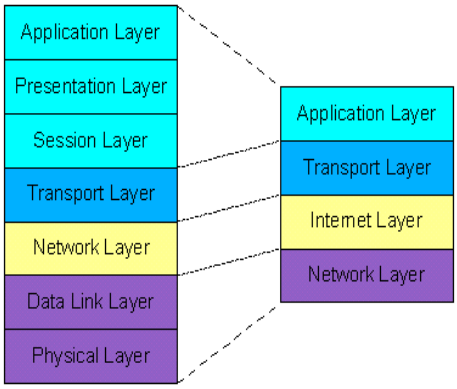
2.3 ADMINISTRACIÓN DE LA RED EN GNU/LINUX

2.3.1 Introducción a TCP/IP

2.3.1.1¿Qué es TCP/IP?

TCP/IP es la base de Internet, y sirve para enlazar computadoras que utilizan diferentes sistemas operativos, incluyendo PC, minicomputadoras y computadoras centrales sobre redes de área local (LAN) y área extensa (WAN). TCP/IP fue desarrollado y utilizado por primera vez en 1972 por el departamento de defensa de los Estados Unidos, ejecutándolo en ARPANET, una red de área extensa del departamento de defensa. TCP/IP son en realidad dos protocolos de comunicación entre ordenadores independientes uno del otro.

El modelo de comunicaciones de la **OSI/ISO** (OSI, Open Systems Interconnection Reference Model, ISO, International Standards Organization), es un modelo teórico de referencia para la arquitectura de red. Existen siete capas de comunicación donde cada una tiene una interfaz para comunicarse con la anterior y la posterior.



TCP es un protocolo orientado a conexión, es decir, es responsable de la transferencia fiable de cada uno de los caracteres (bytes u octetos) que recibe del nivel superior correspondiente. En consecuencia, utiliza números de secuencia y aceptaciones/rechazos. En la torre de protocolos de OSI se corresponde con el nivel 4 o capa de transporte.

Una alternativa al TCP la conforma el protocolo **UDP** (User Datagram Protocol), el cual trata los datos como un mensaje (datagrama) y envía paquetes. Es un protocolo sin conexión y tiene la ventaja de que ejerce una menor sobrecarga a la red que las conexiones de TCP, pero la comunicación no es fiable (los paquetes pueden no llegar o llegar duplicados).

Por otro lado, **IP** (Internet Protocol), define el protocolo que permite identificar las redes y establecer los caminos entre los diferentes ordenadores. Los datos en una red basada en IP son enviados en bloques conocidos como paquetes o datagramas (en el protocolo IP estos términos se suelen usar indistintamente). IP es un protocolo perteneciente a la capa 3 del modelo OSI. IP no provee ningún mecanismo para determinar si un paquete alcanza o no su destino, si se necesita fiabilidad, ésta es proporcionada por los protocolos de la capa de transporte, como TCP. Las cabeceras IP contienen las direcciones de las máquinas origen y destino, direcciones que serán usadas por los enrutadores para decidir el tramo de red por el que reenviarán los paquetes.

Existe otro protocolo alternativo de capa 3 llamado **ICMP** (Internet Control Message Protocol). ICMP se utiliza para mensajes de error o control. Por ejemplo, si uno intenta conectarse a un equipo (host), el ordenador local puede recibir un mensaje ICMP indicando “host unreachable”. ICMP también puede ser utilizado para extraer información sobre una red.

En resumen, TCP/IP es una familia de protocolos (que incluyen IP, TCP, UDP) que proveen un conjunto de funciones a bajo nivel utilizadas por la mayoría de las aplicaciones. Existe actualmente una nueva versión del protocolo **IPv6**, también llamado IPng (IP next generation) que reemplaza al IPv4. Este protocolo mejora notablemente el anterior en temas tales como mayor número de nodos, control de tráfico, seguridad o mejoras en aspectos de routing .

2.3.1.2 Conceptos de TCP/IP

- **Nodo**: Se denomina nodo (host) a una máquina que se conecta a la red (en un sentido amplio un nodo puede ser un ordenador, una impresora, una torre (rack) de CD, etc, es decir, un elemento activo y diferenciable en la red que recalama o presta algún servicio y/o comparte información.

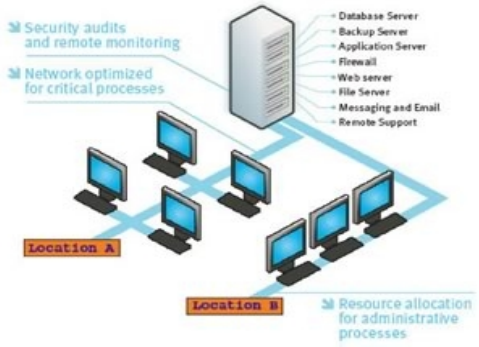
- **Dirección de red Ethernet** (*Ethernet address* o *MAC address*): un número de 48 bits que se encuentra en el dispositivo físico (hardware) del controlador (NIC) de red Ethernet y es grabado por el fabricante del mismo (este número debe ser único en el mundo, por lo que cada fabricante de NIC tiene un rango preasignado).

- Host name: cada nodo debe tener además un único nombre en la red. Ellos pueden ser sólo nombres o bien utilizar un esquema de nombres jerárquico basado en dominios (*hierarchical domain naming scheme*). Los nombres de los nodos deben ser únicos, lo cual resulta fácil en pequeñas redes, más dificultoso en redes extensas e imposible en Internet si no se realiza algún control.

- **Dirección de Internet** (*IP address*): está compuesto por cuatro números en el rango 0-255 separados por puntos (por ejemplo 192.168.0.1) y es utilizado universalmente para identificar los ordenadores sobre una red o Internet. La traslación de nombres en direcciones IP es realizada por un servidor DNS (*Domain Name System*) que transforma los nombres de nodo (legibles por humanos) en direcciones IP.

- **Puerto** (*port*): identificador numérico “del buzón” en un nodo que permite que un mensaje (TCP, UDP) pueda ser leído por una aplicación concreta dentro de este nodo (por ejemplo, dos máquinas que se comuniquen por telnet lo harán por el puerto 23, pero las dos mismas máquinas pueden tener una comunicación ftp por el puerto 21). Se pueden tener diferentes aplicaciones comunicándose entre dos nodos a través de diferentes puertos simultáneamente.

- **Nodo *router*** (*gateway*): es un nodo que realiza encaminamientos. Un *router*, según sus características, podrá transferir información entre dos redes de protocolos similares o diferentes y puede ser además selectivo.



- ***Domain Name System*** (DNS): permite asegurar un único nombre y facilitar la administración de las bases de datos que realizan la traslación entre nombre y dirección de Internet, y se estructuran en forma de árbol. Para ello, se especifican dominios separados por puntos, de los que el más alto (de derecha a izquierda) describe una categoría, institución o país (COM, comercial, EDU, educación, GOV, gubernamental, MIL, militar (gobierno), ORG, sin fin de lucro, XX dos letras por país, ...). El segundo nivel representa la organización, el tercero y restantes departamentos, secciones o divisiones dentro de una organización.

- **DHCP, bootp**: DHCP y bootp son protocolos que permiten a un nodo cliente obtener información de la red (tal como la dirección IP del nodo). Muchas organizaciones, con gran cantidad de máquinas, utilizan este mecanismo para facilitar la administración en grandes redes o donde existe una gran cantidad de usuarios móviles.

- **ARP, RARP**: en algunas redes (como por ejemplo IEEE 802 LAN que es el estándar para Ethernet), las direcciones MAC e IP son descubiertas automáticamente a través de dos protocolos miembros de *Internet protocol suite: Address Resolution Protocol* (ARP) y *Reverse Address Resolution Protocol* (RARP). ARP utiliza mensajes (*broadcast messages*) para determinar la dirección Ethernet correspondiente a una dirección de red particular (IP). RARP utiliza mensajes de tipo *broadcast* (mensaje que llega a todos los nodos) para determinar la dirección de red asociada con una dirección hardware en particular. RARP es especialmente importante en máquinas sin disco, en las cuales la dirección de red generalmente no se conoce en el momento del inicio (*boot*).

2.3.1.3 Direccionamiento IP

Una **dirección IP** es un número que identifica de manera lógica y jerárquica a una interfaz de un dispositivo (habitualmente una computadora) dentro de una red que utilice el protocolo IP, que corresponde al nivel de red o nivel 3 del modelo de referencia OSI.

Los sitios de Internet que por su naturaleza necesitan estar permanentemente conectados, generalmente tienen una *dirección IP fija,* es decir, no cambia con el tiempo. Los servidores de correo, DNS, FTP públicos, y servidores de páginas WEB necesariamente deben contar con una dirección IP fija o estática, ya que de esta forma se permite su localización en la red.

En su versión 4, una dirección IP se representa mediante un número binario de 32 bits (Ipv4). Las direcciones IP se pueden expresar como números de notación decimal: se dividen los 32 bits de la dirección en cuatro octetos. El valor decimal de cada octeto puede ser entre 0 y 255 (el número binario de 8 bits más alto es 11111111 y esos bits, de derecha a izquierda, tienen valores decimales de 1,2,4,8,16,32,64 y 128, lo que suma 255 en total). En la expresión de direcciones Ipv4 en decimal se separa cada octeto por un carácter ".". Cada uno de estos octetos puede estar comprendido entre 0 y 255, salvo algunas excepciones.

Hay tres clases de direcciones IP que una organización puede recibir de parte de la Internet Corporation for Assigned Names and Numbers (ICANN): clase A, clase B y clase C. En la actualidad, ICANN reserva las direcciones de clase A para los gobiernos de todo el mundo (aunque en el pasado se le hayan otorgado a empresas de gran envergadura como, por ejemplo, Hwelett Packard) y las direcciones de clase B para las medianas empresas. Se otorgan direcciones de clase C para todos los demás solicitantes. Cada clase de red permite una cantidad fija de equipos (hosts).

- En una red de clase A, se asigna el primer octeto para identificar la red, reservando los tres últimos octetos (24 bits) para que sean asignados a los hosts, de modo que la cantidad máxima de hosts es 2^{24} (menos dos: las direcciones reservadas de broadcast [tres últimos octetos a 255] y de red [tres últimos octetos a 0], es decir, 16 777 214 hosts.

- En una red de clase B, se asignan los dos primeros octetos para identificar la red, reservando los dos octetos finales (16 bits) para que sean asignados a los hosts, de modo que la cantidad máxima de hosts es 2^{16} (menos dos), o 65 534 hosts.

- En una red de clase C, se asignan los tres primeros octetos para identificar la red, reservando el octeto final (8 bits) para que sea asignado a los hosts, de modo que la cantidad máxima de hosts es 2^{8} (menos dos), o 254 hosts.

Clase	Rango	Máscara de Red	Broadcast
A	**1.0.0.0 – 127.255.255.255**	**255.0.0.0**	**X.255.255.255**
B	**128.0.0.0 – 191.255.255.255**	**255.255.0.0**	**X.X.255.255**
C	**192.0.0.0 – 223.255.255.255**	**255.255.255.0**	**X.X.X.255**
D	**224.0.0.0 – 239.255.255.255**	Dirección de multicast	
E	**240.0.0.0 – 255.255.255.255**	Reversado para uso futuro	

- La dirección 0.0.0.0 es utilizada por las máquinas cuando están arrancando o no se les ha asignado dirección.

