the machine.

```

1 erlerobot

```

/etc/hosts

As we have seen in [5.1 Host Basics](#), in the file /etc/hosts specify the IP along with the name of each machine you want to access by name. The /etc/hosts file always contains the localhost IP address, 127.0.0.1, which is used for interprocess communication(never remove this line). Sometimes contains addresses of additional hosts, which can be contacted without using an external naming service such as DNS (the Domain Name Server).

```

1 127.0.0.1 localhost
2 127.0.1.1 ubuntu-armhf
3
4 10.0.0.1 erle
5 11.0.0.1 erlerobot
6

```

/etc/resolv.conf

In the file / etc / resolv.conf specify what servers use to resolve domain names.

```

1 nameserver 80.58.61.250
2 nameserver 80.58.61.254

```

/etc/nsswitch.conf

Defines the order in which to contact different name services.

```
1 # /etc/nsswitch.conf
2 #
3 # Example configuration of GNU Name Service Switch functionality.
4 # If you have the 'glibc-doc-reference' and 'info' packages installed, try:
5 # `info libc "Name Service Switch"' for information about this file.
6
7 passwd:         compat
8 group:          compat
9 shadow:         compat
10
11 hosts:          files mdns4_minimal [NOTFOUND=return] dns mdns4
12 networks:       files
13
14 protocols:      db files
15 services:       db files
16 ethers:         db files
17 rpc:            db files
18
19 netgroup:       nis
```

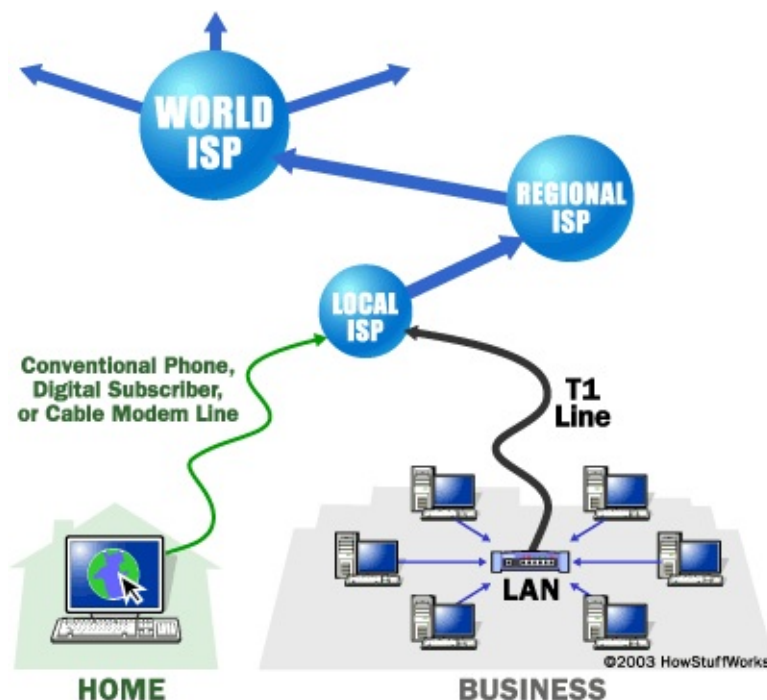
Internet/Intranet applications

The Linux system is a great platform for offering networking services. In this section, we will try to give an overview of most common network servers, applications and how Internet works.

Internet basics

So what is "the Internet"? The Internet is a gigantic collection of millions of computers, all linked together on a computer network.

One of the greatest things about the Internet is that nobody really owns it. It is a global collection of networks, both big and small. These networks connect together in many different ways to form the single entity that we know as the Internet. In fact, the very name comes from this idea of interconnected networks.



The network allows all of the computers to communicate with one another. A home computer may be linked to the Internet using a phone-line modem, DSL or cable modem that talks to an **Internet service provider (ISP)**. A computer in a business or university will usually have a **network interface card (NIC)** that directly connects it to a **local area network (LAN)** inside the business. The business can then connect its LAN to an ISP using a high-speed phone line like a **T1 line**. A T1 line can handle approximately 1.5 million bits per second, while a normal phone line using a modem can typically handle 30,000 to 50,000 bits per second. ISPs then connect to larger ISPs, and the largest ISPs maintain fiber-optic "backbones" for an entire nation or region. **Backbones** are typically fiber optic trunk lines. The trunk line has multiple fiber optic cables combined together to increase the capacity. Backbones around the world are connected through fiber-optic lines, undersea cables or satellite links (see An Atlas of Cyberspaces for some interesting backbone maps). In this way, every computer on the Internet is connected to every other computer on the Internet.

The Function of an Internet Router

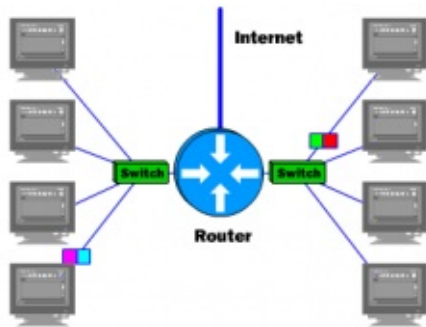
All the Internet networks rely on NAPs, backbones and routers to talk to each other.

A **router** is a device that forwards data packets between computer networks. The routers determine where to send information from one computer to another. Routers are specialized computers that send your messages and those of every other Internet user speeding to their destinations along thousands of pathways. A router has two separate, but related, jobs:

- It ensures that information doesn't go where it's not needed. This is crucial for keeping large volumes of data from clogging the connections of "innocent bystanders."
- It makes sure that information does make it to the intended destination.

In performing these two jobs, a router is extremely useful in dealing with two separate computer networks. It joins the two networks, passing information from one to the other. It also protects the networks from one another, preventing the traffic on one from unnecessarily spilling over to the other. Regardless of how many networks are attached, the basic operation and

function of the router remains the same. Since the Internet is one huge network made up of tens of thousands of smaller networks, its use of routers is an absolute necessity.



Clients and servers

In general, all of the machines on the Internet can be categorized as two types: servers and clients. Those machines that provide services (like Web servers or FTP servers) to other machines are **servers**. And the machines that are used to connect to those services are **clients**.

A server machine may provide one or more services on the Internet. For example, a server machine might have software running on it that allows it to act as a Web server, an e-mail server and an FTP server. Clients that come to a server machine do so with a specific intent, so clients direct their requests to a specific software server running on the overall server machine. For example, if you are running a Web browser on your machine, it will most likely want to talk to the Web server on the server machine. Your Telnet application will want to talk to the Telnet server, your e-mail application will talk to the e-mail server, and so on...

URL: Uniform Resource Locator

When you use the Web or send an e-mail message, you use a domain name to do it. For example, the Uniform Resource Locator (URL) "<http://www.erlerobot.com>" contains the domain name erlerobot.com. So does this e-mail address: example@erlerobot.com. Every time you use a domain name, you use the Internet's DNS servers to translate the human-readable domain name into the machine-readable IP address.

