Administración de red

Remo Suppi Boldrito

PID_00238583

UOC
Universitat Oberta de Catalunya

Índice

Introducción

TCP/IP

V. Cerf *et al.* (1976) en su documento original "Specification of Internet Transmission Control Program" utiliza por primera vez las siglas TCP.

El sistema operativo UNIX (GNU/Linux) se toma como ejemplo de una arquitectura de comunicaciones estándar. Desde el mítico UUCP (servicio de copia entre sistemas operativos UNIX) hasta las redes actuales, UNIX siempre ha mostrado su versatilidad en aspectos relacionados con la comunicación y el intercambio de información. Con la introducción de redes de ordenadores (área local LAN, área amplia WAN o las más actuales, área metropolitana MAN) ofreciendo enlaces multipunto a diferentes velocidades (56kbits/seg hasta 1/10Gbit/seg), han ido surgiendo nuevos servicios basados en protocolos más rápidos, portables entre diferentes ordenadores y mejor adaptados, como el TCP/IP [*transport control program (capa 4 del modelo OSI)/ internet protocol* (capa 3 del modelo OSI)].

Muchos cambios se han producido en los últimos años con el incremento de dispositivos conectados a Internet (que ha tenido un crecimiento exponencial y se han agotado las direcciones de dispositivos IPv4), derivados de la proliferación de teléfonos inteligentes (*smartphones*), tabletas (pero también televisores, consolas de video juegos, coches, etc.), así como las necesidades de ancho de banda que han provocado evoluciones que todavía tardarán algunos años en estabilizarse.

Entre ellas, la más directa es el cambio de protocolo IP que será en su versión 6 (IPv6) y que obligará a cambiar todos los sistemas a los operadores y usuarios para admitir este nuevo protocolo de conexión (y que no es compatible con el anterior si bien pueden coexistir en forma segmentada y a través de puentes de interconexión).

El otro concepto significativo en los entornos de las comunicaciones es el de *Internet de las cosas* ('*Internet of Things*'), que se atribuye a Auto-ID Labs (http://www.autoidlabs.org) con base en el Instituto Tecnológico de Massachusetts, MIT. La idea es muy simple (si bien su aplicación puede resultar compleja): si todos los artículos (ropa, alimentos, libros, coches, neveras, etc.) tuvieran un dispositivo de identificación y fuera posible su seguimiento remoto, la vida cotidiana de los seres humanos del planeta se transformaría. Esto implicaría que cada artículo podría ser identificado y gestionado remotamente con notables avances en determinados campos pero con muchas incertezas, por otro lado. Por ejemplo, ya no tendríamos que pasar todos los productos por un lector de código de barras en un supermercado para luego volverlos a poner en el carro durante la compra, pero al instante se podría tener información sobre los hábitos de consumo de una persona identificada (por ejemplo, por

su tarjeta de crédito durante el pago), terminaríamos con los artículos desaparecidos pero podríamos tener el seguimiento *on line* del mismo o saber qué hace una persona que lleva unos zapatos con conectividad a la red.

La base de IofT está en IPv6 para poder identificar estos objetos (algo que no se puede hacer con IPv4) y poder codificar entre 100 a 100.000 millones de objetos, así como seguir su movimiento. De acuerdo a los datos de las grandes consultoras en el 2020 habrá entre 25.000 y 30.000 millones de dispositivos conectados a Internet y será necesario desarrollar nuevas tecnologías (o hacer evolucionar las actuales), además de Wi-Fi, Bluetooth, RDFI, NFC, HSDPA como por ejemplo ZigBee, UWB, etc. (las llamadas PAN –Personal Area Network–, para lo cual hay un grupo de trabajo IEEE 802.15 especializado en *wireless personal area networks*, WPAN) para que consuman menos energía y la comunicación sea efectiva. [HeMa, LeCe, daCos, Com, Mal, Cis, Gar, KD].

1. Introducción a TCP/IP (TCP/IP suite)

El protocolo TCP/IP sintetiza un ejemplo de estandarización y una voluntad de comunicación a nivel global.

El protocolo TCP/IP es en realidad un conjunto de protocolos básicos que se han ido agregando al principal para satisfacer las diferentes necesidades en la comunicación ordenador-ordenador como son TCP, UDP, IP, ICMP, ARP. [Mal96]

La utilización más frecuente de TCP/IP para el usuario en la actualidad son la conexión remota a otros ordenadores (telnet, SSH[1]), la utilización de ficheros remotos (NFS[2]) o su transferencia (FTP[3], HTTP[4]).

(1)Del inglés *secure shell*.

(2)Del inglés *network file system*.

(3)Del inglés *file transfer protocol*.

(4)Del inglés *hipertext markup protocol*.

1.1. Servicios sobre TCP/IP

Los servicios TCP/IP tradicionales (y que hoy en día continúan existiendo o han evolucionado) más importantes son [Gar98]:

a) Transferencia de archivos: el FTP permite a un usuario de un ordenador obtener/enviar archivos de un ordenador hacia otro ordenador. Para ello, el usuario deberá tener una cuenta, en el ordenador remoto, identificarse a través de su nombre (*login*) y una palabra clave (*password*) o en ordenadores donde existe un repositorio de información (software, documentación...), el usuario se conectará como anónimo (*anonymous*) para transferir (leer) estos archivos a su ordenador. Esto no es lo mismo que los más recientes sistemas de archivos de red, NFS (o protocolos *netbios* sobre tcp/ip), que permiten virtualizar el sistema de archivos de una máquina para que pueda ser accedido en forma interactiva sobre otro ordenador. Hoy en día es un protocolo poco utilizado ya que protocolos como WebDAV sobre http han permitido que la transferencia de archivos se realice de forma más simple y sin aplicaciones específicas más allá de un navegador y un servidor de http.

b) Conexión (*login*) remota: el protocolo de terminal de red (telnet) permite a un usuario conectarse a un ordenador remotamente. El ordenador local se utiliza como terminal del ordenador remoto y todo es ejecutado sobre este permaneciendo el ordenador local invisible desde el punto de vista de la sesión. Este servicio en la actualidad se ha reemplazado por el SHH por razones de seguridad. En una conexión remota mediante telnet, los mensajes circulan tal cual (texto plano), o sea, si alguien "observa" los mensajes en la red, equivaldrá a mirar la pantalla del usuario. SSH codifica la información (que significa un coste añadido a la comunicación), que hace que los paquetes en la red sean

ilegibles a un observador extraño. Con la mejora de las velocidades de red y de los procesadores de los dispositivos no tiene prácticamente impacto sobre la comunicación, en cambio, es extremadamente necesario cuando se envía información confidencial o se desea mantener la privacidad.

Utilización típica de *TCP/IP remote login*

Hoy en día el *telnet* ha sido sustituido por *ssh*, ya que es inseguro:

ssh remote_host

```
GNU/Linux 4.4.0-31-generic x86_64
Last Login: Fri Sep 30 08:00:03 2016 from sys.nteum.org
$
```

c) eMail: este servicio permite enviar mensajes a los usuarios de otros ordenadores. Este modo de comunicación se ha transformado en un elemento vital en la vida de los usuarios y permiten que los *e-mails* (correos electrónicos) sean enviados a un servidor central para que después puedan ser recuperados por medio de programas específicos (clientes) o leídos a través de una conexión web.

El avance de la tecnología y el bajo coste de los ordenadores han permitido que determinados servicios se hayan especializado en ello y se ofrecen configurados sobre determinados ordenadores trabajando en un modelo cliente-servidor. Un servidor es un sistema que ofrece un servicio específico para el resto de la red. Un cliente es otro ordenador que utiliza este servicio. Todos estos servicios generalmente son ofrecidos dentro de TCP/IP:

- **Sistemas de archivos en red** (NFS): permite a un sistema acceder a los archivos sobre un sistema remoto en una forma más integrada que FTP. Los dispositivos de almacenamiento (o parte de ellos) son exportados hacia el sistema que desea acceder y este los puede "ver" como si fueran dispositivos locales. Este protocolo permite a quien exporta poner las reglas y las formas de acceso, lo que (bien configurado) hace independiente el lugar donde se encuentra la información físicamente del sitio donde se "ve" la información.

- **Impresión remota**: permite acceder a impresoras conectadas a otros ordenadores.

- **Ejecución remota**: permite que un usuario ejecute un programa sobre otro ordenador. Existen diferentes maneras de realizar esta ejecución: o bien a través de un comando (*rsh*, ssh, *rexec*) o a través de sistemas con RPC[5] que permiten a un programa en un ordenador local ejecutar una función de un programa sobre otro ordenador. Los mecanismos RPC han sido objeto de estudio y existen diversas implementaciones, pero las más comunes son Xerox's Courier y Sun's RPC (esta última adoptada por la mayoría de los UNIX).

(5)Del inglés *remote procedure call.*

- **Servidores de nombre** (*name servers*): en grandes instalaciones existen un conjunto de datos que necesitan ser centralizados para mejorar su utilización, por ejemplo, nombre de usuarios, palabras clave, direcciones de red, etc. Todo ello facilita que un usuario disponga de una cuenta para todas las máquinas de una organización. Por ejemplo, Sun's Yellow Pages (NIS en las versiones actuales de *Sun*) está diseñado para manejar todo este tipo de datos y se encuentra disponible para la mayoría de UNIX. El DNS[6] es otro servicio de nombres pero que guarda una relación entre el nombre de la máquina y la identificación lógica de esta máquina (dirección IP).

- **Servidores de terminales gráficas** (*network-oriented window systems*): permiten que un ordenador pueda visualizar información gráfica sobre un display que está conectado a otro ordenador. El más común de estos sistemas en GNU/Linux es X Window y funciona a través de un *display manager* (dm) o en el caso de sistemas Windows® los *terminal Services* (ahora conocidos como *Remote Desktop Services*) permiten a un usuario acceder a las aplicaciones y datos almacenados en otro ordenador mediante la red.

[6]Del inglés *domain name system.*

No obstante, en la última década han proliferado los servicios que se ofrecen en TCP/IP para dar respuesta a las necesidades tanto de los usuarios individuales como a los servicios de grandes instalaciones. Entre los más importantes para sistemas *nix podemos enumerarlos siguientes:

Nota

En este apartado describiremos los protocolos/servicios habituales y el resto quedarán para la siguiente asignatura (*Administración Avanzada Gnu/Linux*).

- **autofs, amd**. Montado automático de discos por la red.

- **arpwatch**. Base de datos sobre las direcciones físicas de los controladores Ethernet (direcciones MAC) que se ven en una red y las direcciones lógicas asociadas a ella (IP).

- **audit**. Permite guardar información remota con fines de auditoría.

- **bluetooth**. Comunicaciones a través del protocolo del mismo nombre.

- **bootparamd/tftp**. Permite a máquinas sin disco obtener los parámetros de red y SO.

- **cups**. Servicio de impresión por red.

- **cvs/subversion**. Sistema de versiones concurrentes.

- **inetd/xinetd**. *Superdaemon* de red que centraliza un conjunto de servicios y aplica filtros.

- **imap/pop**. Servicio de *Internet Message Access Protocol* para el acceso al sistema de correo remoto.

- **dhcp**. *Dynamic Host Configuration* provee de información sobre parámetros de la red a clientes en una subred.

- **firewall**. *Packet Filtering firewall* utilizado para gestionar y filtrar todos los paquetes IP en el *kernel* de Linux (por ejemplo, iptables/shorewall).

- **heartbeat**. *High Availability Services* para incrementar la redundancia en servicios críticos.

- **http**. Servicio que implementa el protocolo http (y otros) para la gestión de servicios web.

- **ipmi**. Gestión remota de máquinas a través de OpenIPMI.

- **ipsec**, **kerberos**, **ssl/ssh**. Protocolos/servicios para permitir comunicaciones codificadas y su autentificación.

- **iscsi**. Gestión y acceso a discos vía el protocol iSCSI sobre una red.

- **ldap**. *Lightweight Directory Access Protocol*, provee servicios de acceso a información en redes de gran tamaño.

- **named**. *Domain Name System* (por ejemplo, bind).

- **netdump**. Envío de información sobre la red para situaciones de diagnóstico cuando un *kernel* deja de funcionar.

- **netfs/nfs/smb**. Montado de disco en red: *Network File System* (NFS), Windows (SMB).

- **ntalk**. Servicio de chat.

- **ntpd**. Servicios de sincronización de relojes.

- **portmap**. *DARPA2RPC- number-mapper* utilizado por *Remote Procedure Call* (RPC) en protocolos como NFS.

- **proxy**. Servicios de *Proxy Caching Server* para http y ftp (por ejemplo, squid).

- **rsync**. *Remote Synchronization* para servicios de resguardo de archivos (*backups*).

- **snmpd**. Implementa *Simple Network Management Protocol* para obtener información/gestionar dispositivos conectados a un red.

- **sqld**. Servicio de bases de datos por red (por ejemplo, mysqld/postgresqld).
- **vncserver**. Servicio para *Virtual Network Computing* y utilización de *Remote Desktop Sharing*.
- **ypbind/ypserv**. Servicios que implementan NIS en sistemas GNU/Linux.
- **zeroconf DNS**. Publica y obtiene información de la red cuando no existe un servicio de DNS (por ejemplo, mdnsresponder).

1.2. ¿Qué es TCP/IP?

TCP/IP son en realidad dos protocolos de comunicación entre ordenadores independientes uno del otro.

Por un lado, TCP[7], define las reglas de comunicación para que un ordenador (*host*) pueda 'hablar' con otro (si se toma como referencia el modelo de comunicaciones OSI/ISO se describe la capa 4, ver tabla siguiente). TCP es orientado a conexión, es decir, equivalente a un teléfono, y la comunicación se trata como un flujo de datos (*stream*).

(7)Del inglés *transmission control protocol*.

Por otro lado, IP[8], define el protocolo que permite identificar las redes y establecer los caminos entre los diferentes ordenadores. Es decir, encamina los datos entre dos ordenadores a través de las redes. Corresponde a la capa 3 del modelo OSI/ISO y es un protocolo sin conexión (ver tabla siguiente). [Com01, Rid00, Dra99]

(8)Del inglés *internet protocol*.

Una alternativa al TCP la conforma el protocolo UDP[9], el cual trata los datos como un mensaje (*datagrama*) y envía paquetes. Es un protocolo sin conexión[10] y tiene la ventaja de que ejerce una menor sobrecarga en la red que las conexiones de TCP, pero la comunicación no es fiable (los paquetes pueden no llegar o llegar duplicados).

(9)Del inglés *user datagram protocol*.

(10)El ordenador destino no debe necesariamente estar escuchando cuando un ordenador establece comunicación con él.

Existe otro protocolo alternativo llamado ICMP[11]. 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. ICMP es similar a UDP, ya que maneja mensajes (datagramas), pero es más simple que UDP, ya que no posee identificación de puertos[12] en el encabezamiento del mensaje.

(11)Del inglés *internet control message protocol*.

(12)Son buzones donde se depositan los paquetes de datos y desde donde las aplicaciones servidoras leen dichos paquetes.

En el modelo de comunicaciones de la OSI[13]/ISO[14], es un modelo teórico adoptado por muchas redes. Existen siete capas de comunicación, de las que cada una tiene una interfaz para comunicarse con la anterior y la posterior:

(13)Del inglés *open systems interconnection reference model*.

Nivel	Nombre	Utilización
7	Aplicación	SMTP[15], el servicio propiamente dicho
6	Presentación	Telnet, FTP implementa el protocolo del servicio
5	Sesión	Generalmente no se utiliza
4	Transporte	TCP, UDP transformación de acuerdo a protocolo de comunicación
3	Red	IP permite encaminar el paquete (*routing*)
2	Link	Controladores (*drivers*) transformación de acuerdo al protocolo físico
1	Físico	Ethernet, ADSL... envía del paquete físicamente

(14)Del inglés *international standards organization.*

(15)Del inglés *simple mail transfer protocol.*

Aunque el modelo OSI es el de referencia, muchas veces se prefiere utilizar el modelo TCP/IP que tiene cuatro capas de abstracción (según se define en el RFC 1122) y que frecuentemente se compara con el modelo OSI de siete capas para establecer las equivalencias. El modelo TCP/IP y los protocolos relacionados son mantenidos por la Internet Engineering Task Force (IETF) y la subdivisión en capas que realizan es:

- **Capa 4 o capa de aplicación**. Aplicación, asimilable a las capas 5 (sesión), 6 (presentación) y 7 (aplicación) del modelo OSI. La capa de aplicación debía incluir los detalles de las capas de sesión y presentación OSI. Se creó una capa de aplicación que maneja aspectos de representación, codificación y control de diálogo.
  De acuerdo al diseño de TCP/IP, esta capa maneja los protocolos de alto nivel y aspectos de presentación/codificación y diálogo.

- **Capa 3 o capa de transporte**. Transporte, asimilable a la capa 4 (transporte) del modelo OSI.
  Esta capa está vinculada a protocolos y a parámetros de calidad de servicio con relación a la confiabilidad y control de errores. Es por ello por lo que define dos protocolos *Transmission Control Protocol* (TCP), orientado a comunicación y libre de errores, y *User Datagram Protocol* (UDP), que es sin conexión y por lo tanto, puede haber paquetes duplicados o fuera de orden.

- **Capa 2 o capa de Internet**. Internet, asimilable a la capa 3 (red) del modelo OSI.
  El objetivo de esta capa es enviar paquetes origen desde cualquier red y que estos lleguen a su destino, independientemente de la ruta escogida y de las redes que deben recorrer para llegar. El protocolo específico de esta capa se denomina Internet Protocol (IP), el cual debe decidir la mejor ruta y adecuación de los paquetes para obtener el objetivo deseado.

- **Capa 1 o capa de acceso al medio**. Acceso al medio, asimilable a la capa 2 (enlace de datos o link) y a la capa 1 (física) del modelo OSI.
  Esta capa se ocupa de todos los aspectos que requiere un paquete IP para realizar realmente un enlace físico e incluye los detalles de tecnología (p. ej., LAN, WAN) y los de las capas físicas y de enlace de datos del modelo OSI.

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. [KD, Dra]

Algunos de los protocolos que utilizan los servicios mencionados han sido diseñados por Berkeley, Sun u otras organizaciones. Ellos no forman oficialmente parte de *Internet Protocol Suite*, IPS). Sin embargo, son implementados utilizando TCP/IP y por lo tanto, considerados como parte formal de IPS. Una descripción de los protocolos disponibles en Internet puede consultarse la RFC 1011 (ver referencias sobre RFC [IET]), que lista todos los protocolos disponibles.

Como se mencionó anteriormente, existe una nueva versión del protocolo IPv6, 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*.

1.3. Dispositivos físicos (*hardware*) de red

Desde el punto de vista físico (capa 1 del modelo OSI), el hardware más utilizado para LAN es conocido como Ethernet (o FastEthernet o GigaEthernet). Sus ventajas son su bajo coste, velocidades aceptables: 10, 100, o 1.000 megabits (1.000 megabits = 1Gbits por segundo y actualmente ya hay dispositivos comerciales a 10Gbits/seg) y facilidad en su instalación.

Existen tres modos de conexión en función del tipo de cable de interconexión: grueso (*thick*), fino (*thin*) y par trenzado (*twisted par*).

Las dos primeras están obsoletas (utilizan cable coaxial), mientras que la última se realiza a través de cables (pares) trenzados y conectores similares a los telefónicos (se conocen como RJ45). La conexión par trenzado es conocida como 10baseT, 100baseT o 1000baseT (según la velocidad) y utiliza repetidores llamados *hubs* como puntos de interconexión. La tecnología Ethernet utiliza elementos intermedios de comunicación (*hubs, switchs, routers*) para configurar múltiples segmentos de red y dividir el tráfico para mejorar las prestaciones de transferencia de información. Normalmente, en las grandes instituciones estas LAN Ethernet están interconectadas a través de fibra óptica utilizando tecnología FDDI[16] que es mucho más cara y compleja de instalar, pero se pue-

(16)Del inglés *fiber distributed data interface.*

den obtener velocidades de transmisión equivalentes a Ethernet y no tienen la limitación de la distancia de esta (FDDI admite distancias de hasta 200 km). Su coste se justifica para enlaces entre edificios o entre segmentos de red muy congestionados. [Rid00, KD00]

En el caso de las conexiones domésticas o para pequeñas empresas, la tecnología de redes que ha tenido gran difusión es a través de conexiones por medio de un cable telefónico de cobre (DSL, VDSL o ASDL[17] *Asymmetric Digital Subscriber Line*) o últimamente, por fibra óptica (FTTH, *Fiber to the Home*), donde en cualquier caso es necesario un dispositivo (*router*) que convierta estos protocolos en Ethernet (o en protocolos inalámbricos Wifi). La velocidad es uno de los parámetros de selección entre un tipo u otro de tecnología y el coste comienza a ser asumible para conexiones particulares, sobre todo por la incorporación de servicios de banda ancha, como TV digital o *Video on Demand*, a los ya habituales. Sobre ASDL encontramos opciones desde 12Mbits a 50Mbits, dependiendo de la ubicación de la central y el cliente, y sobre fibra podemos encontrar velocidades entre 30Mbits y 300Mbits. Se debe tener en cuenta que para este tipo de conexiones los valores que se dan son generalmente de bajada, mientras que de subida suelen ser aproximadamente 1/10 del valor de bajada (si bien en algunas operadoras de fibra óptica ofrecen simetría –igual velocidad de subida que de bajada–).

(17)Del inglés *Digital Subscriber Line, Very high bit-rate Digital Subscriber Line, Asymmetric Digital Subscriber Line*.

Existe además otro tipo de hardware menos común, pero no menos interesante como es ATM[18] (no obstante, hoy en día con la disponibilidad y bajo coste de las redes Ethernet de Gigabit ha quedado relegada a pocas instalaciones muy específicas). Este hardware permite montar redes de área local (LAN) con una calidad de servicio elevada y es una buena opción cuando deben montarse redes de alta velocidad y baja latencia, como por ejemplo aquellas que involucren distribución de vídeo en tiempo real.

(18)Del inglés *asynchronous transfer mode*.

Existe otro hardware soportado por GNU/Linux para la interconexión de ordenadores, entre los cuales podemos mencionar: *Frame Relay* o X.25, utilizada en ordenadores que acceden o interconectan WAN y para servidores con grandes necesidades de transferencias de datos (hoy en día utilizado solo con finalidades académicas); *Packet Radio*, interconexión vía radio utilizando protocolos como AX.25, NetRom o Rose, o dispositivos dialing up, que utilizan líneas series, lentas pero muy baratas, a través de un módem analógico o digital (RDSI, DSL, ADSL, etc.).

Para virtualizar la diversidad de hardware sobre una red, TCP/IP define una interfaz abstracta mediante la cual se concentrarán todos los paquetes que serán enviados por un dispositivo físico (lo cual también significa una red o un segmento de esta red). Por ello, por cada dispositivo de comunicación en la máquina tenderemos una interfaz correspondiente en el kernel del sistema operativo.

Ethernet

Ethernet en GNU/Linux se llaman con ethx (donde en todas, x indica un número de orden comenzando por 0) o algunas distribuciones las denominan emx o ibx –estas para tarjetas de Infinibad–, la interfaz a líneas series (módem) se llaman por pppx (para PPP) o slx (para SLIP), para FDDI son fddix. Estos nombres son utilizados por los comandos para configurar sus parámetros y asignarles el número de identificación que posteriormente permitirá comunicarse con otros dispositivos en la red.

En GNU/Linux significa tener que incluir los módulos adecuados para el dispositivo (NIC[19]) adecuado (en el *kernel* o como módulos). Esto supone compilar el kernel después de haber escogido con, por ejemplo, `make menuconfig` el NIC adecuado, indicándole como interno o como módulo (en este último caso se deberá compilar el módulo adecuado también).

[19] Del inglés *network interface card*.

ifconfig -a

Para ver las interfaces de red disponibles hay que aplicar el comando *ifconfig -a*. Este comando muestra todas las interfaces/parámetros por defecto de cada una.

Si bien este comando continua funcionando, es obsoleto (en la mayoría de las distribuciones) y se recomienda utilizar `ip address` o simplemente `ip a`.

Los dispositivos de red se pueden mirar en el directorio */dev*, que es donde existe un archivo (especial, ya sea de bloque o de caracteres según su transferencia), que representa a cada dispositivo hardware [KD, Dra]. También se puede observar en */proc/net* los dispositivos configurados y toda la información (dinámica) de su estado y configuración. Por ejemplo */proc/net/dev* me da la información instantánea de los paquetes enviados y recibidos y su estatus por cada interfaz (se han omitido las columnas con cero):

Interface	Receive			Transmit		
	bytes	packets	errs	bytes	packets	errs
lo	530998295	1272798	0	530998295	1272798	0
eth1	201192	5984	0	2827048	66035	0
eth0	18723180	24837	0	1123703	16419	0

2. Conceptos en TCP/IP

Como se ha observado, la comunicación significa una serie de conceptos que ampliaremos a continuación [Mal, Com]:

- **Internet/intranet**: el término *intranet* se refiere a la aplicación de tecnologías de Internet (red de redes) dentro de una organización, básicamente para distribuir y tener disponible información dentro de la compañía. Por ejemplo, los servicios ofrecidos por GNU/Linux como servicios Internet e intranet incluyen correo electrónico, WWW, *news*, etc. y generalmente son montados sobre direcciones IP privadas (se verá más adelante), por lo cual estas máquinas no son reconocidas desde Internet y su salida hacia Internet es a través de un *router* (en la mayoría de los casos pasa lo mismo con máquinas que tenemos en un entorno doméstico). Si es necesario que alguno de estos servicios tenga acceso desde fuera de la institución, habrá que poner servicios de Proxy o *routers* que puedan redireccionar los paquetes hacia el servidor interno.

- **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 tableta, un teléfono, una impresora, una torre (*rack*) de CD, etc.), es decir, un elemento activo y diferenciable en la red que reclama o presta algún servicio y/o comparte información.

- **Dirección de red Ethernet** (*Ethernet address* o *MAC address*[20]): un número de 48 bits (por ejemplo 00:88:40:73:AB:FF –en octal– 0000 0000 1000 1000 0100 0000 0111 0011 1010 1011 1111 1111 –en binario–) 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). Se utiliza en la capa 2 del modelo OSI y es posible tener 2^{48} o 281.474.976.710.656 direcciones MAC posibles.

[20] Del inglés, *media access control*.

- **Host name**: cada nodo debe tener además un único nombre en la red. Ellos pueden ser solo 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. Los nombres deben ser de un máximo de 32 caracteres entre a-zA-Z0-9.-, y que no contengan espacios o # comenzando por un carácter alfabético.

Nota

Nombre de la máquina: **more /etc/hostname**.

- **Dirección de Internet** (*IP address*): está compuesto por un conjunto de números y dependerá de la versión del protocolo IP, y se utiliza universalmente para identificar los ordenadores sobre una red o Internet. Para la versión 4 (IPV4) está formado por cuatro números en el rango 0-255 (32 bits), separados por puntos (por ejemplo, 192.168.0.1) lo cual posibilita 4.294.967.296 (2^{32}) direcciones de host diferentes, lo cual se ha mostrado insuficiente y sobre todo porque cada individuo dispone de más de un ordenador, tableta, teléfono, PDA, etc. Para la versión 6 (IPv6) la dirección es de 128 bits y se agrupan en cuatro dígitos hexadecimales formando 8 grupos, por ejemplo, fe80:0db8:85a3:08d3:1319:7b2e:0470:0534 es una dirección IPv6 válida. Se puede comprimir un grupo de cuatro dígitos si este es nulo (0000). Por ejemplo:
  fe80:0db8:0000:08d3:1319:7b2e:0470:0534=
  fe80:0db8::08d3:1319:7b2e:0470:0534
  Siguiendo esta regla, si dos grupos consecutivos son nulos, se pueden agrupar por ejemplo: fe80:0db8:0000:0000:0000:0000:0470:0534=fe80:0db8::0470:0534. Se debe tener cuidado ya que fe80:0000:0000:0000:1319:0000:0000:0534 no se puede resumir como fe80::1319::0534 porque no se conoce la cantidad de grupos nulos de cada lado. También los ceros iniciales se pueden omitir quedando fe80:0db8::0470:0534 como fe80:db8::470:534. En IPv6, por lo tanto, se admiten 340.282.366.920.938.463.463.374.607.431.768.211.456 direcciones (2^{128} o 340 sextillones de direcciones), lo cual significa aproximadamente $6{,}7 \times 10^{17}$ (670 mil billones) de direcciones por cada milímetro cuadrado de la superficie de la Tierra, que se demuestra un número suficientemente grande como para no tener las limitaciones de IPv4.
  La traslación de nombres en direcciones IP la realiza un servidor DNS (*Domain Name System*) que transforma los nombres de nodo (legibles por humanos) en direcciones IP (`named` es uno de los servicios que en GNU/Linux realiza esta conversión).

Nota

Dirección IP de la máquina: **more /etc/hosts**.

- **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 ssh lo harán por el puerto 22, pero estas mismas máquinas pueden tener una comunicación http por el puerto 80). Se pueden tener diferentes aplicaciones comunicándose entre dos nodos a través de diferentes puertos simultáneamente.

Nota

Puertos preasignados en UNIX: **more /etc/services**. Este comando muestra los puertos predefinidos por orden y según soporten TCP o UDP.

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

Nota

Visualización de la configuración del routing:
`netstat -r`.

- ***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, o en casos especiales tres letras CAT lengua y cultura catalana...). El segundo nivel representa la organización, el tercero y restantes departamentos, secciones o divisiones dentro de una organización (por ejemplo, www.uoc.edu o pirulo@nteum.remix.cat). Los dos primeros nombres (de derecha a izquierda, *uoc.edu* en el primer caso, *remix.cat* (en el segundo) deben ser asignados (aprobados) por el Internet Network Information Center (NIC, órgano mundial gestor de Internet) y los restantes pueden ser configurados/asignados por la institución. Una regla que rige estos nombres es la FQDN (*fully qualified domain name*) que incluye el nombre de un ordenador y el nombre de dominio asociado a ese equipo. Por ejemplo, si tenemos el *host* que tiene por nombre "nteum" y el nombre de dominio "remix.cat.", el FQDN será "«nteum.remix.cat.". En los sistemas de nombre de dominio de zonas, y más especialmente en los FQDN, los nombres de dominio se especificarán con un punto al final del nombre.