- La dirección que tiene su parte de host a cero sirve para definir la red en la que se ubica. Se denomina **dirección de red**.

- La dirección que tiene su parte de host a unos sirve para comunicar con todos los hosts de la red en la que se ubica. Se denomina **dirección de broadcast**.

- Las direcciones 127.x.x.x se reservan para pruebas de retroalimentación. Se denomina **dirección de bucle local** o **loopback**.

Hay ciertas direcciones en cada clase de dirección IP que no están asignadas y que se denominan **direcciones privadas**. Las direcciones privadas pueden ser utilizadas por los hosts que usan traducción de dirección de red (**NAT)** para conectarse a una red pública o por los hosts que no se conectan a Internet. En una misma red no puede existir dos direcciones iguales, pero sí se pueden repetir en dos redes privadas que no tengan conexión entre sí o que se vean a través de NAT. Las direcciones privadas son:

- Clase A: 10.0.0.0 a 10.255.255.255 (8 bits red, 24 bits hosts)
- Clase B: 172.16.0.0 a 172.31.255.255 (16 bits red, 16 bits hosts)
- Clase C: 192.168.0.0 a 192.168.255.255 (24 bits red, 8 bits hosts)

A partir de 1993, ante la previsible futura escasez de direcciones IPv4 debido al crecimiento exponencial de hosts en Internet, se empezó a introducir el sistema CIDR (Classless Inter-Domain Routing o Encaminamiento Inter-Dominios sin Clases), que pretende en líneas generales establecer una distribución de direcciones más fina y granulada, calculando las direcciones necesarias y "desperdiciando" las mínimas posibles, para solventar el problema que la distribución por clases había estado gestando. Este sistema es, de hecho, el empleado actualmente para la delegación de direcciones.

Muchas aplicaciones requieren conectividad dentro de una sola red, y no necesitan conectividad externa. En las redes de gran tamaño a menudo se usa TCP/IP. Por ejemplo, los bancos pueden utilizar TCP/IP para conectar los cajeros automáticos que no se conectan a la red pública, de manera que las direcciones privadas son ideales para ellas. Las direcciones privadas también se pueden utilizar en una red en la que no hay suficientes direcciones públicas disponibles.

Las direcciones privadas se pueden utilizar junto con un servidor de traducción de direcciones de red (NAT) para suministrar conectividad a todos los hosts de una red que tiene relativamente pocas direcciones públicas disponibles. Según lo acordado, cualquier tráfico que posea una dirección destino dentro de uno de los intervalos de direcciones privadas no se enrutará a través de Internet.

Direcciones Ipv6

La función de la dirección IPv6 es exactamente la misma a su predecesor IPv4, pero dentro del protocolo IPv6. Está compuesta por 8 segmentos de 2 bytes cada uno, que suman un total de 128bits, el equivalente a unos 3,4x10^38 hosts direccionables. La ventaja con respecto a la dirección IPv4 es obvia en cuanto a su capacidad de direccionamiento.

Su representación suele ser hexadecimal y para la separación de cada par de octetos se emplea el símbolo ":". Un bloque abarca desde 0000 hasta FFFF. Algunas reglas acerca de la representación de direcciones IPv6 son:

- Los ceros iniciales, como en IPv4, se pueden obviar.

  Ejemplo: 2001:0123:0004:00ab:0cde:3403:0001:0063 > 2001:123:4:ab:cde:3403:1:63

- Los bloques contiguos de ceros se pueden comprimir empleando "::". Esta operación sólo se puede hacer una vez.

  Ejemplo: 2001:0:0:0:0:0:0:4 -> 2001::4.

2.3.1.4 Encaminamiento de paquetes

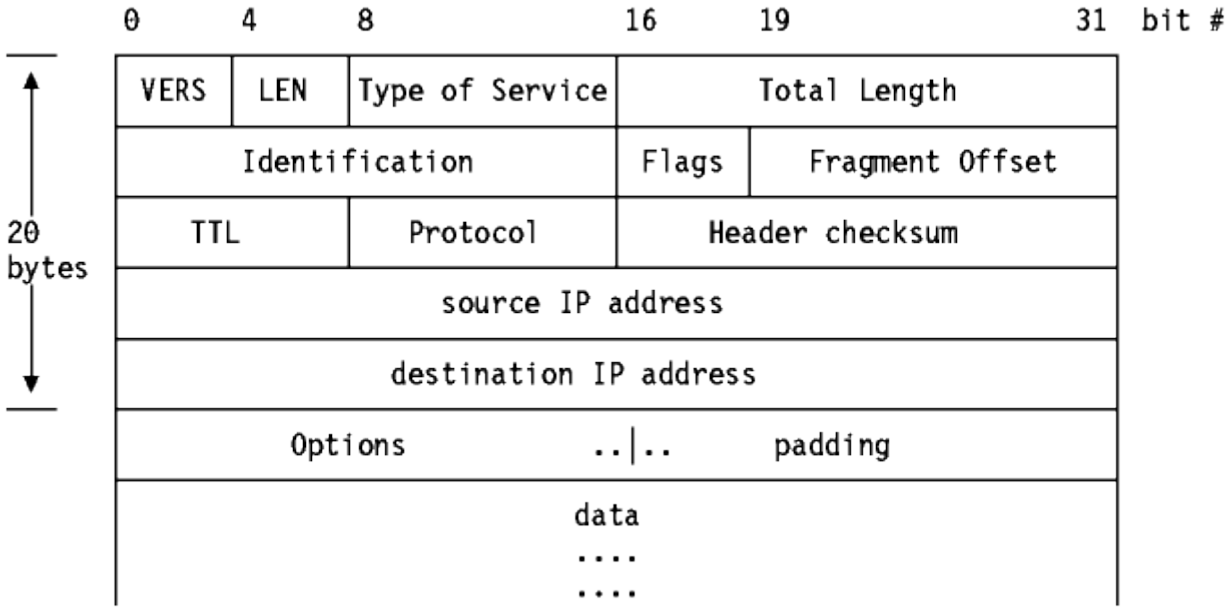
El protocolo IP proporciona los mecanismos necesarios para el transporte de los segmentos creados por TCP. A las unidades creadas por IP se les llama datagramas de IP, que son transportadas por la red a través de rutas de manera independiente unas de otras. IP es un protocolo en el nivel 3 de OSI, su función es hacer todo lo que se pueda para entregar un datagrama al host destino. IP no garantiza la entrega fiable de los datagramas al host destino, los datagramas se pueden destruir en el camino debido a:

- Errores en los bits durante la transmisión por el medio.
- Que un encaminador congestionado descartó el datagrama debido a la falta de espacio en buffer.
- Temporalmente, no había camino hasta el destino.
- etc.

Todas las funciones que aseguran la fiabilidad del envío y la entrega de datos se ha concentrado en la capa de TCP como vimos con anterioridad. IP sólo hace lo mejor por entregar estos datagramas de un extremo a otro.

Los datagramas IP están formados por palabras de 32 bits. La cabecera de cada Datagrama tiene un mínimo (y tamaño más frecuente) de cinco palabras y un máximo de quince.
Los campos que lo forman son los siguientes:

VERS: versión de IP que emplea para construir el Datagrama.



LEN : Tamaño de la cabecera en palabras.

TOS: Tipo de servicio. Su estructura es:

Prioridad	D	T	R	Sin uso

La prioridad (0=normal, 7=control de red) permite implementar algoritmos de control de congestión más eficientes. Los tipos D, T y R solicitan un tipo de transporte dado: D = Procesamiento con retardos cortos, T = alto desempeño y R = alta confiabilidad. Nótese que estos bits son solo “sugerencias”, no es obligatorio para la red cumplirlos.

Longitud total: Mide en bytes la longitud de todo el Datagrama.

Identificación: Número de 16 bits que identifica el Datagrama, que permite implementar números de secuencias y que permite reconocer los diferentes fragmentos de un mismo Datagrama pues todos ellos comparten este número.

Banderas: Un campo de tres bits donde el primero está reservado. El segundo indica cómo se fragmenta el datagrama.

Desp. de Fragmento: A un trozo de datos se le llama bloque de Fragmento. Este campo indica el tamaño del desplazamiento en bloques de fragmento con respecto al Datagrama original.

TTL: Tiempo de vida del datagrama, especifica el número de segundos que se permite al Datagrama circular por la red antes de ser descartado.

Protocolo: Especifica qué protocolo de alto nivel se empleó para construir el mensaje transportado en el campo de datos del Datagrama. Algunos valores posibles son: 1=ICMP, 6=TCP, 17=UDP, 88=IGRP (Protocolo de Enrutamiento de Pasarela Interior de CISCO).

Checksum: Es un campo de 16 bits que se calcula haciendo el complemento a uno de cada palabra de 16 bits del encabezado, sumándolas y haciendo su complemento a uno. Esta suma hay que recalcularla en cada nodo intermedio debido a cambios en el TTL o por fragmentación.

Dirección IP de la Fuente

Dirección IP del Destino

Opciones IP: Existen hasta 40 bytes en la cabecera del Datagrama IP que pueden llevar una o más opciones.

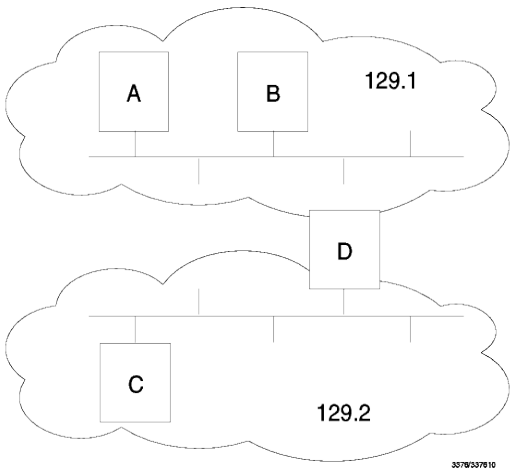
Una función importante de la capa IP es el **encaminamiento.** Proporciona los mecanismos básicos para interconectar distintas redes físicas. Esto significa que un *host* puede actuar simultáneamente como host normal y como "router". Un host sólo contiene información acerca de posibles destinos:

- Los hosts conectados directamente a la red física a la que está conectado el host.
- Los hosts o redes para las que se le han dado "definiciones" específicas.
- Los hosts o redes para las que el host ha recibido un mensaje ICMP *redirect.*
- Un destino por defecto para todo lo demás.

Si el host destino está conectado a una red a la que también está conectado el host fuente, un datagrama IP puede ser enviado directamente, simplemente encapsulando el datagrama IP en una trama. Es lo que se llama *encaminamiento directo.*

El *encaminamiento indirecto* ocurre cuando el host de destino no está en una red conectada directamente al host fuente. La única forma de alcanzar el destino es a través de uno o más "routers". La dirección del primero de ellos (el primer salto) se llama ruta indirecta (Gateway o puerta de enlace). La dirección del primer salto es la única información que necesita el host fuente: el "router" que reciba el datagrama se responsabiliza del segundo salto, y así sucesivamente.