Top-level domain names, also called first-level domain names, include .COM, .ORG, .NET, .EDU and .GOV. Within every top-level domain there is a huge list of second-level domains. For example, in the .COM first-level domain there is: Yahoo and Microsoft. Every name in the .COM top-level domain must be unique. The left-most word, like www, is the host name. It specifies the name of a specific machine (with a specific IP address) in a domain. A given domain can, potentially, contain millions of host names as long as they are all unique within that domain.

Ports and HTTP

Any server machine makes its services available using numbered ports (one for each service that is available on the server). For example, if a server machine is running a Web server and a file transfer protocol (FTP) server, the Web server would typically be available on port 80, and the FTP server would be available on port 21. Clients connect to a service at a specific IP address and on a specific port number. Once a client has connected to a service on a particular port, it accesses the service using a specific protocol. Protocols are often text and simply describe how the client and server will have their conversation. Every Web server on the Internet conforms to the **hypertext transfer protocol (HTTP)**.

HTTP is an application protocol for distributed, collaborative, hypermedia information systems. Hypertext is structured text that uses logical links (hyperlinks) between nodes containing text. HTTP is the protocol to exchange or transfer hypertext.

Cookies

A cookie is a piece of text that a Web server can store on a user's hard disk. Cookies allow a Web site to store information on a user's machine and later retrieve it. The pieces of information are stored as name-value pairs.

Cookies evolved because they solve a big problem for the people who implement Web sites. In the broadest sense, a cookie allows a site to store state information on your machine. This information lets a Web site remember what state your browser is in. An ID is one simple piece of state information; if an ID exists on your machine, the site knows that you have visited before. The state is, "Your browser has visited the site at least one time," and the site knows your ID from that visit.

The World Wide Web (WWW)

The Internet has much to offer in terms of information on almost any subject matter imaginable and interaction with people and organizations from all over the world. Much of this access and interaction make use of the environment which is popularly known as the World Wide Web (WWW) or web. The WWW is an interlinked network of systems(web servers) offering multimedia services and information. A user can access these using what is known as web browser software, such as Mozilla or Chrome web browsers.

Servers

A server is a system (software and suitable computer hardware) that responds to requests across a computer network to provide, or help to provide, a network service. Servers can be run on a dedicated computer, which is also often referred to as "the server", but many networked computers are capable of hosting servers. In many cases, a computer can provide several services and have several servers running.



If you want to get into a bit more detail on the process of getting a Web page onto your computer screen, here are the basic steps that occurred behind the scenes:

The browser broke the URL into three parts:

- The protocol ("http")
- The server name ("www.erlerobot.com")
- The file name ("tutorials.htm")

The browser communicated with a name server to translate the server name "www.erlerobot.com" into an IP Address, which it uses to connect to the server machine. The browser then formed a connection to the server at that IP address on port 80. Following the HTTP protocol, the browser sent a GET request to the server, asking for the file ["http://www.erlerobot.com/tutorials.htm"](http://www.erlerobot.com/tutorials.htm).

The server then sent the HTML text for the Web page to the browser. (Cookies may also be sent from server to browser in the header for the page.) The browser read the HTML tags and formatted the page onto your screen.

Server types

Standalone server

Offering a service to users can be approached in two ways. A daemon or service can run in standalone mode, or it can be dependent on another service to be activated.

Network services that are heavily and/or continuously used, usually run in the standalone mode: they are independent program daemons that are always running. They are most likely started up at system boot time, and they wait for requests on the specific connection points or ports for which they are set up to listen. When a request comes, it is processed, and the listening continues until the next request. A web server is a typical example: you want it to be available 24 hours a day, and if it is too busy it should create more listening instances to serve simultaneous users.

An example of a standalone network service on your home computer might be the named (name daemon), a caching name server. Standalone services have their own processes running, you can check any time using `ps` :

```
root@erlerobot:~# ps auxw | grep named
```

```
root@erlerobot:~#  
root@erlerobot:~# ps auxw | grep named  
root      863   1.0  0.2  1516   556 tty00    S+   00:02   0:00 grep --color=auto named  
root@erlerobot:~#
```

However, there are some services that you can use on your PC, even if there is no server process running for that services. Examples could be the FTP service, the secure copy service or the finger service. Those services have the Internet Daemon (inetd) listening in their place.

(x)inetd

On your home PC, things are usually a bit calmer. You may have a small network, for instance, and you may have to transfer files from one PC to another from time to time, using FTP or Samba. In those cases, starting all the services which you only need occasionally and having them run all the time would be a waste of resources. So in smaller setups, you will find the necessary daemons dependent on a central program, that listen on all the ports of the services for which it is responsible.

This super-server, the Internet services daemon, is started up at system initialization time. There are two common implementations: inetd and xinetd (the extended Internet services daemon). One or the other is usually running on every Linux system:

```
ps -ef | grep inet
```

```
root@erlerobot:~# ps -ef | grep inet  
root      866   833   0 00:06 tty00    00:00:00 grep --color=auto inet  
root@erlerobot:~#
```

The services for which the Internet daemon is responsible, are listed in its configuration file, `/etc/inetd.conf`, for inetd, and in the directory `/etc/xinetd.d` for xinetd. Commonly managed services include file share and print services, SSH, FTP, telnet, the Samba configuration daemon, talk and time services.

As soon as a connection request is received, the central server will start an instance of the required server. Thus, in the example below, when user bob starts an FTP session to the local host, an FTP daemon is running as long as the session is active:

```
ps auxw | grep ftp
```



```
root@erlerobot:~# ps auxw | grep ftp
root      855  0.0  0.2  1516  556 tty00    S+   00:01   0:00 grep --color=au
to ftp
root@erlerobot:~#
```

Of course, the same happens when you open connections to remote hosts: either a daemon answers directly, or a remote (x)inetd starts the service you need and stops it when you quit.

Web

The Apache Web Server

Apache is by far the most popular web server, used on more than half of all Internet web servers. Most Linux distributions include Apache. Apache's advantages include its modular design, SSL support, stability and speed. Given the appropriate hardware and configuration it can support the highest loads.

On Linux systems, the server configuration is usually done in the `/etc/httpd` directory. The most important configuration file is `httpd.conf`; it is rather self-explanatory. Should you need help, you can find it in the `httpd` man page or on the Apache website.

Web browsers

A number of web browsers, both free and commercial, exist for the Linux platform. Netscape Navigator as the only decent option has long been a thing of the past, as Mozilla/Firefox or Google Chrome offer a competitive alternative running on many other operating systems, like MS Windows and MacOS X as well.

Amaya is the W3C browser. Opera is a commercial browser, compact and fast. Many desktop managers offer web browsing features in their file manager, like nautilus.

Among the popular text based browsers are lynx and links. You may need to define proxy servers in your shell, by setting the appropriate variables. Text browsers are fast and handy when no graphical environment is available, such as when used in scripts.

Proxy servers

What is a proxy server?

A proxy server is a server (a computer system or an application) that acts as an intermediary for requests from clients seeking resources from other servers. A client connects to the proxy server, requesting some service, such as a file, connection, web page, or other resource available from a different server and the proxy server evaluates the request as a way to simplify and control its complexity.