  En algunas redes en las que no se dispone de DNS o no se tiene acceso a él, se puede utilizar el archivo */etc/hosts* (que deberá estar en todos los dispositivos de la red) y cumplirá el mismo objetivo, o también se puede instalar el servicio mDNS (por ejemplo, Avahi) que implementa lo que se denomina *zero-configuration networking* (*zeroconf*) para la configuración por multicast de DNS/DNS-SD. Este servicio permite a los programas publicar y descubrir servicios y *hosts* sobre una red local. Por ejemplo, un usuario puede conectar su ordenador a la red local y automáticamente el servicio mDNS descubrirá impresoras, archivos, *hosts*, usuarios o servicios (un equivalente a este servicio de sistemas Mac-W es Bonjour).

  En una política de apertura, el ICANN ha regulado la petición/inclusión de nuevos dominios de orden superior genéricos (gTLDs) orientados por sectores, por ejemplo, ciudades (.barcelona, .miami...), alimentación (.beer, .coffee...), geocultural, música y arte, educación, informática, etc.

Nota

Dominio y quién es nuestro servidor de DNS: **more /etc/default domain**; **more /etc/resolv.conf**.

- **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, la máscara, el *gateway*, DNS, etc.). 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 IP son descubiertas automáticamente a través de dos protocolos miembros de IPS: ARP[21] y RARP[22]. ARP utiliza mensajes (*broadcast messages*) para determinar la dirección Ethernet (especificación MAC de la capa 3 del modelo OSI) correspondiente a una

[21] Del inglés *address resolution protocol*.

[22] Del inglés *reverse address resolution protocol*.

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*).

- **Biblioteca de sockets**: en UNIX toda la implementación de TCP/IP forma parte del kernel del sistema operativo (o bien dentro del mismo o como un módulo que se carga en el momento del inicio como el caso de GNU/Linux con los controladores de dispositivos). La forma de utilizarlas por un programador es a través de la API[23] que implementa ese operativo. Para TCP/IP, la API más común es la Berkeley Socket Library (Windows utiliza una librería equivalente que se llama Winsocks). Esta biblioteca permite crear un punto de comunicación (*socket*), asociar este a una dirección de un nodo remoto/puerto (*bind*) y ofrecer el servicio de comunicación (a través de *connect*, *listen*, *accept, send, sendto, recv, recvfrom,* por ejemplo). La biblioteca provee, además de la forma más general de comunicación (familia AF INET), comunicaciones más optimizadas para casos en que los procesos se comunican en la misma máquina (familia AF UNIX). En GNU/Linux, la biblioteca de sockets es parte de la biblioteca estándar de C, Libc, (Libc6 en las versiones actuales), y soporta AF_INET, AF_UNIX en sus protocolos TCP/UDP, así como una serie de otros protocolos, como por ejemplo, AF_IPX (redes Novell), AF_X25 (para X.25), AF_ATMPVC-AF_ATMSVC (para ATM), AF_ROSE (para el *amateur radio protocol*), etc.

[23] Del inglés *application programming interface.*

Nota

Tablas de arp: arp -a.

3. ¿Cómo se asigna una dirección Internet?

Esta dirección en sus orígenes era asignada por el Internet Network Information Center, que fue el organismo gubernamental de Internet responsable de los nombres de dominio y las direcciones IP (hasta el 18/9/1998), y posteriormente, este rol fue asumido por la Internet Corporation for Assigned Names and Numbers (ICANN).

Para este apartado deberemos separar las definiciones en relación al protocolo IPv4 y el protocolo IPv6.

3.1. IPv4

La dirección IP en IPv4 tiene dos campos: el izquierdo representa la identificación de la red y el derecho la identificación del nodo. Considerando lo mencionado anteriormente (4 números entre 0-255, o sea 32 bits o cuatro bytes), cada byte representa o bien la red o bien el nodo. La parte de red es asignada por el ICANN y la parte del nodo es asignada por la institución o el proveedor.

Existen algunas restricciones: **0** (por ejemplo, 0.0.0.0) en el campo de red está reservado para el *routing* por defecto y **127** (por ejemplo, 127.0.0.1) es reservado para la autorreferencia (*local loopback* o *local host*), **0** en la parte de nodo se refiere a esta red (por ejemplo, 192.168.0.0) y **255** es reservado para paquetes de envío a todas las máquinas (*broadcast*) (por ejemplo, 198.162.255.255).

En las diferentes asignaciones se puede tener diferentes tipos de redes o direcciones:

- **Clase A** (*red.host.host.host*): 1.0.0.1 a 126.254.254.254 (126 redes, 16 millones de nodos) definen las grandes redes. El patrón binario es: **0** + 7 bits red + 24 bits de nodos.

- **Clase B** (*red.red.host.host*): 128.1.0.1 a 191.255.254.254 (16K redes, 65K nodos) generalmente se utiliza el primer byte de nodo para identificar subredes dentro de una institución). El patrón binario es **10** + 14 bits de red + 16 bits de nodos.

- **Clase C** (*red.red.red.host*): 192.1.1.1 a 223.255.255.254 (2 millones de bits de redes, 254 de nodos). El patrón binario es **110** + 21 bits red + 8 bits de nodos.

- **Clase D y E** (*red.red.red.host*): 224.1.1.1 a 255.255.255.254 reservado para *multicast* (desde un nodo a un conjunto de nodos que forman parte de un grupo) y propósitos experimentales.

Redes privadas

Máquinas que se conectan entre ellas sin tener conexión con el exterior.

Algunos rangos de direcciones han sido reservados para que no correspondan a redes públicas, sino a redes privadas y los mensajes no serán encaminados a través de Internet, lo cual es comúnmente conocido como Intranet. Estas son:

- Clase A: 10.0.0.0 hasta 10.255.255.255
- Clase B: 172.16.0.0 hasta 172.31.0.0
- Clase C: 192.168.0.0 hasta 192.168.255.0.

Un concepto asociado a una dirección IP es el de máscara, que luego nos permitirá definir subredes y realizar el encaminamiento de los paquetes en forma automática entre estas subredes. Para ello, es necesario definir una máscara de red que serán los bits significativos de la subred y que nos permitirá definir si dos IP están dentro de la misma red o no. Por ejemplo, en una dirección IPv4 estática determinada (p. ej., 192.168.1.30), la máscara de red 255.255.255.0 (es decir, 11111111111111111111111100000000 en representación binaria) indica que los primeros 24 bits de la dirección IP corresponden a la dirección de red y los otros 8 son específicos a la máquina. En IPv6 y dado que son 128 bits, solo se expresará la cantidad de 1 (notación que también se utiliza en IPv4), para facilitar su lectura. En el ejemplo anterior, para IPv4 sería 24 y generalmente se pondría 192.168.1.30/24 y en IPv6, por ejemplo, para la dirección fe80:0db8::0470:0534 se podría expresar la máscara como fe80:0db8::0470:0534/96, donde indica que para esta dirección los 96 primeros bits corresponden a la red.

La dirección de *broadcast* es especial, ya que cada nodo en una red escucha todos los mensajes (además de su propia dirección). Esta dirección permite que datagramas, generalmente información de *routing* y mensajes de aviso, puedan ser enviados a una red y todos los nodos del mismo segmento de red los puedan leer. Por ejemplo, cuando ARP busca encontrar la dirección Ethernet correspondiente a una IP, este utiliza un mensaje de *broadcast*, el cual es enviado a todas las máquinas de la red simultáneamente. Cada nodo en la red lee este mensaje y compara la IP que se busca con la propia y le retorna un mensaje al nodo que hizo la pregunta si hay coincidencia. Por ejemplo, en una red 192.168.1.0/24, la dirección de broadcast es 192.168.1.255.

3.2. IPv6

Los tipos de direcciones IPv6 pueden identificarse tomando en cuenta los rangos definidos por los primeros bits de cada dirección (el valor que se especifica después de la barra es la máscara equivalente al número de bits que no se consideran de esa dirección):

- **::/128**. La dirección con todo ceros se utiliza para indicar la ausencia de dirección, y no se asigna ningún nodo.

- **::1/128**. Representa la dirección de *loopback* que puede usar un nodo para enviarse paquetes a sí mismo (corresponde con 127.0.0.1 de IPv4). No puede asignarse a ninguna interfaz física.

- **::1.2.3.4/96**. La dirección IPv4 compatible se usa como un mecanismo de transición en las redes duales IPv4/IPv6 (no utilizado).

- **::ffff:0:0/96**. La dirección IPv4 mapeada se usa como mecanismo de transición.

- **fe80::/10**. El prefijo del *link local* especifica que la dirección solo es válida en el enlace físico local.

- **fec0::**. El prefijo local (*site-local prefix*) especifica que la dirección solo es válida dentro de una organización local. La RFC 3879 lo declaró obsoleto y deben ser sustituidos por direcciones Local IPv6 Unicast.

- **ff00::/8**. El prefijo de multicast se usa para las direcciones multicast.

Hay que resaltar que no existen las direcciones de difusión (*broadcast*) en IPv6, y la funcionalidad puede emularse utilizando la dirección multicast FF01::1/128.

Si la dirección es una dirección IPv4 empotrada, los últimos 32 bits pueden escribirse en base decimal como ::ffff:192.168.1.1 o ::ffff:c0a8:0101 y no confundir con ::192.168.89.9 o ::c0a8:0101.

El formato ::ffff:1.2.3.4 se denomina dirección IPv4 mapeada, y el formato ::1.2.3.4, dirección IPv4 compatible.

Las direcciones IPv4 pueden ser transformadas fácilmente al formato IPv6. Por ejemplo, si la dirección decimal IPv4 es 158.109.64.1 (en hexadecimal, 0x9e6d4001), puede ser convertida a 0000:0000:0000:0000:0000:0000:9e6d:4001 o ::9e6d:4001. En este caso se

puede utilizar la notación mixta IPv4 compatible que sería::158.109.64.1. Este tipo de dirección IPv4 compatible se está utilizando muy poco, aunque los estándares no la han declarado obsoleta.

Cuando lo que se desea es identificar un rango de direcciones diferenciable por medio de los primeros bits, se añade este número de bits tras el carácter de barra "/". Por ejemplo:

- fe80:0db8::0470:0534/96 sería equivalente a fe80:0db8::

- fe80:0db8::0674:9966/96 sería equivalente a fe80:0db8:: y también a fe80:0db8::0470:0534/96

Las direcciones IPv6 se representan en el DNS mediante registros AAAA (también llamados *registros de quad-A*, ya que tienen cuatro veces la longitud de los registros A para IPv4) especificados por la RFC 3363 (existe otra visión llamada A6, pero si bien es más genérica también es más compleja y puede complicar aún más la transición entre ipv4 e ipv6).

Uno de los grandes problemas de la transición hacia IPv6 es el agotamiento de las direcciones IPv4 y los problemas que este está ocasionando, y es por ello por lo que algunos países (como India o China) ya han comenzado su migración. Existen una serie de mecanismos que permitirán la convivencia y la migración progresiva, tanto de las redes como de los equipos de usuario, que puede clasificarse en tres grupos: doble pila, túneles, traducción.

Problemas de la transición hacia IPv6

Por lo que se refiere a Europa, podéis consultar la pàgina web de RIPE (http://www.ripe.net/internet-coordination/ipv4-exhaustion/ipv4-available-pool-graph) para haceros una idea del problema.

La doble pila hace referencia a una solución de nivel IP con doble pila (RFC 4213), que implementa las pilas de ambos protocolos, IPv4 e IPv6, en cada nodo de la red. Cada nodo con doble pila en la red tendrá dos direcciones de red, una IPv4 y otra IPv6, y como ventajas presenta que es fácil de desplegar y muy soportado, y como inconvenientes, que la topología de red requiere dos tablas de encaminamiento. Los túneles permiten conectarse a redes IPv6 "saltando" sobre redes IPv4 encapsulando los paquetes IPv6 en paquetes IPv4 (teniendo como siguiente capa IP el protocolo número 41 y por ello el nombre de proto-41). Existen muchas tecnologías de túneles disponibles y se diferencian en el método para determinar la dirección a la salida del túnel. La traducción es necesaria cuando un nodo que solo soporta IPv4 intenta comunicar con un nodo que solo soporta IPv6 y básicamente se agrupan en "con estado" (NAT-PT (RFC 2766), TCP-UDP Relay (RFC 3142), Socks-based Gateway (RFC 3089) o "sin estado" (Bump-in-the-Stack, Bump-in-the-API (RFC 276)).

3.3. Subredes y encaminamiento (*routing*)

Dos conceptos complementarios a lo descrito anteriormente es el de **subredes** y **routing** entre ellas. Subredes significa subdividir la parte del nodo en pequeñas redes dentro de la misma red para, por ejemplo, mejorar el tráfico. Una subred toma la responsabilidad de enviar el tráfico a ciertos rangos de

direcciones IP extendiendo el mismo concepto de redes A-B-C, pero solo aplicando esta redirección en la parte nodo de la IP. El número de bits que son interpretados como identificador de la subred es dado por una máscara de red (*netmask*) que es un número de 32 bits (igual que la IP).

Quien definirá que paquetes van hacia un lado u otro será el *host* que cumpla con el papel de enrutador (*router*) y que interconectará varios segmentos de redes/redes entre sí. Generalmente se conoce al *router* como puerta de enlace (*gateway*) y se utiliza como el *host* que ayuda a alcanzar el exterior (por ejemplo, Internet) desde la red local.

Para obtener el identificador de la subred, se deberá hacer una operación lógica **Y** (AND) entre la máscara y la IP, lo cual dará la IP de la subred.

Por ejemplo, sea una institución que tiene una red clase B con número 172.17.0.0, y su *netmask* es, por lo tanto, 255.255.0.0. Internamente, esta red está formada por pequeñas redes (una planta del edificio, por ejemplo). Así, el rango de direcciones es reasignado en 20 subnets 172.17.1.0 hasta 172.17.20.0. El punto que conecta todas estas subredes (*backbone*) tiene su propia dirección, por ejemplo 172.17.1.0.

Estas subredes comparten el mismo IP de red, mientras que el tercero es utilizado para identificar cada una de las subredes dentro de ella (por eso se utilizará una máscara de red 255.255.255.0).

El segundo concepto, *routing*, representa el modo en que los mensajes son enviados a través de las subredes.

Por ejemplo, sean tres departamentos con subredes Ethernet:

- Compras (subred 172.17.2.0).
- Clientes (subred 172.17.4.0).
- Recursos humanos, RR.HH., (subred 172.17.6.0).
- Backbone con FFDI (subred 172.17.1.0).

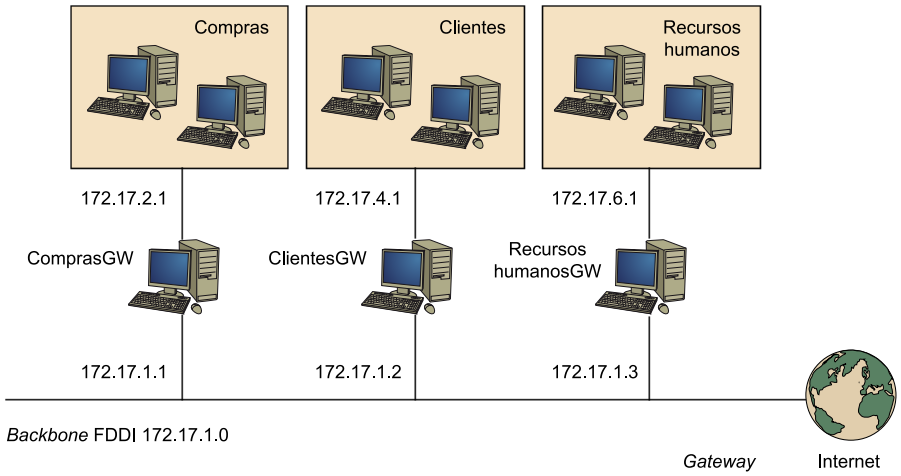
Para encaminar los mensajes entre los ordenadores de las tres redes, se necesitarán tres puertas de intercambio (*gateways*), que tendrán cada una dos interfaces de red para cambiar entre Ethernet y FFDI. Estas serán:

- CromprasGW IPs:172.17.2.1 y 172.17.1.1,
- ClientesGW IPs:172.17.4.1 y 172.17.1.2
- RRHHGW IPs:172.17.6.1 y 172.17.1.3,

es decir, una IP hacia el lado de la subnet y otra hacia el backbone.

Cuando se envían mensajes entre máquinas de compras, no es necesario salir al *gateway*, ya que el protocolo TCP/IP encontrará la máquina directamente. El problema está cuando la máquina Compras0 quiere enviar un mensaje a RRHH3. El mensaje debe circular por los dos gateways respectivos. Cuando Compras0 "ve" que RRHH3 está en otra red, envía el paquete a través del gateway ComprasGW, que a su vez se lo enviará a RRHHGW y que a su vez se lo enviará a RRHH3. La ventaja de las subredes es clara, ya que el tráfico entre todas las máquinas de compras, por ejemplo, no afectará a las máquinas de clientes o de recursos humanos (si bien significa un planteamiento más complejo y caro a la hora de diseñar, y construir la red).

Figura 1. Configuración de segmentos y gateways en una Intranet



IP utiliza una tabla para hacer el routing de los paquetes entre las diferentes redes y en la cual existe un routing por defecto asociado a la red 0.0.0.0. Todas las direcciones que coinciden con esta, ya que ninguno de los 32 bits son necesarios, son enviadas por el *gateway* por defecto (*default gateway*) hacia la red indicada. Sobre comprasGW, por ejemplo, la tabla podría ser:

Dirección	Máscara	Gateway	Interfaz
172.17.1.0	255.255.255.0	-	fddi0
172.17.4.0	255.255.255.0	172.17.1.2	fddi0
172.17.6.0	255.255.255.0	172.17.1.3	fddi0
0.0.0.0	0.0.0.0	172.17.2.1	fddi0
172.17.2.0	255.255.255.0	-	eth0

El '-' significa que la máquina está directamente conectada y no necesita *routing*. El procedimiento para identificar si se realiza el *routing* o no, se lleva a cabo a través de una operación muy simple con dos AND lógicos (subred AND mask y origen AND mask) y una comparación entre los dos resultados. Si son

iguales no hay *routing*, sino que se debe enviar la máquina definida como *gateway* en cada máquina para que esta realice el *routing* del mensaje. Por ejemplo, un mensaje de la 172.17.2.4 hacia la 172.17.2.6 significará:

```
172.17.2.4 AND 255.255.255.0 = 172.17.2.0
172.17.2.6 AND 255.255.255.0 = 172.17.2.0
```

Como los resultados son iguales, no habrá *routing*. En cambio, si hacemos lo mismo con 172.17.2.4 hacia 172.17.6.6 podemos ver que habrá un *routing* a través del 172.17.2.1 con un cambio de interfaz (eth0 a ffdi0) a la 172.17.1.1 y de esta hacia la 172.17.1.2 con otro cambio de interfaz (fddi0 a eth0) y luego hacia la 172.17.6.6. El *routing*, por defecto, se utilizará cuando ninguna regla satisfaga la coincidencia. En caso de que dos reglas coincidan, se utilizará aquella que lo haga de modo más preciso, es decir, la que menos ceros tenga. Para construir las tablas de *routing*, se puede utilizar el comando `route` durante el arranque de la máquina, pero si es necesario utilizar reglas más complejas (o *routing* automático), se puede utilizar el RIP[24] o entre sistemas autónomos, el EGP[25] o también el BGP[26]. Estos protocolos se implementan en el comando `gated`.

(24)Del inglés *routing information protocol.*

(25)Del inglés *external gateway protocol.*

(26)Del inglés *border gateway protocol.*

Para instalar una máquina sobre una red existente, es necesario, por lo tanto, disponer de la siguiente información obtenida del proveedor de red o de su administrador:

- Dirección IP del nodo
- Dirección de la red IP
- Dirección de *broadcast*
- Dirección de máscara de red
- Dirección de *router*
- Dirección del DNS

Si se construye una red que nunca tendrá conexión a Internet, se pueden escoger las direcciones que se prefieran, pero es recomendable mantener un orden adecuado en función del tamaño de red que se desee tener y para evitar problemas de administración dentro de dicha red. A continuación, se verá cómo se define la red y el nodo para una red privada (hay que ser cuidadoso, ya que si se tiene la máquina conectada a la red, se podría perjudicar a otro usuario/*host* que tuviera asignada esta dirección).

4. ¿Cómo se debe configurar la red?

4.1. Configuración de la interfaz (NIC)

En este apartado se verá con un poco más de detalle lo que ya se mencionó en el subapartado 4.6 del módulo "Nivel Usuario" sobre la configuración de la red (ethernet) y de la red Wifi.

Una vez cargado el kernel de GNU/Linux, este ejecuta el *daemon* `systemd` (equivalente al proceso `init` en versiones anteriores) y que de acuerdo a la configuración en */etc/systemd* (o en *usr/lib/systemd*) configurará el sistema. Esta inicialización ejecutará un *script* que permitirá la configuración de la red.

Ejemplo

En Debian se ejecuta `/etc/init.d/networking` para la configuración de la interfaz de red (si bien en las últimas versiones se ejecuta systemd, el *script* de configuración de red continúa siendo este; no obstante cambiará en futuras versiones para adecuarlo totalmente a los *scripts* de inicialización de este nuevo *daemon*). El *script* está solo una vez (*/etc/init.d*) y, de acuerdo a los servicios deseados en ese estado, se crea un enlace en el directorio correspondiente a la configuración del nodo-estado (para los servicios de red se puede ver que se arrancan en */etc/rcS.d*, que es el directorio que contiene todos los *scrips* que se arrancan durante el *boot* en la etapa inicial de monousuario durante la inicialización del hardware). Los parámetros por defecto para cada *script* del */etc/init.d/* se encuentran en un directorio específico (por ejemplo, en Debian es */etc/default/* y en Fedora, */etc/sysconfig*).

Los dispositivos de red se crean automáticamente cuando se inicializa el hardware correspondiente. Por ejemplo, el controlador de Ethernet crea las interfaces eth[0..n] secuencialmente cuando se localiza el hardware correspondiente. En el caso de que las interfaces no estén numeradas correctamente (suele ocurrir cuando trabajamos con máquinas virtuales clonadas de otras máquinas virtuales), es necesario borrar el archivo */etc/udev/rules.d/70*net** (o equivalente) que preserva la numeración de los dispositivos de red a través del mecanismo de *udev* (que veremos más adelante).

Nota

Consultar **man ifconfig** o **man ip** para las diferentes opciones del comando.

A partir de este momento, se puede configurar la interfaz de red, lo cual implica dos pasos: asignar la dirección de red al dispositivo e inicializar los parámetros de la red al sistema. El comando utilizado para ello es el `ip` (si bien también se puede utilizar el comando `ifconfig` *–interfaz configure–* aunque este es obsoleto). Un ejemplo será:

```
ip addr add 192.168.110.23/24 dev eth0
```

Lo cual indica configurar el dispositivo eth0 con dirección IP 192.168.110.23 y máscara de red 255.255.255.0. Para eliminar la IP asignada, ejecutamos la orden `ip addr del 192.168.110.23/24 dev eth1` (debemos acostum-

brarnos a utilizar el comando `ip`, ya que podremos tener algunas diferencias si configuramos algunas cosas con `ifconfig` –que está obsoleto– y otras con `ip`). Para habilitar la interfaz podemos ejecutar `ip link set eth1 up` y para deshabilitarla `ip link set eth1 down`.

```
ifconfig eth0 192.168.110.23 netmask 255.255.255.0 up
```

En sistemas operativos de la rama Debian todavía se sigue utilizando la orden `ifconfig` o `ifdown/ifup`. Estos últimos son utilizados por los *script* de inicialización para configurar-desactivar la red en forma más simple utilizando el archivo */etc/network/interfaces* para obtener todos los parámetros necesarios (consultar `man interfaces` para su sintaxis).

Orden ifconfig

Se ha anunciado que en futuras distribuciones la instrucción `ifconfig` no estará disponible. Por ejemplo, ya existe una distribución de Ubuntu Docker que no incorpora por defecto el paquete *net-tools*, que es el que contiene `ifconfig`, `ifup`, `ifdown`, entre otros.

En GNU/Linux existen diferentes formas de configurar la red para que el administrador no deba en cada *boot* tener que introducir los parámetros de configuración. En Debian, una de las formas es a través de los comandos mencionados anteriormente (`ifup`, `ifdown`) que se ejecutan automáticamente, y el archivo */etc/network/interfaces*. Si se decide utilizar este método, no es necesario hacer nada más.

Para modificar los parámetros[27] de red de la interfaz eth0, se puede hacer:

(27)Consultar *man interfaces* en la sección 5 del manual para más información del formato.

- **ifdown eth0**. Detiene todos los servicios de red. También se puede ejecutar sobre `eth0./etc/init.d/networking stop`, que detiene todos los servicios sobre todas las interfaces de red (o `systemctl stop networking`).

- `vi /etc/network/interfaces` (o el editor que se prefiera, pero es aconsejable el `vi` ya que está en todos los sistemas *nix). Permite editar y modificar los parámetros correspondientes.

- **ifup eth0**. Esta orden permite poner en marcha los servicios de red sobre eth0 (o `/etc/init.d/networking start` o `systemctl networking start`).

Supongamos que desea configurar sobre Debian una interfaz eth0 que tiene una dirección IP fija 192.168.0.123 y con 192.168.0.1 como puerta de enlace (*gateway*). Se debe editar */etc/network/interfaces* de modo que incluya una sección como:

```
auto eth0
iface eth0 inet static
      address 192.168.0.123
      netmask 255.255.255.0
      gateway 192.168.0.1
```

Si tiene instalado el paquete `resolvconf` (en algunas distribuciones, como por ejemplo en Ubuntu, viene instalado por defecto) puede añadir líneas para especificar la información relativa al DNS. Por ejemplo:

```
iface eth0 inet static
      address 192.168.0.123
      netmask 255.255.255.0
      gateway 192.168.0.1
      dns-search remix.cat
      dns-nameservers 195.238.2.21 195.238.2.22
```

En lugar de `auto eth0` también se puede llegar a utilizar, por ejemplo, la orden `allow-hotplug eth0`, que indica la interfaz que se podrá activar con `ifup --allow=hotplug eth0`. Las líneas que comiencen con `allow-` son utilizadas para identificar interfaces que podrían ser activadas por diferentes subsistemas (`allow-auto` y `auto` son sinónimos).

Recordar que si la máquina dispone de varias interfaces de red, se puede repetir la sección previa con el dispositivo correspondiente, pero las tres últimas líneas (*gateway*, *dns-search* y *dns-nameservers*) solo deberán estar **una** vez, ya que son comunes para todas las interfaces.

Después de activar la interfaz, los argumentos de las opciones *dns-search* y *dns-nameservers* quedan disponibles para la inclusión en */etc/resolv.conf*. El argumento remix.cat de la opción *dns-search* corresponde al argumento de la opción *search* en *resolv.conf* y los argumentos 195.238.2.21 y 195.238.2.22 de la opción *dns-nameservers* corresponden a los argumentos de las opciones *nameserver* en *resolv.conf*. En caso de que el paquete *resolvconf* no esté instalado, se puede modificar manualmente el archivo */etc/resolv.conf* (y si está instalado y no se utiliza el */etc/network/interfaces* para configurar los DNS, se puede modificar los archivos que se encuentran en */etc/resolvconf.d*).

Nota

Sobre */etc/resolv.conf*, deberéis consultar el manual para las opciones de DNS

`man resolv.conf`.

Una configuración equivalente por DHCP (es decir, un servidor de DHCP que nos pasará los parámetros de configuración de la red) se simplifica a:

```
auto eth0
iface eth0 inet dhcp
```

Si bien el paquete *net-tools* (que incluye órdenes como `arp`, `ifconfig`, `ip-tunnel`, `iwconfig`, `nameif`, `netstat`, `route`) ya no se instala por defecto en alguna distribuciones, en sistemas operativos de la rama Debian todavía se continua utilizando (sobre todo para los *scripts* de inicialización), pero también incluye el comando `ip` (del paquete *iproute2*) que es mucho más versátil y potente (permite establecer túneles, *routing* alternativos, etc.). No obstante, la sintaxis es un poco más compleja. A continuación veremos una serie de ejemplos para la configuración de la red por el comando `ip`:

- **ip addr add 192.168.1.1 dev eth0**. Define una IP a eth0.
- **ip addr show**. Muestra la configuración.
- **ip addr del 192.168.1.1/24 dev eth0** elimina IP de eth0.
- **ip route add default via 192.168.0.1**. Agrega un *gateway*.
- **ip link set eth1 up**. Activa interfaz.
- **ip link set eth1 down**. Desactiva interfaz.
- **ip route show**. Muestra el *routing*.
- **ip route add 10.10.20.0/24 via 192.168.50.100 dev eth0**. Agrega una regla.
- **ip route del 10.10.20.0/24**. Borra una regla.
- **up ip route add 10.10.20.0/24 via 192.168.1.1 dev eth1**. Se agrega para definir una regla estática en */etc/network/interfaces*.