En el siguiente ejemplo vemos que el host A tiene un ruta directa con B y D, y una indirecta con C. El host D es un "router" entre las redes 129.1 y 129.2



Un host puede distinguir si una ruta es directa o indirecta examinando el número de red destino y subred de la dirección IP. Si coinciden con una de las direcciones de red o subred del host fuente, la ruta es directa. En este caso, el host necesita ser capaz de direccionar correctamente el objetivo usando un protocolo inferior en el modelo OSI a IP. Esto se puede hacer automáticamente usando un protocolo como ARP (Address Resolution Protocol). Para rutas indirectas, el único conocimiento requerido es la dirección IP de un "router" que conduzca a la red de destino.

La implementacioens de IP pueden soportar también rutas explícitas, es decir, una ruta a una dirección IP concreta. Esto es habitual en las conexiones que usan SLIP ("Serial Line Internet Protocol") que no proporciona un mecanismo para que dos hosts se informen mutuamente de sus direcciones IP. Tales rutas pueden tener incluso el mismo número de red que el host, por ejemplo en subredes compuetas de enlaces punto a punto. En general, sin embargo, la información de encaminamiento se genera sólo mediante los números de red y de subred.

Cada host guarda el conjunto de mapeados entre las direcciones IP de destino y las direcciones IP del siguiente salto para ese destino en una tabla llamada **tabla de encaminamiento IP.** En esta tabla se pueden encontrar tres tipos de mapeado:

- Rutas directas, para redes conectadas localmente.
- Rutas indirectas, para redes accesibles a través de uno o más "routers".
- Una ruta por defecto, que contiene la dirección IP de un "router" que todas las direcciones IP no contempladas en las rutas directas e indirectas han de usar.

Veamos un ejemplo gráfico:

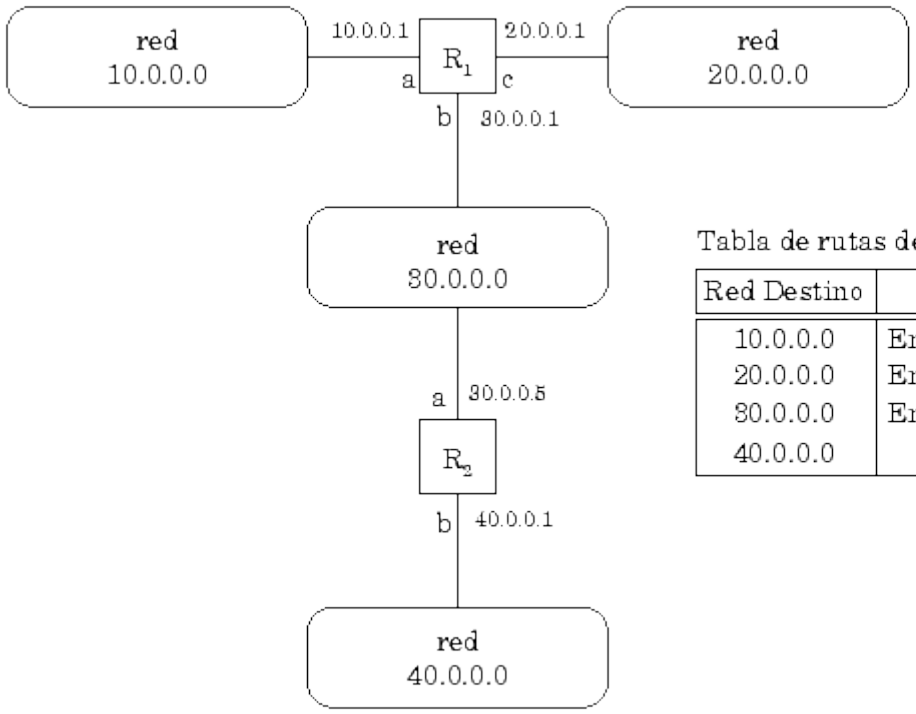


Tabla de rutas de R_1

Red Destino	Enrutar a:	iF
10.0.0.0	Entrega directa	a
20.0.0.0	Entrega directa	c
30.0.0.0	Entrega directa	b
40.0.0.0	30.0.0.5	b

2.3.2 Configuración de parámetros de Red

2.3.2.1 Configuración de la red en Debian

Las herramientas tradicionales de configuración de red a bajo nivel en sistemas GNU/Linux son los programas *ifconfig* y *route* que vienen en el paquete *net-tools.* Estas herramientas han sido oficialmente reemplazadas por *ip* que viene en el paquete *iproute2.* El programa *ip* funciona con Linux 2.2. y superior y es más poderoso que las herramientas anteriores. Sin embargo, las herramientas anteriores aún funcionan y resultan más familiares para muchos usuarios.

Veamos una ilustración de cómo cambiar la dirección IP de la interfaz *eth1* y convertir a *eth1* en una ruta de red. Empezamos ejecutando **ifconfig** y **route** sin argumentos para mostrar el estado actual de todas las interfaces de red y encaminamiento. Estas herramientas vienen en el paquete *netbase* instalado por defecto en nuestro sistema base.

```
debian:~# ifconfig
eth1      Link encap:Ethernet  HWaddr 00:00:1A:00:03:E2
          inet addr:192.168.1.11  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::200:1aff:fe00:3e2/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2625 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1075 errors:0 dropped:0 overruns:0 carrier:0
          collisions:11 txqueuelen:1000
          RX bytes:168018 (164.0 KiB)  TX bytes:51128 (49.9 KiB)
          Interrupt:5 Base address:0xe800

debian:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.1.0     *               255.255.255.0   U     0      0        0 eth1
default         192.168.1.1     0.0.0.0         UG    0      0        0 eth1
```

ifconfig es una utilidad de línea de comandos que permite obtener y configurar las interfaces de red de un equipo. Si no se proporcionan argumentos, ifconfig muestra el estado de las interfaces de red que se encuentran activas. Si se proporciona una interfaz como argumento, ifconfig muestra el estado de dicha interfaz.

Para modificar la dirección IP de una diercción de red introduciríamos lo siguiente:

```
ifconfig eth1 nueva_dirección_IP netmask máscara_red gw gateway broadcast dir_broadcast
```

Para añadir una entrada en la tabla de encaminamiento:

```
route add -net red_destino netmask máscara_red gw gateway dev inferfaz_ethernet
```

Un sistema Debian a veces necesita identificarse por su nombre. Para este propósito el kernel guarda el **hostname** (nombre de la máquina). El script de inicio /etc/init.d/hostname.sh establece el nombre de la máquina duante el arranque (con el comando *hostname*) usando el nombre almacenado en /etc/hostname. Este archivo únicamente debería contener el nombre de la máquina y no un nombre de dominio completo. Para mostrar el nombre actual de la máquina basta ejecutar *hostname* sin argumentos.

Las máquinas son referencias por el nombre de dominio y por su dirección IP. **DNS** es un sistema cliente-servidor en donde los sistemas de resolución de nombres (llamados también traductores de direcciones) consultan a los servidores de nombres con objeto de asociar los nombres de dominio con las direcciones IP y otras propiedades de las máquinas.

La tarea de averiguar las direcciones IP asociadas con un nombre de dominio particular es la función de un sistema de resolución (resolver). El más utilizado es el conjunto de funciones de la biblioteca C GNU que llevan este nombre. La forma que el sistema de resolución de la biblioteca C resuelve los nombres viene dada por la línea *hosts* del archivo de configuración */etc/nsswitch.conf.* Esta línea lista los servicios que deberían usarse para resolver un nombre: por ejemplo, *dns, files, nis, nisplus*.

```
hosts:          files dns
```

Si se utiliza el servicio *files* el comportamiento del sistema de resolución también viene regido por el archivo de configuración */etc/hosts. Hosts* es un fichero simple de texto que asocia direcciones IPs con nombres de hosts.

```
127.0.0.1       localhost.localdomain       localhost
192.168.1.25    servidor
192.168.1.50     ap
```

Si se utiliza el servicio *dns,* el comportamiento del sistema de resolución también viene dado por el archivo de configuración */etc/resolv.conf.* Una de las funciones importantes del archivo *resolv.conf* consiste en listar las direcciones IP de los servidores de nombres que se contactarán para resolver el nombre. Esta lista a menudo depende del entorno de red que puede cambiar de tanto en tanto mientras la máquina está funcionando. Programas tales como *pppd* y *dhclient* son capaces de manipular resolv.conf para añadir y eliminar líneas, pero estas características no siempre funcionan adecuadamente y entran en conflicto entre sí.

```
debian:/etc# cat resolv.conf
nameserver 80.58.61.250
nameserver 80.58.61.254
```

Configuración de la red a alto nivel en Debian

A fin de facilitar la configuración de la red, Debian proporciona una herramienta estándar de configuración de red de alto nivel que consiste en los progarmas *ifup*, *ifdown* y el archivo */etc/network/interfaces.* Configuraremos una interfaz mediante:

```
ifdown eth1
nano /etc/network/interfaces    # Modificamos a nuestro antojo
ifup eth1
```

Para configurar una IP estática editaremos el fichero */etc/network/interfaces* de modo que incluya un fragmento como el siguiente:

```
iface eth1 inet static
      address 192.168.1.11
      netmask 255.255.255.0
      network 192.168.1.0
      broadcast 192.168.1.255
      gateway 192.168.1.1
      # dns-* options ar implemented by the resolvconf package, if installed
      dns-nameservers 80.58.61.254
```

A continuación comentaremos las entradas del archivo: En este equipo hay una entrada del primer dispositivo de red Ethernet, la cual se configura de forma estática.

- ***address***: es la dirección IP de la tarjeta en cuestión.
- ***netmask***: es la máscara de subred asociada a la dirección.
- ***network****:* es la dirección de red asociada a la dirección.
- ***broadcast***: es la dirección de broadcast de la red. La dirección de broadcast es aquella que se utiliza para enviar un paquete a todos los equipos conectados en la red. Se halla aplicando una OR entre la máscara de red invertida y la dirección de red.
- ***gateway****:* en este campo colocaremos la dirección del Gateway que nos da acceso a otras redes (normalmente, Internet).

Si en lugar de una configuración estática queremos configurar la interfaz de red dinámicamente utilizando DHCP escribiremos:

```
iface eth1 inet dhcp
```

Posteriormente en otro apartado analizaremos en detalle el protocolo DHCP.

Cuando la máquina arranca, en la carga de configuraciones iniciales, se lanza el script de inicio */etc/rcS.d/S40networking* que ejecuta el comando ifup -a. Esto activa todas las interfaces físicas que aparecen en las secciones *auto* de */etc/network/interfaces*. Por ejemplo, siempre se desea por lo menos que la interfaz local (loopback) *lo* se active en el arranque. Por lo tanto nuestro */etc/network/interfaces* incluirá las siguientes líneas:

```
auto lo
iface lo inet loopback
```

Probando la red:

Ping es un programa útil para verificar si una instalación TCP/IP es satisfactoria. Su nombre proviene de *Packet Internet Groper*. El comando ping trabaja enviando múltiples paquetes IP a un destino específico. Cada paquete enviado es una solicitud de respuesta. La respuesta de salida para un ping contiene la ratio de éxito y el tiempo de ida y vuelta al destino. Cada paquete enviado es una solicitud de respuesta. La respuesta de salida para un ping contiene la radio de éxito y el tiempo de ida y vuelta al destino. A partir de esta información se puede determinar si hay conectividad con un destino. El comando ping se utiliza para probar la función de transmisión/recepción de una tarjeta de red, la configuración TCP/IP y la conectividad de una red.