Proxy configuration

If you have the proxy server name and port, it should be rather obvious to feed that information into your browser. However, many (command line) applications depend on the variables `http_proxy` and `ftp_proxy` for correct functioning. For your convenience, you might want to add a line like the following to your `~/.bashrc`.

You can edit it with `vi` text editor by typing:

```
vi ~/.bashrc
```

Then add

```
export http_proxy=http://username:password@proxy_server_name:port_number
```

For instance:

```
export http_proxy=http://willy:Appelsi3ntj3@proxy:80
```

If you do not need to give a username and password, simply leave out everything before the "@" sign, this sign included.

- For more info about proxy serves you can read: http://www.brennan.id.au/11-Squid_Web_Proxy.html
- Here you can find answers to common questions about proxy serves: <http://www.cyberciti.biz/faq/linux-unix-set-proxy-environment-variable/>

Mail

Electronic mail transport has been one of the most prominent uses of networking since the first networks were devised.

Servers

Sendmail is the standard mail server program or Mail Transport Agent for UNIX platforms. It is robust, scalable, and when properly configured with appropriate hardware, handles thousands of users without blinking. More information about how to configure Sendmail is included with the sendmail and sendmail-cf packages. For more information use `man sendmail` and `man aliases`.

Qmail is another mail server, gaining popularity because it claims to be more secure than Sendmail. While Sendmail is a monolithic program, Qmail consists of smaller interacting program parts that can be better secured. Postfix is another mail server which is gaining popularity.

These servers handle mailing lists, filtering, virus scanning and much more. Free and commercial scanners are available for use with Linux. Examples of mailing list software are Mailman, Listserv, Majordomo and EZmlm. See the web page of your favorite virus scanner for information on Linux client and server support. Amavis and Spamassassin are free implementations of a virus scanner and a spam scanner.

Remote mail servers

The most popular protocols to access mail remotely are POP3 and IMAP4. IMAP and POP both allow offline operation, remote access to new mail and they both rely on an SMTP server to send mail.

While POP is a simple protocol, easy to implement and supported by almost any mail client, IMAP is to be preferred because:

- It can manipulate persistent message status flags.
- It can store as well as fetch mail messages.
- It can access and manage multiple mailboxes.
- It supports concurrent updates and shared mailboxes.
- It is also suitable for accessing Usenet messages and other documents.

IMAP works both on-line and off-line.

it is optimized for on-line performance, especially over low-speed links.

Mail user-agents

There are plenty of both text and graphical E-mail clients, we'll just name a few of the common ones. Pick your favorite.

The UNIX mail command has been around for years, even before networking existed. It is a simple interface to send messages and small files to other users, who can then save the message, redirect it, reply to it etcetera.

While it is not commonly used as a client anymore, the mail program is still useful, for example to mail the output of a command to somebody.

- Sending mail:

```
mail example@erlerobot < doc.txt
```

```
mail -s "Hello Erle" example@erlerobot.com
```

```
echo "This will go into the body of the mail." | mail -s "Hello Erle" example@erlerobot.com
```

- Reading Mail:

In normal usage `mail` is given no arguments and checks your mail out of the post office, then prints out a one line header of each message found. The current message is initially the first message (numbered 1) and can be printed using the ``print` command (which can be abbreviated `p`). You can move among the messages much as you move between lines in `ed`(1), with the commands `+` and `-` moving backwards and forwards, and simple numbers.

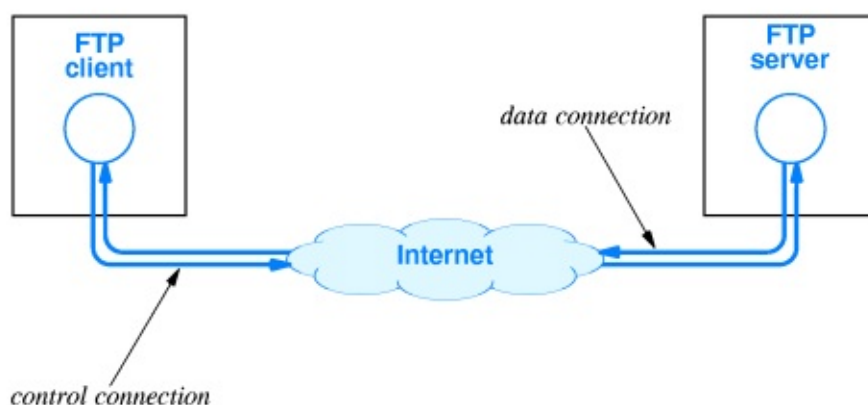
For those users who prefer a graphical interface to their mail (and a tennis elbow or a mouse arm), there are hundreds of options. The most popular for new users are Mozilla Mail/Thunderbird, which has easy anti-spam configuring options, and Evolution, the MS Outlook clone. Kmail is popular among KDE users.

For more info and mail installation/configuration, you can visit: <http://www.ajpdsoft.com/modules.php?name=News&file=article&sid=506#instalarmailx>

FTP

FTP (File Transfer Protocol) is used to transfer files between two computers over a network and Internet. FTP can transfer files between any computers that have an Internet connection, and also works between computers using totally different operating systems.

FTP requires a client (FTP program) installed on your PC to connect to your host, or server. Once you are logged in, you are presented with a directory. If files already exist on your host you can select the download to option and pull files off the FTP server to a specified directory on your computer. If you select the upload option, you must also select your PC directory that contains the files you wish to upload. The files are then copied from one location to the other. It's like copying files from one directory or folder on your PC, except that one of the folders could be on the East coast and the other folder on the West coast.



Transferring files from a client computer to a server computer is called "uploading" and transferring from a server to a client is "downloading".

FTP servers

On a Linux system, an FTP server is typically run from xinetd, using the WU-ftp server, although the FTP server may be configured as a stand-alone server on systems with heavy FTP traffic. See the exercises.

Other FTP servers include among others vsftpd, Ncftpd and Proftpd.

Most Linux distributions contain the anonftp package, which sets up an anonymous FTP server tree and accompanying configuration files.

FTP clients

Most Linux distributions include ncftp, an improved version of the common UNIX `ftp` command, which you may also know from the Windows command line. The ncftp program offers extra features such as a nicer and more comprehensible user interface, file name completion, append and resume functions, bookmarking, session management and more.

Warning FTP is insecure!