Otra forma de configurar la red (recomendada para usuarios con movilidad y configuraciones estándar) es a través del paquete **Network Manager** (NM). Este paquete consta de una interfaz gráfica (**nm-connection-editor**) para la configuración de los dispositivos de red (y puede coexistir con las configuraciones en */etc/network/interfaces*) o se puede configurar a través de los archivos, teniendo en cuenta que hay que desactivar las interfaces que queremos que gestione el NM en */etc/network/interfaces*. El NM no gestionará interfaces definidas en */etc/network/interfaces* siempre y cuando */etc/NetworkManager/NetworkManager.conf* contenga:

```
[main]
plugins=ifupdown,keyfile
[ifupdown]
managed=false
```

Se deberá cambiar `managed=true` si se desea que NM gestione las interfaces definidas en */etc/network/interfaces*. Siempre que para su configuración se modifique el archivo */etc/NetworkManager/NetworkManager.conf* y luego de modificarlo se deberá recargar este con `nmcli` con `reload` (para aspectos detallados de la configuración consultar `man NetworkManager.conf` o https://wiki.gnome.org/Projects/NetworkManager/SystemSettings). En algunas situaciones el NM puede generar conflictos con algunos dispositivos de red que hayan sido configurados previamente con el NM y luego se desee realizar la

configuración a través de */etc/network/interfaces*, por lo cual se recomienda desinstalar el NM o eliminar los archivos de configuración de la interfaz correspondiente del directorio */etc/NetworkManager/system-connections/*.

Para configuraciones de otros dispositivos de red, como conexiones con PPP (punto a punto) con módem PSTN (*Public Switched Telephone Network*), módem (genérico) ADSL o compatibles con PPPOE (*Point to Point Protocol Over Ethernet*) o PPTP con (*Point-to-Point Tunneling Protocol*), podéis consultar la información del dispositivo específico y http://qref.sourceforge.net/quick/ch-gateway.es.html o http://debian-handbook.info/browse/es-ES/stable/sect.network-config.html#sect.roaming-network-config.

4.1.1. Configuraciones avanzadas de la red

Es necesario en GNU/Linux diferenciar entre una interfaz física y una interfaz lógica. Una interfaz física es lo que hasta ahora hemos denominado interfaz (por ejemplo, **eth0**) y una interfaz lógica es un conjunto de valores (a veces llamados *perfiles*) que pueden asignarse a los parámetros variables de una interfaz física. Las definiciones *iface* en */etc/network/interfaces* son, en realidad, definiciones de interfaces lógicas no de interfaces físicas, pero si no se indica nada una interfaz física, será configurada, por defecto, como interfaz lógica.

No obstante, si tenemos un ordenador portátil que utilizamos en sitios diferentes (por ejemplo, casa y trabajo) y necesitamos configuraciones diferentes para la red de cada sitio, podemos hacer uso de las definiciones de interfaz lógica. Primero se deben definir dos interfaces lógicas, como casa y trabajo (en lugar de **eth0** como se hizo anteriormente):

```
iface casa inet static
           address 192.168.1.30
           netmask 255.255.255.0
           gateway 192.168.1.1

iface trabajo inet static
           address 158,.109.65.66
           netmask 255.255.240.0
           gateway 158.109.64.1
```

De esta manera, la interfaz física **eth0** se puede activar para la red de casa con `ifup eth0=casa` y para reconfigurarla para el trabajo con *ifdown eth0; ifup eth0=trabajo*.

El mecanismo es muy potente y se puede ampliar a través de la configuración en función de una serie de condicionantes, utilizando una sección mapping. La sintaxis de una sección *mapping* es la siguiente:

```
   mapping patrón
```

```
            script nombre_script
            [map script]
```

El *script* llamado en la sección *mapping* será ejecutado con el nombre de la interfaz física como argumento y con el contenido de todas las líneas map de la sección. Antes de finalizar, el *script* mostrará el resultado de la transformación por la salida estándar.

Por ejemplo, la siguiente sección mapping hará que ifup active la interfaz eth0 como interfaz lógica casa:

```
    mapping eth0
            script /usr/local/sbin/echo-casa
```

donde */usr/local/sbin/echo-casa* es:

```
#!/bin/sh
echo casa
```

Esto puede ser útil si tenemos, por ejemplo, dos tarjetas de red diferentes (una para casa y otra para el trabajo). El directorio */usr/share/doc/ifupdown/examples/* contiene un *script* de transformación que se puede usar para seleccionar una interfaz lógica, basándose en la dirección MAC (*Media Access Controller*). Primero se debe instalar el *script* en un directorio apropiado con:

```
    install -m770 /usr/share/doc/ifupdown/examples/get-mac-address.sh /usr/local/sbin/
```

A continuación se puede añadir una sección como la siguiente en el archivo */etc/network/interfaces*:

```
    mapping eth0
            script /usr/local/sbin/get-mac-address.sh
            map 02:23:45:3C:45:3C casa
            map 00:A3:03:63:26:93 trabajo
```

Existen otros programas de transformación más sofisticados, como **guessnet**, entre otros. Por ejemplo, en el caso de **guessnet**, se debe instalar el paquete y declarar una sección en */etc/network/interfaces*:

```
    mapping eth0
            script guessnet-ifupdown
            map casa
            map trabajo
```

Ahora al hacer `ifup eth0`, `guessnet` verificará si eth0 tiene que activarse como casa o trabajo, utilizando la información almacenada en las definiciones de las interfaces lógicas.

Es posible también configurar los dispositivos de red para el arranque en caliente a través del paquete `hotplug`. Este tipo de configuración es útil cuando utilizamos dispositivos removibles, como por ejemplo, un módem USB. Más información en [DR][GRD].

4.1.2. Configuración de la red en IPv6

En relación con la configuración de IPv6, los sistemas GNU/Linux incorporan esta funcionalidad a través de su implementación en el *kernel* o a través de módulos (en Debian se incluyen en el *kernel* y algunas arquitecturas específicas a través de un módulo llamado ipv6). Herramientas básicas como `ping` y `traceroute` tienen sus equivalentes IPv6, `ping6` y `traceroute6`, disponibles en los paquetes `iputils-ping` y `iputils-tracepath`, respectivamente. Con el archivo */etc/network/interfaces* una red IPv6 se configura en forma similar a IPv4 (verificar previamente que su router es compatible con IPv6 que retransmita datos a la red IPv6 global):

```
iface eth0 inet6 static
    address fe80:0db8::0470:0534
    netmask 64
    # Desactivar autoconfiguración
    # autoconf 0
    # El enrutador se configura automáticamente
    # y no tiene dirección fija (accept_ra 1). Si la tuviera:
    # gateway 2001:db8:1234:5::1 map trabajo
```

Las subredes IPv6 generalmente tienen una máscara de red de 64 bits, lo cual significa que existen 264 direcciones diferentes dentro de la subred y permite a un método llamado *autoconfiguración de direcciones sin estado* (SLAAC, *Stateless Address Autoconfiguration*) selecionar una dirección basada en la dirección MAC de la interfaz de red. De forma predeterminada, si SLAAC está activado en la red, el *kernel* encontrará *routers* IPv6 automáticamente y configurará las interfaces de red.

Este tipo de configuración puede tener consecuencias en la privacidad, ya que si se cambia de red frecuentemente, sería fácil la identificación del dispositivo en estas redes. Las extensiones de privacidad de IPv6 solucionan este problema y asignarán direcciones adicionales generadas aleatoriamente a la interfaz, las cambiarán periódicamente y las utilizarán para conexiones salientes, mientras que las conexiones entrantes podrán utilizar las direcciones generadas por SLAAC. Un ejemplo de esta configuración es activar en */etc/network/interfaces*:

```
iface eth0 inet6 auto
```

```
     # Preferir las direcciones asignadas
      # aleatoriamente para conexiones salientes.
     privext 2
```

Si no se dispone de una conexión IPv6, el método alternativo es utilizar un túnel sobre IPv4. Es importante que los proveedores de túneles IPv6 se ajusten a la RFC 3053 (RFC que describe el procedimiento para solicitar la creación de un túnel IPv6 en un *host* llamado Punto de Presencia o PoP). Existe una gran cantidad de proveedores con implementaciones propias y en base a diferentes objetivos de negocio. Dos de los que se pueden probar son Hurricane Electric Free IPv6 Tunnel Broker o SixXS.

4.1.3. Configuración de red en (estilo) Fedora

Red Hat y Fedora utilizan diferente estructura de ficheros para la configuración de la red: */etc/sysconfig/network*. Por ejemplo, para la configuración estática de la red:

```
NETWORKING=yes


HOSTNAME=my-hostname
   Nombre del host definido por el cmd hostname
FORWARD_IPV4=true
   True para NAT firewall gateways y routers. False para cualquier otro caso
GATEWAY="XXX.XXX.XXX.YYY"
   Dirección IP de la Puerta de salida a Internet.
```

Para configuración por DHCP se debe quitar la línea de *gateway*, ya que será asignada por el servidor. Y en caso de incorporar NIS debe agregarse una línea con el servidor de dominio: **NISDOMAIN=NISProject1**.

Para configurar la interfaz **eth0** en */etc/sysconfig/network-scripts/ifcfg-eth0* (reemplazar las X con los valores adecuados):

```
DEVICE=eth0
BOOTPROTO=static
BROADCAST=XXX.XXX.XXX.255
IPADDR=XXX.XXX.XXX.XXX
NETMASK=255.255.255.0
NETWORK=XXX.XXX.XXX.0
ONBOOT=yes    Activará la red en el boot
```

También a partir de FC3 se pueden agregar:

```
TYPE=Ethernet
HWADDR=XX:XX:XX:XX:XX:XX
GATEWAY=XXX.XXX.XXX.XXX
```

```
IPV6INIT=no
USERCTL=no
PEERDNS=yes
```

O si no para configuración por DHCP:

```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=dhcp
```

Para deshabilitar DHCP, hay que cambiar `BOOTPROTO=dhcp` a `BOOTPROTO=none`. Cualquier cambio en estos ficheros deberá reiniciar los servicios con `service network restart` (o sino también se puede reiniciar con `/etc/init.d/network restart`).

Para cambiar el nombre del *host* se deben seguir estos tres pasos:

1) El comando `hostname nombre-nuevo`.

2) Cambiar la configuración de la red en */etc/sysconfig/network* editando: HOSTNAME=nombre-nuevo.

3) Restaurando los servicios (o haciendo un *reboot*):

- `service network restart` (*o /etc/init.d/network restart*).
- Si la versión del sistema operativo tiene instalado por defecto `systemd` se podrá ejecutar `systemctl restart network`.

Hay que verificar si el nombre está dado de alta en el archivo */etc/hosts*. El hostname puede ser cambiado en tiempo de ejecución recurriendo a la instrucción `sysctl -w kernel.hostname="nombre-nuevo"`.

4.1.4. Configuración de una red Wi-Fi (inalámbrica)

Ya se comentó en el módulo "Nivel Usuario" la configuración de la red inalámbrica, pero en este apartado se darán algunos detalles complementarios/alternativos. Para la configuración de interfaces Wi-Fi se utilizan básicamente el paquete **wireless-tools** (además de `ifconfig` o `ip`). Este paquete utiliza el comando `iwconfig` para configurar una interfaz inalámbrica, pero también se puede hacer a través del */etc/network/interfaces* (algunas distribuciones incluyen el paquete *iw*, que es el equivalente a *iproute2*, y que reemplazará a *wireless-tools*; Debian en la última versión incluye ambos).

Ejemplo: Configurar una WiFi en Debian (similar en FC)

En este caso mostraremos los pasos para cargar los *drivers* de una tarjeta Intel Pro/Wireless 2200BG como método general, pero en los *kernels* actuales estos *drivers* ya están incluidos en el *kernel*, por lo cual no es necesario realizar estos pasos previos aunque sirve como modo de ejemplo. Normalmente, el software que controla las tarjetas se divide en dos partes: el módulo software que se cargará en el *kernel* a través del comando `modprobe` y el *firmware*, que es el código que se cargará en la tarjeta y que nos da el fabricante (consultar la página de Intel para este modelo o el proyecto Wireless Kernel http://wireless.kernel.org/en/users/Drivers/iwlwifi). Como estamos hablando de módulos, es interesante utilizar el paquete de Debian **module-assistant**, que nos permite crear e instalar fácilmente un módulo (otra opción sería instalar las fuentes y crear el módulo correspondiente). El software (lo encontramos en la página del fabricante y lo denomina ipw2200) lo compilaremos e instalaremos utilizando el comando `m-a` del paquete `module-assistant`.

```
aptget install module-assistant      instalación del paquete
m-a -t update
m-a -t -f get ipw2200
m-a -t -build ipw2200
m-a -t install ipw2200
```

Desde la dirección indicada por el fabricante (en su documentación), se descarga la versión del *firmaware* compatible con la versión del *driver*, se descomprime e instala en */usr/lib/hotplug/firmaware* (donde X.X es la versión del el *firmaware*):

```
tar xzvf ipw2200fw2.4.tgz C /tmp/fwr/
cp /tmp/fwr/*.fw /usr/lib/hotplug/firmware/
```

Con esto se copiarán tres archivos (`ipw2200-bss.fw`, `ipw2200-ibss.fw` y `ipw2200-sniffer.fw`). Luego se carga el módulo con `modprobe ipw2200`, se reinicia el sistema (`reboot`) y a continuación, desde la consola, podemos hacer `dmesg | grep ipw`, este comando nos mostrará algunas líneas similares a las que se muestran a continuación y que indicarán que el módulo está cargado (se puede verificar con `lsmod`):

```
ipw2200: Intel(R) PRO/Wireless 2200/2915 Network Driver, git1.0.8
ipw2200: Detected Intel PRO/Wireless 2200BG Network Connection
...
```

Luego se descarga el paquete `wireless-tools` que contiene `iwconfig` y, entre otras, con `aptget install wireless-tools` y ejecutamos `iwconfig`; saldrá algo parecido a:

```
eth1 IEEE 802.11b ESSID:"Nombre-de-la-Wifi"
```

```
Mode:Managed Frequency:2.437 GHz
Access Point:00:0E:38:84:C8:72
Bit Rate=11 Mb/s TxPower=20 dBm
Security mode:open
...
```

A continuación, se deberá configurar el archivo */etc/network/interfaces*. Seguir el procedimiento indicado en el módulo "Nivel Usuario".

4.2. Configuración del name resolver

El siguiente paso es configurar el *name resolver*, que convierte nombres tales como *nteum.remix.cat* en 192.168.110.23.

La forma como el sistema de resolución de nombres actúa 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** (ver `man -s 5 nsswitch.conf`). Si se utiliza el servicio **dns**, el comportamiento del sistema de resolución también viene dado por el archivo de configuración */etc/resolv.conf* que contiene las direcciones IP de los servidores de nombres. Su formato es muy simple (una línea de texto por sentencia). Existen tres palabras clave para tal fin: *domain* (dominio local), *search* (lista de dominios alternativos) y *name server* (la dirección IP del *domain name server*).

4.2.1. Ejemplo de */etc/resolv.conf*

```
domain remix.com
search remix.com piru.com
nameserver 8.8.8.8
nameserver 8.8.4.4
```

Esta lista de servidores de nombre a menudo dependen del entorno de red, que puede cambiar dependiendo de dónde esté o se conecte la máquina (en este ejemplo se han configurado dos servidores de nombres que son los Google DNS Public disponibles sin coste y que presentan muy buenas prestaciones y son seguros).

Los programas de conexión a líneas telefónicas (pppd) u obtención de direcciones IP automáticamente (dhclient) son capaces de modificar *resolv.conf* para insertar o eliminar servidores, pero esta característica no siempre funciona adecuadamente y a veces puede entrar en conflicto y generar configuraciones erróneas. El paquete **resolvconf** (por defecto en algunas distribuciones) soluciona de forma adecuada el problema y permite una configuración simple de

los servidores de nombre en forma dinámica. **resolvconf** está diseñado para funcionar sin que sea necesaria ninguna configuración adicional, no obstante, puede requerir alguna intervención para lograr que funcione adecuadamente.

En Debian es un paquete opcional y puede instalarse con los procedimientos habituales. Esto modificará la configuración de */etc/resolv.conf*, que será reemplazada por un enlace a */etc/resolvconf/run/resolv.conf* y el **resolvconf** utilizará un archivo que será generado dinámicamente en */etc/resolvconf/run/resolv.conf*. Se debe tener en cuenta que el **resolvconf** solo es necesario para modificar dinámicamente los *nameserver*, pero si en nuestra red los *nameservers* no cambian frecuentemente, el */etc/resolv.conf* puede ser lo más adecuado.

Cuando **resolvconf** está instalado no es necesario modificar */etc/resolv.conf*, ya que será automáticamente regenerado por el sistema. Si es necesario definir *nameservers*, se podría agregar como *dns-nameservers* en */etc/network/interfaces* (generalmente después de la línea de *gateway*).

4.2.2. El archivo *host.conf*

El archivo */etc/host.conf* contiene información específica para la resolución de nombres. Su importancia reside en indicar dónde se resuelve primero la dirección o el nombre de un nodo. Esta consulta puede hacerse al servidor DNS o a tablas locales dentro de la máquina actual (*/etc/hosts*), y su formato incluye las siguientes palabras claves *order*, *trim*, *multi*, *nospoof*, *spoof* y *reorder*, siendo las más habituales las siguientes:

- **order**. Indica el orden de cómo se realizará la búsqueda del nombre (por ejemplo, *bind*, *hosts*, *nis*).

- **multi**. Que puede ser *on*, por lo cual retornará todas las direcciones válidas para un *host* que esté en */etc/hosts file* en lugar de solo devolver el primero (*off*). Si está en *on* puede causar retardos importantes cuando */etc/hosts* sea de gran tamaño.

- **nospoof/spoofalert/spoof** (*on/off*). Relacionado con cuestiones de seguridad con relación a prevenir el *hostname spoofing* realizado por algunas aplicaciones.

Ejemplo de *letc/host.conf*

```
order hosts,bind
multi on
```

Esta configuración indica que primero se verifique el */etc/hosts* antes de solicitar una petición al DNS y también indica (2.ª línea) que retorne todas las direcciones válidas que se encuentren en */etc/hosts*. Por lo cual, el archivo */etc/hosts* es donde se colocan las direcciones locales o también sirve para acceder a nodos sin tener que consultar al DNS.

La consulta es mucho más rápida, pero tiene la desventaja de que si el nodo cambia, la dirección será incorrecta. En un sistema correctamente configurado, solo deberán aparecer los nodos locales y una entrada para la interfaz *loopback*.

4.2.3. El archivo *letc/hosts*

Este archivo actúa como servidor de nombres y es especialmente útil en una red local no exista una alta variabilidad de las IP asignadas a los nombres:

```
127.0.0.1 localhost
192.168.168.254 nteum.remix.local nteum
# Nodos red privada
192.168.168.1 nodo1.remix.local nodo1
192.168.168.2 nodo2.remix.local nodo2
# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Para el nombre de una máquina pueden utilizarse *alias*, que significa que esa máquina puede llamarse de diferentes maneras para la misma dirección IP. En referencia a la interfaz *loopback*, este es un tipo especial de interfaz que permite realizar a nodo conexiones consigo misma (por ejemplo, para verificar que el subsistema de red funciona sin acceder a la red). Por defecto, la dirección IP 127.0.0.1 ha sido asignada específicamente al *loopback* (un comando `ssh 127.0.0.1` conectará con la misma máquina). Su configuración es muy fácil (la realiza generalmente el *script* de inicialización de red). En muchos sistemas (Debian, incluido) se agrega en el archivo */etc/hosts* la siguiente línea: `127.0.1.1 <host_name>`. Esta se crea en aquellos sistemas que no tienen una IP permanente y para evitar que algún programa genere problemas (por ejemplo Gnome). La etiqueta <*host_name*> coincidirá con el nombre de la máquina definido en */etc/hostname*. Para un sistema con IP definida se debe comentar esta línea y se aconseja poner un dominio del tipo *fully qualified domain name (FQDN)* (el que figura en el DNS) o sino se tiene que insertar uno definido como <*host_name*>.<*domain_name*> en lugar de solo poner <*host_name*> (para máquinas que no tienen un registro en un DNS se aconseja poner como domino local o *localdomain*, por ejemplo nteum.local o nteum.localdomain).

Como se mencionó anteriormente, desde la versión 2 de la biblioteca **libc** existe un reemplazo importante con respecto a la funcionalidad del archivo *host.conf*. Esta mejora incluye la centralización de información de diferentes

servicios para la resolución de nombres, lo cual presenta grandes ventajas para el administrador de red. Toda la información de consulta de nombres y servicios ha sido centralizada en el archivo */etc/nsswitch.conf*, el cual permite al administrador configurar el orden y las bases de datos de modo muy simple.

En este archivo cada servicio aparece uno por línea con un conjunto de opciones, donde, por ejemplo, la resolución de nombres de nodo es una de ellas. En este se indica que el orden de consulta de las bases de datos para obtener el IP del nodo o su nombre será primero el servicio de DNS, que utilizará el archivo */etc/resolv.conf* para determinar la IP del nodo DNS, y en caso de que no pueda obtenerlo, utilizará el de las bases de datos local (*/etc/hosts*). Otras opciones para ello podrían ser *nis, nisplus*, que son otros servicios de información que se describirán en unidades posteriores. También se puede controlar por medio de acciones (entre []) el comportamiento de cada consulta, por ejemplo:

```
hosts: files mdns4_minimal [NOTFOUND=return] dns mdns4
```

Esto indica que cuando se realice la consulta al mdns4, si no existe un registro para esta consulta, retorne al programa que la hizo con un cero. Puede utilizarse el '!' para negar la acción, por ejemplo:

```
hosts dns [!UNAVAIL = return] files
```

4.3. Configuración del *routing*

Otro aspecto que hay que configurar es el *routing*. Si bien existe el tópico sobre su dificultad, generalmente se necesitan unos requerimientos de *routing* muy simples. En un nodo con múltiples conexiones, el *routing* consiste en decidir dónde hay que enviar y qué se recibe. Un nodo simple (una sola conexión de red) también necesita *routing*, ya que todos los nodos disponen de un *loopback* y una conexión de red (por ejemplo, Ethernet, PPP o SLIP). Como se explicó anteriormente, existe una tabla llamada *routing table*, que contiene filas con diversos campos, pero con tres de ellos sumamente importantes: **dirección de destino**, **interfaz** por donde saldrá el mensaje y **dirección IP**, que efectuará el siguiente paso en la red (*gateway*).

Nota

Consulta de tablas de *routing*: `route -n` o también `netstat -r` (o también `ip route`).

El comando *route* permite modificar esta tabla para realizar las tareas de *routing* adecuadas. Cuando llega un mensaje, se mira su dirección destino, se compara con las entradas en la tabla y se envía por la interfaz, en la que la dirección coincide mejor con el destino del paquete. En caso contrario, si está especificado un *gateway*, el mensaje se enviará por la interfaz adecuada.

Consideremos, por ejemplo, que nuestro nodo está en una red clase C con dirección 192.168.110.0 y tiene una dirección 192.168.110.23; y el *router* con conexión a Internet es 192.168.110.3.

En primer lugar, debemos configurar la interfaz:

```
ifconfig eth0 192.168.110.23 netmask 255.255.255.0 up
```

Más adelante, debemos indicar que todos los paquetes con direcciones 192.168.0.* deben ser enviados al dispositivo de red:

```
route add -net 192.1 ethernetmask 255.255.255.0 eth0
```

El *-net* indica que es una ruta de red pero también puede utilizarse *-host* 192.168.110.3. Esta configuración permitirá conectarse a todos los nodos dentro del segmento de red (192.1), pero ¿qué pasará si se desea conectar con otro nodo fuera de este segmento? Sería muy difícil tener todas las entradas adecuadas para todas las máquinas a las cuales se quiere conectar. Para simplificar esta tarea, existe el *default route*, que se utiliza cuando la dirección destino no coincide en la tabla con ninguna de las entradas. Una posibilidad de configuración sería:

```
route add default gw 192.168.110.3 eth0
```

Nota

El gw es la IP o nombre de un *gateway* o nodo *router*.

Una forma alternativa de hacerlo es:

```
ifconfig eth0 inet down     deshabilito la interfaz
ifconfig
lo Link encap:Local Loopback  ... (no mostrará ninguna entrada para eth0)
route   ... (no mostrará ninguna entrada en la tabla de rutas)
```

Luego se habilita la interfaz con una nueva IP y una la nueva ruta:

```
      ifconfig eth0 inet up 192.168.0.111 \
      netmask 255.255.0.0 broadcast 192.168.255.255
      route add -net 10.0.0.0 netmask 255.0.0.0 \
      gw 192.168.0.1 dev eth0
```

La barra (\) indica que el comando continúa en la siguiente línea. El resultado:

```
ifconfig
      eth0 Link encap:Ethernet HWaddr 08:00:46:7A:02:B0
      inet addr:192.168.0.111 Bcast: 192.168.255.255 Mask:255.255.0.0
      UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
```

```
        ...
        lo Link encap:Local Loopback
        inet addr:127.0.0.1 Mask:255.0.0.0
        ...
  route
  Kernel IP routing table
        Destination Gateway Genmask Flags Metric Ref Use Iface
        192.168.0.0 * 255.255.0.0 U 0 0 0 eth0
        10.0.0.0 192.168.0.1 255.0.0.0 UG 0 0 0 eth0
```

Para más información ver los comandos *ifconfig*(8) y *route*(8). Como hemos indicado anteriormente el comando `route` forma parte del paquete *net-tools*, el cual para algunas distribuciones es obsoleto. Es por ello que el nuevo paquete *iproute2* incluye bajo el mismo comando `ip` todas las posibilidades para configurar el *routing*, por ejemplo: `ip route`.

default via 10.0.2.2 dev eth0
10.0.2.0/24 dev eth0 proto kernel scope link src 10.0.2.15
172.16.1.0/24 dev eth1 proto kernel scope link src 172.16.1.1
172.16.2.0/24 via 172.16.1.2 dev eth1

Para añadir una ruta `ip route add` y para borrarla `ip ro del`, por ejemplo:

```
ip route add 10.0.0.0/16 via 172.16.1.1
```

```
ip route del 10.0.0.0/16 via 172.16.1.1
```

Para determinar por dónde se hará el *routing* de un paquete para una determinada IP se puede ejecutar:

```
ip route get 8.8.8.8.
```

8.8.8.8 via 10.0.2.2 dev eth0 src 10.0.2.15

Para agregar un *default gateway* se puede ejecutar:

```
ip route add default via 10.0.2.15.
```

Para visualizarlo ejecutamos `ip r` (es equivalente a `ip route show`) y mostrará:

default via 10.0.2.2 dev eth0

4.4. Configuración de servicios

El siguiente paso en la configuración de la red es la configuración de los servidores y servicios que permitirán a otro usuario acceder a la máquina local o a sus servicios. Los programas servidores utilizarán los puertos para escuchar las peticiones de los clientes, que se dirigirán a este servicio como *IP:port*.

En la mayoría de los sistemas GNU/Linux existe dos alternativas: `inetd` (original y obsoleto) y `xinetd` (reemplazo para *inetd* y con mejores prestaciones/seguridad).

4.4.1. Configuración de inetd

Los servidores pueden funcionar de dos maneras diferentes: *standalone* (en esta manera el servicio escucha en el puerto asignado y siempre se encuentra activo) o a través del `inetd`. Hay que tener en cuenta que en muchas instalaciones este paquete *inetd* no está instalado por defecto y se deberá ejecutar la instrucción `apt-get install inetutils-inetd`. No obstante, se recomienda instalar el paquete `xinetd` (`apt-get install xinetd`) que presenta mejores prestaciones/seguridad.

El **inetd** es un servidor que controla y gestiona las conexiones de red de los servicios especificados en el archivo */etc/inetd.conf*, el cual, ante una petición de servicio, pone en marcha el servidor adecuado y le transfiere la comunicación.

Dos archivos importantes necesitan ser configurados. Son estos: */etc/services* y */etc/inetd.conf*. En el primero se asocian los servicios, los puertos y el protocolo y en el segundo, qué programas servidores responderán ante una petición a un puerto determinado. El formato de */etc/services* es *name port/protocol aliases*, donde el primer campo es nombre del servicio, el segundo, el puerto donde atiende este servicio y el protocolo que utiliza, y el siguiente, un alias del nombre. Por defecto existen una serie de servicios que ya están preconfigurados. A continuación se muestra un ejemplo de */etc/services* (# indica que lo que existe a continuación es un comentario):

```
tcpmux          1/tcp           # TCP port service multiplexer
echo            7/tcp
echo            7/udp
discard         9/tcp           sink null
discard         9/udp           sink null
systat          11/tcp          users
...
ftp             21/tcp
ssh             22/tcp          # SSH Remote Login Protocol
```

```
ssh             22/udp       # SSH Remote Login Protocol
telnet          23/tcp
                             # 24 - private
smtp            25/tcp       mail
...
```

El archivo */etc/inetd.conf* es la configuración para el servicio maestro de red (*inetd server daemon*). Cada línea contiene siete campos separados por espacios: *service socket_type proto flags user server_path server_args*, donde service es el servicio descrito en la primera columna de */etc/services*, *socket_type* es el tipo de socket (valores posibles *stream*, *dgram*, *raw*, *rdm*, o *seqpacket*), proto es el protocolo válido para esta entrada (debe coincidir con el de */etc/services*), flags indica la acción que tomar cuando existe una nueva conexión sobre un servicio que se encuentra atendiendo a otra conexión (*wait* le dice a inetd no poner en marcha un nuevo servidor o *nowait* significa que inetd debe poner en marcha un nuevo servidor). User será el usuario con el cual se identificará quién ha puesto en marcha el servicio, *server_path* es el directorio donde se encuentra el servidor y *server_args* son argumentos posibles que serán pasados al servidor. Un ejemplo de algunas líneas de */etc/inetd.conf* es (recordar que # significa comentario, por lo cual, si un servicio tiene # antes de nombre, significa que no se encuentra disponible):