Como ayuda para diagnosticar la conectividad básica de red, muchos protocolos de red admiten un protocolo de eco. Los protocolos de eco se utilizan para verificar el enrutamiento de los paquetes de protocolo. El comando ping envía un paquete al host destino y luego espera un paquete de respuesta de ese host. Los resultados de este protocolo de eco pueden ayudar a evaluar la confiabilidad de ruta a host, las demoras en la ruta y si se puede acceder al host, o si éste está funcionando.
En el siguiente ejemplo hacemos un ping a 127.0.0.1. La red 127.0.0.0 se reserva para las pruebas de loopback. Si el ping tiene éxito, TCP/IP está bien instalado y funcionando en este computador.

```
debian:/etc# ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.431 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.043 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.044 ms
...
```

El comando **traceroute** es la herramienta ideal para descubrir por dónde se envían los datos en una red. El comando traceroute es similar el ping salvo que en lugar de probar la conectividad de extremo a extremo, traceroute verifica cada paso en el proceso.

```
apt-get install traceroute
traceroute www.debian.org
traceroute to www.debian.org (194.109.137.218), 30 hops max, 40 byte packets
 1  * * *
 2  * * *
 3  10.8.0.95 (10.8.0.95)  390.507 ms  223.149 ms  299.011 ms
 4  t3-4.mpd01.mad04.atlas.cogentco.com (149.6.83.229)  47.221 ms  49.052 ms *
 5  v3493.mpd01.mad05.atlas.cogentco.com (130.117.2.69)  48.734 ms  48.205 ms  48.584 ms
 6  t2-1.mpd02.mad05.atlas.cogentco.com (130.117.2.42)  48.647 ms  48.511 ms  48.747 ms
 7  t4-3.mpd01.par02.atlas.cogentco.com (130.117.2.249)  94.398 ms  94.396 ms  93.964 ms
 8  t2-2.mpd02.par01.atlas.cogentco.com (130.117.2.81)  96.580 ms  93.958 ms  93.597 ms
 9  t3-2.mpd02.lon01.atlas.cogentco.com (130.117.2.162)  85.708 ms  85.478 ms  85.243 ms
10  t4-2.mpd01.ams03.atlas.cogentco.com (130.117.1.37)  93.340 ms  92.402 ms  92.597 ms
```

```
11  ams-ix.tc2.xs4all.net (195.69.144.166)  92.271 ms  93.586 ms  91.838 ms
12  0.so-7-0-0.xr1.3d12.xs4all.net (194.109.5.9)  93.580 ms 0.so-7-0-0.xr2.3d12.xs4all.net (194.109.5.13)
92.016 ms 0.so-7-0-0.xr1.3d12.xs4all.net (194.109.5.9)  93.408 ms
13  0.so-2-0-0.cr1.3d12.xs4all.net (194.109.5.74)  103.591 ms  115.143 ms 0.so-3-0-0.cr1.3d12.xs4all.net
(194.109.5.58)  148.954 ms
14  klecker.debian.org (194.109.137.218)  92.865 ms  92.303 ms  93.355 ms
```

Desde nuestra ubicación, existen 14 saltos para alcanzar www.debian.org.

2.3.2.2 Servidor de parámetros de Red (DHCP)

Una de las tareas más tediosas del administrador de redes, es la de recorrer los puestos de trabajo para configurar números de IP ya sea por la llegada de un equipo nuevo, o por modificaciones en la estructura de la red.

El uso de IPs estáticas en redes bajo nuestra tutela conlleva siempre inconvenientes de mantenimiento sin importar las dimensiones de la misma; Los nuevos nodos que se conecten a la red o los cambios vitales como servidores de nombres, pasarela (gateway), servidores WINS, etc. nos obliga a tener que movernos a todos los equipos y actualizar de uno en uno la nueva configuración. Dado que cuanto más grande es la red, más difícil es mantenerla, el IETF decidió resolver el problema desarrollando el **DHCP (Protocolo para Configuración Dinámica de Terminales**). Este protocolo, basado en otro llamado **BOOTP (Bootstrap Protocol)** pero con mejores características, está descrito en el RFC 2131 y en anteriores como el 1541 y el 1531, ya obsoletos por la publicación del primero.

El protocolo DHCP incluye tres métodos de asignación de direcciones IP:

- **Asignación manual o estática**: cuando el servidor tiene a su disposición una tabla que empareja direcciones MAC con direcciones IP, creada manualmente por el administrador de la red. Sólo clientes con una dirección MAC válida recibirán una dirección IP del servidor.
- **Asignación automática**: Asigna una dirección IP de forma permanente a una máquina cliente la primera vez que hace la solicitud al servidor DHCP y hasta que el cliente la libera. Se suele utilizar cuando el número de clientes no varía demasiado.
- **Asignación dinámica**: el único método que permite la reutilización dinámica de las direcciones IP. El administrador de la red determina un rango de direcciones IP y cada computadora conectada a la red está configurada para solicitar su dirección IP al servidor cuando la tarjeta de interfaz de red se inicializa. El procedimiento usa un concepto muy simple en un intervalo de tiempo controlable. Esto facilita la instalación de nuevas máquinas clientes a la red.

Un servidor DHCP puede proveer de una configuración opcional a la computadora cliente. Dichas opciones están definidas en RFC 2132. Esas opciones son: Dirección del servidor DNS, nombre DNS, puerta de enlace de la dirección IP, dirección de broadcast, máscara de subred, tiempo máximo de espera del ARP, MTU para la interfaz, servidores NIS, Dominios NIS, servidores NTP, servidor SMTP, servidor TFTP, nombre del servidor WINS.

DHCP usa los mismos puertos asignados por el IANA (*Autoridad de Números Asignados en Internet* según siglas en inglés) en BOOTP, que son el 67/UDP para las computadoras servidor y 68/UDP para los clientes.

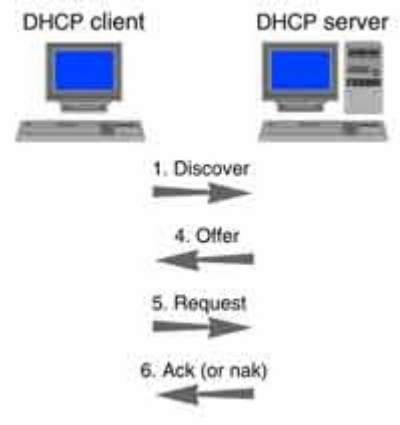
El protocolo de mensajes intercambiados es como sigue:

1 DHCP Discover: Los clientes emiten peticiones masivamente en la subred local para encontrar un servidor disponible, mediante un paquete de broadcast. El router puede ser configurado para redireccionar los paquetes DHCP a un servidor DHCP en una subred diferente. La implementación cliente crea un paquete UDP con destino 255.255.255.255 y requiere también su última dirección IP conocida, aunque esto no es necesario y puede llegar a ser ignorado por el servidor.

2 DHCP Offer: el servidor determina la configuración basándose en la dirección del soporte físico de la computadora cliente. El servidor especifica la dirección IP.

3 DHCP Request: El cliente selecciona la configuración de los paquetes recibidos de DHCP Offer. Una vez más, el cliente solicita la dirección IP específica que le indicó el servidor.

4 DHCP Acknowledge: El servidor confirma el pedido y lo publica masivamente en la subred. Se espera que el cliente configure su interface de red con las opciones que se le han otorgado.



Otros mensajes son:

DHCP Release: los clientes envían una petición al servidor DHCP para liberar su dirección DHCP.
DHCP Nak: El servidor envía al cliente un mensaje indicando que el contrato ha terminado o que la dirección IP asignada no es válida.
DHCP Inform: El cliente envía una petición al servidor de DHCP, para solicitar más información que la que el servidor ha enviado con el DHCPACK original; o para repetir los datos para un uso particular.

A continuación haremos un análisis y configuración de un servidor DHCP (**dhcpd**) para nuestro dispositivo Debian. Lo primero que hay que hacer es instalar el dhcpd, que se hace con un simple:

```
apt-get install dhcpd
```

Una vez terminada la instalación, editamos el archivo de configuración */etc/dhcpd.conf* en el que introducimos los parámetros de nuestro servidor dhcp. Por ejemplo en nuestro caso queda tal que así:

```
subnet 192.168.1.0 netmask 255.255.255.0 {

option domain-name "dispositivodebian.org";
option domain-name-servers 80.58.61.250;
option subnet-mask 255.255.255.0;
default-lease-time 3600;
max-lease-time 7200;
range dynamic-bootp 192.168.1.60 192.168.1.80;
option routers 192.168.1.1;
option broadcast-address 192.168.1.255;
}
```

Este servidor DHCP ofrecerá a los clientes que se conecten a la red una IP en el rango 192.168.1.60–192.168.1.80. Retiene la dirección durante 3600 segundos si el cliente no solicita un intervalo de tiempo específico. Así mismo, el tiempo máximo permitido para retener la dirección será de 7200 segundos. Como nombre de dominio especificamos “dispositivodebian.org”, 192.168.1.1 como puerta de enlace, 255.255.255.0 como máscara de subred al ser direcciones de clase C y 80.58.61.250 como servidor DNS.

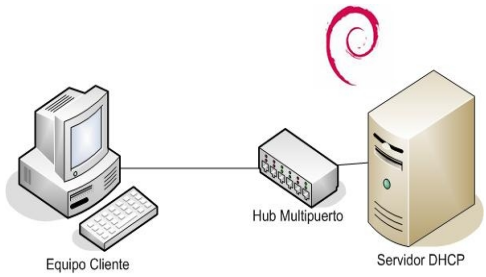
Con esto ya sólo nos resta reiniciar el servidor dhcp mediante:

```
/etc/init.d/dhcpd restart
```

Y ya tenemos el servidor configurado y funcionando. Vamos a hacer una prueba de DHCP en un escenario como el siguiente, para ello en el equipo cliente levantamos la interfaz de red:

```
root@fran-desktop:/etc/network# ifup eth1
Internet Systems Consortium DHCP Client V3.0.3
Copyright 2004-2005 Internet Systems Consortium.
All rights reserved.
For info, please visit http://www.isc.org/products/DHCP

Listening on LPF/eth0/00:11:d8:42:de:86
Sending on   LPF/eth0/00:11:d8:42:de:86
Sending on   Socket/fallback
DHCPDISCOVER on eth0 to 255.255.255.255 port 67 interval 8
DHCPOFFER from 192.168.1.11
DHCPREQUEST on eth0 to 255.255.255.255 port 67
DHCPACK from 192.168.1.11
bound to 192.168.1.60 -- renewal in 1572 seconds.
```



También podemos asignar direcciones IP estáticas basadas en la MAC del cliente, como se muestra en el siguiente ejemplo.

```
host clientedebian {
hardware ethernet 00:11:D8:42:DE:86;
fixed-address 192.168.1.90;
}
```

Pueden conocerse las MAC de las máquinas conectadas a la red ejecutando el comando **arp** desde consola.