Don't use the File Transfer Protocol for non-anonymous login unless you know what you are doing. Your user name and password might be captured by malevolent fellow network users. Use secure FTP instead; the sftp program comes with the Secure SHell suite, see Section

To install and configure ftp in your Linux pc visit: <https://help.ubuntu.com/10.04/serverguide/ftp-server.html>

FTP examples

Use the following syntax to connect to transfer files to and from a remote network ftp site:

```
ftp ftp.example.com
ftp 1.2.3.4
ftp user@ftp.example.com
```

FTP command options:

```
ftp
```

Type the ls command at ftp> prompt for listing current file:

```
ftp> ls
```

To change directory on the remote machine use cd command:

```
ftp> cd dirName
```

To copy one file at a time from the remote ftp server to the local system use `get` command:

```
get fileName
get fileName newFileName
```

In this example, download file resume.pdf in the current remote directory to (or on top of) a file with the same name, resume.pdf, in your current local directory:

```
ftp> get resume.pdf
```

In this example, copies file data.tar.gz in the current remote directory to (or on top of) a file named backup.tar.gz in your current local directory:

```
ftp> get data.tar.gz backup.tar.gz
```

To change directory on your local system, enter:

```
ftp> lcd /path/to/new/dir
ftp> lcd /tmp
```

Print local directory:

```
ftp> lpwd
```

To delete a file in the current remote directory use delete command:

```
ftp> delete fileName
ftp> delete output.jpg
```

To copy one file at a time from the local systems to the remote ftp server, enter:

```
ftp> put fileName
```

Type quit or bye, enter:

```
ftp> quit
```

or

```
ftp> bye
```

For more examples visit this [website](#)

Authentication services

Traditional

Traditionally, users are authenticated locally, using the information stored in `/etc/passwd` and `/etc/shadow` on each system. But even when using a network service for authenticating, the local files will always be present to configure system accounts for administrative use, such as the root account, the daemon accounts and often accounts for additional programs and purposes.

These files are often the first candidates for being examined by hackers, so make sure the permissions and ownerships are strictly set as should be:

```
root@erlerobot:~# ls -l /etc/passwd /etc/shadow
```

```
-rw-r--r-- 1 root root 1256 May 22 2014 /etc/passwd
-rw-r----- 1 root shadow 800 May 22 2014 /etc/shadow
```

PAM

Linux can use PAM, the Pluggable Authentication Module, a flexible method of UNIX authentication. Advantages of PAM:

A common authentication scheme that can be used with a wide variety of applications.

PAM can be implemented with various applications without having to recompile the applications to specifically support PAM.

Great flexibility and control over authentication for the administrator and application developer.

Application developers do not need to develop their program to use a particular authentication scheme. Instead, they can focus purely on the details of their program.

The directory `/etc/pam.d` contains the PAM configuration files (used to be `/etc/pam.conf`). Each application or service has its own file. You can edit or show the content of this files by typing:

```
vi pam.d
```

```
Netrw Directory Listing (netrw v143)
/etc/pam.d
Sorted by name
Sort sequence: [V]$, \<core\W(\.ld\+l)\l=b, \.h$, \.c$, \.cpp$, \~l=*$, *, \.o$, \
Quick Help: <F1>:help ->go up dir D:delete R:rename S:sort-by X:exec

./
chfn
chpasswd
chsh
common-account
common-auth
common-password
common-session
common-session-noninteractive
cron
cups
login
newusers
other
passwd
polkit-1
"pam.d" is a directory 8,1 Top
```

vi pam.conf

```
1 #
2 # /etc/pam.conf
3 #
4 #
5 # NOTE
6 #
7 #
8 # NOTE: Most programs use a file under the /etc/pam.d/ directory to setup the
9 # PAM service modules. This file is used only if that directory does not exist.
10 #
11
12 # Format:
13 # serv. module ctrl module [path] ...[args..]
14 # name type flag
15
"pam.conf" 15L, 552C 1,1 All
```

Each line in the file has four elements:

Module:

- **auth**: provides the actual authentication (perhaps asking for and checking a password) and sets credentials, such as group membership or Kerberos tickets.
- **account**: checks to make sure that access is allowed for the user (the account has not expired, the user is allowed to log in at this time of day, and so on).
- **password**: used to set passwords.
- **session**: used after a user has been authenticated. This module performs additional tasks which are needed to allow access (for example, mounting the user's home directory or making their mailbox available).

The order in which modules are stacked, so that multiple modules can be used, is very important.

- Control Flags: tell PAM which actions to take upon failure or success. Values can be required, requisite, sufficient or optional.
- Module Path: path to the pluggable module to be used, usually in `/lib/security`.
- Arguments: information for the modules.

Shadow password files are automatically detected by PAM.

More information can be found in the pam man pages or at the [Linux-PAM project](#) homepage.

LDAP

The Lightweight Directory Access Protocol is a client-server system for accessing global or local directory services over a network. On Linux, the OpenLDAP implementation is used. It includes slapd, a stand-alone server; slurpd, a stand-alone LDAP replication server; libraries implementing the LDAP protocol and a series of utilities, tools and sample clients.

LDAP directory service is based on a client-server model. One or more LDAP servers contain the data making up the LDAP directory tree or LDAP backend database. An LDAP client connects to an LDAP server and asks it a question. The server responds with the answer, or with a pointer to where the client can get more information (typically, another LDAP server). No matter what LDAP server a client connects to, it sees the same view of the directory; a name presented to one LDAP server references the same entry it would at another LDAP server. This is an important feature of a global directory service, like LDAP.

The main benefit of using LDAP is the consolidation of certain types of information within your organization. For example, all of the different lists of users within your organization can be merged into one LDAP directory. This directory can be queried by any LDAP-enabled applications that need this information. It can also be accessed by users who need directory information.

Since LDAP is an open and configurable protocol, it can be used to store almost any type of information relating to a particular organizational structure. Common examples are mail address lookups, central authentication in combination with PAM, telephone directories and machine configuration databases.

See your system specific information and the man pages for related commands such as `ldapmodify` (LDAP modify entry and LDAP add entry tools) and `ldapsearch` (LDAP search tool) for details.

- More information can be found in the [LDAP Linux HOWTO](#), in this webpage you will also find installing and configuring steps.
- Also this [website](#) should be interesting, for managing the LDAP from the command line.

Remote execution of applications

In this chapter we will talk about remote execution of applications. There are a couple of different ways to execute commands or run programs on a remote machine and have the output, be it text or graphics, sent to your workstation.

Rsh, rlogin and telnet

The `rlogin` and `rsh` commands for remote login and remote execution of commands are inherited from UNIX. While seldom used because they are blatantly insecure, they still come with almost every Linux distribution for backward compatibility with UNIX programs.

Telnet, on the other hand, is still commonly used, often by system and network administrators. Telnet is one of the most powerful tools for remote access to files and remote administration, allowing connections from anywhere on the Internet. Combined with an X server, remote graphical applications can be displayed locally. There is no difference between working on the local machine and using the remote machine.

Because the entire connection is unencrypted, allowing telnet connections involves taking high security risks. For normal remote execution of programs, Secure SHell or `ssh` is advised. We will discuss the secure method later in this section.

However, `telnet` is still used in many cases. For more information use `man telnet`, and for installing them visit this website <http://www.cyberciti.biz/faq/how-do-i-turn-on-telnet-service-on-for-a-linuxfreebsd-system/>

The X Window System

The X Window System was developed at MIT in the late 1980s, rapidly becoming the industry standard windowing system for Unix graphics workstations. The software is freely available, very versatile, and is suitable for a wide range of hardware platforms. Any X environment consists of two distinct parts, the X server and one or more X clients. It is important to realise the distinction between the server and the client. The server controls the display directly and is responsible for all input/output via the keyboard, mouse or display. The clients, on the other hand, do not access the screen directly - they communicate with the server, which handles all input and output. It is the clients which do the "real" computing work - running applications or whatever. The clients communicate with the server, causing the server to open one or more windows to handle input and output for that client.