```
...
telnet   stream   tcp   nowait   root   /usr/sbin/tcpd   /usr/sbin/in.telnetd
ftp      stream   tcp   nowait   root   /usr/sbin/tcpd   /usr/sbin/in.ftpd
# fsp    dgram    udp   wait     root   /usr/sbin/tcpd   /usr/sbin/in.fspd
shell    stream   tcp   nowait   root   /usr/sbin/tcpd   /usr/sbin/in.rshd
login    stream   tcp   nowait   root   /usr/sbin/tcpd   /usr/sbin/in.rlogind
# exec   stream   tcp   nowait   root   /usr/sbin/tcpd   /usr/sbin/in.rexecd ...
...
```

A partir de Debian Woody 3.0 r1, la funcionalidad de inetd ha sido reemplazada por **xinetd** (recomendable ya que presenta más opciones con relación a la configuración de servicios y seguridad), el cual necesita el archivo de configuración */etc/xinetd.conf* (ver subapartado siguiente) y el directorio */etc/xinetd.d*.

Además de la configuración de inetd o xinetd, la configuración típica de los servicios de red en un entorno de escritorio o servidor básico podría incluir además:

Ved también

Para ver más sobre la configuración típica de los servicios de red en un entorno de escritorio o servidor básico, podéis ver el módulo de servidores (módulo 2) de la asignatura *Administración avanzada de sistemas GNU-Linux*.

- **ssh**: conexión interactiva segura como reemplazo de telnet e incluye dos archivos de configuración */etc/ssh/ssh_config* (para el cliente) y */etc/ssh/sshd_config* (para el servidor).

- **exim**: agente de transporte de correo (MTA), incluye los archivos de configuración: */etc/exim/exim.conf*, */etc/mailname*, */etc/aliases*,

/etc/email-addresses.

- **fetchmail**: daemon para descargar el correo de una cuenta POP3, */etc/fetchmailrc*.
- **procmail**: programa para filtrar y distribuir el correo local, *~/.procmailrc*.
- **tcpd**: servicios de filtros de máquinas y dominios habilitados y deshabilitados para conectarse al servidor (*wrappers*): */etc/hosts.allow, /etc/hosts.deny*.
- **DHCP**: servicio para la gestión (servidor) u obtención de IP (cliente), */etc/dhcp3/dhclient.conf* (cliente), */etc/default/dhcp3-server* (servidor), */etc/dhcp3/dhcpd.conf* (servidor). Los directorios/nombre pueden variar en función del paquete instalado.
- **DNS**: servicio de nombres, por ejemplo, `dnsmaq` en */etc/dnsmasq.d*.
- **CVS**: sistema de control de versiones concurrentes, */etc/cvs-cron.conf*, */etc/cvs-pserver.conf*.
- **NFS**: sistema de archivos de red, */etc/exports*.
- **Samba**: sistema de archivos de red y compartición de impresoras en redes Windows, */etc/samba/smb.conf*.
- **CUPS**: sistema de impresión, */etc/cups/**
- **Apache2**: servidor de web, */etc/apache2/**.
- **squid**: servidor *proxy-caché, /etc/squid/**.

4.4.2. Configuración del xinetd

Xinetd es un *super-daemon* que mejora notablemente la eficacia y prestaciones de *inetd* y *tcp-wrappers*. Una de las grandes ventajas de Xinetd es que puede hacer frente a ataques de DoA[28] a través de mecanismos de control para los servicios basados en la identificación de direcciones del cliente, en tiempo de acceso y tiempo de conexión (*logging*). No se debe pensar que Xinetd es el más adecuado para todos los servicios (por ejemplo, SSH es mejor que se ejecute como *daemon*), ya que muchos de ellos generan una gran sobrecarga al sistema y disponen de mecanismos de acceso seguros que no crean interrupciones en la seguridad del sistema [Xin].

[28] Del inglés *denial-of-access*.

Su configuración es muy simple y se realiza a través del archivo */etc/xinetd.conf* (el cual puede incluir más archivos del directorio */etc/xinetd.d/* para una mejor estructuración). En este archivo encontraremos una sección de *defaults* (pará-

metros que se aplicarán a todos los servicios) y *service* (o archivos que las contienen de *service*), que serán los servicios que se pondrán en marcha a través de Xinetd. Un ejemplo típico de la configuración podría ser que normalmente los *defaults* se colocan en *xinetd.conf* y los *services* en archivos separados en el directorio */etc/xinetd.d*, pero en este ejemplo lo hemos puesto todo junto:

```
# xinetd.conf
# Se aplican a todos los servidores y pueden modificarse para cada servicio
defaults
{
   instances   = 10
   log_type   = FILE /var/log/service.log
   log_on_success  = HOST PID
   log_on_failure  = HOST RECORD
   }

# El nombre del servicio debe encontrarse en /etc/services para obtener el puerto correcto
# Si se trata de un servidor/puerto no estándar, usar "port = X"
service ftp
{
   socket_type  = stream
   protocol   = tcp
   wait   = no
   user   = root
   server   = /usr/sbin/proftpd
   }

service ssh
{
   socket_type  = stream
   protocol   = tcp
   wait   = no
   user   = root
   port   = 22
   server   = /usr/sbin/sshd server_args = -i
   }
# This is the tcp version.
service echo
{
   disable  = yes
   type  = INTERNAL
   id  = echo-stream
   socket_type = stream
   protocol = tcp
   user  = root
   wait  = no
   }
```

```
# This is the udp version.
service echo
{
   disable  = yes
   type  = INTERNAL
   id  = echo-dgram
   socket_type = dgram
   protocol = udp
   user  = root
   wait  = yes
   }
```

Los servicios comentados (#) no estarán disponibles. En la sección *defaults* se pueden insertar parámetros tales como el número máximo de peticiones simultáneas de un servicio, el tipo de registro (*log*) que se desea tener, desde qué nodos se recibirán peticiones por defecto, el número máximo de peticiones por IP que se atenderán, o servicios que se ejecutarán como superservidores (*imapd* o *popd*), como por ejemplo:

```
default {
   instances                        = 20
   log_type                         = SYSLOG
   authpriv log_on_success  = HOST
   log_on_failure                   = HOST
   only_from                        = 192.168.0.0/16
   cps                              = 25 30
   enabled = imaps
   }
```

Un parámetro interesante es *cps*, que limita el número de las conexiones entrantes con 2 argumentos: el primero el número de conexiones por segundo a manejar por el servicio y si es mayor, el servicio quedará deshabilitado por el número de segundos indicado en el segundo argumento. Por defecto son 50 conexiones y un intervalo de 10 segundos que se consideran parámetros adecuados para contener una ataque de DoS.

La sección service, una por cada servicio, puede contener parámetros específicos –y a veces muy detallados– del servicio,como por ejemplo:

```
service imapd
{
   socket_type                      = stream
   wait                             = no
   user                             = root
   server                           = /usr/sbin/imapd
   only_from                        = 0.0.0.0/0 #allows every client
```

```
    no_access                          = 192.168.0.1
    instances                          = 30
    log_on_success                    += DURATION USERID
    log_on_failure                    += USERID
    nice                               = 2
    redirect                           = 192.168.1.1 993
    #Permite redireccionar el tráfico del port 993
    #hacia el nodo 192.168.1.1
    bind                               = 192.168.10.4
    #Permite indicar a qué interfaz está asociado el servicio para evitar
    # problemas de suplantación de servicio.
    }
```

4.5. Configuración adicional: protocolos y *networks*

Existen otros archivos de configuración que en la mayoría de los casos no se utilizan pero que pueden ser interesantes. El */etc/protocols* es un archivo que relaciona identificadores de protocolos con nombres de protocolos, así, los programadores pueden especificar los protocolos por sus nombres en los programas.

Ejemplo de */etc/protocols*

```
ip          0     IP   # internet protocol, pseudo protocol number
#hopopt     0     HOPOPT   # IPv6 Hop-by-Hop Option [RFC1883]
icmp        1     ICMP   # internet control message protocol
```

El archivo */etc/networks* tiene una función similar a */etc/hosts*, pero con respecto a las redes, indica nombres de red con relación a su dirección IP (el comando *route* o *ip route* mostrará el nombre de la red y no su dirección en este caso).

Ejemplo de */etc/networks*

```
loopnet 127.0.0.0
localnet 192.168.0.0
amprnet 44.0.0.0 ...
```

4.6. Aspectos de seguridad

Es importante tener en cuenta los aspectos de seguridad en las conexiones a red, ya que una fuente de ataques importantes se produce a través de la red. Ya se hablará más sobre este tema en la unidad correspondiente a seguridad; sin embargo, hay unas cuantas recomendaciones básicas que deben tenerse en cuenta para minimizar los riesgos inmediatamente antes y después de configurar la red de nuestro ordenador:

a) No activar servicios en */etc/inetd.conf* o */etc/xinetd.conf* que no se utilizarán, insertar un "#" antes del nombre para evitar fuentes de riesgo.

b) Modificar el archivo */etc/ftpusers* para denegar que ciertos usuarios puedan tener conexión vía ftp con su máquina.

c) Modificar el archivo */etc/securetty* para indicar desde qué terminales (un nombre por línea), por ejemplo: tty1 tty2 tty3 tty4, se permite la conexión del superusuario (*root*). Desde los terminales restantes, *root* no podrá conectarse.

d) Si se utiliza *inetd* (*xinetd* lo incluye por defecto y solo se debe configurar), entonces hay que agregar el *daemon* `tcpd`. Este servidor es un *wrapper* que permite aceptar-negar un servicio desde un determinado nodo y se coloca en el archivo */etc/inetd.conf* como intermediario de un servicio. El **tcpd** verifica unas reglas de acceso en dos archivos: */etc/hosts.allow /etc/host.deny*

Si se acepta la conexión, pone en marcha el servicio adecuado pasado como argumento, por ejemplo, la línea del servicio de ftp antes mostrada en *inetd.conf: ftp stream tcp nowait root /usr/sbin/tcpd /usr/sbin/in.ftpd*. *tcpd* primero busca el archivo */etc/hosts.allow* y luego */etc/hosts.deny*. El archivo *hosts.deny* contiene la información sobre cuáles son los nodos que no tienen acceso a un servicio dentro de esta máquina. En general, una configuración restrictiva es ALL: ALL, ya que solo se permitirá el acceso a los servicios desde los nodos declarados en el archivo */etc/hosts.allow*.

e) El archivo */etc/hosts.equiv* permite el acceso a esta máquina sin tener que introducir una clave de acceso (*password*). Se recomienda no usar este mecanismo y aconsejar a los usuarios no utilizar el equivalente desde la cuenta de usuario a través del archivo *.rhosts*.

f) En Debian es importante configurar */etc/security/access.conf*, el archivo que indica las reglas de quién y desde dónde se puede conectar (*login*) a esta máquina. Este archivo tiene una línea por orden con tres campos separados por ':' del tipo *permiso: usuarios:origen*. El primero será un + o - (acceso denegado), el segundo un nombre de usuario/s, grupo o *user@host*, y el tercero un nombre de un dispositivo, nodo, dominio, direcciones de nodo o de redes, o ALL.

Ejemplo de *access.conf*

Este comando no permite root logins sobre tty1:

```
ALL EXCEPT root:tty1 ...
```

Permite acceder a *u1*, *u2*, *g1* y todos los de dominio remix.cat:

```
+:u1 u2 g1 .remix.com:ALL
```

4.7. Opciones del IP

Existen una serie de opciones sobre el tráfico IP que es conveniente mencionar. Su configuración se realiza a través de la inicialización del archivo correspondiente en directorio */proc/sys/net/ipv4/*. El nombre del archivo es el mismo

que el del comando y para activarlos se debe poner un 1 dentro del archivo, y un 0 para desactivarlo. Por ejemplo, si se quiere activar *ip_forward*, se debería ejecutar:

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

Los más utilizados son: *ip_forward* utilizado para el routing entre interfaces o con *IP Masquerading; ip_default_ttl,* que es el tiempo de vida para un paquete IP (64 milisegundos por defecto) *ip_bootp_agent* variable lógica (booleana), que acepta paquetes (o no) con dirección origen del tipo 0.b.c.d y destino de este nodo, *broadcast* o *multicast*.

Estas opciones también se pueden modificar estáticamente a través del archivo */etc/sysctl.conf* (mirar en */etc/sysctl.d/* para variables adicionales y *sysctl.conf* (5) para mayor información).

En caso de que no sea necesario trabajar con IPv6, se pueden deshabilitar haciendo:

```
echo "net.ipv6.conf.all.disable_ipv6 = 1" >> /etc/sysctl.conf
```

Y para mirar los cambios:

```
sysctl -p
net.ipv6.conf.all.disable_ipv6 = 1
```

4.8. Comandos para la solución de problemas con la red

Si tiene problemas en la configuración de la red, se puede comenzar verificando la salida de los siguientes comandos para obtener una primera idea:

```
ifconfig                          (o también ip a)
cat /proc/pci
cat /proc/interrupts
dmesg | more
```

Para verificar la conexión a la red, se pueden utilizar los siguientes comandos (se debe tener instalado `netkit-ping`, `traceroute`, `dnsutils`, `iptables` y `net-tools`):

```
ping uoc.edu                          # verificar la conexión a Internet
traceroute uoc.edu                    # rastrear paquetes IP
ifconfig                              # (o ip address) verificar la configuración del host
route -n (o ip route)                 # verificar la configuración de la ruta
dig [@dns.uoc.edu] www.uoc.edu        # verificar registros de www.uoc.edu
                                      # sobre el servidor dns.uoc.edu
iptables -L -n |less                  # verificar filtrado de paquetes (kernel >=2.4)
```

```
netstat -a (o ss -a)                    # muestra todos los puertos abiertos
netstat -l --inet (o ss -l --query=inet) # muestra los puertos en escucha
netstat -ln --tcp (o ss -ln --query=tcp) # mostrar puertos tcp en escucha (numérico)</p>
```

Entre toda ella ampliaremos la explicación de `netstat` (forma parte del paquete *net-tools*) ya que es una de las herramientas que nos muestra un informe instantáneo de la actividad de red. Cuando se ejecuta sin parámetros, nos mostrará todas las conexiones abiertas, que puede ser una lista demasiado detallada, ya que incluye algunos puntos de conexión que no son de red (la comunicación de *dbus*, por ejemplo). No obstante, cuenta con gran cantidad de parámetros para definir la salida. Entre los más utilizados podemos enumerar:

-t. Muestra conexiones TCP

-u. Muestra conexiones UDP

-a. Muestra puertos en Listen (conexiones entrantes)

-n. Muestra IP, puertos e ID del usuario

-p. Muestra procesos involucrados (útil cuando se ejecuta como *root*)

-c. Actualiza dinámicamente la lista de conexiones

Es habitual ejecutar como `netstat -tupan`, pero es recomendable analizar el resto de parámetros para profundizar en los conceptos de monitorización de la red. El comando `ss` (`socket statistics`) es el que se incluye en *iproute2* y que presenta un conjunto de opciones. Entre las más interesantes destacan:

-n, --numeric: no resuelve los nombres de los servicios.

-r, --resolve: intenta resolver los números de puertos y direcciones.

-a, --all: todos los *sockets*.

-l, --listening: todos los *sockets* que están escuchando.

-e, --extended: información detallada.

-m, --memory: información de memoria utilizada.

-p, --processes: proceso que utilizan los *sockets*.

-i, --info: información interna de TCP.

-s, --summary: resumen

Por ejemplo:

```
ss -o state established      # muestra las conexiones establecidas.
ss --query=tcp src *:ssh # muestra las conexiones tcp en que la fuente sea ssh.
ss -ant | awk '{print $NF}' | grep -v '[a-z]' | sort | uniq -c  # muestra el número y
tipo de conexiones activas.
ss -a     # muestra un resumen del estado de los sockets
ss -tlnp     # muestra los puertos en escucha (listen) y el PID del programa
```

5. Configuración del DHCP

5.1. DHCP: cliente y servidor

DHCP son las siglas de *dynamic host configuration protocol*. Su configuración es muy simple y sirve para que, en lugar de configurar cada nodo de una red individualmente, se pueda hacer de forma centralizada y su administración sea más fácil. La configuración de un cliente es muy fácil, ya que solo se debe instalar el siguiente paquete: *dhcp-client* (por ejemplo, en Debian *isc-dhcp-client*) agregando la palabra *dhcp* en la entrada correspondiente a la interfaz que se desea que funcione bajo el cliente *dhcp* (por ej., */etc/network/interfaces* debe tener `iface eth0 inet dhcp`).

La configuración del servidor requiere un poco más de atención, pero no presenta complicaciones. Primero, para que el servidor pueda servir a todos los clientes DHCP (incluido Windows).

En algunas distribuciones no está habilitado el poder enviar mensajes a la dirección 255.255.255.255. Para probarlo, ejecútese:

```
route add -host 255.255.255.255 dev eth0
```

Si aparece el siguiente mensaje *255.255.255.255: Unknown host*, debe añadirse la siguiente entrada en */etc/hosts*: *255.255.255.255 dhcp* e intentar nuevamente:

```
route add -host dhcp dev eth0
```

La instalación en, p. ej, Debian es `apt-get install isc-dhcp-server`. En general, la configuración de *dhcpd* se puede realizar haciendo la modificación del archivo */etc/dhcp/dhcpd.conf*. Un ejemplo de este archivo es:

```
# Ejemplo de /etc/dhcp/dhcpd.conf:
default-lease-time 1200;
max-lease-time 9200;
option domain-name "remix.cat";
deny unknown-clients;
deny bootp;
option broadcast-address 192.168.11.255;
option routers 192.168.11.254;
option domain-name-servers 192.168.11.1, 192.168.168.11.2;
subnet 192.168.11.0 netmask 255.255.255.0 {
   not authoritative;
```

```
        range 192.168.11.1 192.168.11.254

        host marte {
           hardware ethernet 00:00:95:C7:06:4C;
           fixed address 192.168.11.146;
           option host-name "marte";
           }
        host saturno {
           hardware ethernet 00:00:95:C7:06:44;
           fixed address 192.168.11.147;
           option host-name "saturno";
           }
     }
```

Esto permitirá al servidor asignar el rango de direcciones 192.168.11.1 al 192.168.11.254 tal y como se describe cada nodo. Si no existe el segmento *host* { ... } correspondiente, se asignan aleatoriamente. Las IP son asignadas por un tiempo mínimo de 1.200 segundos y máximo de 9.200 (en caso de no existir estos parámetros, se asignan indefinidamente).

Ejecutaremos */etc/init.d/isc-dhcpserver start* para poner en marcha el servidor o también

```
service isc-dhcp-server start
```

(o `systemctl start isc-dhcp-server`). Con `/usr/sbin/dhcpd -d -f` se podrá ver la actividad del servidor sobre la consola del sistema.

5.2. DHCP Relay

Como el protocolo DHCP se realiza sin contar con una dirección de origen y utilizando *broadcast* como destino de las solicitudes (evento de *discovery* generado por el cliente DHCP), no se puede hacer un *routing* hacia otras redes (lo cual es habitual si tenemos las redes segmentadas por otras máquinas haciendo de *routers*), ya que el protocolo DHCP considera que el servidor y el cliente están en la misma red. Para llegar desde una red a otra al servidor de dhcp (y así evitar tener que poner un servidor por cada subred) se puede utilizar un agente intermedio llamado DHCP relay. Un DHCP relay funciona en la máquina que hace de *router* y que recibe las solicitudes de los clientes en formato de *broadcast* y las reenvía como *unicast* a la dirección del servidor DHCP. Su configuración es muy simple y solo se debe instalar el paquete

```
apt-get install isc-dhcp-relay
```

y a continuación configurar el archivo */etc/default/isc-dhcp-relay* agregando *SERVERS="176.16.1.1"* que es la dirección donde se deberán enviar las peticiones. El archivo contiene otras opciones, como definir la interfaz dónde escuchar las peticiones (si no se indica es en todas) y otra de opciones para pasarle al servidor de DHCP.

6. Múltiples IP sobre una interfaz

Existen algunas aplicaciones donde es útil configurar múltiples direcciones IP a un único dispositivo de red. Los ISP[29] utilizan frecuentemente esta característica para proveer de características personalizadas (por ejemplo, de World Wide Web y FTP) a sus usuarios. Para ello, el *kernel* debe estar compilado con las opciones de *Network Aliasing* e *IP* (estas opciones ya están activadas por defecto en los kernels actuales).

[29] Del inglés *internet service providers*

Los alias son anexados a dispositivos de red virtuales asociados al nuevo dispositivo con un formato tal como:

```
dispositivo: número virtual
```

Por ejemplo:

```
eth0:0, ppp0:8
```

Una configuración habitual sería en */etc/network/interfaces* asignar por ejemplo tres IP a **eth0**:

```
auto eth0
allow-hotplug eth0
iface eth0 inet static
    address 192.168.1.42
    netmask 255.255.255.0
    gateway 192.168.1.1

auto eth0:0
allow-hotplug eth0:0
iface eth0:0 inet static
    address 192.168.1.43
    netmask 255.255.255.0

auto eth0:1
allow-hotplug eth0:1
iface eth0:1 inet static
    address 192.168.1.44
    netmask 255.255.255.0
```

También se puede hacer a través de la línea de comandos:

```
ifconfig eth0 192.168.1.42 netmask 255.255.255.0 up
ifconfig eth0:0 192.168.1.43 netmask 255.255.255.0 up
```

```
..
```

Lo cual significa que tendremos dos IP 192.168.1.42 y 192.168.1.43 para la misma interfaz. Para borrar un alias, agregar un '-' al final del nombre (por ejemplo, *ifconfig eth0:0- 0*).

Un caso típico es que se desee configurar una única tarjeta Ethernet para que sea la interfaz de distintas subredes IP. Por ejemplo, supongamos que se tiene una máquina que se encuentra en una red LAN 192.168.0.0/24, y se desea conectar la máquina a Internet usando una dirección IP pública proporcionada con DHCP usando su tarjeta Ethernet existente. Por ejemplo, se puede hacer como en el ejemplo anterior o también editar el archivo */etc/network/interfaces*, de modo que incluya una sección similar a la siguiente:

```
iface eth0 inet static
   address 192.168.0.1
   netmask 255.255.255.0
   network 192.168.0.0
   broadcast 192.168.0.255

iface eth0:0 inet dhcp
```

La interfaz **eth0:0** es una interfaz virtual y al activarse, también lo hará su padre, **eth0**.

Una interface "alias" podría no tener *gateway* ni *dns-nameservers* y la asignación dinámica de IP también es posible.

7. IP *Network Address Translation* (NAT)

El NAT es un recurso para que un conjunto de máquinas configuradas en una red interna puedan utilizar una única dirección IP como *gateway*. Esto permite que los nodos en una red local, por ejemplo, puedan salir hacia Internet (es decir, aquellos que utilizan una IP privada, por ejemplo, 198.162.10.1); pero no pueden aceptar llamadas o servicios del exterior directamente, sino a través de la máquina que tiene la IP real.

El IP *Network Address Translation* (NAT) es el reemplazo de lo que se conocía como *IP Masquerade* y forma parte del *kernel* (el cual debe estar configurado con IP_ADVANCED_ROUTER, IP_MULTIPLE_TABLES _IP_ROUTE_NAT, IP_FIREWALL y IP_ROUTE_FWMARK, pero es la configuración habitual de los *kernels* actuales).

Hoy en día son pocos los servicios que no pueden ejecutarse en una red privada con acceso del exterior y otros deberán ser configurados en modo PASV (pasivo) para que funcionen; sin embargo, WWW, telnet o irc funcionan adecuadamente.

El caso más habitual, hoy en día, es tener un conjunto de máquinas virtualizadas, con una de ellas haciendo de *gateway* y, por ejemplo, conectada al *host* a través de NAT, y las otras en una red interna privada (que a su vez harán NAT con el *gateway*) pero puede extrapolarse a dispositivos físicos de acuerdo a la configuración del hardware de que dispongamos.

Como vimos, y de acuerdo a la RFC 1918, se pueden utilizar como IP privadas los siguientes rangos de direcciones (IP/ Mask): 10.0.0.0/255.0.0.0, 172.16.0.0/255.240.0.0, 192.168.0.0/255.255.0.0.

Los nodos que deben ser ocultados (*masqueraded*) estarán dentro de esta segunda red. Cada una de estas máquinas debería tener la dirección de la máquina que realiza el *masquerade* como *default gateway* o *router*. Sobre esta máquina (*gateway*) podemos configurar una interfaz para que "mire" hacia la red interna (por ejemplo, **eth1**) y otra, hacia la red externa (por ejemplo, **eth0**):

- *Network route* para Ethernet considerando que la red tiene un IP=192.168.1.0/255.255.255.0:

```
route add -net 192.168.1.0 netmask 255.255.255.0 eth0
```

- *Default route* para el resto de Internet:

```
route add default eth0
```

- Todos los nodos sobre la red 192.168.1/24 serán *masqueraded*:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Para automatizar el proceso y no tener que ejecutarlo durante cada boot del sistema, podemos hacer:

- Modificar el archivo */etc/sysctl.conf* con `net.ipv4.ip_forward=1` (o con el comando `sysctl net.ipv4.ip_forward=1`).

- Agregar la regla (se puede incluir "source -s" o no):

```
iptables -t nat -A POSTROUTING -s 192.168.0.0/24 -o eth0 \
-j MASQUERADE
```

- Guardar reglas de iptables: `iptables-save > /etc/default/iptables`

- Descargar *script* de init 'iptables' https://github.com/Sirtea/iptables-init-debian.

- Iniciar reglas y test: `/etc/init.d/iptables start`.

- Para mirar las reglas: `iptables -L -n` y `iptables -t nat -L -n`.

- Para verificar que todo esté bien desde una máquina interna ejecutamos la orden `ping google.com`.

- Si es necesario instalarlas en runlevel 2: `update-rc.d [-n] iptables defaults` (mirar donde están las reglas */etc/default/iptables*).

Es necesario recordar que se deberán agregar las reglas de *routing* adecuadas en caso de que sea necesario redirigir los paquetes hacia interfaces internas. Una herramienta útil para observar por dónde están pasando los paquetes y detectar los problemas es el `tcpdump`, que es muy fácil de utilizar.

Existe otra forma de realizar NAT a través del paquete `iproute2` (disponible en Debian). Este paquete contiene un conjunto de herramientas que reemplaza las funcionalidades de `ifconfig`, `route` y `arp`. La herramienta principal del paquete `iproute2` es `ip`, su sintaxis es `ip [opciones] objeto [comando [argumentos]]`. La forma más habitual es el comando `ip route` que administra las entradas de `route` con las tablas de encaminamiento que maneja el núcleo [DR]. Por ejemplo:

- `ip route add 10.0.0.0/24 via 192.168.0.2`. Agregar un camino hacia la red 10.0.0/24 por la puerta de enlace 192.168.0.2.

- `ip route add nat 158.109.64.1 via 192.168.0.3`. Para aplicar NAT antes de transmitir.

Es decir, `ip route add nat <extaddr>[/<masklen>] via <intaddr>` permitirá que un paquete de entrada destinado a *ext-addr* (la dirección visible desde fuera de Internet) transcriba su dirección destino a *int-addr* (la dirección de su red interna por medio de su *gateway/firewall*). El paquete se encamina de acuerdo a la tabla local de *route*. Se pueden trasladar direcciones simples o bloques:

- `ip route add nat 158.109.64.1 via 192.109.0.2`. La dirección interna 192.109.0.2 será accesible como 158.109.64.1.

- `ip route add nat 158.109.64.32/24 via 192.109.0.0`. El bloque 192.109.0.0-31 se trasladará a 158.109.64.32-63.

Este nuevo paquete (`iproute2`) reemplaza los comandos del paquete original (`net-tools`) y las equivalencias son:

Net-tools	Iproute2	Definición
ifconfig(8)	ip addr	(IP or IPv6) configuración de un dispositivo
route(8)	ip route	Entrada en la tabla de *routing*
arp(8)	ip neigh	Entrada en la tabla de ARP o NDISC
ipmaddr	ip maddr	Dirección *multicast*
iptunnel	ip tunnel	Tunel sobre IP
nameif(8)	ifrename(8)	Renombrado de *network interfaces*
mii-tool(8)	ethtool(8)	Parámetros del dispositivo Ethernet

8. Udev - *Linux dynamic device management*

udev es el mecanismo que reemplaza el antiguo *device file system* (DevFS) y permite identificar un dispositivo en base a sus propiedades, como por ejemplo, marca, modelo, dispositivo, y además se ejecuta en espacio de usuario. udev permite reglas para especificar qué nombre se le da a un dispositivo y hacer que este, por ejemplo, se visualice como un */dev/mi_nombre*.

udev está formado por unos servicios del *kernel services* (que le transfieren los eventos) y el *daemon udevd* (que traslada estos eventos en acciones que pueden ser configuradas mediante reglas). La inicialización se lleva a cabo mediante */etc/rcS.d/udev*, la configuración se almacena en */etc/udev/udev.conf* y las reglas son obtenidas de */run/udev/rules.d*, */etc/udev/rules.d* o */lib/udev/rules.d* (Debian utiliza en su mayor parte */lib/udev/rules.d*).

Uno de los aspectos interesantes de **udev** es que fue diseñado para responder a eventos de *hotplug*, como los que se definen en las interfaces de red con la etiqueta *allow-hotplug* y las definiciones de las reglas para los dispositivos de red se almacenan en */etc/udev/rules.d/70-persistent-net.rules*. Este mecanismo tiene algunos inconvenientes cuando se trabaja con máquinas virtuales clonadas o se cambia un dispositivo, ya que la configuración del dispositivo original está almacenada en este archivo y el sistema lo reconocerá como un nuevo dispositivo y, por lo tanto, lo reenumerará a continuación. Para solucionar este inconveniente, se debe borrar */etc/udev/rules.d/70-persistent-net.rules* y rearrancar el sistema. El comando para realizar la gestión de las reglas y el mecanismo `udev` es `udevadm` (por ejemplo, `udevadm info -a -n /dev/sda`).

Nota

Podéis obtener más información al respecto en [HeMa] http://debian-handbook.info/browse/stable/sect.hotplug.html.

Para escribir reglas podéis consultar http://www.reactivated.net/writing_udev_rules.html.

9. *Bridging*

El *bridging* es un método para compartir conexiones; por ejemplo, la conexión a Internet entre dos o más ordenadores, lo que es útil si no se dispone de un *router* con más de un puerto de Ethernet o está limitado por este número. Básicamente, se utiliza para que un ordenador que está conectado a Internet pueda, por otro dispositivo de red, conectarse a un segundo ordenador que no tiene conexión a Internet y proveerle de acceso a Internet. En sistemas físicos puede parecer extraño, pero es habitual cuando tenemos un conjunto de máquinas virtualizadas y queremos que todas tengan acceso a Internet.

El comando a utilizar es el `brctl` y está incluido en el paquete **bridge-utils** que deberemos instalar (`apt-get install bridge-utils`). Este nos permitirá configurar y utilizar una nueva interfaz (*bridge*), de la misma forma que **eth0**, **eth1**, etc., la cual no existe físicamente (es virtual) y, de forma transparente, obtendrá los paquetes de una interfaz y los trasladará a la otra.

La forma más simple de crear una interfaz es mediante `brctl addbr br0` y se le pueden agregar los dispositivos físicos con `brctl addif br0 eth0 eth1`.

Para hacerlo a través de */etc/network/interfaces*:

```
iface eth0 inet manual
 iface eth1 inet manual
 # Bridge setup
 iface br0 inet dhcp
        bridge_ports eth0 eth1
```

Si necesitamos configurarla con IP estática, cambiar en *br0*:

```
iface br0 inet static
        bridge_ports eth0 eth1
        address 192.168.1.2
        broadcast 192.168.1.255
        netmask 255.255.255.0
        gateway 192.168.1.1
```

Nota

Podéis obtener más información sobre configuración y parámetros (libvirt, wifi, configuración avanzada) en https://wiki.debian.org/BridgeNetworkConnections.

10. Sistema de nombres de dominio (*Domain Name System, DNS*)

La funcionalidad del servicio de DNS es convertir nombres de máquinas (legibles y fáciles de recordar por los usuarios) en direcciones IP o viceversa.

A la consulta de cuál es la IP de *nteum.remix.world*, el servidor responderá 192.168.0.1 (esta acción es conocida como *mapping*); del mismo modo, cuando se le proporcione la dirección IP, responderá con el nombre de la máquina (conocido como *reverse mapping*).

El *Domain Name System* (DNS) es una arquitectura arborescente que evita la duplicación de la información y facilita la búsqueda. Por ello, un único DNS no tiene sentido sino como parte del árbol. Una de las aplicaciones más utilizadas que presta este servicio se llama *named*, se incluye en la mayoría de distribuciones de GNU/Linux (`/usr/sbin/named`) y forma parte de un paquete llamado `bind` (actualmente versión 9.x), coordinado por el ISC (Internet Software Consortium). El DNS es simplemente una base de datos, por lo cual, es necesario que las personas que la modifiquen conozcan su estructura, ya que, de lo contrario, el servicio quedará afectado. Como precaución debe tenerse especial cuidado en guardar las copias de los archivos para evitar cualquier interrupción en el servicio. Los servidores DNS, que pueden convertir la mayoría de los nodos de DNS en su IP correspondiente y se les llama servidores DNS recursivos. Este tipo de servidor no puede cambiar los nombres de los nodos de DNS que hay, simplemente se preguntan otros servidores DNS la IP de un nodo DNS dado. Los servidores DNS autorizados pueden gestionar/cambiar las direcciones IP de los nodos DNS que gestionan y normalmente son con los cuales contactará un servidor DNS recursivo con el fin de conocer la IP de un nodo DNS dado. Una tercera variante de los servidores DNS son los DNS caché, que simplemente almacenan la información obtenida de otros servidores DNS recursivos.

10.1. Servidor de nombres caché

En primer lugar, se configurará un servidor de DNS para resolver consultas, que actúe como caché para las consultas de nombres (*resolver, caching only server*). Es decir, la primera vez consultará al servidor adecuado porque se parte de una base de datos sin información, pero las veces siguientes responderá el servidor de nombres caché, con la correspondiente disminución del tiempo de respuesta. Para instalar un servidor de esta características, instalamos los paquetes `bind9 y dnsutils` (p. ej., en Debian `apt-get bind9 dnsutils`).

Como se puede observar en el directorio */etc/bind* encontraremos una serie de archivos de configuración entre ellos el principal es named.conf que incluye a otros *named.conf.options, named.conf.local, named.conf.default-zones* (podemos encontrar que esta configuración puede variar entre distribuciones pero es similar). El archivo */etc/bind/named.conf.options* contiene las opciones generales para el servidor y los otros dos archivos nos servirán para configurar nuestras zonas en el servidor de nombres que veremos en el apartado siguiente. Para configurar nuestro *caching only server* debemos considerar que las pregunta en primer lugar las resolveremos nosotros pero como es evidente que no tenemos la respuesta se deberá redirigir la pregunta a servidores de DNS en la red. Para ello, es interesante considerar los de servicios como OpenDNS (http://www.opendns.com/opendns-ip-addresses/) que son las IP: 208.67.222.222 y 208.67.220.220 o bien servicios como los de Google (https://developers.google.com/speed/public-dns/?csw=1) que responden a las direcciones IP: 8.8.8.8 y 8.8.4.4. A continuación realizamos la modificación del archivo */etc/bind/named.conf.options* para quitar los comentarios de la sección *forwarders* poniendo lo siguiente:

```
forwarders {
   // OpenDNS servers
   208.67.222.222;
   208.67.220.220;
   // Podríamos incluir nuestro ISP/router -verificar la IP-
 192.168.1.1;
};
```

Se podrían agregar en el mismo archivo al final y reemplazando las que hay (se puede dejar esta configuración para un segundo paso) opciones de seguridad para que solo resuelva las peticiones de nuestra red:

```
// Security options
listen-on port 53 { 127.0.0.1; 192.168.1.100; };
allow-query { 127.0.0.1; 192.168.1.0/24; };
allow-recursion { 127.0.0.1; 192.168.1.0/24; };
allow-transfer { none; };
// Las siguientes dos opciones ya se encuentran por defecto en el
// archivo pero si no tenemos IPv6 podemos comentar la última.
auth-nxdomain no; # conform to RFC1035
// listen-on-v6 { any; };
```

Se puede verificar la configuración con `named-checkconf`. A continuación modificamos el archivo */etc/resolv.conf* para añadir `nameserver 127.0.0.1` para que las preguntas a DNS la librería *gethostbyname(3)* la realice al propio *host* y el archivo */etc/nsswitch.conf* verificando que la línea *hosts* quede como: `hosts: files dns` indicará el orden que se resolverán las peticiones (primero localmente al */etc/hosts* y luego al servidor DNS). Por último solo resta reiniciar

el servidor con `service bind9 restart` (o `systemctl restart bind9`) y hacer las pruebas de funcionamiento. En nuestro caso hemos ejecutado `dig www.debian.org` –se han omitido líneas para resumir la información–:

```
; <<>> DiG 9.9.5-9+deb8u6-Debian <<>> debian.org
...
;; QUESTION SECTION:
;debian.org. IN A
...
;; ANSWER SECTION:
debian.org. 300 IN A 149.20.20.22
debian.org. 300 IN A 128.31.0.62
...
;; Query time: 227 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Fri Jun 17 10:48:57 BST 2016
```

Donde podemos observar que la *query* ha tardado 227 milisegundos. Si la realizamos por segunda vez (ya ha quedado la consulta almacenada en nuestro servidor caché):

```
; <<>> DiG 9.9.5-9+deb8u6-Debian <<>> debian.org
...
;; QUESTION SECTION:
;debian.org. IN A

;; ANSWER SECTION:
debian.org. 289 IN A 149.20.20.22
debian.org. 289 IN A 140.211.15.34
...
;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Fri Jun 17 10:49:08 BST 2016
```

En este caso la consulta ha tardado 0 milisegundos (es decir, un tiempo inferior al que puede contabilizar el comando, que es de 0,001 segundos). verificando que el servidor caché ha funcionado. También podemos hacer la consulta inversa con `nslookup 130.89.148.14` lo cual nos dará el nombre de la máquina que tiene asignada esta IP (en este caso es el servidor al cual está asignado el dominio www.debian.org)

```
Server: 127.0.0.1
Address: 127.0.0.1#53
Non-authoritative answer:
14.148.89.130.in-addr.arpa name = klecker4.snt.utwente.nl.
Authoritative answers can be found from:
. nameserver = l.root-servers.net.
```

```
. nameserver = b.root-servers.net.
. nameserver = j.root-servers.net.
...
```

Es importante considerar si se desea instalar el servidor de DNS en una máquina virtual, al ser un servicio que se debe acceder desde fuera, el servidor DNS (y la mayoría de los servidores) no puede estar en NAT sobre un *host* sino que debe tener IP propia del segmento de red en la cual presta el servicio y por lo tanto el adaptador de red de la máquina virtual debe estar en modo puente (*bridged*). Para configurar los clientes dentro de nuestra red bastará con modificar */etc/resolv.conf* si son GNU/Linux (o modificar */etc/network/interfaces* para agregarlos como `dns-nameservers IP` si es que tiene instalado el paquete `resolvconf`) y en Windows modificar las propiedades IPv4 del adaptador para introducir la IP de nuestro servidor DNS caché.

10.2. Configuración de un dominio propio

DNS posee una estructura en forma de árbol y el origen es conocido como "." (ver `/etc/bind/db.root`). Bajo el "." existen los TLD (*Top Level Domains*) como **org**, **com**, **edu**, **net**, etc. Cuando se busca en un servidor, si este no conoce la respuesta, se buscará recursivamente en el árbol hasta encontrarla. Cada "." en una dirección (por ejemplo, nteum.remix.world) indica una rama del árbol de DNS diferente y un ámbito de consulta (o de responsabilidad) diferente que se irá recorriendo en forma recursiva de izquierda a derecha.

Otro aspecto importante, además del dominio, es el `in-addr.arpa` (*inverse mapping*), el cual también está anidado como los dominios y sirve para obtener nombres cuando se consulta por la dirección IP. En este caso, las direcciones se escriben al revés, en concordancia con el dominio. Si nteum.remix.world es la 192.168.0.30, entonces se escribirá como 30.0.168.192, en concordancia con nteum.remix.world para la resolución inversa de IP -> nombre. En este ejemplo utilizaremos un servidor de DNS atiende a una red interna (192.168.0.0/24) y luego tiene una IP pública pero a a fines del ejemplo y poder hacer las pruebas internas la configuraremos en un IP privada de otro segmento (172.16.0.80) y un dominio ficticio (remix.world) pero en el caso de llevar este servidor a producción se debería cambiar por la IP externa que tenga y el dominio correspondiente que se haga obtenido y tengamos a nuestra disposición.

Para configurar un servidor de nombres propio en primer lugar cambiaremos el archivo */etc/bind/named.conf* para definir dos zonas (una interna y otra externa). El archivo quedará como (observad que hemos comentado la línea que incluye named.conf.default-zones ya que la incluiremos después):

```
include "/etc/bind/named.conf.options";
include "/etc/bind/named.conf.local";
//include "/etc/bind/named.conf.default-zones";
include "/etc/bind/named.conf.internal-zones";
```

```
include "/etc/bind/named.conf.external-zones";
```

Modificamos el archivo *named.conf.internal-zones* para indicarle los archivos donde se encontrarán realmente los nombres de las máquinas que en este caso será dos: uno para para resolución nombre -> IP llamado remix.world.lan y otro para la resolución inversa IP -> nombre llamado 0.168.192.db.

```
view "internal" {
   match-clients {
      localhost;
      192.168.0.0/24;
   };
// set zone for internal
   zone "remix.world" {
      type master;
      file "/etc/bind/remix.world.lan";
      allow-update { none; };
   };
// set zone for internal reverse
   zone "0.168.192.in-addr.arpa" {
      type master;
      file "/etc/bind/0.168.192.db";
      allow-update { none; };
   };
include "/etc/bind/named.conf.default-zones";
};
```

De la misma forma modificamos el archivo *named.conf.external-zones* para indicarle los archivos que resolverán esta zona y que también serán dos: remix.world.wan y 80.0.16.172.db para la resolución directa e inversa respectivamente.

```
view "external" {
// define for external section
   match-clients { any; };
// allow any query
   allow-query { any; };
// prohibit recursion
   recursion no;
// set zone for external
   zone "remix.world" {
      type master;
      file "/etc/bind/remix.world.wan";
      allow-update { none; };
   };
// set zone for external reverse
   zone "80.0.16.172.in-addr.arpa" {
```

```
        type master;
        file "/etc/bind/80.0.16.172.db";
        allow-update { none; };
    };
};
```

También se debe modificar el *named.conf.options*:

```
options {
    directory "/var/cache/bind";
// ...
// forwarders {
// 158.109.0.9;
// 158.109.0.1;
// };
// query range you permit
    allow-query { localhost; 192.168.0.0/24; };
// the range to transfer zone files
    allow-transfer { localhost; 192.168.0.0/24; };
// recursion range you allow
// allow-recursion { localhost; 192.168.0.0/24; };

    dnssec-validation auto;
    auth-nxdomain no; # conform to RFC1035
//listen-on-v6 { any; };
};
```

Ahora es necesario definir los archivos que realmente resolverán las IP/nombres de las zonas. Comenzamos por *remix.server.lan*:

```
$TTL 86400
@ IN SOA nteum.remix.world. root.remix.world. (
    2013050601 ;Serial
    3600 ;Refresh
    1800 ;Retry
    604800 ;Expire
    86400 ;Minimum TTL
)

    IN NS nteum.remix.world.
    IN A 192.168.0.30
    IN MX 10 nteum.remix.world.

nteum IN A 192.168.0.30
```

Se debe tener en cuenta el "." al final de los nombres de dominio. El significado de los registros indican: **SOA** (*Start of Authority*) debe estar en todos los archivos de zona al inicio, después de `TTL` (*Time to Live*) que indica el tiempo en segundos que los recursos de una zona serán válidos, el símbolo `@` significa el origen del dominio; **NS**, el servidor de nombres para el dominio, **MX** (*Mail eXchange*), indica a donde se debe dirigir el correo para una determinada zona, y **A** es la Ip que se debe asignar a un nombre. Para el archivo de resolución inversa *0.168.192.db*:

```
$TTL 86400
@ IN SOA nteum.remix.world. nteum.remix.world. (
   2013050601 ;Serial
   3600 ;Refresh
   1800 ;Retry
   604800 ;Expire
   86400 ;Minimum TTL
)

   IN NS nteum.remix.world.
   IN PTR remix.world.
   IN A 255.255.255.0
30 IN PTR nteum.remix.world.
```

Donde podemos observar que los registros tienen un formato como el siguiente: `<último-dígito-IP> IN PTR <FQDN-of-system>` donde PTR (*Registro PoinTeR*) crea un apuntador hacia el nombre de la IP que coincide con el registro. De la misma forma procedemos para la zona externa con el archivo *remix.world.wan*:

```
$TTL 86400
@ IN SOA nteum.remix.world. root.remix.world. (
   2013050601 ;Serial
   3600 ;Refresh
   1800 ;Retry
   604800 ;Expire
   86400 ;Minimum TTL
)

   IN NS nteum.remix.world.
   IN A 172.16.0.82
   IN MX 10 nteum.remix.world.
nteum IN A 172.16.0.82
```

Y su resolución inversa *80.0.16.172.db*:

```
$TTL 86400
@ IN SOA nteum.remix.world. nteum.remix.world. (
```

```
    2013050601 ;Serial
    3600 ;Refresh
    1800 ;Retry
    604800 ;Expire
    86400 ;Minimum TTL
)

    IN NS nteum.remix.world.
    IN PTR remix.world.
    IN A 255.255.255.248
82 IN PTR nteum.remix.world.
```

Ahora solo restaría hacer los cambios oportunos en el archivo */etc/resolv.conf* para incluir `nameserver 192.168.0.30` y reiniciar el servidor con la instrucción `service bind9 restart` (o `systemctl restart bind9`). Se debe mirar */var/log/syslog* para observar si hay errores en el arranque del servidor y corregirlos hasta que su reinicio esté libre de estos (los más comunes son los símbolos utilizados como comentarios, los puntos al final de dominio o espacios antes de los registros A-PTR). Posteriormente podemos probar con la orden `dig nteum.remix.world` y tendremos una salida como:

```
; <<>> DiG 9.9.5-9+deb8u6-Debian <<>> nteum.remix.world
...
;; QUESTION SECTION:
;nteum.remix.world. IN A
;; ANSWER SECTION:
nteum.remix.world. 86400 IN A 192.168.0.30
;; AUTHORITY SECTION:
remix.world. 86400 IN NS nteum.remix.world.
;; Query time: 0 msec
...
```

Si hacemos la petición inversa con `dig -x 192.168.0.30` tendremos:

```
...
;; QUESTION SECTION:
;30.0.168.192.in-addr.arpa. IN PTR
;; ANSWER SECTION:
30.0.168.192.in-addr.arpa. 86400 IN PTR nteum.remix.world.
;; AUTHORITY SECTION:
0.168.192.in-addr.arpa. 86400 IN NS nteum.remix.world.
;; ADDITIONAL SECTION:
nteum.remix.world. 86400 IN A 192.168.0.30
...
```

Un aspecto interesante es poner alias lo cual significa incluir un registro CNAME en *remix.world.lan* como `ftp IN CNAME nteum.remix.world.` Dado que es un servicio esencial es necesario muchas veces disponer de un servicio secundario (*slave*) y bind9 permite crear muy fácilmente servicios de este tipo a partir del servidor primario por transferencia de las tablas (http://www.server-world.info/en/note?os=Debian_8&p=dns&f=5).

10.3. Otros DNS

Un servidor de DNS muy interesante, y que se incluye en muchas distribuciones, es **Dnsmasq** que permite en forma simple configurar un servidor DNS (*forwarder*) y un servidor DHCP en el mismo paquete para redes pequeñas/medianas. Este puede atender peticiones de nombres de máquinas que no están en los DNS globales e integra el servidor de DHCP para permitir que máquinas gestionadas por el DHCP aparezcan en el DNS con nombres ya configurados y además soporta gestión de IP dinámicas y estáticas para el DHCP y permite el acceso BOOTP/TFTP para máquinas sin disco y que hacen su inicialización por red.

La forma de configurar rápidamente un DNS para máquinas de dominio propio y que haga de *forwarder* para los dominios externos es utilizando las instrucciones `apt-get update; apt-get install dnsmasq`. Si lo que se desea es un simple DNS, entonces ya está configurado teniendo en cuenta que en el archivo */etc/resolv.conf* tendremos alguna cosa como `nameserver 8.8.8.8`. Con esto podemos probar los externos con `dig debian.org @localhost` (o simplemente `dig debian.org`) pero también responderá a todas las máquinas que tengamos definidas en */etc/hosts* como por ejemplo si tenemos una línea como `192.168.1.37 nteum.remix.world nteum` podremos ejecutar `dig nteum @localhost` y nos responderá con la IP correspondiente. Es por ello que si la red es simple podemos agregar las máquinas en este archivo pero si es una red más compleja podemos utilizar el siguiente archivo de configuración */etc/dnsmasq.conf* que incluye una gran cantidad de opciones para organizar los dominios internos y otros parámetros no solo para la configuración del DNS sino también para el servidor de DHCP. [DNSMQ]

MaraDNS es un servidor DNS que puede funcionar como DNS recursivo (caché) o como *authoritative name server* y que se caracteríza por su extremadamente fácil configuración. MaraDNS ha sido optimizado para un pequeño número de dominios en forma simple y eficiente y que permite crear un dominio propio y servirlo sin grandes esfuerzos aunque contiene todos los elementos para configurarlo en forma similar a bind9. En su diseño se ha puesto especial cuidado a la seguridad y utiliza una librería de gestión de cadenas de caracteres que resisten a los principales ataques a los DNS por desbordamiento (*buffer overflows*)[Mara]. EL paquete MaraDNS incluye un DNS autorizado (`maradns`), un servidor DNS recursivo con posibilidades de caché y que se llama Deadwood. La decisión de poner un servidor autorizado o recursivo dependerá de la función que se busque. Si simplemente es para obtener otros sitios en Inter-

net se deberá configurar un servidor recursivo, en cambio se se desea registrar dominios y se tienen una red de ordenadores dentro de este dominio hay que configurar un servidor DNS con autoridad.

Como pruebas de este servicio configuraremos diferentes opciones de MaraDNS para Debian. Aunque en algunas distribuciones antiguas de Debian el paquete estaba incorporado, las versiones incluidas daban problemas. Por eso, la mejor opción es descargarlo y compilarlo desde la página web del autor. Para ello debemos obtener el archivo desde http://maradns.samiam.org/download.html, descomprimirlo (`tar xjvf maradns-x.x.xx.tar.bz2` donde x.x.xx es la versión del software descargado) y dentro de directorio ejecutar `./configure; make` y si no hay errores ejecutar `make install` (esta orden puede dar algunos errores en el sentido de que no consigue generar algunos directorios –básicamente de manuales–, en cuyo caso habrá que crearlos manualmente). Los servidores se instalarán en */usr/local/sbin* y otros archivos complementarios en */usr/local/bin*. Los archivos de configuración ubicados en */etc/mararc y /etc/dwood3rc* y el directorio de configuración */etc/maradns*.

En primer lugar para configurar un DNS recursivo y caché modificar el archivo */etc/dwood3rc* para incluir:

```
bind_address="127.0.0.1"
chroot_dir = "/etc/maradns"
#upstream_servers = {}
#upstream_servers["."]="8.8.8.8, 8.8.4.4"
recursive_acl = "127.0.0.1/16" # Quien puede utilizar la caché

# Parámetros de configuración del servidor
maxprocs = 8 # Maximum number of pending requests
handle_overload = 1 # Send SERVER FAIL when overloaded
maradns_uid = 99 # UID Deadwood runs as
maradns_gid = 99 # GID Deadwood runs as
maximum_cache_elements = 60000

# File cache (readable and writable by the maradns_uid/gid)
cache_file = "dw_cache"
```

También es posible que Deadwood pueda contactar con otros DNS recursivos antes que con los root DNS servers. Para ello quitar el comentario de las líneas upstream_servers (en este caso se han puestos los dns públicos de Google). Luego podremos ejecutar `/usr/local/sbin/Deadwood` o si se quiere ejecutar como *daemon* se deberá utilizar el programa `duende` incluido en el paquete: `/usr/local/bin/duende /usr/local/sbin/Deadwood`.

Para configurar un DNS autorizado debemos modificar el archivo */etc/mararc* con los siguiente valores: cvs2 el dominio que gestionará este servidor y el archivo que tendrá la información en este caso db.local, el directorio donde se encontrará los archivos de configuración (/etc/maradns) y la IP de donde responderá.

```
csv2 = {}
csv2["remix.world."] = "db.local"
ipv4_bind_addresses = "127.0.0.1"
chroot_dir = "/etc/maradns"
```

El archivo */etc/maradns/db.local* simplemente tendrá una línea por cada registro que deseamos que transforme el dns por ejemplo, `nteum.remix.world.192.168.0.30 ~`. Luego se debe poner en marcha el servidor[30] o si se desea ejecutar como *daemon* habrá que usar entonces el programa `/usr/local/bin/duende /usr/local/sbin/maradns`. Una de las cuestiones interesantes es cómo puedo dar servicio de forma authoritative para mis dominios en una Intranet pero además ser recursivo cuando la petición sea a dominios externos? Es decir tengo unos pocos nombres (FQDN) que pertenecen a mi dominio (por ejemplo remix.world) sobre mi Intranet y deseo al mismo tiempo asegurar la resolución de dominio FQDN externos (por ejemplo debian.org). La solución planteada por el autor (Sam Trenholme) pasa por una configuración combinada de MaraDNS 2.x y Deadwood 3.x poniendo en este caso MaraDNS que escuche sobre localhost (127.0.0.1) mientras Deadwood sobre la IP del servidor (192.168.1.37 en nuestro caso). Los archivos de configuración */etc/mararc* y */etc/dwood3rc* serán:

(30) /usr/local/sbin/maradns

```
# Archivo /etc/mararc
# Simplemente indicar que escuche sobre localhost y el dominio que será responsable
ipv4_bind_addresses = "127.0.0.1"
timestamp_type = 2
verbose_level = 3
hide_disclaimer = "YES"
csv2 = {}
csv2["remix.world."] = "db.local"
chroot_dir = "/etc/maradns"

#Archivo /etc/dwood3rc
# Indicarle que además de nuestro dominio donde queremos consultar el resto
root_servers = {}
root_servers["remix.world."] = "127.0.0.1"
root_servers["."]="198.41.0.4, 192.228.79.201, 192.33.4.12, 199.7.91.13,"
root_servers["."]+="192.203.230.10, 192.5.5.241, 192.112.36.4, 128.63.2.53, "
root_servers["."]+="192.36.148.17, 192.58.128.30, 193.0.14.129, 199.7.83.42, "
root_servers["."]+="202.12.27.33"
# Donde escuchará la peticiones = IP del servidor Deadwood
bind_address="192.168.1.37"
```

```
# Habilitar la respuesta de IP privadas
filter_rfc1918=0
# IPs que podrán hacer peticiones
recursive_acl = "192.168.1.0/24"
# Caché de disco para Deadwood
cache_file = "dw_cache"
# Directorio raíz
chroot_dir = "/etc/maradns"
```

Finalmente en nuestras máquinas poner como */etc/resolv.conf* la IP de nuestro servidor `nameserver 192.168.1.37` y poner en marcha ambos servidores (luego de verificar que funciona se pueden poner como *daemons* y bajar el nivel de log con `verbose_level = 1`.

11. NIS (YP)

Con el fin de facilitar la administración y dar comodidad al usuario en redes de diferentes tamaños que ejecutan GNU/Linux (o algún otro sistema operativo con soporte para este servicio), se ejecutan servicios de *Network Information Service*, NIS (o *Yellow Pages*, YP, en la definición original de Sun). GNU/Linux puede dar apoyo como cliente/servidor de NIS. La información que se puede distribuir en NIS es la siguiente: usuarios (*login names*), contraseñas (*passwords*, `/etc/passwd`), directorios de usuario (*home directories*) e información de grupos (*group information*, `/etc/group`), redes, hosts, etc., lo cual presenta la ventaja de que, desde cualquier máquina cliente o desde el mismo servidor, el usuario se podrá conectar con la misma cuenta y contraseña y al mismo directorio (aunque el directorio deberá ser montado anteriormente sobre todas las máquinas cliente por NFS o mediante el servicio de `automount`). [Miq, Kuk]

La arquitectura NIS es del tipo cliente-servidor, es decir, existe un servidor que dispondrá de todas las bases de datos y unos clientes que consultarán estos datos en forma transparente para el usuario. Por ello, se debe pensar en la posibilidad de configurar servidores "de refuerzo" (llamados secundarios) para que los usuarios no queden bloqueados ante la caída del servidor principal. Es por ello que la arquitectura se denomina realmente de múltiples servidores (*master+mirrors-clients*).

11.1. Configuración del servidor

Para instalar el servidor en una máquina (servidor nis) debemos instalar el paquete *nis* (el cual hará la instalación de todas las dependencias necesarias) con `apt-get install nis`. Durante la instalación nos solicitará un dominio NIS, bajo el cual agrupará todas las máquinas a las que dará servicio. Es recomendable que este dominio NIS no coincida con el dominio Internet ni con el nombre del *host*. Por ejemplo, si el servidor tiene como nombre.dominio *nteum.remix.world* se podría poner como dominio nis NISnteum (se debe tener en cuenta que es sensible a mayúsculas, así que NISNteum es diferente de Nisnteum). Se puede consultar este nombre de dominio con la orden `nisdomainname`, (o también se puede consultar */proc/sys/kernel/domainname*) y se puede reconfigurar con `dpkg-reconfigure nis`. Luego se debe modificar el archivo */etc/default/nis* para modificar la línea `NISSERVER=master` para indicarle el rol de servidor. También (si bien se puede hacer en un segundo paso) se debe ajustar el archivo */etc/ypserv.securenets* para modificar la línea que indica `0.0.0.0 0.0.0.0` que da acceso a todo el mundo y restringirlo a nuestra red, por ejemplo `255.255.255.0 192.168.1.0`.

Por último, se debe modificar el */etc/hosts* para que tengamos la máquina con el nombre definido en */etc/hostname* (se puede cambiar/visualizar con el comando `hostname` o editando el fichero) con una línea FQND como por ejemplo `192.168.1.30 nteum.remix.world nteum`. La configuración del servidor se realiza con el comando `/usr/lib/yp/ypinit -m`; en algunas distribuciones es necesario verificar que existe el archivo `/etc/networks`, que es imprescindible para este *script*. Si este archivo no existe, cread uno vacío con `touch /etc/networks`; este *script* nos solicitará cuáles serán los servidores NIS indicándonos la máquina local por defecto y deberemos terminar con el atajo Crtl+D. Por otro lado, también se puede ejecutar el cliente `ypbind` sobre el servidor; así pues, todos los usuarios entran por NIS indicando en el archivo */etc/default/nis* que exista `NISCLIENT=true`. Para reiniciar el servicio se ejecutará la orden `service nis restart` (o `systemctl restart nis`).

Considerad que a partir de este momento los comandos para cambiar la contraseña o la información de los usuarios, como `passwd`, `chfn` o `adduser`, no son válidos. En su lugar, se deberán utilizar comandos tales como `yppasswd`, `ypchsh` y `ypchfn`. Si se cambian los usuarios o se modifican los archivos mencionados, habrá que reconstruir las tablas de NIS ejecutando el comando `make` en el directorio `/var/yp` para actualizar las tablas.

La configuración de un servidor esclavo es similar a la del maestro, excepto si `NISSERVER = slave` en `/etc/default/nis`. Sobre el maestro se debe indicar que distribuya las tablas automáticamente a los esclavos, poniendo `NOPUSH = "false"` en el archivo `/var/yp/Makefile`. Ahora se debe indicar al maestro quién es su esclavo por medio de la ejecución de:

```
/usr/lib/yp/ypinit -m
```

e introduciendo los nombres de los servidores esclavos. Esto reconstruirá los mapas pero no enviará los archivos a los esclavos. Para ello, sobre el esclavo, ejecutad:

```
/etc/init.d/nis stop (o systemctl stop nis)

/usr/lib/yp/ypinit -s nombre_master_server

/etc/init.d/nis start (o systemctl start nis)
```

Así, el esclavo cargará las tablas desde el maestro. También se podría poner en el directorio `/etc/cron.d` el archivo NIS con un contenido similar a (recordad hacer un `chmod 755 /etc/cron.d/nis`):

```
20 *  * * *  root  /usr/lib/yp/ypxfr_1perhour >/dev/null 2>&1
40 6 * * *    root  /usr/lib/yp/ypxfr_1perday  >/dev/null 2>&1
```

```
55 6,18 * * *    root  /usr/lib/yp/ypxfr_2perday  >/dev/null 2>&1
```

Con ello, se garantizará que todos los cambios del maestro se transfieran al servidor NIS esclavo.

Iniciad el servidor ejecutando el comando `/etc/init.d/nis restart` (o `systemctl restart nis`). Esta sentencia iniciará el servidor (`ypserv`) y el *password daemon* (`yppasswdd`), cuya activación se podrá consultar con la orden `ypwich -d domain`.

Actualización de las tablas NIS

Es recomendable que después de usar `adduser o useradd` para agregar un nuevo usuario sobre el servidor, ejecutéis `cd /var/yp; make` para actualizar las tablas NIS (y siempre que se cambie alguna característica del usuario, por ejemplo la palabra clave con el comando `passwd`, solo cambiará la contraseña local y no la de NIS).

Para probar que el sistema funciona y que el usuario dado de alta está en el NIS, podéis hacer `ypmatch userid passwd`, donde `userid` es el usuario dado de alta con `adduser` antes y después de haber hecho el `make`. Para verificar el funcionamiento del sistema NIS podéis usar el *script* de http://tldp.org/HOWTO/NIS-HOWTO/verification.html que permite una verificación más detallada del NIS.

También se puede probar en la misma máquina que un usuario se puede conectar a un domino NIS haciendo: `adduser usuario` y respondiendo a las preguntas, luego hacemos `cd /var/yp; make`. En este momento lo veremos tanto en */etc/passwd* como en NIS con `ypcat passwd`. Luego lo podemos borrar con el comando `userdel usuario` (no utilizar el comando `deluser` sino el `userdel`) que solo lo borrará del */etc/passwd* y no del NIS (hay que tener cuidado de no hacer un `make`, ya que este también lo borrará del NIS). Luego nos podemos conectar como `ssh usuario@localhost` y con ello podremos verificar que el NIS funciona, ya que el acceso solo se podrá hacer por NIS porque por */etc/passwd* no existe este usuario. Con relación a NIS+ (una versión más segura de NIS existente en algunos Unix –por ejemplo Solaris/OpenSolaris o derivados, http://wiki.illumos.org/–). Su desarrollo fue parado en el 2012 dada la falta de interés/recursos de la comunidad, no así NIS/YP, que sigue actualizado y presente en todas las distribuciones (http://www.linux-nis.org/nisplus/)

11.2. Configuración del cliente

Un cliente local es el que anexa el ordenador a un dominio NIS ya existente ejecutando `apt-get install nis`.

El procedimiento de instalación del paquete NIS solicitará un dominio (`NIS domainname`). Este es un nombre que describirá el conjunto de máquinas que utilizarán el NIS y que hemos introducido previamente en el servidor (NISnteum en nuestro caso, véase el subapartado anterior). Puede ocurrir que como todavía no se ha configurado la dirección del servidor NIS, el cliente intentará localizarlo enviando un mensaje de *broadcast*. Si no lo encuentra dará (después de un tiempo breve) un mensaje similar al siguiente:

[....] Starting NIS services: ypbind[....] binding to YP server...failed (backgrounded).

El cliente NIS usa el comando `ypbind` para encontrar un servidor para el dominio especificado, ya sea mediante `broadcast` (no aconsejado por inseguro) o buscando el servidor indicado en el archivo de configuración `/etc/yp.conf` (recomendable) que puede tener básicamente dos tipos de configuración como las mostradas abajo –solo se debe indicar una de ellas– .

Sintaxis del archivo `/etc/yp.conf`

`domain nisdomain server hostname`: indica que se utiliza el *hostname* para el dominio `nisdomain`. En nuestro caso podría ser `domain NISnteum server 192.168.1.30` Se podría tener más de una entrada de este tipo para un único dominio.

`ypserver hostname`: indica que se utiliza *hostname* como servidor introduciendo la dirección IP del servidor NIS. Si se indica el nombre, asegúrese de que se puede encontrar la IP por DNS o que la misma figura en el archivo `/etc/hosts`, ya que, de otro modo, el cliente se bloqueará.

Para iniciar el servicio se debe ejecutar:

`/etc/init.d/nis stop` *para detenerlo*

`/etc/init.d/nis start` *para iniciarlo*

o también con el comando `service nis start|stop` o con el nuevo comando `systemctl start|stop nis`

A partir de este momento, el cliente NIS estará funcionando. Ello se puede confirmar con `rpcinfo -u localhost ypbind`, que mostrará las dos versiones del protocolo activo o, cuando el servidor esté funcionando, se puede utilizar el comando `ypcat mapname` (por ejemplo, `ypcat passwd`, que mostrará los usuarios NIS definidos en el servidor) donde las relaciones entre *mapnames* y las tablas de la base de datos NIS están definidas en `/var/yp/nicknames`.

11.3. ¿Qué recursos se deben especificar para utilizar el NIS?

A partir de la inclusión de lib6 en las distribuciones de GNULinux (esto es Debian5 o FC4) es necesario orientar la consulta del *login* a las bases de datos adecuadas con los pasos siguientes:

1) Verificar el fichero `/etc/nsswitch.conf` y asegurarse de que las entradas `passwd`, `group`, `shadow` y `netgroup` son similares a:

```
passwd: compat nis
group: compat nis
shadow: compat nis
netgroup: nis
hosts: files dns nis
```

2) Consultad la sintaxis de este archivo en `man nsswitch.conf`.

3) En algunas configuraciones puede ser necesario agregar al final del fichero */etc/pam.d/common-session* (si no existe ya la línea)

```
session optional pam_mkhomedir.so skel=/etc/skel umask=077
```

para crear de forma automática el directorio home si no existe pero no es lo habitual ya que generalmente se desea que este se monte del servidor NFS.

4) Con esta configuración se podrá realizar una conexión local (sobre el cliente NIS) a un usuario que no esté definido en el fichero `/etc/passwd`, es decir, un usuario definido en otra máquina (*ypserver*). Por ejemplo, se podría hacer `ssh -l user localhost`, donde `user` es un usuario definido en *ypserver*.

12. Servicios de conexión remota: telnet y ssh

12.1. Telnet y telnetd

Telnet es un comando (cliente) utilizado para comunicarse interactivamente con otro *host* que ejecuta el *daemon* `telnetd`, aunque ha caído en desuso ya que la conexión no es cifrada y su contenido puede interceptarse capturando los paquetes de la red. El comando `telnet` se puede ejecutar como `telnet host` o interactivamente como `telnet`, el cual pondrá el *prompt* "telnet>" y luego, por ejemplo, `open host`. Una vez establecida la comunicación, se deberá introducir el usuario y la contraseña bajo la cual se desea conectar al sistema remoto. Se dispone de diversos comandos (en modo interactivo) como `open`, `logout` y `mode` (se deben definir las características de visualización), `close`, `encrypt`, `quit`, `set` y `unset` o se pueden ejecutar comando externos con "!". Se puede utilizar el archivo `/etc/telnetrc` para definiciones por defecto o `.telnetrc` para definiciones de un usuario particular (deberá estar en el directorio *home* del usuario).

El *daemon* `telnetd` es el servidor de protocolo telnet para la conexión interactiva. Generalmente, es el *daemon* `inetd` o `xinetd` que pone en marcha `telned`; se recomienda incluir un `wrapper tcpd` (que utiliza las reglas de acceso en `host.allow` y `host.deny`) en la llamada al `telnetd` dentro del archivo `/etc/inetd.conf` (o en *xinetd.conf* o el archivo correspondiente en *xinetd.d*).

Se debe recordar que, si bien el par `telnet-telnetd` puede funcionar en modo *encrypt* en las últimas versiones (transferencia de datos encriptados, que deben estar compilados con la opción correspondiente), es un comando que ha quedado en el olvido por su falta de seguridad (transmite el texto en claro sobre la red, lo que permite la visualización del contenido de la comunicación desde otra máquina, por ejemplo, con el comando `tcpdump`); no obstante, puede utilizarse en redes seguras o situaciones controladas.

Si no está instalado, se puede utilizar (Debian) `apt-get install telnetd` y después verificar que se ha dado de alta, bien en `/etc/inetd.conf`, bien en `/etc/xinetd.conf` (o en el directorio en que estén definidos los archivos; por ejemplo, `/etc/xinetd.d` según se indique en el archivo anterior con la sentencia `include /etc/xinetd.d`). El fichero `xinetd.conf` o también el archivo `/etc/xinetd.d/telnetd` deberán incluir una sección como[31]:

[31] Cualquier modificación en xinetd.conf deberá arrancar nuevamente el servicio con `service xinetd restart`.

```
service telnet {
disable = no
flags = REUSE
```

```
socket_type = stream
wait = no
user = root
server = /usr/sbin/in.telnetd
log_on_failure += USERID
}
```

Telnet-SSL

Se recomienda que en lugar de usar `telnetd` se utilice `telnet-SSL(d)`, que reemplaza a `telnet(d)` utilizando encriptación y autentificación por SSL, o que se utilice SSH (el cual ya es un estándar en todas las distribuciones y entornos) .

12.2. SSH, *Secure shell*

Un cambio aconsejable hoy en día es utilizar `ssh` (comando que ya está en todos los sistemas operativos incluido en Windows con putty o moabterm) en lugar de telnet, rlogin o rsh. Estos tres últimos comandos son inseguros (excepto SSLTelnet) por varias razones: la más importante es que todo lo que se transmite por la red, incluidos nombres de usuarios y contraseñas, es en texto plano (aunque existen versiones de telnet-telnetd encriptados, debe coincidir que ambos los sean), de modo que cualquiera que tenga acceso a esa red o a algún segmento de la misma puede obtener toda esta información y luego suplantar la identidad del usuario. La segunda es que estos puertos (telnet, rsh, etc.) son el primer lugar donde un *cracker* intentará conectarse. El protocolo ssh (en su versión OpenSSH) provee una conexión encriptada y comprimida mucho más segura que, por ejemplo, telnet (es recomendable utilizar la versión 2.0 o versiones superiores del protocolo). Todas las distribuciones actuales incorporan el cliente ssh y el servidor sshd por defecto. Es necesario tener actualizada la librería OpenSSL, utilizada por estos programas, ya que recientemente se encontró un problema de seguridad llamado *heartbleed* y que ha sido rápidamente solucionado en la mayoría de las distribuciones GNU/Linux (https://es.wikipedia.org/wiki/Heartbleed).

12.2.1. ssh

Para ejecutar el comando, haced:

```
ssh -l user hostname o ssh user@hostname
```

A través de SSH se pueden encapsular otras conexiones como X11 o cualquier otra TCP/IP. Si se omite el parámetro `-l`, el usuario se conectará con el mismo usuario local y en ambos casos el servidor solicitará la contraseña para validar la identidad del usuario. SSH soporta diferentes modos de autentificación (consultad `man ssh`) basados en algoritmo RSA y clave pública. Si el cliente no está instalado, hacer `apt-get install openssh-client`.

Usando el comando `ssh-keygen -t rsa|dsa`, se pueden crear las claves de identificación de usuario. El comando crea en el directorio del `.shh` del usuario los ficheros[32] `id_rsa` y `id_rsa.pub`, las claves privada y pública, respectivamente. El usuario podría copiar la pública (`id_rsa.pub`) en el archivo `$HOME/.ssh/authorized_keys` del directorio `.ssh` del usuario de la máquina remota. Este archivo podrá contener tantas claves públicas como sitios desde donde se quiera conectar a esta máquina de forma remota. La sintaxis es de una clave por línea aunque las líneas tendrán un tamaño considerable. Después de haber introducido las claves públicas del usuario-máquina en este archivo, este usuario se podrá conectar sin contraseña desde esa máquina. También para realizar esta copia se puede utilizar el comando `ssh-copy-id máquina` que copiará las llaves en forma automática y es mucho más simple de utilizar.

[32] Por ejemplo, para el algoritmo de encriptación RSA.

En forma normal (si no se han creado las claves), se le preguntará al usuario una contraseña, pero como la comunicación será siempre encriptada, nunca será accesible a otros usuarios que puedan escuchar sobre la red. Para mayor información, consultad `man ssh`. Para ejecutar remotamente un comando, simplemente haced:

```
ssh -l login_name host_comando_remoto
```

Por ejemplo: `ssh -l user localhost ls -al`

12.2.2. sshd

El `sshd` es el servidor (*daemon*) para el `ssh` (si no están instalados, se puede hacer con `apt-get install openssh-client openssh-server`. Juntos reemplazan a `rlogin`, `telnet`, `rsh` y proveen una comunicación segura y encriptada entre dos *hosts* inseguros de la red. Este se arranca generalmente a través de los archivos de inicialización (`/etc/init.d` o `/etc/rc`) y espera conexiones de los clientes. El `sshd` de la mayoría de las distribuciones actuales soporta las versiones 1, 2 y 3 del protocolo SSH. Cuando se instala el paquete, se crea una clave RSA específica del *host* y cuando el *daemon* se inicia, crea otra, la RSA para la sesión, que no se almacena en el disco y que cambia cada hora. Cuando un cliente inicia la comunicación, genera un número aleatorio de 256 bits que está encriptado con las dos claves del servidor y enviado. Este número se utilizará durante la comunicación como clave de sesión para encriptar la comunicación que se realizará a través de un algoritmo de encriptación estándar. El usuario puede seleccionar cualquiera de los algoritmos disponibles ofrecidos por el servidor. Existen algunas diferencias (más seguro) cuando se utiliza la versión 2 (o 3) del protocolo. A partir de este momento, se inician algunos de los métodos de autentificación de usuario descritos en el cliente o se le solicita la contraseña, pero siempre con la comunicación encriptada. Para mayor información, consultad `man sshd`.

Tanto ssh como sshd disponen de un gran conjunto de parámetros que pueden ser configurados según se necesite en los archivos `/etc/ssh/ssh_config` y `/etc/ssh/sshd_config`. En el servidor probablemente algunas de las opciones más utilizadas son `PermitRootLogin yes|no` para permitir la conexión del usuario root o no, `IgnoreRhosts yes` para evitar leer los archivos `$HOME/.rhosts` y `$HOME/.shosts` de los usuarios, `X11Forwarding yes` para permitir que aplicaciones Xwindows en el servidor se visualicen en la pantalla del cliente (muy útil en la administración remota de servidores con el comando `ssh -X host_a_administrar`).

12.2.3. Túnel sobre SSH

Muchas veces tenemos acceso a un servidor `sshd` pero por cuestiones de seguridad no podemos acceder a otros servicios que no están encriptados (por ejemplo, un servicio de consulta de mail POP3 o un servidor de ventanas X11) o simplemente se quiere conectar a un servicio al cual solo se tiene acceso desde el entorno de la empresa. Para ello, es posible establecer un túnel encriptado entre la máquina cliente (por ejemplo, con Windows y un cliente ssh llamado `putty` de software libre) y el servidor con `sshd`. En este caso, al vincular el túnel con el servicio, el servicio verá la petición como si viniera de la misma máquina. Por ejemplo, si queremos establecer una conexión para POP3 sobre el puerto 110 de la máquina remota (y que también tiene un servidor `sshd`) haremos:

```
ssh -C -L 1100:localhost:110 usuario-id@host
```

Este comando pedirá la contraseña para el usuario-id sobre *host* y una vez conectado, se habrá creado el túnel. Cada paquete que se envíe desde la máquina local sobre el puerto 1100 se enviará a la máquina remota *localhost* sobre el puerto 110, que es donde escucha el servicio POP3 (la opción `-C` comprime el tráfico por el túnel).

Hacer túneles sobre otros puertos es muy fácil. Por ejemplo, supongamos que *solo* tenemos acceso a un *remote proxy server* desde una máquina remota (*remote login*), no desde la máquina local; en este caso, se puede hacer un túnel para conectar el navegador a la máquina local. Consideremos que tenemos *login* sobre una máquina pasarela (*gateway*), que puede acceder a la máquina llamada *proxy*, la cual ejecuta el *Squid Proxy Server* sobre el puerto 3128. Ejecutamos:

```
ssh -C -L 8080:proxy:3128 user@gateway
```

Después de conectarnos, tendremos un túnel escuchando sobre el puerto local 8080 que reconducirá el tráfico desde *gateway* hacia *proxy* al 3128. Para navegar de forma segura solo se deberá hacer `http://localhost:8080/`.

12.3. VPN SSL (vía tun driver)

OpenSSH v4.3 introduce una nueva característica que permite crear *on-the-fly* VPN vía el túnel driver (tun) como un puente entre dos interfaces en diferentes segmentos de red a través de Internet y por lo cual un ordenador (B en nuestro caso) "quedará dentro" de la red del otro ordenador (A) a pesar de estar dentro de otra red. Para analizar este mecanismo, consideraremos que el ordenador A y B están conectados a la red vía ethernet y a su vez a Internet a través de una *gateway* que hace NAT. A tiene IP sobre su red (privada) 10.0.0.100, y B sobre su red (privada) 192.168.0.100 y como dijimos cada uno tiene acceso a internet NAT *gateway*.

A través de OpenSSH conectaremos B a la red de A vía una interfaz túnel ssh. Como hemos definido A ya tiene IP sobre la red A (10.0.0.100) y que se ve desde el gateway externo como dirección 1.2.3.4, es decir esta es la IP pública del servidor `ssh` de A (el router de A deberá hacer un forwarding de 1.2.3.4:22 a 10.0.0.100:22 para que externamente se pueda contactar con servidor `ssh` de A). B deberá tener una IP sobre la red A que será su interfaz tun0 y que le asignaremos 10.0.0.200. B debe también debe tener acceso al servidor `ssh` de A (o bien acceso directo o el puerto 22 debe ser forwarded sobre la red A en la dirección 1.2.3.4). Cuando el túnel esté creado B accederá directamente a la red de A lo cual significa que B "estará" dentro de la red de A).

Los pasos de configuración son los siguientes:

1) Activar el IP forwarding: `echo 1 > /proc/sys/net/ipv4/ip_forward`

2) Túnel: Sobre B `ssh -w 0:0 1.2.3.4` se puede utilizar como opciones además `-NTCf`. Esto crea un tunnel interface tun0 entre cliente (B) y el servidor (A), recordar que A tiene dirección pública (1.2.3.4). Se deberá tener acceso de root a ambos sistemas para que puedan crear las interfaces (en A verificar que se tiene en `sshd_config PermitRootLogin yes` y `PermitTunnel yes`). Se recomienda utilizar SSH *keys* (PKI) y por lo cual se deberá hacer el cambio `PermitRootLogin without-password` y si sobre el cliente no se quiere dar acceso de root se puede utilizar el comando `sudo` (ver `man sudoers`).

3) Verificar las interfaces: `ip addr show tun0`

4) Configurar las interfaces:

Ordenador A: `ip link set tun0 up ip addr add 10.0.0.100/32 peer 10.0.0.200 dev tun0`

Ordenador B: `ip link set tun0 up ip addr add 10.0.0.200/32 peer 10.0.0.100 dev tun0`

5) Probar: sobre B `ping 10.0.0.100` y sobre A `ping 10.0.0.200`

6) Routing directo: tenemos un link que conecta B en la red A pero es necesario inicializar el routing.

Sobre B: `ip route add 10.0.0.0/24 via 10.0.0.200` (para permitir enviar desde B a cualquier máquina de la red A).

7) Routing inverso: También para que los paquetes vuelvan a B sobre A debemos hacer `arp -sD 10.0.0.200 eth0`

El routing nos asegurará que otras máquinas que están en la red A puedan enviar paquetes a B a través de A. Podemos probar haciendo desde B `ping 10.0.0.123`. Si se desea que todos los paquetes de B vayan a través de A deberemos cambiar el default *gateway* de B pero esto no es simple ya que el propio túnel va por internet. Primero debemos crear un *host-based route* hacia la máquina A (todos los paquetes excepto los que crean el *link* deben ir por el túnel -pero obviamente no los que crean el túnel-). Sobre B: `ip route add 1.2.3.4/32 via 192.168.0.1` En este caso 192.168.0.1 es el *default gateway* para B, es decir el *gateway* sobre la red B que provee conectividad a internet y que nos servirá para mantener el túnel. Luego se podemos cambiar el *default gateway* sobre B: `ip route replace default via 10.0.0.1`. Con lo cual tendremos que 192.168.0.1 es el *default gateway* de la red B y 10.0.0.1 *default gateway* de la red A. Ya que la máquina B está conectada a la red de A le estamos indicando utilizar la red A como *default gateway* en lugar de *default gateway* sobre la red B (como sería habitual). Se puede verificar esto con el comando `tracepath google.com`.

12.4. Túneles encadenados

Muchas veces el acceso a redes internas solo es posible a través de un servidor SSH (que están en la DMZ) y desde esta es posible la conexión hacia servidores SSH internos para luego llegar a la máquina SSH de nuestro interés (y si queremos copiar archivos o hacer el *forwarding* de X11 puede ser todo un reto). La forma de hacerlo es primero conectarnos a la M1, luego de esta a la M2 (a la cual solo es posible conectarse desde M1) y desde esta a la M3 (a la cual solo es posible conectarse desde M2). Una forma de facilitar la autentificación es utilizar el *agent forwarding* con el parámetro -A y poniendo nuestra llave pública en todos los *$HOME/.ssh/authorized_keys* así cuando vayamos ejecutando los diferentes comandos la petición de llaves puede ser *forwarded* hacia la máquina anterior y así sucesivamente. los comandos serán `ssh -A m1.org` y desde esta `ssh -a m2.org` y a su vez `ssh -a m3.org`. Para mejorar esto podemos automatizarlo con `ssh -A -t m1.org ssh -A -t m2.org ssh -A m2.org`

(se ha incluido la opción -t para que `ssh` le asigne una pseudo-tty y solo veremos la pantalla final). El applet SSHmenu puede ayudar a automatizar todo esto (http://sshmenu.sourceforge.net/).

Una forma de gestionar efectivamente esta conexión "multisaltos" es a través de la opción `ProxyCommand` de `ssh` (ver `man ssh_config`) que nos permitirá conectarnos de forma más eficiente. Para ello definiremos los siguientes comandos en *$HOME/.ssh/config*:

```
Host m1
   HostName m1.org
Host m2
   ProxyCommand ssh -q m1 nc -q0 m2.org 22
Host m3
   ProxyCommand ssh -q m2 nc -q0 m3.org 22
```

Donde el primer comando (cuando hagamos `ssh m1`) nos conectará a m1.org. Cuando ejecutemos `ssh m2` se establecerá una conexión ssh sobre m1 pero el comando ProxyCommand utilizará en comando nc para extender la conexión sobre m2.org. Y el tercero es una ampliación del anterior por lo cual cuando ejecutemos `ssh m3` es como si ejecutáramos una conexión a m1 y luego ampliáramos esta a m2 y luego a m3.[Muh]

13. Servicios de transferencia de ficheros: FTP

El FTP (*File Transfer Protocol*) es un protocolo cliente/servidor (bajo TCP) que permite la transferencia de archivos desde y hacia un sistema remoto. Un servidor FTP es un ordenador que ejecuta el *daemon* `ftpd`.

Algunos sitios que permiten la conexión anónima bajo el usuario *anonymous* son generalmente repositorios de programas. En un sitio privado, se necesitará un usuario y una contraseña para acceder. También es posible acceder a un servidor ftp mediante un navegador y generalmente hoy en día los repositorios de programas se sustituyen por servidores web (p. ej. Apache) u otras tecnologías como Bittorrent (que utiliza redes *peer to peer*, P2P) o servidores web con módulos de WebDav o el propio comando `scp` (*secure copy*) que forma parte del paquete openssh-client o el par `sftp/sftpd` que forman parte de los paquetes `openssh-client` y `openssh-server` respectivamente. No obstante, se continúa utilizando en algunos casos y Debian, por ejemplo, permite el acceso con usuario/contraseña o la posibilidad de subir archivos al servidor (si bien con servicios web-dav también es posible hacerlo).

El protocolo (y los servidores/clientes que lo implementan) de ftp por definición no es encriptado (los datos, usuarios y contraseñas se transmiten en texto claro por la red) con el riesgo que ello supone. Sin embargo, hay una serie de servidores/clientes que soportan SSL y por lo tanto encriptación.

13.1. Cliente ftp (convencional)

Un cliente ftp permite acceder a servidores ftp y hay una gran cantidad de clientes disponibles. El uso del ftp es sumamente simple; desde la línea de comando, ejecutad:

```
ftp nombre-servidor
```

O también `ftp` y luego, de forma interactiva, `open nombre-servidor`

El servidor solicitará un nombre de usuario y una contraseña (si acepta usuarios anónimos, se introducirá *anonymous* como usuario y nuestra dirección de correo electrónico como contraseña) y a partir del *prompt* del comando (después de algunos mensajes), podremos comenzar a transferir ficheros.

El protocolo permite la transferencia en modo ASCII o binario. Es importante decidir el tipo de fichero que hay que transferir porque una transferencia de un binario en modo ASCII inutilizará el fichero. Para cambiar de un modo a otro, se deben ejecutar los comandos `ascii` o `binary`. Los coman-

dos útiles del cliente ftp son el `ls` (navegación en el directorio remoto), `get nombre_del_fichero` (para descargar ficheros) o `mget` (que admite `*`), `put nombre_del_fichero` (para enviar ficheros al servidor) o `mput` (que admite `*`); en estos dos últimos se debe tener permiso de escritura sobre el directorio del servidor. Se pueden ejecutar comandos locales si antes del comando se inserta un "`!`". Por ejemplo, `!cd /tmp` significará que los archivos que bajen a la máquina local se descargarán en `/tmp`. Para poder ver el estado y el funcionamiento de la transferencia, el cliente puede imprimir marcas, o *ticks*, que se activan con los comandos `hash` y `tick`. Existen otros comandos que se pueden consultar en la hoja del manual (`man ftp`) o haciendo `help` dentro del cliente.

Enlace de interés

En la Wikipedia disponéis de una comparativa de diversos clientes FTP:

http://en.wikipedia.org/wiki/Comparison_of_FTP_client_software.

13.2. Servidores FTP

El servidor tradicional de UNIX se ejecuta a través del puerto 21 y es puesto en marcha por el *daemon* `inetd` (o `xinetd`, según se tenga instalado). En `inetd.conf` conviene incluir el *wrapper* `tcpd` con las reglas de acceso en `host.allow` y el `host.deny` en la llamada al `ftpd` por el `inetd` para incrementar la seguridad del sistema. Cuando recibe una conexión, verifica el usuario y la contraseña y lo deja entrar si la autentificación es correcta. Un FTP *anonymous* trabaja de forma diferente, ya que el usuario solo podrá acceder a un directorio definido en el archivo de configuración y al árbol subyacente, pero no hacia arriba, por motivos de seguridad. Este directorio generalmente contiene directorios `pub/, bin/, etc/` y `lib/` para que el *daemon* de ftp pueda ejecutar comandos externos para peticiones de `ls`. El *daemon* `ftpd` soporta los siguientes archivos para su configuración:

Ved también

El tema de la seguridad del sistema se estudia en el módulo "Administración de seguridad".

- **/etc/ftpusers:** lista de usuarios que no son aceptados en el sistema. Un usuario por línea.
- **/etc/ftpchroot:** lista de usuarios a los que se les cambiará el directorio base `chroot` cuando se conecten. Necesario cuando deseamos configurar un servidor anónimo.
- **/etc/ftpwelcome:** anuncio de bienvenida.
- **/etc/motd:** noticias después del *login*.
- **/etc/nologin:** mensaje que se muestra después de negar la conexión.
- **/var/log/ftpd:** *log* de las transferencias.

Si en algún momento queremos inhibir la conexión al ftp, se puede incluir el archivo `/etc/nologin`. El `ftpd` muestra su contenido y termina. Si existe un archivo `.message` en un directorio, el `ftpd` lo mostrará cuando se acceda al mismo.

Es importante prestar atención a que los usuarios habilitados únicamente para utilizar el servicio ftp no dispongan de un *shell* a la entrada correspondiente de dicho usuario en `/etc/passwd` para impedir que este usuario tenga conexión, por ejemplo, por `ssh` o `telnet`. Para ello, cuando se cree el usuario, habrá que indicar, por ejemplo:

```
useradd -d/home/nteum -s /bin/false nteum
```

y luego: `passwd nteum`,

lo cual indicará que el usuario nteum no tendrá *shell* para una conexión interactiva (si el usuario ya existe, se puede editar el fichero `/etc/passwd` y cambiar el último campo por `/bin/false`). A continuación, se debe agregar como última línea `/bin/false` en `/ect/shells`. En [Mou] se describe paso a paso cómo crear tanto un servidor ftp seguro con usuarios registrados como un servidor ftp anonymous para usuarios no registrados. Dos de los servidores no estándares más comunes son el ProFTPD y Vsftpd (ambos incluidos en Debian por ejemplo).

Enlaces de interés

Para saber más sobre ProFTPD y Vsftpd podéis visitar las siguientes páginas http://www.proftpd.org y https://security.appspot.com/vsftpd.html.

Enlaces de interés

Para configurar un servidor FTP en modo encriptado (TSL) o para tener acceso *anonymous* podéis consultar: http://www.debian-administration.org/articles/228. Por otro lado, para saber más sobre la instalación y configuración de PureFtpd podéis consultar: http://www.debian-administration.org/articles/383.

14. Servicios de archivos (NFS, *Network File System*)

El sistema NFS permite a un servidor exportar un sistema de archivo para que pueda ser utilizado de forma interactiva desde un cliente. El servicio se compone básicamente de un servidor (básicamente representado por `nfsd*` y un cliente (representado por `rpc.mountd`) que permiten compartir un sistema de archivo (o parte de él) a través de la red. En la última versión de NFSv4 se incluyen una serie de *daemons* más como *idmapd*, *statd*, además de una serie de módulos para las nuevas funcionalidades de esta versión. En la distribución Debian, instalad `apt-get install nfs-common` (será necesario disponer del paquete *rpcbin* que como se comentó anteriormente es el nuevo *portmap* y generalmente ya está instalado) para el cliente, mientras que el servidor necesita `apt-get install nfs-kernel-server`. El servidor (en Debian) se gestiona a través del *script*`/etc/init.d/nfs-kernel-server` o simplemente con