2.3.2.3 Servidor Proxy DNS

El sistema global de **DNS** se encarga de traducir los nombres de dominio como por ejemplo www.debian.org a una dirección IP como por ejemplo *194.109.137.218.*

En los inicios de Internet, todos los nombres existentes estaban en un archivo centralizado conocido como HOSTS.TXT. Este archivo contenía los nombres de todos los dominios conocidos. Cuando el archivo HOSTS.TXT creció a un tamaño imposible de manejar, surgió la necesidad de crear un nuevo sistema escalable para manejar los nombres de dominio. Es por esto que, en 1983, surgió la primera especificación del nuevo sistema de nombres de dominio en el RFC833.

Para la operación práctica del sistema DNS se utilizan tres componentes principales:

* Los clientes DNS (resolvers), un programa cliente DNS que se ejecuta en la computadora del usuario y que genera peticiones DNS de resolución de nombres a un servicio DNS (*Por ejemplo: ¿qué dirección IP corresponde a nombre.dominio?*)

* Los servidores DNS (*name servers*), que contestan las peticiones de los clientes, los servidores recursivos tienen la capacidad de reenviar la petición a otro servidor si no disponen de la dirección solicitada;

* Y las zonas de autoridad, porciones del espacio de nombres de dominio que almacenan los datos. Cada zona de autoridad abarca al menos un dominio y posiblemente sus subdominios, si estos últimos no son delegados a otras zonas de autoridad.

El DNS consiste en un conjunto jerárquico de servidores DNS. Cada dominio o subdominio tiene una o más zonas de autoridad que publican la información acerca del dominio y los nombres de servicios de cualquier dominio incluido. La jerarquía de las zonas de autoridad coincide con la jerarquía de los dominios. Al inicio de esa jerarquía se encuentran los servidores raíz: los servidores que responden cuando se busca resolver un dominio de primer y segundo nivel.

Bind es el servidor DNS más popular en entornos Linux. BIND (acrónimo de Berkeley Internet Name Domain) es una implementación del protocolo DNS y provee una implementación libre de los principales componentes del Sistema de Nombres de Dominio. Otros tipos de servidores DNS son: PowerDNS, MaraDNS, djbdns ó MyDNS (Que utiliza MySQL para almacenar los registros) y pdnsd.

pdnsd es el que analizaremos y utilizaremos en nuestro dispositivo Debian. Pdnsd es un proxy DNS caché en disco. Si nuestro dispositivo lo incorpora podemos hacer de servidor DNS para el resto de la red, además evitaría tener que resolver la dirección haciendo uso de los recursos DNS de Internet cada vez que una dirección fuera solicitada, ya que lo haría sólo la primera vez y el resto se serviría desde el disco duro donde estuviera cacheado. Además permite tener varios servidores DNS (de distintos proveedores, servicios, etc..) que pueden ser sondeados para obtener los datos, evitando puntos único de fallo, etc.
Para instalarlo teclearmos lo siguiente:

```
apt-get install pdnsd
```

Mientras que para ejecutarlo escribiremos:

```
pdnsd --daemon -p /var/run/pdnsd.pid
```

Donde --daemon indica que el proceso se ejecutará como demonio en *background.* La opción -p escribe el pid del proceso en el fichero indicado.

Podemos ver más opciones de argumentos en línea de ejecución con:

```
pdnsd --help
```

El archivo de configuración de pdnsd es */etc/pdnsd.conf* . Dicho archivo está dividido en secciones:

La sección **global { }** define parámetros que afectan a todo el comportamiendo del proceso. Los más importantes:

server_ip: definimos la dirección IP donde pdnsd escuchará las peticiones de los clientes. También es posible hacerlo mediante: interface=IP.
server_port: por defecto el 53.
cache_dir: el directorio donde se almacenará la cache. Por defecto es */var/cache/pdnsd*

La sección **server { }** especifica servidores de nombres que pdnsd utilizará para recabar información, en orden de aparación, si el primero falla se preguntará al siguiente, y así sucesivamente. Las opciones destacables aquí son:

ip: la ip del servidor.
uptest: determina el método que se utiliza para comprobar que el servidor está disponible. (por ejemplo: ping)
ping_timeout: el tiempo en milisegundos que considera un ping fallido.
interval: especifica cada cuánto tiempo se realiza una prueba de test al servidor.
purge_cache: si está en off los datos de la caché sólo se limpiarán cuando la memoria reservada para ello esté llena.

La sección **rr { }** especifica resultados dns que están almacenados localmente sin necesidad de preguntar a un servidor remoto (el del ISP por ejemplo). Util para darle nombres DNS a equipos de nuestra red LAN.

La sección **neg { }** nos permite especificar nombres dns que serán denegados por pdnsd.

La sección **source { }** le permite a pdnsd leer de un archivo con formato como */etc/hosts* nombres y direcciones que podrán ser consultadas por los clientes. Es una forma más fácil de definir nombres de equipos de nuestra LAN que utilizando rr { }. Por defecto file = “/etc/hosts”. Nota: los nombres son interpretados en la notación absoluta (nombres.xxx)

A modo de ejemplo listamos el archivo de configuración de pdnsd en nuestro caso:

```
global {
        perm_cache=512;
        cache_dir="/var/cache/pdnsd";
        max_ttl=604800;
        run_as="pdnsd";
        paranoid=on;
        status_ctl=on;
        server_port=53;
        server_ip="192.168.0.1";
}

source {
        ttl=86400;
        owner="localhost.";
        serve_aliases=on;
        file="/etc/hosts";
}

server {
        ip="80.58.61.250";
        timeout=30;
        interval=30;
        uptest=ping;
        ping_timeout=50;
        purge_cache=off;
}
```

Y con esto ya tenemos funcionando nuestro servidor DNS para el resto de los equipos de la red.

Por últmo, deberíamos volver a configurar nuestro servidor DHCP modificando el fichero */etc/dhcpd.conf* y decirle que nuestro servidor DNS es ahora 192.168.0.1.

2.3.3 Configuración de Linux como Router

2.3.3.1 Enrutamiento de Paquetes en Debian

Linux, además de los servicios que hemos visto hasta ahora, puede ejercer otras actividades, algunas de ellas sin necesidad de software añadido. Únicamente apoyándose en el kernel. Una de estas actividades es el routing o enrutamiento. Así pues, podemos convertir una de nuestras máquinas más antiguas en un poderoso router multifunción con la ayuda de nuestro sistema operativo.

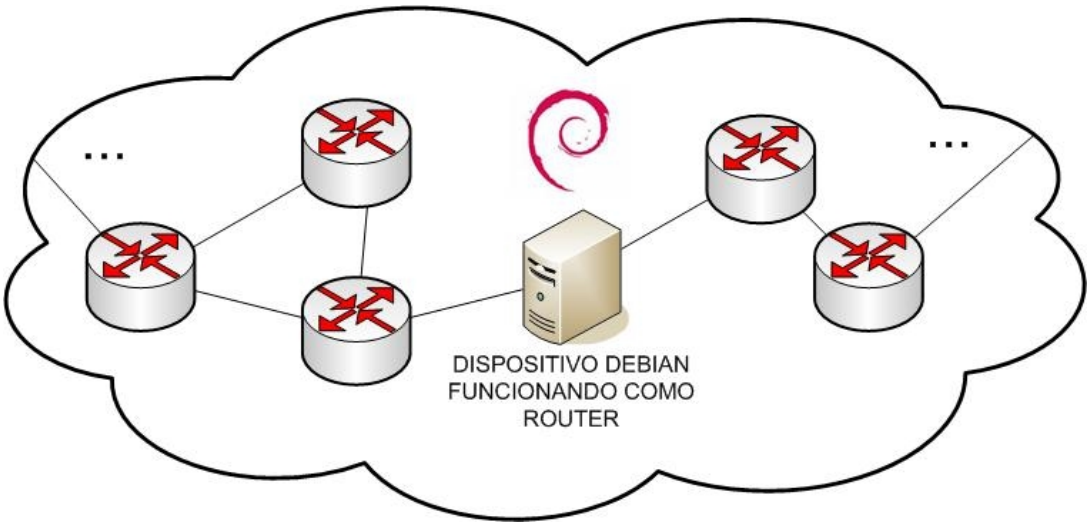
En este apartado veremos cómo realizar tareas de enrutamiento en GNU/Linux (para ser más exactos, enrutamiento estático). Para ello, contaremos con un sistema que dispondrá de dos interfaces de red, ambas ethernet. Una de las interfaces conectará nuestro router a la LAN interna, mientras que la otra interfaz nos conectará a la red WAN. Nuestro objetivo será interconectar las dos redes a las que pertenecen cada uno de los dispositivos, de forma que puedan comunicarse entre sí equipos pertenecientes a cada una de ellas.

Lo primero que vamos a hacer es definir el significado de **router**. Un router o encaminador es un dispositivo de interconexión de redes de ordenadores que opera en la capa 3 (nivel de red) del modelo OSI. Este dispositivo interconecta segmentos de red o redes enteras. Hacen pasar paquetes de datos entre redes tomando como base la información de la capa de red.

Algunos fabricantes de Routers son las empresas: Cisco, Lucent o Nortel Networks. Un Router **Cisco** también utiliza un sistema operativo (Cisco IOS) con sus comandos especiales al igual que Unix o Windows, emplea diferentes paquetes (Firewall, NAT) de la misma forma que cualquier computadora, y obviamente existen diferentes modelos dependiendo de su uso; solo que el funcionamiento de un producto **Router** como tal, es únicamente transmitir información para que los datos que viajan a través de una red lo hagan de la manera más eficiente posible y Cisco ha sido una de las empresas que mejor lo ha hecho para la Red de Redes **"Internet"** .

La configuración de red Linux utiliza el mismo protocolo que Internet (TCP/IP), lo que hace que el hecho de dar servicios de Internet (WEB, FTP, email) e interconectar máquinas Linux en una red de área local (LAN) no suponga prácticamente ningún esfuerzo, ya que en el proceso no se necesitan máquinas traductoras. Además, este hecho de realizar encaminamiento por software (sistema operativo con esta capacidad) presenta las ventajas de mayor rapidez, fiabilidad y sencillez en la configuración.

Montar nuestro propio router con viejas piezas de repuesto tiene algunas ventajas frente a comprar un router hardware comercial. Además de ser mucho más económico, otra clara ventaja es el control y flexibilidad sobre la conexión. Para una red pequeña, un servidor utilizando Linux es capaz de encaminar toda la información en la red. Pero si se tiene una Red de cómputo compleja, en un entorno empresarial es recomendable utilizar un producto especializado para esta labor.