In short, the X Window System allows a user to log in into a remote machine, execute a process (for example, open a web browser) and have the output displayed on his own machine. Because the process is actually being executed on the remote system, very little CPU power is needed in the local one. Indeed, computers exist whose primary purpose is to act as pure X servers. Such systems are called X terminals.

- [Here](#) you can find more information about the X window System.
- A free port of the X Window System exists for Linux and can be found at: <http://www.xfree86.org/>

Telnet and X

If you would want to use `telnet` to display graphical applications running on a remote machine, you first need to give the remote machine access to your display (to your X server) using the `xhost` command.

By typing a command similar to the one below in a terminal window on your local machine:

```
xhost +remote.machine.com
```

After that, connect to the remote host and tell it to display graphics on the local machine by setting the environment variable `DISPLAY`:

```
export DISPLAY="local.host.com:0.0"
```

After completing this step, any application started in this terminal window will be displayed on your local desktop, using remote resources for computing, but your local graphical resources (your X server) for displaying the application.

This procedure assumes that you have some sort of X server (XFree86, X.org, Exceed, Cygwin) already set up on the machine where you want to display images. The architecture and operating system of the client machine are not important as long as they allow you to run an X server on it.

Mind that displaying a terminal window from the remote machine is also considered to be a display of an image.

SSH suite

SSH (Secure SHell) is a protocol which facilitates secure communications between two systems using a client / server architecture that allows users to connect to a remote host.

Most UNIX and Linux systems now run Secure SHell in order to leave out the security risks that came with Telnet. Most Linux systems will run a version of OpenSSH, an Open Source implementation of the SSH protocol, providing secure encrypted communications between untrusted hosts over an untrusted network. In the standard setup X connections are automatically forwarded, but arbitrary TCP/IP ports may also be forwarded using a secure channel.

The SSH client connects and logs into the specified host name. The user must provide his identity to the remote machine as specified in the `sshd_config` file, which can usually be found in `/etc/ssh`. The configuration file is rather self-explanatory and by defaults enables most common features. Should you need help, you can find it in the `sshd` man pages.

When the user's identity has been accepted by the server, the server either executes the given command, or logs into the machine and gives the user a normal shell on the remote machine. All communication with the remote command or shell will be automatically encrypted.

The session terminates when the command or shell on the remote machine exits and all X11 and TCP/IP connections have been closed.

When connecting to a host for the first time, using any of the programs that are included in the SSH collection, you need to establish the authenticity of that host and acknowledge that you want to connect:

```
ssh 11.0.0.2
```

If you just want to check something on a remote machine and then get your prompt back on the local host, you can give the commands that you want to execute remotely as arguments to `ssh`:

```
ssh 11.0.0.2 who
```

```
root@erlerobot:~# ssh 11.0.0.2 who
ssh: connect to host 11.0.0.2 port 22: No route to host
```

The command `uname` prints operating system name:

```
root@erlerobot:~# uname -n
erlerobot
```

X11 and TCP forwarding

X11 forwarding is when you use SSH to forward X windows to your local machine. If the X11 Forwarding entry is set to yes on the target machine and the user is using X applications, the `DISPLAY` environment variable is set, the connection to the X11 display is automatically forwarded to the remote side in such a way that any X11 programs started from the shell will go through the encrypted channel, and the connection to the real X server will be made from the local machine. The user should not manually set `DISPLAY`. Forwarding of X11 connections can be configured on the command line or in the `sshd` configuration file.

The X server

This procedure assumes that you have a running X server on the client where you want to display the application from the

remote host. The client may be of different architecture and operating system than the remote host, as long as it can run an X server, such as Cygwin (which implements an X.org server for MS Windows clients and others) or Exceed, it should be possible to set up a remote connection with any Linux or UNIX machine.

For more information about this topics you can visit: https://wiki.archlinux.org/index.php/Secure_Shell

Server authentication

The ssh client/server system automatically maintains and checks a database containing identifications for all hosts it has ever been used with. Host keys are stored in `$HOME/.ssh/known_hosts` in the user's home directory. Additionally, the file `/etc/ssh/ssh_known_hosts` is automatically checked for known hosts. Any new hosts are automatically added to the user's file. If a host's identification ever changes, ssh warns about this and disables password authentication to prevent a Trojan horse from getting the user's password. Another purpose of this mechanism is to prevent man-in-the-middle attacks which could otherwise be used to circumvent the encryption. In environments where high security is needed, sshd can even be configured to prevent logins to machines whose host keys have changed or are unknown.

Secure remote copying

The SSH suite provides `scp` command. `scp` uses ssh for data transfer, uses the same authentication and provides the same security as ssh. In other words, `scp` copies files between hosts on a network. For example:

```
scp <file> root@Ip:<destination directory>
```

```
scp prueba.txt root@11.0.0.2:/var/tmp/
```

`scp` can only be used for transferring files, and it is non-interactive (i.e., everything has to be specified on the command line). `sftp` is more elaborate, and allows interactive commands to do things like creating directories, deleting directories and files (all subject to system permissions, of course), etc.

```
sftp 11.0.0.2

sftp> cd /var/tmp

sftp> get Sch*

sftp> bye
```

This are complicated commands, you can learn more [here](#).

Authentication keys

The `ssh-keygen` command generates, manages and converts authentication keys for ssh. It can create RSA keys for use by SSH protocol version 1 and RSA or DSA keys for use by SSH protocol version 2.

Normally each user wishing to use SSH with RSA or DSA authentication runs this once to create the authentication key in `$HOME/.ssh/identity`, `id_dsa` or `id_rsa`. Additionally, the system administrator may use this to generate host keys for the system.

Normally this program generates the key and asks for a file in which to store the private key. The public key is stored in a file with the same name but `.pub` appended. The program also asks for a passphrase. The passphrase may be empty to indicate no passphrase (host keys must have an empty passphrase), or it may be a string of arbitrary length.

There is no way to recover a lost passphrase. If the passphrase is lost or forgotten, a new key must be generated and copied to the corresponding public keys.


```
root@erlerobot:~# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa): /root/.ssh/id_rsa
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
ff:b1:7b:39:71:33:82:91:fa:ea:d4:96:0a:4f:c1:58 root@erlerobot
The key's randomart image is:
+---[ RSA 2048]-----+
|
|      E .
|      +  o
|      .So. o
|      oo....o.
|      . oo+. .+o
|      = oo o+
|      .=. +O .
+-----+
root@erlerobot:~#
```

Extra concepts

VNC

VNC or Virtual Network Computing is in fact a remote display system which allows viewing a desktop environment not only on the local machine on which it is running, but from anywhere on the Internet and from a wide variety of machines and architectures, including MS Windows and several UNIX distributions. You could, for example, run MS Word on a Windows NT machine and display the output on your Linux desktop. VNC provides servers as well as clients, so the opposite also works and it may thus be used to display Linux programs on Windows clients. VNC is probably the easiest way to have X connections on a PC. The following features make VNC different from a normal X server or commercial implementations:

No state is stored at the viewer side: you can leave your desk and resume from another machine, continuing where you left. When you are running a PC X server, and the PC crashes or is restarted, all remote applications that you were running will die. With VNC, they keep on running.