```
service nfs-kernel-server start|stop
```

(o `systemctl start|stop nfs-kernel-server`). El servidor utiliza un archivo (`/etc/exports`) para gestionar el acceso remoto a los sistemas de archivo y el control de los mismos. Sobre el cliente (u otro usuario a través de `sudo`), el root puede montar el sistema remoto a través del comando:

```
mount -t nfs Ipserver:directorio-remoto directorio_local
```

y a partir de este momento, el directorio-remoto se verá dentro de directorio local (este debe existir antes de ejecutar el `mount`). Esta tarea en el cliente se puede automatizar utilizando el archivo de *mount* automático (`/etc/fstab`) incluyendo una línea; por ejemplo:

```
remix.world:/home /home nfs defaults 0 0.
```

Esta sentencia indica que se montará el directorio `/home` del *host* `remix.world` en el directorio local `/home`. Además, este sistema de archivo se montará con los parámetros por defecto (ver `man mount` apartado `mount options for ntfs` y man nfs para opciones específicas para NFSv4). Los últimos dos ceros indican que el sistema de archivos no debe ser `dumped` y que no se activará el `fsck` sobre él. El archivo `/etc/exports` sirve de ACL (lista de control de acceso) de los sistemas de archivo que pueden ser exportados a los clientes. Cada línea contiene un sistema de ficheros (*filesystem*) por exportar seguido de los clientes que lo pueden montar, separados por espacios en blan-

co. A cada cliente se le puede asociar un conjunto de opciones para modificar el comportamiento (consultad *man exports* para ver un lista detallada de las opciones). Un ejemplo de esto podría ser:

```
# Ejemplo de /etc/exports
/          master(rw) trusty(rw,no_root_squash)
/projects  proj*.local.domain(rw)
/usr       .local.domain(ro) @trusted(rw)
/pub       (ro,insecure,all_squash)
/home      195.12.32.2(rw,no_root_squash) www.first.com(ro)
/user      195.12.32.2/24(ro,insecure)
/home      192.168.1.0/24(rw,sync,fsid=0,no_root_squash,no_subtree_check)
```

La primera línea exporta el sistema de archivos entero (`/`) a `master` y `trusty` en modo lectura/escritura. Además, para `trusty` no hay *uid squashing* (el root del cliente accederá como root a los archivos root del servidor, es decir, los dos root son equivalentes a pesar de ser de máquinas diferentes y se utiliza para máquinas sin disco). La segunda y tercera líneas muestran ejemplos de '*****' y de *netgroups* (indicados por `@`). La cuarta línea exporta el directorio `/pub` a cualquier máquina del mundo, solo de lectura, permite el acceso de clientes NFS que no utilizan un puerto reservado para el NFS (opción `insecure`) y todo se ejecuta bajo el usuario *nobody* (opción `all squash`). La quinta línea especifica un cliente por su IP y en la sexta, se especifica lo mismo pero con máscara de red (/24) y con opciones entre paréntesis sin espacio de separación. Solo puede haber espacios entre los clientes habilitados. La última línea exporta el directorio */home* a todas las máquinas de la red 192.168.0.* en modo syncronizado, de lectura/escritura, con acceso del *root* remoto, *fsid* es la identificación de sistema de archivo y *no_subtree_check* indica que no se hará la verificación de la ruta/archivo en una petición sobre el servidor.

Dos comandos útiles para trabajar con el nfs son el `exportfs` (muestra y nos permite actualizar las modificaciones que se hayan realizado sobre */etc/exports*) y `nfsiostat/nfsstat` que nos permitirá obtener estadísticas de funcionamiento sobre NFS y observar su funcionamiento.

15. OpenLdap (Ldap)

LDAP significa *Lightweight Directory Access Protocol* y es un protocolo para acceder a datos basados en un servicio X.500. Este se ejecuta sobre TCP/IP y el directorio es similar a una base de datos que contiene información basada en atributos. El sistema permite organizar esta información de manera segura y utiliza réplicas para mantener su disponibilidad, lo que asegura la coherencia y la verificación de los datos accedidos-modificados.

El servicio se basa en el modelo cliente-servidor, donde existen uno o más servidores que contienen los datos; cuando un cliente se conecta y solicita información, el servidor responde con los datos o con un puntero a otro servidor de donde podrá extraer más información; sin embargo, el cliente solo verá un directorio de información global [Mou, Mal]. Para importar y exportar información entre servidores `ldap` o para describir una serie de cambios que se aplicarán al directorio, el formato utilizado se llama LDIF (LDAP *Data Interchange Format*). LDIF almacena la información en jerarquías orientadas a objetos que luego serán transformadas al formato interno de la base de datos. Un archivo LDIF tiene un formato similar a:

```
dn: o = UOC, c = SP o: UOC
objectclass: organization
dn: cn = Remix Nteum, o = UOC, c = SP
cn: Remix Nteum
sn: Nteum
mail: nteumuoc.edu
objectclass: person
```

Cada entrada se identifica con un nombre indicado como `DN` (*Distinguished Name*). El DN consiste en el nombre de la entrada más una serie de nombres que lo relacionan con la jerarquía del directorio y donde existe una clase de objetos (`objectclass`) que define los atributos que pueden utilizarse en esta entrada. LDAP provee un conjunto básico de clases de objetos: **grupos** (incluye listas desordenadas de objetos individuales o grupos de objetos), **localizaciones** (tales como países y su descripción), **organizaciones y personas**. Una entrada puede, además, pertenecer a más de una clase de objeto, por ejemplo, un individuo es definido por la clase persona, pero también puede ser definido por atributos de las clases `inetOrgPerson`, `groupOfNames` y `organization`.

La estructura de objetos del servidor (llamado *schema*) determina cuáles son los atributos permitidos para un objeto de una clase (que se definen en el archivo */etc/ldap/schema* como `inetorgperson.schema, nis.schema, opeldap.schema, corba.schema`, etc.). Todos los datos se representan como un par atributo = valor, donde el atributo es descriptivo de la información

que contiene; por ejemplo, el atributo utilizado para almacenar el nombre de una persona es `commonName`, o cn, es decir, una persona llamada Remix Nteum se representará por cn: Remix Nteum y llevará asociado otros atributos de la clase persona como `givenname: Remix surname: Nteum mail: nteum@uoc.edu`. En las clases existen atributos obligatorios y optativos y cada atributo tiene una sintaxis asociada que indica qué tipo de información contiene el atributo, por ejemplo, `bin` (*binary*), `ces` (*case exact string*, debe buscarse igual), `cis` (*case ignore string*, pueden ignorarse mayúsculas y minúsculas durante la búsqueda), `tel` (*telephone number string*, se ignoran espacios y '-') y `dn` (*distinguished name*). Un ejemplo de un archivo en formato LDIF podría ser:

```
dn: dc = UOC, dc = com
objectclass: top
objectclass: organizationalUnit

dn: ou = groups, dc = UOC, dc = com
objectclass: top
objectclass: organizationalUnit
ou: groups

dn: ou = people, dc = UOC, dc = com
objectclass: top
objectclass: organizationalUnit
ou: people

dn: cn = Remix Nteum, ou = people, dc = UOC, dc = com
cn: Remix Nteum
sn: Nteum
objectclass: top
objectclass: person
objectclass: posixAccount
objectclass: shadowAccount
   uid:remix userpassword:{crypt}p1pss2ii(0pgbs*do& = )eksd uidnumber:104
   gidnumber:100
   gecos:Remix Nteum
   loginShell:/bin/bash
   homeDirectory: /home/remix
   shadowLastChange:12898
   shadowMin: 0
   shadowMax: 999999
   shadowWarning: 7
   shadowInactive: -1
   shadowExpire: -1
   shadowFlag: 0

dn:
cn = unixgroup, ou = groups, dc = UOC, dc = com
```

```
objectclass: top
objectclass: posixGroup
cn: unixgroup
   gidnumber: 200
   memberuid: remix
   memberuid: otro-usuario
```

Las líneas largas pueden continuarse debajo comenzando por un espacio o un tabulador (formato LDIF). En este caso, se ha definido la base DN para la institución `dc = UOC, dc = com`, la cual contiene dos subunidades: `people` y `groups`. A continuación, se ha descrito un usuario que pertenece a `people` y a `group`. Una vez preparado el archivo con los datos, este debe ser importado al servidor para que esté disponible para los clientes LDAP. Existen herramientas para transferir datos de diferentes bases de datos a formato LDIF [Mal]. Sobre Debian, se debe instalar el paquete `slapd` y `ldap-utils`, que es el servidor de OpenLdap y un conjunto de utilidades para acceder a servidores locales y remotos. Durante la instalación, solicitará un passwd para gestionar la administración del servicio, se generará una configuración inicial y se pondrá en marcha el servicio que se puede verificar con el comando `slapcat` que tomando la información disponible (FQDN de */etc/hosts*) generará algo como:

```
   dn: dc=nteum,dc=org
   objectClass: top
   objectClass: dcObject
   objectClass: organization
   o: nteum.org
   dc: nteum
...

   dn: cn=admin,dc=nteum,dc=org
   objectClass: simpleSecurityObject
   objectClass: organizationalRole
   cn: admin
   description: LDAP administrator userPassword::
    e1NTSEF9TU9jVE1qWllPVFBmd2FiZWJtSjcrY0pYd2wvaTk5aUc=
...
```

Con los comandos `ldapadd` y `ldapdelete` se pueden agregar/borrar registros de la tabla del servicio utilizando normalmente archivos de texto (sobre todo para el `add`) donde estén definidos los nuevos registros. Si bien no es un procedimiento complejo para gestionar el servidor puede ser más adecuado, en los primeros pasos, utilizar algunas de las aplicaciones que permiten gestionar el servidor de un forma más amigable como por ejemplo `phpldapadmin` (administración por medio apache+php), `jxplorer` (aplicación en java), `gosa` o

`lat` (todos ellos en la mayoría de las distribuciones). En nuestro caso instalaremos `phpldapadmin` que combina simplicidad y permite gestionar la mayoría de las opciones del servidor Ldap.

Para instalarlo ejecutamos `apt-get install phpldapadmin` el cual ya reiniciará el servidor apache pero sino lo hace podemos reiniciarlo con la orden `/etc/init.d/apache2 restart` (o `systemctl restart apache2`). Para hacer la conexión, en un navegador introducimos la siguiente dirección `http://localhost/phpldapadmin/`. Saldrá la página de bienvenida y en la banda izquierda podremos hacer el *login* (si el *ServerName* de apache no coincide con el LDAP no pedirá como usuario DN por lo cual le deberemos indicar el correcto cn=admin,dc=nteum,dc=org y la contraseña del servidor LDAP). Una vez conectado al servidor, seleccionamos la raíz (dc=nteum,dc=org) y seleccionamos entre los *templates* que aparecen una *Organizational Unit (ou)* a la que llamaremos `users`, repetimos los pasos (opción `create new entry` desde la raíz) y creamos otra *ou* llamada `groups`. Ahora solamente nos queda crear los usuarios y los grupos y asignar los usuarios a sus grupos. Dentro de la unidad organizativa `groups` crearemos los grupos `ventas` (gid=1001) y `compras` (gid=1002) y en la unidad organizativa `users` crearemos los usuarios `juan pirulo` (uid=1001, ventas, id=jpirulo) y `ana pirulo` (uid=1002, compras, id=apirulo). Para los grupos seleccionamos la OU `groups`, hacemos `create new child` y seleccionamos `Posix Group`. Creamos los dos grupos indicados pero deberemos modificar el `gid` ya que los asigna por defecto a partir de 1.000 y nosotros los queremos diferentes.

Repetimos la operación en `users` seleccionando `User Account`. Aquí nos pedirá los datos de los usuarios (nombre, apellido, grupo, contraseñas, *home directory*, *shell*, etc.) y luego deberemos cambiar el `uid`, ya que los asigna por defecto. A continuación podemos observar nuevamente el contenido ejecutando la instrucción `slapcat` los nuevos datos generados. Phpldapadmin se puede configurar con el archivo

/usr/share/phpldapadmin/config/config.php

pero en la mayoría de los casos la configuración por defecto ya es operativa (en algunos casos puede ser necesario adaptar la línea $servers->setValue('server','base',array('dc=nteum,dc=org')); para adecuar a los datos de nuestro servidor).

Para poder configurar un cliente tenemos que instalar los paquetes *libnss-ldap libpam-ldap ldap-utils* ejecutando la orden

```
apt-get install libnss-ldap libpam-ldap ldap-utils,
```

que nos solicitará una serie de información: URI del servidor (ldap://192.168.1.33), el DC (dc=nteum,dc=org), la versión (3), el usuario administrativo (cn=admin, dc=nteum,dc=org) y el `passwd`, nos informará que

hagamos el cambio manual de *nsswitch* (ok), permitir a PAM cambiar los `password` locales (Yes), `enforces login` (No), `admin acount suffix` (cd=admin,dc=nteum, dc=org), y finalmente el `passwd` nuevamente.

A continuación deberemos modificar el archivo */etc/nsswitch.conf* con:

```
passwd: compat ldap
group: compat ldap
shadow: compat ldap
...
netgroup: ldap
```

Por último deberemos modificar el archivo */etc/pam.d/common-password* (quitando de la línea el 'use_authtok')

```
password [success=1 user_unknown=ignore default=die] pam_ldap.so try_first_pass
```

y en */etc/pam.d/common-session* agregar al final (para crear el home directory automáticamente):

```
session optional pam_mkhomedir.so skel=/etc/skel umask=077
```

Luego deberemos reiniciar la máquina (`shutdown -r now`) y ya nos podremos conectar con los usuarios que hemos creado (jpirulo o apirulo) desde la interfaz gráfica. Es interesante configurar LDAP sobre TLS (protocolo cifrado) para dar más seguridad a la conexión (véase los detalles de configuración en http://www.server-world.info/en/note?os=Debian_8&p=openldap&f=4).

16. *Virtual private network* (VPN)

Una VPN[33] es una red que utiliza Internet como transporte de datos, pero impide que estos puedan ser accedidos por miembros externos a ella.

(33)Del inglés *virtual private network*.

Tener una red con VPN significa tener nodos unidos a través de un túnel por donde viaja el tráfico y donde nadie puede interactuar con él. Se utiliza cuando se tienen usuarios remotos que acceden a una red corporativa para mantener la seguridad y privacidad de los datos. Para configurar una VPN, se pueden utilizar diversos métodos SSH (SSL), CIPE, IPSec, PPTP y pueden consultarse en las referencias [Bro] [Wil] del proyecto TLDP que, si bien son un poco antiguas, aún nos son útiles porqué muestran los principios y conceptos básicos de la infraestructura VPN.

Para realizar las pruebas de configuración, en este apartado se utilizará la **OpenVPN**, que es una solución basada en SSL VPN, y se puede usar para un amplio rango de soluciones, por ejemplo, acceso remoto, VPN punto a punto, redes WiFi seguras o redes distribuidas empresariales. OpenVPN implementa OSI layer 2 o 3 utilizando protocolos SSL/TLS y soporta autentificación basada en certificados, tarjetas (*smart cards*), y otros métodos de certificación.

OpenVPN no es un servidor *proxy* de aplicaciones ni opera a través de un *web browser*.

16.1. Instalación y prueba en modo *raw*

En este apartado utilizaremos una máquina Debian y otra Ubuntu, pero es similar en el resto de las distribuciones. Primero hay que instalar en ambas máquinas OpenVPN: `apt-get install openvpn`. En función de la distribución podrá dar algunos errores, ya que intenta arrancar el servicio, pero como todavía no está configurado muestra algunos avisos. Después, se debe probar en modo *raw* si la conectividad entre servidor y cliente funciona o existe algún impedimento (por ejemplo, un *firewall*). Para comprobarlo, deberemos ejecutar lo siguiente:

- Desde el servidor:

```
openvpn --remote ubub --dev tun1 --ifconfig 10.9.8.1 10.9.8.2
```

- Desde el cliente:

```
openvpn -remote deba -dev tun1 -ifconfig 10.9.8.2 10.9.8.1
```

Ambas máquinas deben estar conectadas (en nuestro caso, el nombre de la máquina cliente es *ubub, 10.9.8.2* –Ubuntu– y su IP en el túnel VPN, y *deba, 10.9.8.1*, el nombre de la máquina servidor y la IP en el túnel VPN). La salida será similar en ambas máquinas; por ejemplo, en el servidor:

```
Thu Jun 12 12:50:31 2014 OpenVPN 2.2.1 x86_64-linux-gnu [SSL] [LZO2] [EPOLL] [PKCS11] [eurephia]
      [MH] [PF_INET6] [IPv6 payload 20110424-2 (2.2RC2)] built on Jun 18 2013
Thu Jun 12 12:50:31 2014 IMPORTANT: OpenVPN's default port number is now 1194, based on
     an official port number assignment by IANA.  OpenVPN 2.0-beta16 and earlier used 5000
  as the default port.
Thu Jun 12 12:50:31 2014 NOTE: OpenVPN 2.1 requires '--script-security 2' or higher to call
     user-defined scripts or executables
Thu Jun 12 12:50:31 2014 ******* WARNING *******: all encryption and authentication features
     disabled -- all data will be tunnelled as cleartext
Thu Jun 12 12:50:31 2014 TUN/TAP device tun1 opened
Thu Jun 12 12:50:31 2014 do_ifconfig, tt->ipv6=0, tt->did_ifconfig_ipv6_setup=0
Thu Jun 12 12:50:31 2014 /sbin/ifconfig tun1 10.9.8.1 pointopoint 10.9.8.2 mtu 1500
Thu Jun 12 12:50:31 2014 UDPv4 link local (bound): [undef]
Thu Jun 12 12:50:31 2014 UDPv4 link remote: [AF_INET]172.16.1.2:1194
Thu Jun 12 12:50:34 2014 Peer Connection Initiated with [AF_INET]172.16.1.2:1194
Thu Jun 12 12:50:35 2014 Initialization Sequence Completed
```

Si hacemos un *ping* sobre el servidor (desde otro terminal) veremos que ambas puntas de túnel funcionan:

- ping 10.9.8.1:

```
PING 10.9.8.1 (10.9.8.1) 56(84) bytes of data.
64 bytes from 10.9.8.1: icmp_req=1 ttl=64 time=0.027 ms
--- 10.9.8.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.027/0.045/0.064/0.019 ms
```

- ping 10.9.8.2:

```
PING 10.9.8.2 (10.9.8.2) 56(84) bytes of data.
64 bytes from 10.9.8.2: icmp_req=1 ttl=64 time=0.597 ms
--- 10.9.8.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.597/1.131/1.666/0.535 ms
```

Y también podremos ver la interfaz como *tun1* con `ifconfig`:

```
tun1      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
```

```
inet addr:10.9.8.1  P-t-P:10.9.8.2  Mask:255.255.255.255
UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
RX packets:5 errors:0 dropped:0 overruns:0 frame:0
TX packets:5 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:100
RX bytes:420 (420.0 B)  TX bytes:420 (420.0 B)
```

Para terminar la aplicación, simplemente hay que hacer un *Crtl+C* (y veréis cómo desaparece también la interfaz virtual *tun1*). El siguiente paso después de verificada la conectividad es realizar la configuración.

16.2. VPN con intercambio de llaves estáticas

Para analizar este servicio, utilizaremos una opción de la OpenVPN llamada *OpenVPN for Static key configurations*, que ofrece una forma simple de configurar una VPN ideal para pruebas o para conexiones punto a punto. Sus ventajas son la simplicidad y que no es necesario un certificado X509 PKI[34] para mantener la VPN. Las desventajas son que solo permite un cliente y un servidor. Al no utilizar el mecanismo de PKI (llave pública y llave privada), puede haber igualdad de claves con sesiones anteriores, debe existir, pues, una llave en modo texto en cada *peer* y la llave secreta debe ser intercambiada anteriormente por un canal seguro.

(34)Del inglés *public key infrastructure*.

Recordemos que nuestro túnel VPN tendrá sobre el servidor IP=10.9.8.1 y el cliente con IP=10.9.8.2. La comunicación será encriptada entre el cliente y el servidor sobre UDP port 1194 (que es el puerto por defecto de OpenVPN). Después de instalar el paquete, se deberá generar la llave estática en */etc/openvpn* y copiarla de forma segura al cliente:

```
cd /etc/openvpn
openvpn --genkey --secret static.key
scp static.key ubub:/etc/openvpn
```

En nuestro caso hemos utilizado el comando *secure copy* (`scp`) para transferir el archivo *static.key* al cliente */ubub*) sobre un canal seguro.

El archivo de configuración del servidor */etc/openvpn/tun0.conf*:

```
dev tun0
ifconfig 10.9.8.1 10.9.8.2
secret /etc/openvpn/static.key
```

El archivo de configuración del cliente */etc/openvpn/tun0.conf*:

```
remote deba
dev tun0
ifconfig 10.9.8.2 10.9.8.1
```

```
secret /etc/openvpn/static.key
```

Antes de verificar el funcionamiento de la VPN, debe asegurarse en el *firewall* que el puerto 1194 UDP está abierto sobre el servidor y que la interfaz virtual *tun0* usada por OpenVPN no está bloqueada ni sobre el cliente ni sobre el servidor. Tengamos en mente que el 90 % de los problemas de conexión encontrados por usuarios nuevos de OpenVPN están relacionados con el *firewall*.

Para verificar la OpenVPN entre dos máquinas, hay que recordar de cambiar las IP y el dominio por el que tenga en su configuración, y luego ejecutar tanto del lado servidor como del cliente (*--verb 6* mostrará información adicional a la salida y puede evitarse en posteriores ejecuciones):

```
openvpn --config /etc/openvpn/tun0.conf --verb 6
```

El cual dará una salida similar a:

```
Thu Jun 12 13:59:58 2014 us=359502 OpenVPN 2.2.1 x86_64-linux-gnu [SSL] [LZO2] [EPOLL] [PKCS11]
      [eurephia] [MH] [PF_INET6] [IPv6 payload 20110424-2 (2.2RC2)] built on Jun 18 2013
.
Thu Jun 12 13:59:58 2014 us=361008 TUN/TAP device tun0 opened
Thu Jun 12 13:59:58 2014 us=361049 TUN/TAP TX queue length set to 100
Thu Jun 12 13:59:58 2014 us=361081 do_ifconfig, tt->ipv6=0, tt->did_ifconfig_ipv6_setup=0
Thu Jun 12 13:59:58 2014 us=361113 /sbin/ifconfig tun0 10.9.8.1 pointopoint 10.9.8.2 mtu 1500
Thu Jun 12 13:59:58 2014 us=363861 Data Channel MTU parms [ L:1544 D:1450 EF:44 EB:4 ET:0 EL:0 ]
Thu Jun 12 13:59:58 2014 us=363927 Local Options String: 'V4,dev-type tun,link-mtu 1544,tun-mtu
     1500,proto UDPv4,ifconfig 10.9.8.2 10.9.8.1,cipher BF-CBC,auth SHA1,keysize 128,secret'
Thu Jun 12 13:59:58 2014 us=363947 Expected Remote Options String: 'V4,dev-type tun,link-mtu
     1544,tun-mtu 1500,proto UDPv4,ifconfig 10.9.8.1 10.9.8.2,cipher BF-CBC,auth SHA1,keysize
       128,secret'
..
Thu Jun 12 14:00:37 2014 us=657976 Peer Connection Initiated with [AF_INET]172.16.1.2:1194
Thu Jun 12 14:00:37 2014 us=658000 Initialization Sequence Completed
```

Para verificar su funcionamiento se puede ejecutar, por ejemplo, un *ping 10.9.8.1* desde el cliente, con lo cual veremos su respuesta y además en la consola del servidor un mensaje de la actividad del túnel similar a:

```
Thu Jun 12 14:06:54 2014 us=604089 TUN READ [84]
Thu Jun 12 14:06:54 2014 us=604113 UDPv4 WRITE [124] to [AF_INET]172.16.1.2:1194:  DATA len=124
Thu Jun 12 14:06:55 2014 us=604293 UDPv4 READ [124] from [AF_INET]172.16.1.2:1194:  DATA len=124
Thu Jun 12 14:06:55 2014 us=604342 TUN WRITE [84]
Thu Jun 12 14:06:55 2014 us=604370 TUN READ [84]
Thu Jun 12 14:06:55 2014 us=604395 UDPv4 WRITE [124] to [AF_INET]172.16.1.2:1194:  DATA len=124
```

Para agregar compresión sobre el link, debe añadirse la siguiente línea a los dos archivos de configuración **comp-lzo** y para proteger la conexión a través de un NAT *router/firewall* y seguir los cambios de IP a través de un DNS, si uno de los *peers* cambia, agregar a los dos archivos de configuración:

```
keepalive 10 60
ping-timer-rem
persist-tun
persist-key
```

Nota

Podéis obtener más información al respecto en la web de OpenVPN: http://openvpn.net/index.php/open-source/documentation/miscellaneous/78-static-key-mini-howto.html

Como hemos visto, la ejecución anterior es en consola, por lo que queda la consola bloqueada con la ejecución de la VPN; pero una vez depurada, es necesario poner en marcha como *daemon* y para ello es necesario agregar al archivo de configuración */etc/openvpn/tun0.conf* la palabra *daemon* y cambiar en el archivo */etc/default/openvpn* la línea de AUTOSTART="tun0" para indicarle el nombre de nuestra VPN (también se puede poner "all" que ejecutará todos los archivos de configuración */etc/openvpn/*.conf*). Luego queda ponerla en marcha en ambos lados (`service openvpn start`) y probar con un `ifconfig` que existe el *tun0* y con un `ping` su funcionalidad.

Una vez establecida la conexión, podemos utilizarla para conectarnos como si se tratara de cualquier otra IP; por ejemplo, desde el servidor podemos hacer un `ssh 10.9.8.2` que nos conectará al cliente por `ssh` pero a través del túnel encriptado (con el `ssh` no es necesario, ya que encripta la comunicación, pero se utiliza a modo de ejemplo).

16.3. VPN con TLS

Si bien como prueba de concepto una VPN con llave estática es muy adecuada, ya que tiene una configuración simple y no necesita de infraestructura PKI, tiene los inconvenientes de la escalabilidad (un servidor y un cliente). La llave, que es la misma, existe en texto en cada *peer* y debe ser copiada por un canal previo seguro. La configuración con TLS nos permitirá resolver estos inconvenientes si bien su configuración y puesta en marcha será algo más compleja.

En primer lugar debemos copiar el *script* que nos permitirá generar las llaves:

```
cd /etc/openvpn
mkdir easy-rsa
cp -R /usr/share/doc/openvpn/examples/easy-rsa/2.0/* easy-rsa/
```

Después, modificar */etc/openvpn/easy-rsa/vars* para reflejar su localización:

```
export KEY_COUNTRY="SP"
```

```
export KEY_PROVINCE="BCN"
export KEY_CITY="Bellaterra"
export KEY_ORG="Nteum"
export KEY_EMAIL="root@deba"
```

Y, finalmente, ejecutar:

```
cd /etc/openvpn/easy-rsa;
mkdir keys; touch keys/index.txt; echo 01 > keys/serial
. ./vars
./clean-all
```

Recordar que los archivos *.key son confidenciales, los archivos *.crt y *.csr pueden ser enviados por canales inseguros (por ejemplo, por email) y que cada ordenador deberá tener su par *certificate/key*.

Con esto ya podemos crear la entidad certificadora (CA) que generará los archivos */etc/openvpn/easy-rsa/keys/ca.crt* y */etc/openvpn/easy-rsa/keys/ca.key*:

```
./build-ca
```

A continuación, generaremos una *inntermediate certificate authority certificate/key* que creará los archivos *server.crt* y *server.key* en */etc/openvpn/easy-rsa/keys/* firmados por la CA:

```
./build-key-server server
```

A continuación se deben generar los archivos Diffie-Hellamn (necesarios en la comunicación SSL/TLS) y, seguidamente, las llaves para el cliente (*ubub* en nuestro caso):

```
./build-dh
./build-key ubub
```

Este último paso generará en el directorio */etc/openvpn/easy-rsa/keys/* los archivos *ubub.crt* y *ubub.key*, los cuales deberán ser copiados en el mismo directorio (*/etc/openvpn/easy-rsa/keys/*) del cliente y también el certificado de la ca (*ca.crt*). Con esto ya podemos probar la conexión desde consola.

Se debe tener en cuenta que el primer comando hay que ejecutarlo en el servidor y el segundo en el cliente, y que es una única línea tanto para el servidor como para el cliente:

```
openvpn --dev tun1 --ifconfig 10.9.8.1 10.9.8.2 --tls-server --dh /etc/openvpn/easy-rsa/keys
    /dh1024.pem --ca /etc/openvpn/easy-rsa/keys/ca.crt --cert /etc/openvpn/easy-rsa/keys
 /server.crt --key /etc/openvpn/easy-rsa/keys/server.key --reneg-sec 60 --verb 5
```