En primer lugar necesitaremos activar el reenvío de paquetes. El kernel permite modificar determinados parámetros y variables dinámicamente a través del sistema de archivos **/proc.** Para poder cambiar estos valores tenemos que tener el kernel compilado con la opción CONFIG_SYSCTL, pero lo más probable es que sí lo tengamos ya que viene marcada por defecto. Para activar el reenvio de paquetes haremos:

```
echo “1” > /proc/sys/net/ipv4/ip_forward
```

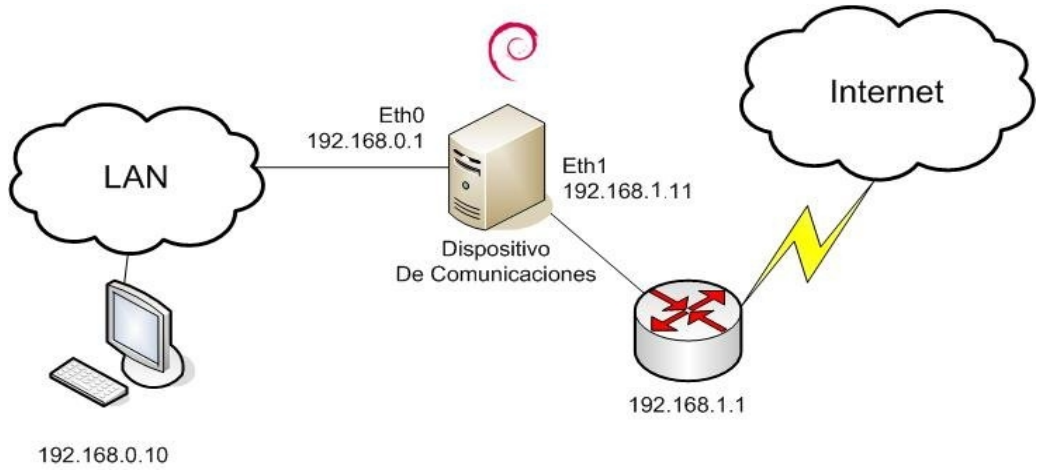
Esto es indispensable para que la máquina actúe como enrutador. También indicar que en Debian hay scripts que asignan estos valores de forma automatica en el arranque (*/etc/sysctl.conf* y */etc/network/options*). Por tanto, en el fichero */etc/network/options* escribiremos:

```
ip_forward = yes
```

Sin embargo, en las recientes versiones de Debian Etch, el archivo /etc/network/options ha dejado de usarse, y en su defecto se utilizaría el archivo */etc/sysctl.conf* y más en concreto la línea *ip_forward*=yes en */etc/network/options* se sustituiría en */etc/sysctl.conf* por la línea:

```
net.ipv4.ip_forward = 1
```

También es necesario asignar una dirección IP distinta a cada interfaz de red de cada dispositivo. En nuestro caso hemos utilizado las direcciones: 192.168.1.11 y 192.168.0.1, tal como vemos en el siguiente gráfico:



Después de esto, para que nuestra máquina Linux ejerza como un router únicamente necesitaremos definir nuestra tabla de enrutamiento. Podemos hacer esto mediante el comando **route**. Para añadir rutas lo haremos con el parámetro 'add' (route add) mientras que para borrarlas lo haremos con 'del' (route del).

En nuestro caso, bastaría con ejecutar estos comandos:

```
route add -net 192.168.0.0 netmask 255.255.255.0 dev eth0
route add -net 192.168.1.0 netmask 255.255.255.0 dev eth1
```

Si consultamos ahora nuestra tabla de enrutamiento, obtendremos el siguiente resultado:

```
debian:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.1.0     *               255.255.255.0   U     0      0        0 eth1
192.168.0.0     *               255.255.255.0   U     0      0        0 eth0
```

A partir de este momento, cualquier máquina que tuviese configurada la dirección de nuestro router como su puerta de enlace, podría establecer comunicación con una máquina de la otra red, aunque ambas máquinas se encuentren en redes diferentes (he aquí la función del router cumplida, interconectar dos o más redes).

Para añadir conexión a Internet desde la red 192.168.0.X haríamos lo siguiente:

```
route add default gateway 192.168.1.1
```

Y la tabla de enrutamiento quedaría como:

```
debian:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.1.0     *               255.255.255.0   U     0      0        0 eth1
192.168.0.0     *               255.255.255.0   U     0      0        0 eth0
default         192.168.1.1     0.0.0.0         UG    0      0        0 eth1
```

Observamos que se ha añadido una nueva ruta a la tabla, que además se ha situado como la puerta de enlace por defecto. ¿Qué significa esto? A partir de este momento, cualquier paquete dirigido a una dirección de las redes conocidas se enviará a través de la interfaz que sabemos está asociada a dicha red. Sin embargo, cuando nos lleguen peticiones a cualquier otra dirección que no pertenezca a estas redes, la enviaremos a la máquina 192.168.1.1 (gateway por defecto), que conocerá a qué red pertenece dicha dirección, o de lo contrario pasará el paquete a otro router que sabrá más acerca de la red destino.

Podemos considerar al router, por tanto, como un punto que conoce hacia dónde deben ir dirigidos los paquetes para que alcancen su destino.

Linux también puede ejercer como un **enrutador dinámico**. El enrutamiento dinámico soluciona el problema de la gestión manual del mismo: en el caso de que se “rompa” una ruta no es necesario reconfigurar manualmente las rutas en cada host. Los protocolos de enrutamiento dinámico se encargan de buscar la ruta adecuada a cada destino.

Existen varias aplicaciones en Linux para realizar rutado dinámico, entre las que destacan:

- **Routed:** routed es uno de los paquetes de rutado standard disponibles en Linux. Soporta RIP únicamente (uno de los protocolos de rutado más viejos todavía en servicio). El RIP es muy sencillo, los routers simplemente hacen un broadcast de sus tablas de rutado a los routers vecinos, lo cual da como resultado una tabla completa de rutado que tiene entradas para cada destino en Internet. Este método es fundamentalmente inseguro, y muy ineficiente más allá de pequeñas redes seguras (en cuyo caso probablemente no sea necesario). Asegurarlo no es posible, se puede configurar un cortafuegos en los puertos 520 y 521, que son los que utiliza RIP para transferir, sin embargo, puede dar como resultado rutas a través de las cuales preferiríamos no atravesar, y además los atacantes pueden seguir falsificando las rutas.

- **Gated:** gated es un software de rutado más avanzado que routed. Soporta versiones de RIP 1 y 2, DCN HELLO, OSPF Versión 2, EGP versión 2, BGP versiones 2 a 4. Actualmente, el protocolo BGP (Border Gateway Protocol) va ganando en popularidad respecto al EGP, y el OSPF (OSPF tiene seguridad incorporada, es muy eficiente y bastante más complicado) sobre el RIP.

- **Zebra:** zebra tiene bastantes maś características que gated, y ostenta una bonita linea de interfaz de comandos al estilo de Cisco. Se ejecuta como un demonio, multi-hilo para el rendimiento, cada protocolo (RIP, OSPF, etc.) tiene su propia configuración, y se pueden ejecutar múltiples protocolos a la vez (aunque podría originar confusiones/problemas). Existe un puerto de configuraciones maestro, y un puerto para cada protocolo.

Como hemos podido ver, las capacidades de Linux en lo que a rutado se refiere son muy amplias, llegando a superar en muchos casos las funcionalidades de los dispositivos dedicados de alto precio.

2.3.3.2 Puentes de Red

Un puente o Bridge es un dispositivo de interconexión de redes/ordenadores que opera en la capa 2 (nivel de enlace de datos) del modelo OSI. Éste interconecta dos segmentos de red (o divide una red en dos segmentos) encargándose del paso de datos de una red a otra, con base en la dirección física de destino de cada paquete. Funciona a través de una tabla de direcciones MAC detectadas en cada segmento que está conectado. Cuando detecta que un nodo de uno de los segmentos está intentando transmitir datos a un nodo del otro, el bridge o puente de red copia el frame para la otra subred. Debido a este mecanismo de aprendizaje automático, los bridges no precisan de configuración manual. Los switches que solemos utilizar en nuestras redes utilizan esta técnica (no así los hubs o concentradores).

Con Linux podemos crear nuestro propio puente de red de una forma sencilla. Así pues, podemos establecer una máquina con Linux de forma que comunique dos redes a un nivel por debajo de la capa de red (en la capa de enlace).

Su puesta en marcha, como hemos comentado ya, es sencilla. Lo primero que debemos hacer es instalar el paquete 'bridge-utils', que nos proporcionará las herramientas necesarias para configurar nuestro puente de red.

En Debian basta ejecutar:

```
apt-get install bridge-utils
```

Ahora nos pondremos a trabajar sobre la configuración. Al igual que hicimos con la configuración de dispositivos Ethernet, la definición del bridge se hará en el archivo */etc/network/interfaces*. En nuestro caso vamos a crear un puente entre la red eth1 y eth0. Para ello, el archivo de configuración quedaría:

```
iface br0 inet static
address 192.168.0.100
netmask 255.255.255.0
bridge_ports eth0 eth1
bridge_stp off
```

Si ahora ejecutamos:

```
ifup br0
```

A partir de este momento, una máquina perteneciente a la red eth1 podría contactar con una de la red eth0, sin ningún tipo de enrutamiento, ya que el bridge se encargaría de gestionarlo todo en el nivel de enlace (capa 2 del modelo OSI). Como nuestro dispositivo funciona en capa 2, para que se puedan ver dos máquinas, ambas deberán estar en la misma subred IP.

En la siguiente captura de ifconfig vemos cómo las dos interfaces eth0 y eth1 ya no tienen asginadas direcciones IP:

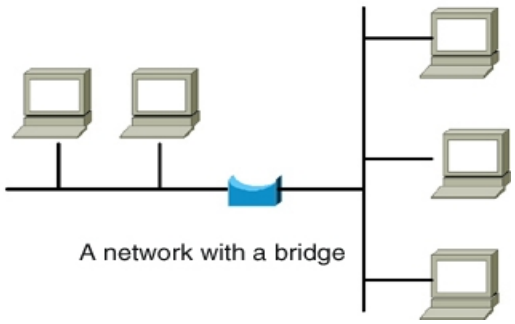
```
br0       Link encap:Ethernet  HWaddr 00:00:1A:00:03:E2
          inet addr:192.168.1.100  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::200:1aff:fe00:3e2/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:57 errors:0 dropped:0 overruns:0 frame:0
          TX packets:49 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:4999 (4.8 KiB)  TX bytes:5347 (5.2 KiB)

eth0      Link encap:Ethernet  HWaddr 00:80:C8:0C:78:C0
          inet6 addr: fe80::280:c8ff:fe0c:78c0/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
```

```
        TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
        Interrupt:12 Base address:0xe400

eth1      Link encap:Ethernet  HWaddr 00:00:1A:00:03:E2
        inet6 addr: fe80::200:1aff:fe00:3e2/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:2127 errors:0 dropped:0 overruns:0 frame:0
        TX packets:2103 errors:0 dropped:0 overruns:0 carrier:0
        collisions:8 txqueuelen:1000
        RX bytes:184557 (180.2 KiB)  TX bytes:231840 (226.4 KiB)
        Interrupt:5 Base address:0xe800
```

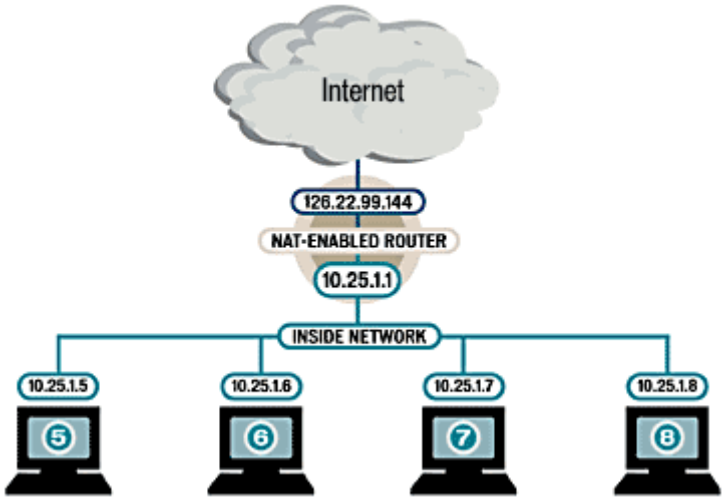
En lugar de ello tenemos una única dirección IP (192.168.1.100) que es accesible desde las dos interfaces.