For more info visit the [VCN website](#)

The rdesktop protocol

In order to ease management of MS Windows hosts, recent Linux distributions support the Remote Desktop Protocol (RDP), which is implemented in the rdesktop client. rdesktop is an open source client for Windows Remote Desktop Services, capable of natively speaking Remote Desktop Protocol (RDP) in order to present the user's Windows desktop. The protocol is used in a number of Microsoft products, including Windows NT Terminal Server, Windows 2000 Server, Windows XP and Windows 2003 Server.

The project's homepage is at <http://www.rdesktop.org/>.

Cygwin

Cygwin is a large collection of GNU and Open Source tools which provide functionality similar to a Linux distribution on Windows. Apart from providing UNIX command line tools and graphical applications, it can also be used to display a Linux desktop on an MS Windows machine, using remote X.

You can read more in the official webpage: <https://www.cygwin.com/>

Security

In the ever-changing world of global data communications, inexpensive Internet connections, and fast-paced software development, security is becoming more and more of an issue. Security is now a basic requirement because global computing is inherently insecure. In this chapter we will matter about Linux security.

Secure services and security tips

The goal is to run as few services as possible. If the number of ports that are open for the outside world are kept to a minimum, this is all the better to keep an overview. If services can't be turned off for the local network, try to at least disable them for outside connections.

A rule of thumb is that if you don't recognize a particular service, you probably won't need it anyway. Also keep in mind that some services are not really meant to be used over the Internet. Don't rely on what should be running, check which services are listening on what TCP ports using the netstat command:

```
netstat -l | grep tcp
```

```
root@erlerobot:~# netstat -l | grep tcp
tcp        0      0 *:http          *:*           LISTEN
tcp        0      0 *:ssh           *:*           LISTEN
tcp6       0      0 [::]:ssh        [::]:*        LISTEN
root@erlerobot:~#
```

Security Tips:

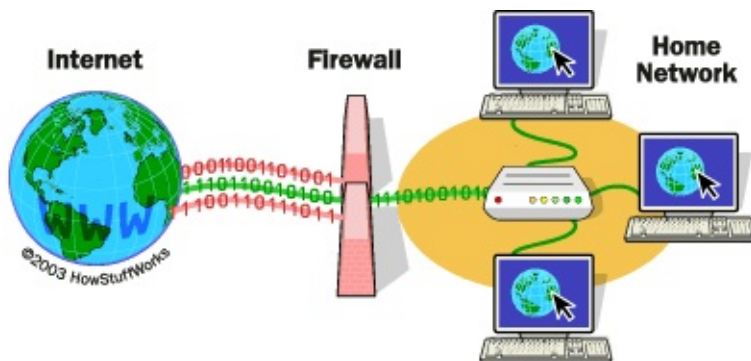
- xec, rlogin and rsh, and telnet just to be on the safe side.
- X11 on server machines.
- No lp if no printer is physically attached.
- No MS Windows hosts in the network, no Samba required.
- Don't allow FTP unless an FTP server is required.
- Don't allow NFS and NIS over the Internet, disable all related services on a stand-alone installation.
- Don't run an MTA if you're not actually on a mail server.
- Do not allow root logins. UNIX developers came up with the su over two decades ago for extra security.
- Direct root access is always dangerous and susceptible to human errors, be it by allowing root login or by using the `su` - command. Rather than using su, it is even better to use `sudo` to only execute the command that you need extra permissions for, and to return afterwards to your own environment.
 - Note: `sudo` allows a permitted user to execute a command as the superuser or another user, as specified in the `sudoers` file.
- Take passwords seriously. Use shadow passwords. Change your passwords regularly.
- Try to always use SSH or SSL. Avoid telnet, FTP and E-mail clients and other client programs which send unencrypted passwords over the network. Security is not only about securing your computer, it is also about securing your passwords.
- Limit resources using quota and/or ulimit.
- The mail for root should be delivered to, or at least read by, an actual person.

Firewall

Firewalls are a means of controlling what information is allowed into and out of your local network. In other words, a firewall is a secure and trusted machine that sits between a private network and a public network.

Typically the firewall host is connected to the Internet and your local LAN, and the only access from your LAN to the Internet is through the firewall. This way the firewall can control what passes back and forth from the Internet and your LAN.

Firewalls can be constructed in quite a variety of ways. The most sophisticated arrangement involves a number of separate machines and is known as a perimeter network. Two machines act as "filters" called chokes to allow only certain types of network traffic to pass, and between these chokes reside network servers such as a mail gateway or a World Wide Web proxy server. This configuration can be very safe and easily allows quite a great range of control over who can connect both from the inside to the outside, and from the outside to the inside



The Linux kernel provides a range of built-in features that allow it to function quite nicely as an IP firewall. The network implementation includes code to do IP filtering in a number of different ways, and provides a mechanism to quite accurately configure what sort of rules you'd like to put in place.

IP Filtering

IP filtering is simply a mechanism that decides which types of IP datagrams will be processed normally and which will be discarded. By discarded we mean that the datagram is deleted and completely ignored, as if it had never been received. You can apply many different sorts of criteria to determine which datagrams you wish to filter; some examples of these are:

- Protocol type: TCP, UDP, ICMP, etc.
- Socket number (for TCP/UDP).
- Datagram type: SYN/ACK, data, ICMP Echo Request, etc.
- Datagram source address: where it came from.
- Datagram destination address: where it is going to.

It is important to understand at this point that IP filtering is a network layer facility. This means it doesn't understand anything about the application using the network connections, only about the connections themselves. For example, you may deny users access to your internal network on the default telnet port, but if you rely on IP filtering alone, you can't stop them from using the telnet program with a port that you do allow to pass through your firewall. You can prevent this sort of problem by using proxy servers for each service that you allow across your firewall. The proxy servers understand the application they were designed to proxy and can therefore prevent abuses, such as using the telnet program to get past a firewall by using the World Wide Web port.

If you want to read more visit the chapter about IP filtering [here](#).

Iptables

Iptables Firewall is a tool that is almost always available in the Linux kernel (for kernel 2.4 and 2.6). His administration is in command line as root.

Here you can find a short introduction to Iptables.