```
openvpn --remote deba --dev tun1 --ifconfig 10.9.8.2 10.9.8.1 --tls-client --ca /etc/openvpn
    /easy-rsa/keys/ca.crt --cert /etc/openvpn/easy-rsa/keys/ubub.crt --key /etc/openvpn
 /easy-rsa/keys/ubub.key --reneg-sec 60 --verb 5
```

La salida será equivalente a la que vimos en el punto anterior y ahora con el `ifconfig` tendremos un dispositivo *tun1*. Desde otra terminal podemos realizar un `ping` para comprobar su funcionalidad. Observar el texto remarcado que se deberá reemplazar con los archivos que generéis para su configuración.

A continuación, crearemos los archivos de configuración en el servidor y el cliente para que arranque en forma automática. Primero, el directorio de *log*

```
mkdir -p /etc/openvpn/log/; touch /etc/openvpn/log/openvpn-status.log
```

y, luego, editaremos */etc/openvpn/server.conf*:

```
port 1194
proto udp
dev tun
ca      /etc/openvpn/easy-rsa/keys/ca.crt    # generated keys
cert    /etc/openvpn/easy-rsa/keys/server.crt
key     /etc/openvpn/easy-rsa/keys/server.key  # keep secret
dh      /etc/openvpn/easy-rsa/keys/dh1024.pem
server 10.9.8.0 255.255.255.0  # internal tun0 connection IP
ifconfig-pool-persist ipp.txt
keepalive 10 120
comp-lzo         # Compression - must be turned on at both end
persist-key
persist-tun
status log/openvpn-status.log
verb 3  # verbose mode
client-to-client
```

A continuación, se deberá modificar la entrada AUTOSTART="server" del archivo */etc/default/openvpn* para iniciar el servicio (`service openvpn start`). Sobre el cliente, crearemos el archivo */etc/openvpn/client.conf*:

```
client
dev tun
port 1194
proto udp
remote deba 1194             # VPN server IP : PORT
nobind
ca /etc/openvpn/easy-rsa/keys/ca.crt
cert /etc/openvpn/easy-rsa/keys/ubub.crt
key /etc/openvpn/easy-rsa/keys/ubub.key
```

```
comp-lzo
persist-key
persist-tun
verb 3
```

También tenemos que modificar la entrada AUTOSTART="client" en el archivo */etc/default/openvpn* para poner en marcha el servicio con la siguiente instrucción: `service openvpn start`. Veremos que se ha creado la interfaz *tun0* y que responde al `ping` de los diferentes comandos sobre la IP del túnel del servidor. A partir de este servidor y con pequeños cambios podremos utilizar las Apps OpenVPN para iOS y para Android o para renviar el tráfico IP a través del túnel VPN.

Nota

Podéis ver los detalles de estas acciones en https://wiki.debian.org/OpenVPN

En ciertas ocasiones es necesario que los clientes puedan configurarse su conexión a la VPN por lo cual una interfaz gráfica puede ayudar. Existen dos paquetes como *pluguins* en el NetworkManager que reducen la complejidad a la hora de configurar los clientes: `network-manager-openvpn` y para los que trabajen de escritorios Gnome, `network-manager-openvpn-gnome`. En primer lugar, se deben generar desde el servidor los certificados para el nuevo cliente y enviárselos por un canal seguro (sobre todo el archivo .key), posteriormente, se debe instalar uno de los dos paquetes (tener en cuenta que el segundo reemplazará al primero si este está instalado). Con estos *pluguins*, y una vez arrancado el NetworkManager, será muy fácil configurar el cliente para que este gestione la conexión a la VPN.

Nota

Podéis obtener más información al respecto en https://wiki.gnome.org/Projects/NetworkManager/Admins.

17. Configuraciones avanzadas y herramientas

Existe un conjunto de paquetes complementarios (o que sustituyen a los convencionales) y herramientas que o bien mejoran la seguridad de la máquina (recomendados en ambientes hostiles), o bien ayudan en la configuración de red (y del sistema en general) de modo más amigable.

Estos paquetes pueden ser de gran ayuda al administrador de red para evitar intrusos o usuarios locales que se exceden de sus atribuciones (generalmente, no por el usuario local, sino a través de una suplantación de identidad) o bien ayudar al usuario novel a configurar adecuadamente los servicios.

En este sentido, es necesario contemplar:

1) Configuración avanzada de TCP/IP: a través del comando *sysctl* es posible modificar los parámetros del kernel durante su ejecución o en el inicio para ajustarlos a las necesidades del sistema. Los parámetros susceptibles de modificar son los que se encuentran en el directorio */proc/sys/* y se pueden consultar con `sysctl -a`. La forma más simple de modificar estos parámetros es a través del archivo de configuración */etc/syscntl.conf* (recordar siempre de hacer una copia de los ficheros de configuración; por ejemplo, se puede hacer con la instrucción `cp /etc/syscntl.conf /etc/syscntl.conf.org` para poder retornar la configuración a su estado anterior en caso de problemas). Después de la modificación, se debe volver a arrancar la red:

```
/etc/init.d/networking restart              (o systemctl restart networking)
```

En este apartado veremos algunas modificaciones para mejorar las prestaciones de la red (mejoras según condiciones) o la seguridad del sistema (consultar las referencias para más detalles) [Mou]:

```
net.ipv4.icmp_echo_ignore_all = 1
```

No responde paquetes ICMP, como por ejemplo el comando `ping`, que podría significar un ataque DoS (*Denial-of-Service*).

```
net.ipv4.icmp_echo_ignore_broadcasts = 1
```

Evita congestiones de red no respondiendo el *broadcast*.

```
net.ipv4.conf.all.accept_source_route = 0
```

```
net.ipv4.conf.lo.accept_source_route = 0
net.ipv4.conf.eth0.accept_source_route = 0
net.ipv4.conf.default.accept_source_route = 0
```

Inhibe los paquetes de *IP Source routing* que podrían representar un problema de seguridad.

```
net.ipv4.tcp_syncookies = 1
net.ipv4.conf.all.accept_redirects = 0
```

Permite rechazar un ataque DoS por paquetes SYNC que consumiría todos los recursos del sistema forzando a hacer un *reboot* de la máquina.

```
net.ipv4.conf.lo.accept_redirects = 0
net.ipv4.conf.eth0.accept_redirects = 0
net.ipv4.conf.default.accept_redirects = 0
```

Útil para evitar ataques con *CMP Redirect Acceptance* (estos paquetes son utilizados cuando el routing no tiene una ruta adecuada) en todas las interfaces.

```
net.ipv4.icmp_ignore_bogus_error_responses = 1
```

Envía alertas sobre todos los mensajes erróneos en la red.

```
net.ipv4.conf.all.rp_filter = 1
net.ipv4.conf.lo.rp_filter = 1
net.ipv4.conf.eth0.rp_filter = 1
net.ipv4.conf.default.rp_filter = 1
```

Habilita la protección contra el *IP Spoofing* en todas las interfaces.

```
net.ipv4.conf.all.log_martians = 1
net.ipv4.conf.lo.log_martians = 1
net.ipv4.conf.eth0.log_martians = 1
net.ipv4.conf.default.log_martians = 1
```

Generará *log* sobre todos los *Spoofed Packets*, *Source Routed Packets* y *Redirect Packets*.

Los siguientes parámetros permitirán que el sistema pueda atender mejor y más rápido las conexiones TCP.

```
net.ipv4.tcp_fin_timeout = 40,          Por defecto, 60.
net.ipv4.tcp_keepalive_time = 3600,      Por defecto, 7.200.
net.ipv4.tcp_window_scaling = 0
net.ipv4.tcp_sack = 0
```

```
net.ipv4.tcp_timestamps = 0,        Por defecto, todos a 1 (habilitados).
```

2) Iptables: **Netfilter** es un conjunto de funcionalidades integradas en el *kernel* de Linux para interceptar y gestionar los paquetes de red. El principal componente de este es `iptables`, que funciona como una herramienta de cortafuegos (*firewall*) permitiendo no solo las acciones de filtro sino también de traslación de direcciones de red (NAT) –como ya se utilizó en el apartado correspondiente de este módulo– o redirecciones/registro de las comunicaciones. Este tema se verá en detalle en la asignatura "Administración avanzada", pero aquí daremos unos mínimos conceptos sobre este paquete. Con el comando iptables podemos gestionar las reglas haciendo, por ejemplo:

```
iptables -L              Para listar las reglas
iptable -t nat -L        Ídem anterior pero las de NAT
iptables -A Type -i Interf -p prot -s SrcIP --source-port Ps -d DestIP --destination-port Pd -j Action
                         Insertar una regla
iptables-save > /etc/iptables.rules
                         Salvar las reglas definidas
iptables-restore < /etc/iptables.rules
                         Restaurar reglas previamente salvadas
```

Un ejemplo del archivo /etc/iptables.rules podría ser (# indica comentario):

```
*filter
# Permitir todo el tráfico de loopback (lo0) y denegar el resto de 127/8
-A INPUT -i lo -j ACCEPT
-A INPUT ! -i lo -d 127.0.0.0/8 -j REJECT
# Aceptar todas las conexiones entrantes previamente establecidas
-A INPUT -m state --state ESTABLISHED, RELATED -j ACCEPT
# Aceptar todo el tráfico saliente
-A OUTPUT -j ACCEPT
# Permitir HTTP y HTTPS desde cualquier lugar
-A INPUT -p tcp --dport 80 -j ACCEPT
-A INPUT -p tcp --dport 443 -j ACCEPT
# Permitir las conexiones de SSH
# Normalmente utiliza el puerto 22, verificarlo en el archivo /etc/ssh/sshd_config.
 -A INPUT -p tcp -m state --state NEW --dport 22 -j ACCEPT
# Responder al ping icmp
-A INPUT -p icmp -m icmp --icmp-type 8 -j ACCEPT
# Rechazar todo el tráfico restante de entrada.
-A INPUT -j REJECT
-A FORWARD -j REJECT
COMMIT
```

La sintaxis puede parecer un poco extraña pero es fácil de aprender. En este caso las reglas se han puesto para que actúe como *firewall* y, como deberán ser cargadas por la instrucción `iptables-restore`, no incluyen la instrucción `iptables` al inicio de cada regla y es necesario el *commit* al final. En este ejemplo solo aceptamos tráfico entrante de *ping*, *http*, *htpps* y *ssh*, y bloqueamos todo el tráfico restante de entrada. En cuanto al tráfico de salida, se deja que salga todo sin restricciones. Se pueden salvar las reglas en un archivo, cargarlas y, por ejemplo desde otra máquina, probar la conectividad o ejecutar algún programa como `nmap` que nos mostrará los puertos abiertos en la máquina configurada con `iptables`.

Nota

Podéis obtener más información al respecto en https://wiki.debian.org/es/iptables.

3) GnuPG: GnuPG es una implementación completa del estándar OpenPGP definido por la RFC 4880. GnuPG (o GPG) permite cifrar y firmar datos de todo tipo y cuenta con un sistema de gestión de llaves muy versátil, así como de los módulos de acceso para todo tipo de directorios de claves públicas. GPG puede funcionar en línea de comandos o integrado a herramientas por medio de sus librerías y también proporciona soporte para S/MIME (*Secure/Multipurpose Internet Mail Extensions*).

Para crear el par de llaves se debe ejecutar `gpg --gen-key` respondiendo a las preguntas que nos hará.

Nota

Podéis obtener más información al respecto en https://wiki.debian.org/Keysigning y en http://moser-isi.ethz.ch/gpg.html.

Para visualizar las llaves creadas, ejecutar

```
gpg --list-keys o gpg -v -fingerprint
```

lo cual nos dará un resultado como:

```
pub   2048R/119EDDEA 2014-06-13
Key fingerprint = 6644 5BF2 9926 441C A211  B684 DA35 616A 119E DDEA
uid                  Remo Suppi (Universitat Autònoma de Barcelona) <Remo.Suppi@uab.cat>
sub   2048R/C8A79D4F 2014-06-13
```

Lo siguiente es crear un certificado de revocación ya que, si bien ahora no es importante, cuando la llave pública esté en un servidor y se quiera destruir (por diferentes cuestiones), la única forma es revocarla. Para ello, debemos ejecutar `gpg -a --gen-revoke` y copiar la información entre [BEGIN] y [END] en un archivo y ponerlo a resguardo, ya que con este se podrá anular la llave pública.

Después, se debe hacer "pública" la llave pública en algún servidor, como por ejemplo *pgp.mit.edu*; para ello, ejecutar

```
gpg --keyserver=x-hkp://pgp.mit.edu -a --send-keys 119EDDEA,
```

donde el último número es el *Key-id* que hemos obtenido de la clave. Para incorporar una llave pública de otro usuario a nuestro *key-ring*, haremos

```
gpg --import <file>
```

o preguntarle al `key-server gpg --search-keys <description>`, con lo cual obtendremos la llave necesaria para luego usarla para encriptar datos para ese usuario, por ejemplo. Usaremos `gpg --delete-key <description>` para borrarla.

GPG resuelve el problema de la necesidad de intercambio de claves con antelación con el mecanismo de llaves públicas y privadas, pero con ello aparece un nuevo problema: si recibo (por ejemplo, desde un servidor de claves) una llave pública de alguien, ¿cómo sé que esta llave pertenece realmente a la persona a la que se dice que pertenece? Cualquiera podría crear una llave en mi nombre y usarlo ¡y nadie se daría cuenta de que no soy yo! Es por ello por lo que se debe ser cuidadoso al aceptar otras llaves y garantizar que esa llave es de una determinada persona y que estoy seguro (comprobándolo con el *Key-ID* y el *fingerprint*). Un mecanismo fácil es firmar una llave y cuantos más usuarios conocidos firmen esta llave más (todos) estaremos seguros de que pertenece al usuario que conocemos (es lo que se denomina *key-signing*). Yo puedo firmar otra llave (con la mía) para garantizar que esa llave pertenece a quien estoy seguro que es, y también si recibo una llave pública de alguien que no sé quién es pero está firmada por varias personas que conozco, entonces puedo confiar en que esta llave pertenece a esa persona (confianza).

Para firmar una llave pública debe estar en el *key-ring* y ejecutar la orden `gpg --edit-key <description>`, donde "description" puede ser el nombre/email o cualquier dato o parte de la llave que queremos firmar. Después entraremos en modo comando e indicamos `sign -> grado de confianza (0-3) -> Y -> passwd -> save`.

Ahora se debería subir nuevamente esta llave pública al servidor con la nueva firma: `gpg -a --send-keys <key-ID>`. Para actualizar las llaves que tenemos en nuestro *key-ring*, debemos hacer `gpg --refresh-keys`. Para encriptar un archivo, deberíamos hacer `gpg -e -a -r <description> file`, el cual se llamará *file.asc*; ya que hemos incluido el *-a* que indica que la salida debe ser ASCII, si no existe, se generará en binario como *file.gpg*. Para desencriptar se deberá hacer `gpg -d -o archivo_salida file.asc` que pedirá el *passwd* y lo generará en *archivo_salida*.

Para utilizarlo en el correo u otras aplicaciones, es conveniente integrar `gpg` con la aplicación utilizada, por ejemplo, para incluir `gpg` en `thunderbird` y poder firmar/encriptar correos deberemos instalar una extensión llamada `enigmail`.

Nota

Podéis obtener más información sobre `enigmail` en su página web.

4) Logcheck: una de las actividades de un administrador de red es verificar diariamente (más de una vez por día) los archivos *log* para detectar posibles ataques/intrusiones o eventos que puedan dar indicios sobre estas cuestiones. Esta herramienta selecciona (de los archivos *log*) información condensada de

problemas y riesgos potenciales y luego la envía al responsable, por ejemplo, a través de un correo. El paquete incluye utilidades para ejecutarse de modo autónomo y recordar la última entrada verificada para las subsiguientes ejecuciones. La lista de archivos que se tienen que monitorizar se almacena en */etc/logcheck/logcheck.logfiles* y la configuración por defecto es adecuada (si se modificó gran parte del archivo */etc/syslog.conf*). Logcheck puede funcionar en tres modalidades:

- *Paranoid*. Este modo es muy detallado y debería limitarse a casos específicos como *firewalls*.
- *Server*. Es el modo por defecto y es el recomendado para la mayoría de los servidores.
- *Workstation*. Es el modo adecuado para estaciones de escritorio.

Esta herramienta permite una configuración total de los filtros y de las salidas si bien puede ser complicado reescribirlos. Las reglas se pueden clasificar en "intento de intrusión" (*cracking*), almacenadas en */etc/logcheck/cracking.d/*; "alerta de seguridad" almacenadas en */etc/logcheck/violations.d/*, y las que son aplicadas al resto de los mensajes.

Nota

Podéis obtener más información al respecto en [HeMa] http://debian-handbook.info/browse/es-ES/stable/sect.supervision.html.

5) **PortSentry** y **Tripwire**. PortSentry forma parte de un conjunto de herramientas que proporcionan servicios de seguridad de nivel de *host* para GNU/Linux. PortSentry, LogSentry y Hostsentry protegen contra escaneos de puertos y detectan indicios de actividad sospechosa. Tripwire es una herramienta que ayudará al administrador notificando sobre posibles modificaciones y cambios en archivos para evitar posibles daños (mayores). Esta herramienta compara las diferencias entre los archivos actuales y una base de datos generada previamente para detectar cambios (inserciones y borrado), lo cual es muy útil para detectar posibles modificaciones de archivos vitales, como por ejemplo, en archivos de configuración.

6) Tcpdump y **Wireshark**: Tcpdump es una herramienta muy potente que está en todas las distribuciones y que nos servirá para analizar los paquetes de red. Este programa permite el volcado del tráfico de red de una red y puede analizar la mayoría de los protocolos utilizados hoy en día (IPv4, ICMPv4, IPv6, ICMPv6, UDP, TCP, SNMP, AFS BGP, RIP, PIM, DVMRP, IGMP, SMB, OSPF, NFS.). Wireshark es otra herramienta (más compleja) que dispone de una interfaz gráfica para analizar paquetes y permite también decodificarlos y analizar su contenido (actúa como *network sniffer*). Ambas herramientas se instalan bajo el procedimiento habitual y vienen preconfiguradas en casi todas las distribuciones.

Dada la importancia de analizar los paquetes hacia y saber adónde van, mostraremos algunos comandos habituales de tcpdump:

- **tcpdump**. Parámetros por defecto -v o -vv para el nivel de información mostrada, -q salida rápida.
- **tcpdump -D**. Interfaces disponibles para la captura.
- **tcpdump -n**. Muestra IP en lugar de direcciones.
- **tcpdump -i eth0**. Captura el tráfico de eth0.
- **tcpdump udp**. Solo los paquetes UDP.
- **tcpdump port http**. Solo los paquetes del puerto 80 (web).
- **tcpdump -c 20**. Solo los 20 primeros paquetes.
- **tcpdump -w capture.log**. Envía los datos a un archivo.
- **tcpdump -r capture.log**. Lee los datos de un archivo.
- **tcpdump host** *www.uoc.edu*. Solo los paquetes que contengan esta dirección.
- **tcpdump src** *192.168.1.100 and dst 192.168.1.2 and port ftp*. Muestra los paquetes ftp que vayan desde 192.168.1.100 a 192.168.1.2.
- **tcpdump -A**. Muestra el contenido de los paquetes.

Nota

Los datos capturados no son texto, por lo cual se puede o bien utilizar el mismo para su lectura u otra herramienta como Wireshark para realizar su lectura y decodificación.

7) Webmin: esta herramienta permite configurar y añadir aspectos relacionados con la red a través de una interfaz web. Si bien se continúa su desarrollo en muchas distribuciones, no se incluye por defecto. Para ejecutarla, una vez instalada desde un navegador hay que llamar a la URL *https://localhost:10000*, que solicitará la aceptación del certificado SSL y el usuario (inicialmente *root*) y su clave (*passwd*).

8) System-config-*: en Fedora (y, también en menor medida, en Debian) existe una gran variedad de herramientas gráficas que se llaman *system-config-"algunacosa"* y donde "alguna-cosa" es para lo cual están diseñadas. En general, si se está en un entorno gráfico, se puede llegar a cada una de ellas por medio de un menú.

9) Otras herramientas:

- **Nmap**: explorar y auditar con fines de seguridad una red.
- **OpenVas**: análisis de vulnerabilidades.
- **Snort**: sistema de detección de intrusos, IDS.
- **Netcat**: utilidad simple y potente para depurar y explorar una red.
- **MTR**: programa que combina la funcionalidad de traceroute y ping y es muy adecuada como herramienta de diagnóstico de red.
- **Hping2**: genera y envía paquetes de ICMP/UDP/TCP para analizar el funcionamiento de una red.

Ved también

Algunas de estas herramientas serán tratadas en el módulo de administración de seguridad en la asignatura *Administración avanzada de sistemas GNU-Linux*.

Actividades

1. Definid los siguientes escenarios de red:
a. Máquina aislada.
b. Pequeña red local (4 máquinas, 1 *gateway*).
c. 2 redes locales segmentadas (2 conjuntos de 2 máquinas, un *router* cada una y un gateway general).
d. 2 redes locales interconectadas (dos conjuntos de 2 máquinas + *gateway* cada una).
e. Una máquina con 2 interfaces conectada a Internet con NAT a un router y a una red privada1, una segunda máquina con dos interfaces conectada a red privada1 y la otra a una red privada2, una tercera máquina conectada a red privada2.
f. 2 máquinas conectadas a través de una red privada virtual.

Indicar las ventajas/desventajas de cada configuración, para qué tipo de infraestructura son adecuadas y qué parámetros relevantes se necesitan (tanto para IPv4 como para IPv6).

2. Utilizando máquinas virtuales, realizar la configuración y monitorización y test de conexión (por ejemplo, `ping`, `dig` y `apt-get update`) de las propuestas del punto anterior y en cada máquina de la arquitectura propuesta.

3. Realizar los experimentos anteriores sobre IPv6 utilizando ping6 y un túnel (por ejemplo, http://www.gogo6.net/freenet6/tunnelbroker) para mostrar la conectividad hacia Internet.

4. Configurad un servidor DNS como caché y con un dominio propio.

5. Configurad un servidor/cliente NIS con dos máquinas exportando los directorios de usuario del servidor por NFS.

6. Configurad un servidor SSH para acceder desde otra máquina sin contraseña.

Bibliografía

Todas las URL han sido visitadas en junio de 2016.

[Apachessl] SSL/TLS Strong Encryption: An Introduction. <http://httpd.apache.org/docs/2.2/ssl/ssl_intro.html#cryptographictech>

[Bro] Bronson, Scott (2001). "VPN PPP-SSH". *The Linux Documentation Project*, 2001.

[Bro] *Comparison of FTP client software* <http://en.wikipedia.org/wiki/Comparison_of_FTP_client_software>

[Cis] Cisco (2005). "TCP/IP Overview White Paper".

[Com] Comer, Douglas (2013). *Internetworking with TCP/IP Volume One*. Addison-Wesley.

[daCos] Costa, F. da. "Intel Technical Books, Rethinking the Internet of Things".

[Deb] Debian.org. *Debian Home*. <http://www.debian.org>

[DNS] Langfeldt, N. *DNS HOWTO*. <http://tldp.org/HOWTO/DNS-HOWTO.html>

[DNSMQ] DNSMasq. *DNS forwarder and DHCP server*. <https://wiki.debian.org/HowTo/dnsmasq>

[DR] Debian. "Debian Reference Manual. Network Setup Chapter 5".

[Gar98] Garbee, Bdale (1998). *TCP/IP Tutorial*. N3EUA Inc.

[Gnu] Gnupg.org. GnuPG Web Site.

[GRD] Debian. "Guía de referencia Debian Capítulo 10. Configuración de red".

[HandB] *Servicios de red: Postfix, Apache, NFS, Samba, Squid, LDAP* <http://debian-handbook.info/browse/es-ES/stable/network-services.html>

[HeMa] Hertzog, R.; Mas, R. (2015). "El libro del administrador Debian 8". ISBN: 979-10-91414-08-1. Libro electrónico disponible en español: https://debian-handbook.info/browse/es-ES/stable/index.html (ISBN: 979-10-91414-09-8)

[IET] IETF. "Repositorio de Request For Comment desarrollado por Internet Engineering Task Force (IETF) en el Network Information Center (NIC)".

[ITE] Instituto de Tecnologías Educativas. *Redes de área local: Aplicaciones y Servicios Linux*. <http://www.ite.educacion.es/formacion/materiales/85/cd/linux/indice.htm>

[KD] Kirch, Olaf; Dawson, Terry (2000). *Linux Network Administrator's Guide*. O'Reilly Associates [como *e-book* (*free*) en Free Software Foundation, Inc].

[Kuk] Kukuk, T. *The Linux NIS(YP)/NYS/NIS+ HOWTO -obs:EOL-*. <http://tldp.org/HOWTO/NIS-HOWTO/verification.html>

[LeCe] Leiner, B.; Cerf, V.; Clark, D. y otros. "Brief History of the Internet".

[Mal] Mallett, Fred (1996). *TCP/IP Tutorial*. FAME Computer Education.

[Mal] Pinheiro Malère, L. E. *Ldap. The Linux Documentation Project*. <http://tldp.org/HOWTO/LDAP-HOWTO/>

[Mara] Trenholme, S. *MaraDNS - A small open-source DNS server*. <http://maradns.samiam.org/>

[Miq] NIS HOWTO. <http://www.linux-nis.org/>

[Mou01] Mourani, Gerhard (2001). *Securing and Optimizing Linux: The Ultimate Solution*. Open Network Architecture, Inc.

[Muh] Transparent Multi-hop SSH. <http://sshmenu.sourceforge.net/articles/transparent-mulithop.html>

[OVPN] OpenVPN. "OpenVPN: Instalación y configuración".

[Proftpd] Proftpd. <http://www.debian-administration.org/articles/228>

[SW] Server-World. <http://www.server-world.info/en/note?os=Debian_8>

[Wil02] Wilson, Matthew D. (2002). "VPN". *The Linux Documentation Project*.

Anexo

Controlando los servicios vinculados a red en FCx

Un aspecto importante de todos los servicios es cómo se ponen en marcha. Fcx (desde la FC6) incluye una serie de utilidades para gestionar los servicios -daemons- (incluidos los de red). Como ya se ha visto en el apartado de administración local, el *runlevel* es el modo de operación que especifica que daemons se ejecutarán. En FC podemos encontrar: runlevel 1 (monousuario), runlevel 2 (multiusuario), runlevel 3 (multiusuario con red), runlevel 5 (X11 más (runlevel 3). Típicamente se ejecuta el nivel 5 o 3 si no se necesitan interfaces gráficas. Para determinar qué nivel se está ejecutando, se puede utilizar `/sbin/runlevel` y para saber qué nivel es el que se arranca por defecto `cat /etc/inittab` | grep :initdefault: que nos dará información como id:5:initdefault: (tambien se puede editar el */etc/inittab* para cambiar el valor por defecto).

Para visualizar los servicios que se están ejecutando, podemos utilizar `/sbin/chkconfig –list` y para gestionarlos podemos utilizar **system-config-services** en modo gráfico o `ntsysv` en la línea de comandos. Para habilitar servicios individuales podemos utilizar `chkconfig`. Por ejemplo, el siguiente comando habilita el servicio `crond` para los niveles 3 y 5: */sbin/chkconfig* ***--level 35 crond on.***

Independientemente de cómo se hayan puesto en marcha los servicios, se puede utilizar `/sbin/service –status-all` o individualmente `/sbin/service crond status` para saber cómo está cada servicio. Y también gestionarlo (start, stop, status, reload, restart); por ejemplo, `service crond stop` para pararlo o `service crond restart` para reiniciarlo.

Es importante **no deshabilitar los siguientes servicios** (a no ser que se sepa lo que se está haciendo): acpid, haldaemon, messagebus, klogd, network, syslogd. Los servicios más importantes vinculados a la red (aunque no se recogen todos, sí la mayoría de ellos en esta lista no exhaustiva) son:

- **NetworkManager**, **NetworkManagerDispatcher**: es un *daemon* que permite cambiar entre redes fácilmente (Wifi y Ethernet básicamente). Si solo tiene una red no es necesario que se ejecute.

- **Avahi-daemon**, **avahi-dnsconfd**: es una implementación de *zeroconf* y es útil para detectar dispositivos y servicios sobre redes locales sin DNS (es lo mismo que *mDNS)*.

- **Bluetooth**, **hcid**, **hidd**, **sdpd**, **dund**, **pand**: Bluetooth red inalámbrica es para dispositivos portátiles (*NO ES* wifi, 802.11). Por ejemplo, teclados, mouse, teléfonos, altavoces/auriculares, etc.

- **Capi**, **isdn**: red basada en hardware *ISDN*.

- **Iptables**: es el servicio de *firewall* estándar de Linux. Es totalmente necesario por seguridad si se tiene conexión a red *(cable, DSL, T1)*.

- **Ip6tables**: es el servicio de *firewall* estándar de Linux pero para el protocolo y redes basadas en Ipv6.

- **Netplugd**: puede monitorizar la red y ejecutar comando cuando su estado cambie.

- **Netfs**: se utiliza para montar automáticamente sistemas de archivos a través de la red (NFS, Samba, etc.) durante el arranque.

- **Nfs, nfslock:** son los daemon estándar para compartir sistemas de archivos a través de la red en sistemas operativos estilo Unix/Linux/BSD.

- **Ntpd**: servidor de hora y fecha a través de la red.

- **Portmap:** es un servicio complementario para NFS (*file sharing*) y/o NIS (*authentication*).

- **Rpcgssd**, **rpcidmapd**, **rpcsvcgssd:** se utiliza para NFS v4 (nueva versión de NFS).

- **Sendmail**: este servicio permite gestionar los mails (MTA) o dar soporte a servicios como IMAP o POP3.

- **Smb**: este daemon permite compartir ficheros sobre sistemas *Windows*.

- **Sshd**: SSH permite a otros usuarios conectarse interactivamente de forma segura a la máquina local.

- **Yum-updatesd:** servicio de actualizaciones por red de FC.

- **Xinetd:** servicio alternativo de **inetd** que presenta un conjunto de características y mejoras, como por ejemplo lanzar múltiples servicios por el mismo puerto (este servicio puede no estar instalado por defecto).