2.3.3.3 Traducción de Direcciones de Red (NAT)

Normalmente, los paquetes viajan en una red desde su origen a su destino, a través de varios enlaces diferentes. Ninguno de estos enlaces altera realmente el paquete: únicamente lo envían un paso adelante. Si uno de estos enlaces hiciese NAT (Network Address Translation), podría alterar el origen o el destino del paquete según pasase a través de él. Cuando un enlace o nodo modifica un paquete, recuerda dichos cambios, de forma que cuando le llega el paquete de respuesta invierte de nuevo las modificaciones.

La traducción de direcciones de red, como su mismo nombre nos indica, es un mecanismo por el cual un determinado nodo realiza modificaciones en la cabecera de un paquete modificando o la dirección de origen o de destino con un propósito específico.



Las razones principales para usar NAT son las siguientes:

- **Compartir una conexión a Internet**: Muchos de los proveedores de servicios de Internet actuales nos proporcionan únicamente una dirección IP al contratar sus servicios. Podemos enviar paquetes desde la dirección privada que queramos, pero sólo obtendremos respuesta a los paquetes de esa dirección IP pública de origen. Si queremos conectar varias máquinas diferentes a Internet haciendo uso únicamente de una dirección IP pública necesitaremos utilizar NAT, más concretamente, source NAT (ya que realmente se modifica la dirección de origen de los paquetes).
  De esta forma, estamos protegiendo nuestras máquinas, ya que en este caso no se encuentran expuestas directamente a Internet, y un posible ataque contra ellas es más complejo.

- **Varios servidores:** otra utilidad muy interesante es la posibilidad de que se alcance desde el exterior de nuestra red alguna máquina interna. Esto suele ser necesario por el mismo motivo que el caso anterior: quizá sólo dispongamos de una IP pública, pero necesitaremos que desde fuera se puedan establecer conexiones al interior. Rescribiendo el destino de los paquetes entrantes podemos conseguirlo. Este tipo de reenvío también es conocido como *port-forwarding*.
  Una variante de esto sería el balanceo de carga, en la cual se toma un cierto número de máquinas, repartiendo los paquetes entre ellas.

- **Proxy transparente:** en algunas ocasiones necesitaremos que un cierto tipo de peticiones se redirijan hacia una aplicación concreta, que tomará cuentas de ellas. Un ejemplo serían los proxies. Imaginemos que deseamos implantar un proxy Web en nuestra red local, pero no deseamos tener que configurar las máquinas de todos nuestros usuarios. Lo mejor en este caso sería hacer uso de un proxy transparente.
  Para hacer esto, haremos uso de NAT, de forma que todas las peticiones Web que lleguen a la puerta de enlace, sean redirigidas a otra máquina, al puerto 3128 (donde escucha squid por defecto). De esta forma habremos instalado un proxy en nuestra red sin que ello afecte a nuestros usuarios.

Podemos diferenciar dos tipos de NAT: Source NAT (SNAT) y Destination NAT (DNAT).

- **Source NAT:** ocurre cuando alteramos el origen del primer paquete, es decir, modificamos el lugar de donde viene la conexión. El Source NAT siempre se hace después del encaminamiento, justo antes de que el paquete salga por el cable. El enmascaramiento (usado para compartir una conexión a Internet) es una forma especializada de SNAT.

- **Destination NAT:** se da cuando alteramos la dirección de destino del primer paquete, es decir, cambiamos la dirección a donde se dirige la conexión. DNAT siempre se hace antes del encaminamiento, cuando el paquete entra por el cable. El port-forwading, el balanceo de carga y el proxy transparente son formas de DNAT.

Por otro lado, existe otra clasificación de NAT: estático y dinámico:

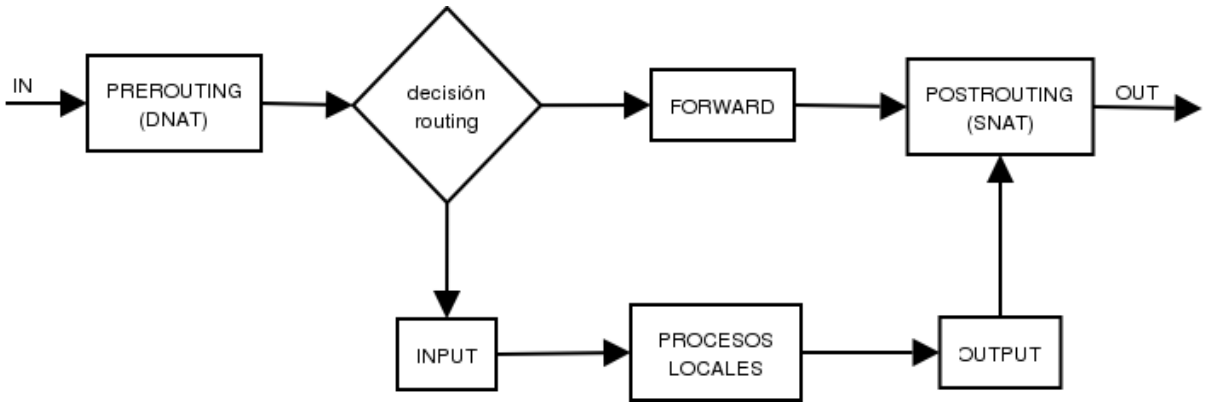
- **NAT Estático:** realiza un mapeo en el que una dirección IP privada se traduce a una correspondiente dirección IP pública de forma unívoca. Normalmente se utiliza cuando un dispositivo necesita ser accesible desde fuera de la red privada.

- **NAT Dinámico:** una dirección IP privada se traduce a un grupo de direcciones públicas. Por ejemplo, si un dispositivo posee la IP 192.168.10.10 puede tomar direcciones de un rango entre la IP 200.85.67.44 y 200.85.67.99. Implementando esta forma de NAT se genera automáticamente un firewall entre la red pública y privada, ya que sólo se permite la conexión que se origina desde la red privada.

- **Sobrecarga:** la forma más utilizada de NAT proviene del NAT dinámico, ya que toma multiples direcciones IP privadas y las traduce a una única dirección IP pública utilizando diferentes puertos. Esto se conoce también como **PAT (Port Address Translation – Traducción de Direcciones por Puerto)**, NAT de única dirección o NAT multiplex pública. Así se evita los conflictos de direcciones entre las distintas redes.

Para hacer uso de NAT bajo Linux necesitaremos habilitar el soporte correspondiente en nuestro núcleo e instalar la herramienta **iptables**, que será la encargada de gestionar nuestras reglas de NAT. Aunque analizaremos en la siguiente sección **Netfilter** e **Iptables** en detalle, para este apartado introduciremos mínimamente algunas reglas con iptables para la realización de NAT. Necesitaremos crear reglas NAT que le digan al núcleo qué conexiones cambiar, y cómo hacerlo.

La tabla de reglas NAT contiene tres listas llamadas cadenas: cada regla se examina por orden hasta que coincide. Las tres cadenas se llaman PREROUTING (Para *Destination NAT*, según los paquetes entran), POSTROUTING (Para *Source NAT,* según los paquetes salen), y OUTPUT (para *Destination NAT* con paquetes generados desde la propia máquina).



En cada uno de los puntos anteriores, cuando un paquete pasa miramos la conexión a la que está asociado. Si es una conexión nueva, comprobamos la cadena correspondiente en la tabla de NAT para ver qué hacer con ella. La respuesta que obtenemos se aplicará a cualquier paquete posterior de esa conexión.

La opción más importante de iptables que mostraremos en esta sección es la opción de selección de tabla, **-t**. Para todas las operaciones de NAT, usaremos *-t nat* para la tabla NAT. Otra opción importante es -A para añadir una nueva regla al final de la cadena o -I para insertarla al principio. Podemos especificar el origen (-s o –source) y el destino (-d o –destination) de los paquetes sobre los que se quiere hacer NAT. Estas opciones pueden ir seguidas por una IP sencilla (192.168.1.1), un nombre (www.debian.org), o una dirección de red (192.168.1.0/24 ó 192.168.1.0/255.255.255.0). Se puede especificar también por qué interfaz de entrada o salida mirar (-i ó –in-interface -o o –out-interface), pero lo que puede especificar depende de en qué cadena se esté poniendo la regla: en PREROUTING sólo se puede elegir la interfaz de entrada, y en POSTROUTING y OUTPUT sólo la de salida.

Una vez que sabemos cómo elegir los paquetes que queremos modificar, para completar nuestra regla, necesitaremos decirle al núcleo exactamente qué queremos que haga con los paquetes.

SOURCE NAT: cambiar la dirección de origen de las conexiones a algo diferente. Esto se hace en la cadena POSTROUTING, justo antes de que sea enviado. Este es un detalle importante, ya que significa que cualquier otro servicio de la máquina Linux (encaminamiento, filtrado de paquetes) verá el paquete sin cambiar. El source nat se especifica indicando -j SNAT, y la opción --to-source especifica una dirección IP, un rango de direcciones IP, y un puerto o rango de puertos opcionales (sólo con los protocolos UDP y TCP).

Veamos algunos ejemplos:

```
## Cambiar la dirección de origen por 1.2.3.4
iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to 1.2.3.4

## Cambiar la dirección de origen a 1.2.3.4, 1.2.3.5 o 1.2.3.6
iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to 1.2.3.4-1.2.3.6

## Cambiar la dirección de origen por 1.2.3.4, puertos 1-1023
iptables -t nat -A POSTROUTING -p tcp -o eth0 -j SNAT --to 1.2.3.4:1-1023
```

ENMASCARAMIENTO: Hay un caso especializado de ***Source NAT*** denominado enmascaramiento (masquerading): sólo debería ser usado en direcciones IP asignadas de forma dinámica. No es necesario escribir la dirección de origen de forma explícita con el enmascaramiento: utilizará la dirección origen de la interfaz por la que el paquete está saliendo. Pero lo más importante aún, si el enlace cae, las conexiones (que se iban a perder de todas maneras) se olvidan, lo que signfica que evita problemas cuando la conexión vuelva a la normalidad con una IP diferente.

Este será el tipo de NAT que utilizaremos en nuestro dispositivo para paquetes salientes.