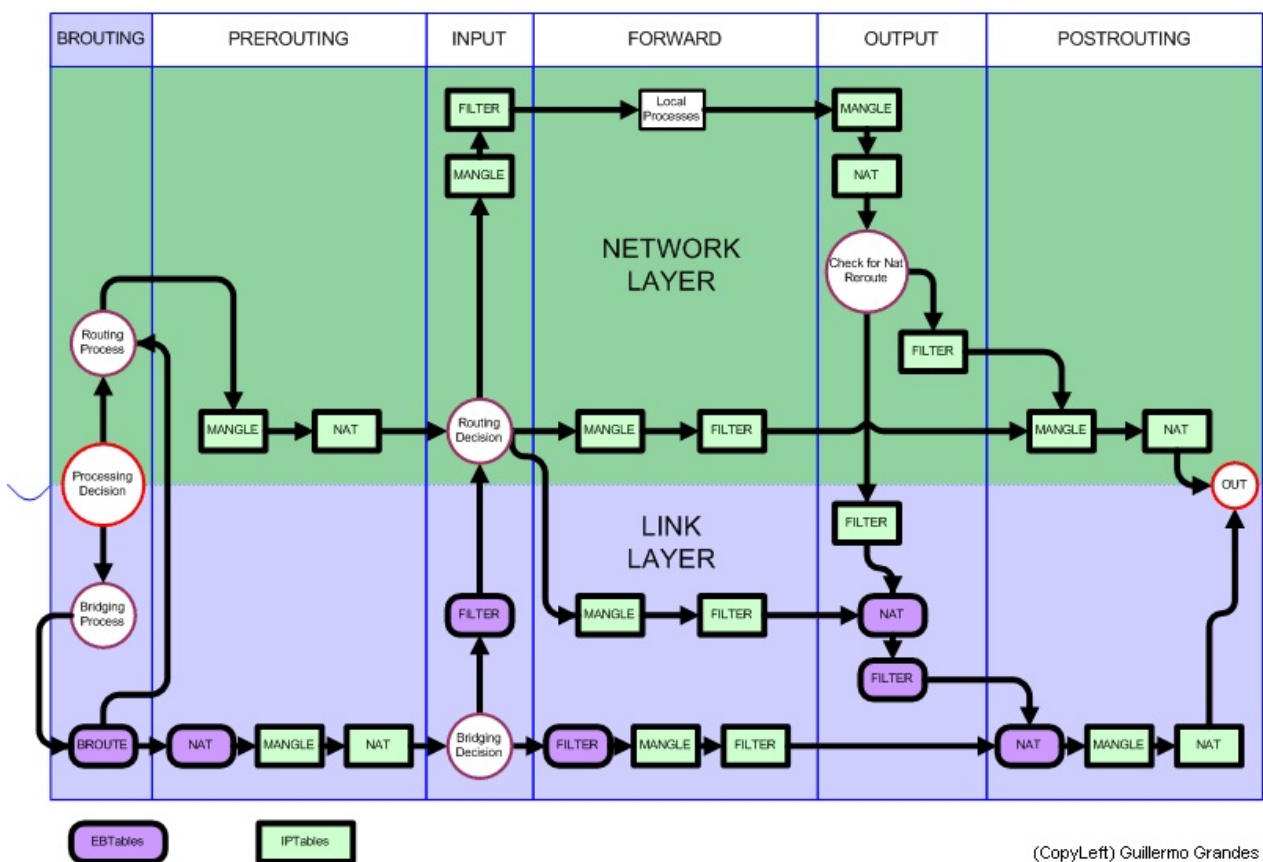
Introduction to IPtables

`iptables` is a powerful **firewall integrated into the Linux kernel** itself and part of the netfilter project (consisting of IPtables, ip6tables, ebtables, arptables and ipset). This firewall is fully configurable:

- command line without installing anything on the computer.
- through a Graphical User Interface(there are already several circulating on the network).

`iptables` is a firewall tool that allows not only filter packets, but also perform network address translation (NAT) for IPv4 or maintain log records. In other words, `iptables` is the name of the user space tool by which administrators can define policies that filter the traffic flowing through the network.

IPtables starts with the system and remains active all the time, adapting and applying to all network traffic rules which have been previously configured.



Types of tables

There are three types of tables in `iptables` :

MANGLE tables:

The mangle tables, are responsible for the modification of packets, such as TTL or TOS for example, at the time you specify. It is normally used so far we will not stop in this section. Just tell that to the Kernel 2.4.17 consisted of 2 strings:

- PREROUTING to modify incoming packets before routing it to the next level IPtables.
- OUTPUT for altering locally generated packets before routing them.

Since kernel 2.4.18, three new brands were introduced in MANGLE tables:

- INPUT to modify incoming packets.

- FORWARD to alter packets that are then routed through Iptables.
- POSTROUTING to change packages that are ready to go.

NAT table:

The NAT (network address translation or network address translation) tables will be consulted when a packet creates a new connection. Allow share a public IP address among many computers, it is essential in many situations by the shortage of IPv4 addresses. Also allow you to add rules that modify the IP addresses of the packets. It contains two main types of rules, SNAT (or IP masquerading) to change the source address and DNAT (or Port Forwarding) for addresses. Amendment contains three chains:

- PREROUTING to modify packets as soon get on your PC.
- OUTPUT for altering locally generated packets before routing them.
- POSTROUTING to update packages that are ready to leave the team.

Filtering tables:

Filtering tables are used by default for the handling of data packets. It is responsible for filtering packages such as the string that is set for each of them to a destination or another, or block. All packets pass through this table, which contains three built-in chains:

- INPUT with which all packets destined for this system traversing this chain (and this is called sometimes LOCAL_INPUT),
- OUTPUT for all packages created by this system pass through this chain (which is also known as the LOCAL_OUTPUT).
- FORWARD (REDIRECTION) with all the packages just go through this system to be routed to its destination walking this chain .

Every network packet received by or sent from a Linux system is subject to at least one table. However, a packet may be subjected to multiple rules within each table before emerging at the end of the chain. The structure and purpose of these rules may vary, but usually seek to identify a packet coming from or going to a particular IP address or set of addresses, when using a particular protocol and network service. Regardless of their destination, when packets match a particular rule in one of the tables, are given a goal (target) or share them.

Every chain has a default policy of ACCEPT , DROP , REJECT , or QUEUE. If any of these rules in the chain apply to the packet, then the packet is treated according to the default policy.

Types Targets

These policies are also known as Targets for the above brands and the most common are:

Target ACCEPT

As once the package complies with all the conditions you have specified, if we use the option-j ACCEPT it will be accepted and not travel more rules in the current chain and any other of the same table. This is important because they clarify the package can be removed in another table.

Target DROP

As once the package complies with all the conditions you have specified, if we use the option-j DROP it will be blocked and will not be processed in any other chain or table. This can have the drawback leave dead sockets and in many cases should use the REJECT target.

Target REJECT

This module has the same effect as 'DROP', except that the sender is sent an ICMP error message "port unreachable"

Target QUEUE

Is used to process packets arriving at a given process, whether accompanied or not addressed to him.

Orders and parameters

Most common orders are:

Order	Meaning
iptables-F	flush (erase, emptying) of all rules.
IPtables-L	list of rules that are being implemented.
iptables-A	add rule.
iptables-D	delete a rule .

Most used parameters are:

Parameter	Meaning
-p [protocol]	protocol to which the packet belongs.
-s [source]	source address of the packet, can be a hostname, a normal IP address or a network address (mask, so address / mask).
-d [destination]	Like the above, it can be a host name, network address, or unique IP address.
-i [interface-entry]	Specify the interface on which the packet is received.
-o [output-interface]	interface by which to send the packet.
[!]-F:	Specifies that the rule refers to second and further fragments of a fragmented packet. If preempts!, Refers only to the first package, or unfragmented packets.
-j [target]	Allows you to choose the target to which to send the packet, ie, the action to perform with him.

For more, you can try typing:

```
iptables --help
```

```

--delete -D chain rulenum          Delete rule rulenum (1 = first) from chain
--insert -I chain [rulenum]        Insert in chain as rulenum (default 1=first)
--replace -R chain rulenum          Replace rule rulenum (1 = first) in chain
--list -L [chain [rulenum]]         List the rules in a chain or all chains
--list-rules -S [chain [rulenum]]   Print the rules in a chain or all chains
--flush -F [chain]                  Delete all rules in chain or all chains
--zero -Z [chain [rulenum]]         Zero counters in chain or all chains
--new -N chain                      Create a new user-defined chain
--delete-chain -X [chain]           Delete a user-defined chain
--policy -P chain target             Change policy on chain to target
--rename-chain -E old-chain new-chain Change chain name, (moving any references)

Options:
--ipv4 -4                          Nothing (line is ignored by iptables-restore)
--ipv6 -6                          Error (line is ignored by iptables-restore)
[!] --proto -p proto                protocol: by number or name, eg. 'tcp'
[!] --source -s address[/mask][...] source specification
[!] --destination -d address[/mask][...] destination specification
[!] --in-interface -i input name[+] network interface name ([+] for wildcard)
--jump -j target                    target for rule (may load target extension)
--goto -g chain                     jump to chain with no return
--match -m match                    extended match (may load extension)
--numeric -n                        numeric output of addresses and ports
[!] --out-interface -o output name[+] network interface name ([+] for wildcard)
--table -t table                    table to manipulate (default: 'filter')
--verbose -v                        verbose mode
--line-numbers                       print line numbers when listing
--exact -x                          expand numbers (display exact values)
[!] --fragment -f                   match second or further fragments only
--modprobe=<command>                try to insert modules using this command
--set-counters PKTS BYTES           set the counter during insert/append
[!] --version -V                    print package version.
root@erlerobot:~#

```

First you may have in to account that if you get this when you use the command line above, you don't need to type `-t filter` in the following commands:

```
--table -t table table to manipulate (default: 'filter')
```

To list or consult our firewall rules:

```
iptables -t filter -L
```

When calling one of the chains, we do this:

```
iptables -t <table> -Action <string>
```

The table is filter, the action can be A,I,P or-D (plus other more specific), With respect to the chain, we have the choice between -INPUT, -OUTPUT , - FORWARD .

example:

```
iptables -t filter -A INPUT
```

To create a basic firewall that blocks incoming connections only, first we build the policies of the three chains of the filter table:

```
iptables -t filter -P INPUT DROP
iptables -t filter -P FORWARD ACCEPT
iptables -t filter -P OUTPUT ACCEPT
```

```
root@erlerobot:~# iptables -P INPUT DROP
root@erlerobot:~# iptables -P FORWARD ACCEPT
root@erlerobot:~# iptables -P OUTPUT ACCEPT
root@erlerobot:~# iptables -L
Chain INPUT (policy DROP)
target     prot opt source                destination
          all  --  anywhere              anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
root@erlerobot:~#
```

Now for an example of a rule that accepts connections to port 80 of the system:

```
iptables -A INPUT -i eth0 -s 0.0.0.0 / 0 -p TCP -dport www -j ACCEPT
```

And here the description of each component of the above command:

- iptables: Iptables command to (do not forget that the rules are a Shell script)
- A: append option to add the rule
- INPUT: package status (to enter is INPUT)
- i eth0: eth0 network interface
- s 0.0.0.0 / 0: address access (either in this case)
- p TCP: port type
- dport: destination port
- j ACCEPT: packet destination (although it is accepted here could be DROP, LOG, REJECT, ..)

For more info you can visit this two websites:

- Iptables man page(<http://ipset.netfilter.org/iptables.man.html>)
- [Iptables basic guides](#)

Practice

Here you find some exercise and quizzes to practice what you have learns in this tutorial.

General Networking

Question 1

This question consist on displaying network information for your workstation. You can read more in [6.2 Network configuration commands](#):

Exercise

Introduce the command necessary to display the IP address. If you don't know the solution feel free to press solution to see the right answer:

```
// Introduce on IP_address the corresponding bash command line  
var IP_address =
```

```
var IP_address= ip addr show;
```

Exercise

Now type the command you should use to display the IP route.

```
// Introduce on IP_route the corresponding bash command line  
var IP_route =
```

```
var IP_route= ip route show;
```

Exercise

Finally introduce the command that shows the nameserver.

```
// Introduce on Serv the corresponding bash command line  
var Serv=
```

```
var Serv= hostname;
```

Question 2

Quiz

Question 1 of 1

Suppose no DNS is available. What would you do to reach your neighbour's machine without typing the IP address all the time?:

- ☐ Edit /etc/hosts file.
- ☐ Use `ifconfig` command.

Suppose no DNS is available. What would you do to reach your neighbour's machine without typing the IP address all the time?:

- ☒ Edit /etc/hosts file.
- ☐ Use `ifconfig` command.

Question 3

Quiz

Question 1 of 1

How would you permanently store proxy information for a text mode browser such as links?:

- ☐ use `chmod` to change proxy servers permission.
- ☐ edit your `~/.bashrc` with a `export` line.

How would you permanently store proxy information for a text mode browser such as links?:

- ☐ use `chmod` to change proxy servers permission.
- ☒ edit your `~/.bashrc` with a `export` line.

Question 4

Quiz

Question 1 of 1

Does your machine run a web server?How to check it?:

- ☐ I check if /etc/httpd file exists.
- ☐ I check it using `ps`.

Does your machine run a web server?How to check it?:

- ☐ I check if /etc/httpd file exists.
- ☒ I check it using `ps`.

Question 5

Exercise

You want to send to *member1@erlerobot.com* a mail containing the *doc.txt* file. Please, enter the command line for doing this. You can find help in *7.4 Mail*. If you don't know press solution.

```
// Introduce on command_line the corresponding bash command line  
var command_line=
```

```
var command_line= mail member1@erlerobot.com < doc.txt;
```

Security

Question 1

Exercise

Type the command line to make a list of open (listening) ports on your machine. For more clues, you can read [9.1 Secure services and security tips](#). Nevertheless you don't know, select solution to see the correct answer.

```
// Introduce on command_line the corresponding bash command line  
var Command_line =
```

```
var Command_line= netstat -l | grep tcp;
```

Question 2

Quiz

Question 1 of 1

How can you see who connected to your system?:

- ☐ using the command `who`
- ☐ using the command `traceroute` for following a packet

How can you see who connected to your system?:

- ☒ using the command `who`
- ☐ using the command `traceroute` for following a packet

Question 3

Quiz

Question 1 of 1

Using Iptables we want to let the packages just go through this system to be routed. Which command line is the correct one? :

- ☐ `iptables -t filter -P FORWARD ACCEPT`
- ☐ `iptables -t filter -P FORWARD DROP`

Using Iptables we want to let the packages just go through this system to be routed. Which command line is the

correct one? :

- ☐ iptables -t filter -P FORWARD ACCEPT
- ☒ iptables -t filter -P FORWARD DROP