Para habilitar el enmascaramiento escribiremos:

```
iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
```

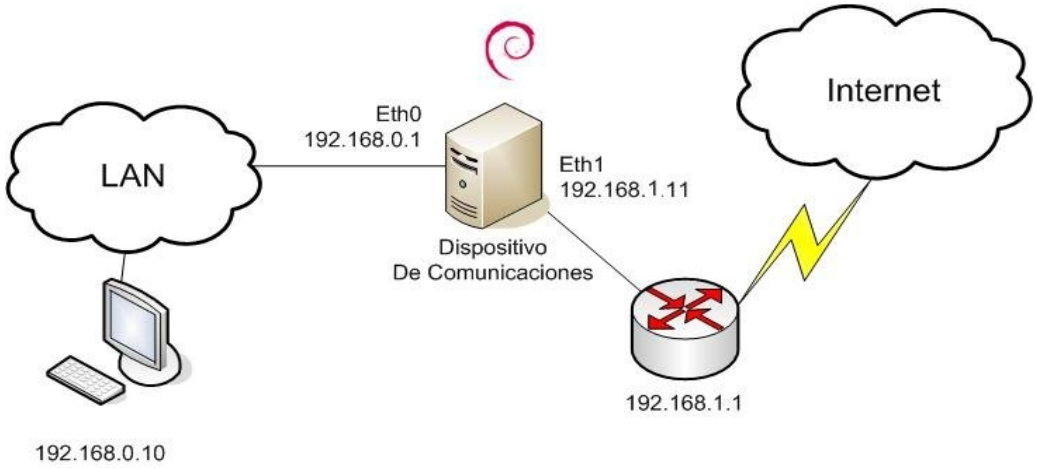
Comprobamos que se ha establecido la regla mediante:

```
debian:~# iptables -t nat -n -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
MASQUERADE  all  --  0.0.0.0/0            0.0.0.0/0

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Una vez realizado esto todos los paquetes que salgan por Eth1 en nuestro dispositivo tendrán como IP origen 192.168.1.11, y un puerto origen asignado por el dispositivo. De esta manera la estación de trabajo en la que estamos (192.168.0.10) podrá acceder a equipos en la red 192.168.1.0 como si fuera 192.168.1.11 y también podrá realizar conexiones salientes a Internet.



DESTINATION NAT: cambio de destino. Se hace en la cadena PREROUTING, según entra el paquete; esto significa que cualquier otro servicio de la máquina con Linux (encaminamiento, filtrado de paquetes) verá el paquete yendo a su destino real (el definitivo). Esto significa que se puede utilizar la opción -i (interfaz de entrada). Para alterar el destino de un paquete generado de forma local (en la máquina que hace el NAT), se debe usar OUTPUT, pero esto es más inusual. Destination NAT se especifica utilizando -j DNAT, y la opción --to-destination especifica una dirección IP, un rango de direcciones IP, y un puerto o rango de puertos opcionales (sólo para los protocolos UDP y TCP).

Veamos algunos ejemplos:

```
## Cambia la dirección de destino por 5.6.7.8
iptables -t nat -A PREROUTING -i eth1 -j DNAT --to 5.6.7.8

## Cambia la dirección de destino por 5.6.7.8, 5.6.7.9 o 5.6.7.10.
iptables -t nat -A PREROUTING -i eth1 -j DNAT --to 5.6.7.8-5.6.7.10
```

```
## Cambia la dirección de destino del tráfico web por 5.6.7.8, puerto 8080.
iptables -t nat -A PREROUTING -p tcp --dport 80 -i eth1 -j DNAT --to 5.6.7.8:8080

## Redirige los paquetes locales que van a 1.2.3.4 hacia el dispositivo loopback.
iptables -t nat -A OUTPUT -d 1.2.3.4 -j DNAT --to 127.0.0.1
```

Volviendo a nuestro esquema, lo más común es querer "redireccionar" algún puerto en nuestro dispositivo Debian porque tenemos algún servidor en la red 192.168.0.0. Si quisiéramos abrir el puerto 22 para controlar por SSH (Hablaremos detenidamente sobre Secure Shelll en el Apartado 2.4.4) tendríamos que ejecutar la siguiente regla:

```
iptables -t nat -A PREROUTING -p tcp --dport 22 -i eth1 -j DNAT --to 192.168.0.10:22
```

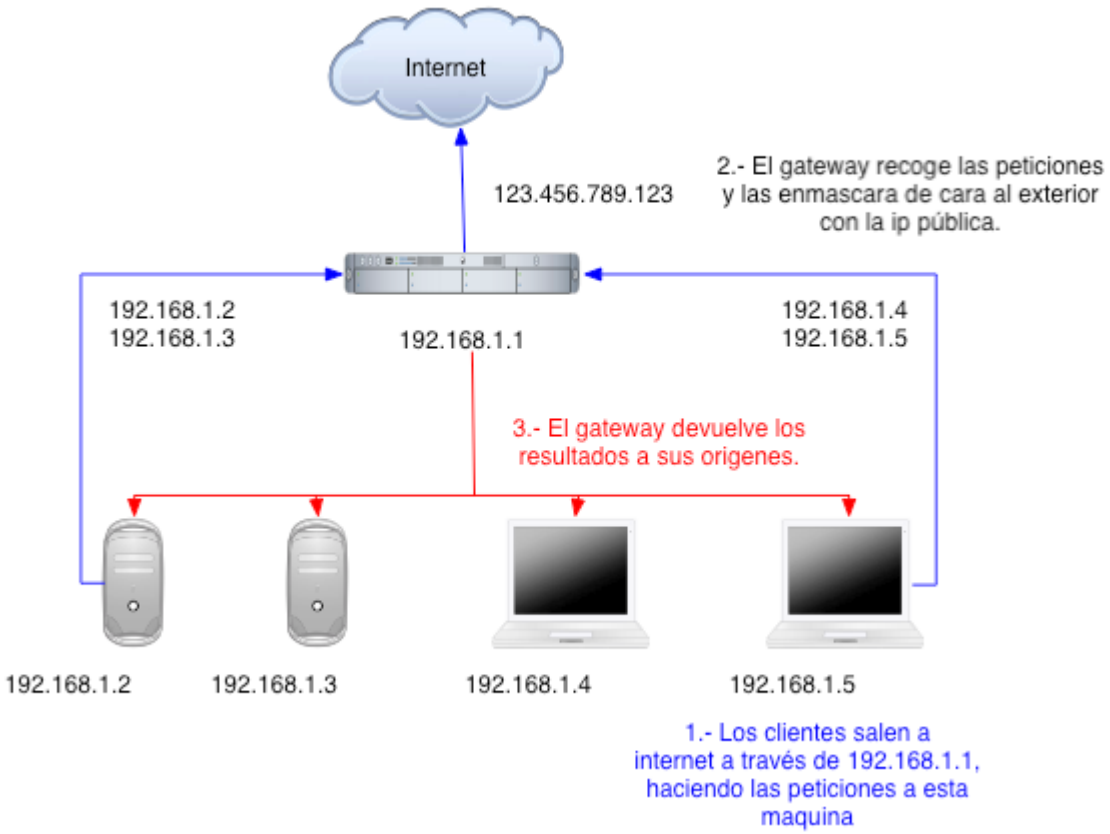
REDIRECCIÓN: hay un caso especializado de *Destination NAT* llamado redirección: es una simple conveniencia que es exactamente lo mismo que hacer DNAT, pero con la dirección de la interfaz de entrada.

Por ejemplo:

```
iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 80 -j REDIRECT --to-port 3128
```

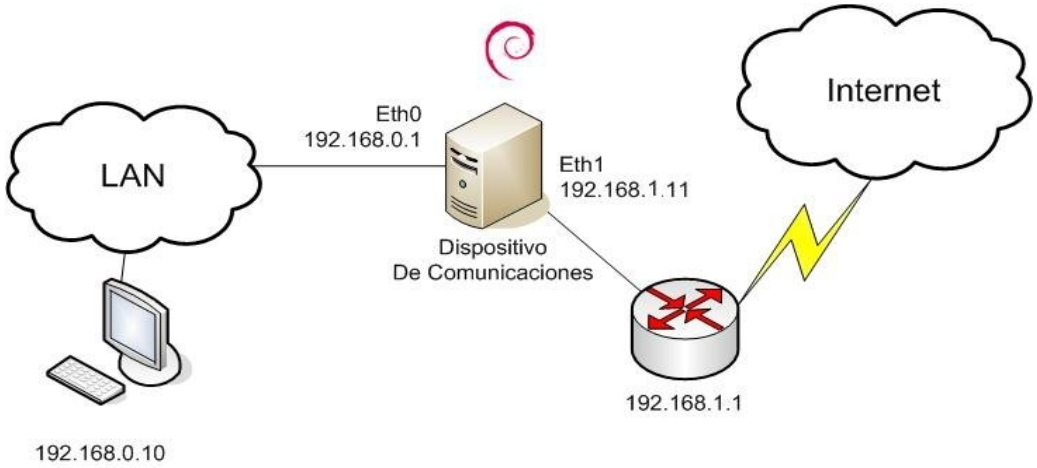
Envía el tráfico que entra dirigido al puerto 80 (web) a un proxy squid (transparente). Veremos también más adelante en qué consiste un Proxy Squid.

No debemos olvidar que todos los cambios que hacemos con iptables desaparecen cuando reiniciamos la máquina; por tanto, debemos crear un script con las reglas de iptables para hacer NAT y que se ejecute automáticamente al incio. Veremos en la siguiente sección cómo realizar dicho script.



Hasta aquí la parte de administración de la red. Llegados a este punto debemos tener nuestro dispositivo Debian enrutando tráfico con un servidor DHCP activado, y un proxy DNS, permitiendo a los equipos que están por detrás de él compartir una conexión a Internet realizando NAT, tanto si pertenecen a la red cableada como a la inalámbrica.

Sin embargo, si volvemos a observar el esquema de conectividad de nuestro dispositivo, vemos que un paquete originado por un PC de la LAN con salida a Internet pasa por dos puntos donde se realiza NAT, (en nuestro dispositivo Debian primero y después en el router que nos proporciona la conexión WAN, que en nuestro caso es tecnología ADSL).



Como hemos comentado, el objetivo principal de hacer NAT es poder compartir una dirección IP pública entre varios ordenadores de una red local que utilizan direcciones privadas. Este objetivo lo cumplimos con el NAT que se realiza en el router WAN, por tanto, no tendría mucho sentido volver a hacer otra traducción de direcciones en nuestro dispositivo Debian, el esquema sería igualmente válido y funcional si hicieramos en nuestra máquina routing de paquetes simplemente.

No obstante, hemos querido desarrollar anteriormente todo el proceso de configuración de NAT para poder utilizar dicha configuración en cualquier escenario, independientemente del dispositivo y tecnología que hubiera a la salida de nuestra máquina Debian. Pero a partir de ahora y en el resto del proyecto no activaremos NAT en nuestro dispositivo ya que no es absolutamente necesario.

Eso sí, para que esto funcione, lo único que habría que hacer es que le tendríamos que decir al router WAN cómo alcanzar la red 192.168.0.0/24, para que sepa dónde tiene que mandar los paquetes que recibe con destino para los ordenadores de esa red. Para ello configuraremos una ruta estática en él. La tabla de enrutamiento de dicho router quedaría tal que así:

```
# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.1.0     *               255.255.255.0   U     0      0        0 br0
192.168.0.0     192.168.1.11    255.255.255.0   UG    0      0        0 br0
default         84.79.128.1     0.0.0.0         UG    0      0        0 nas0
```

Observamos que para llegar a la red 192.168.0.0, envía todos los paquetes a nuestro router (192.168.1.11).

Ya que tenemos resuelta la conectividad y la administración de la red, pasaremos al siguiente punto: dotar de seguridad dichas conexiones.