Introducción a networking

Este capítulo tratará los conceptos generales de networking. Partirá de lo básico y profundizará en temas específicos para comprender el porqué de determinadas configuraciones bajo GNU/Linux. Describiremos el modelo de referencia OSI, la caracterización de la suit de protocolos TCP/IP, y concluiremos con el detalle del esquema cliente-servidor de las redes actuales.

SERVICIO DE ATENCIÓN AL LECTOR: lectores@redusers.com

¿QUÉ ES NETWORKING?

Si queremos entender el correcto funcionamiento de una red de datos es fundamental introducir la teoría que nos proveerá de conceptos e ideas básicos para definir las herramientas que utilizaremos a lo largo del libro. Antes de comenzar a definir conceptos específicos y complicados de entender por primera vez (después se vuelven naturales), empecemos por describir qué es **networking**, bajo qué necesidades surgió esta área de conocimiento informático y su actual utilización e importancia para la sociedad.

Hace algunos años, cuando dos maquinarias relacionadas con las telecomunicaciones necesitaban intercambiar información, las personas transportaban instrucciones de un lugar a otro sin ningún medio de almacenamiento utilizado actualmente. Un famoso ejemplo son las tarjetas perforadas.

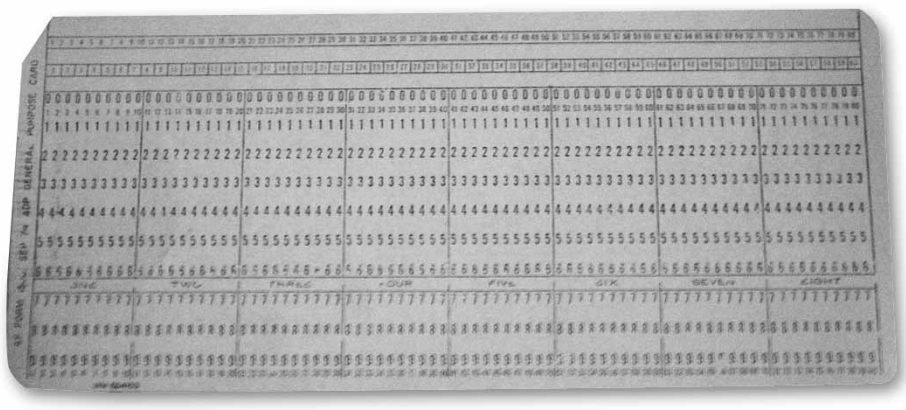

Figura 1. Las tarjetas perforadas fueron reemplazadas por medios de almacenamiento magnéticos y ópticos.

Con el avance de la electrónica fue posible para el matemático **George Stibitz** marcar un hito en la historia de las telecomunicaciones en septiembre de 1940.

III CABLES TWISTED PAIR

Los cables de par trenzado cuentan con un método de cableado en el que dos cables conductores se trenzan uno con el otro con el objetivo de cancelar campos de interferencia electromagnética y ruido proveniente de fuentes externas o de otros cables cercanos.

Mediante el uso de su máquina teletipo fue capaz de enviar instrucciones para la resolución de un problema de New Hampshire a su **calculadora de números complejos** en la ciudad de Nueva York.

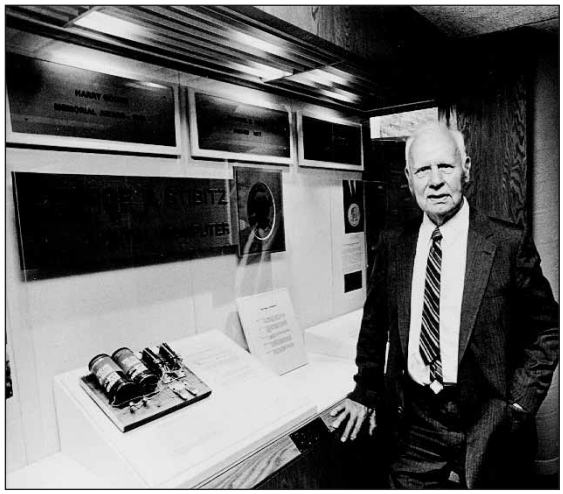

*Figura 2. **George Stibitz** con su calculadora **model K**, llamada así por haber sido construida en una mesa de cocina (**K**itchen en inglés).*

Durante la década del 60 se consolidó la idea de utilizar **datagramas** o **paquetes** que serían utilizados en una red de paquetes intercambiados (*switched network*). Aquí debemos proceder a definir el concepto de datagrama. Éste es un fragmento o paquete enviado con la suficiente información con la que una red puede simplemente intercambiar el paquete (o fragmento o datagrama) hacia el destinatario, y es independiente de la cantidad de fragmentos o paquetes restantes a transmitir. En principio, la idea parece interesante: tenemos una cantidad muy grande de información y en vez de enviarla de manera continua, la dividimos en **n** partes, y la enviamos una por una. La complejidad de este método yace en cómo ordenar los fragmentos de manera correcta en el receptor y qué hacer ante la pérdida de uno o más de éstos.
Consideremos el caso de que por alguna razón (generalmente **física** o de la **capa 1**, como veremos) un paquete no llega a destino. En resumen, la pérdida de paquetes

III IANA

La **IANA** (I*nternet Assigned Numbers Authority)* es la organización encargada de otorgar nombres y números globalmente únicos. O sea, la encargada de otorgar nombres de dominio y direcciones IP en todo el mundo.

Figura 3. El cable de par trenzado tiene 4 pares de cables, o sea 8 cables en total.

fue uno de los primeros **drawbacks** o puntos en contra del método. Bastaba con acercar un campo magnético (por ejemplo, un imán) al cable que interconectaba los equipos o con incrementar la temperatura del ambiente para que se perdiera información. Sin embargo, estos problemas ya fueron solucionados por personas muy inteligentes que tuvieron la inventiva de fabricar cables prácticamente inmunes a campos electromagnéticos externos y también a la temperatura. Un ejemplo es el cable **twisted pair** o de **par trenzado**, como muestra la **Figura 3**. Estos cables vienen con distintos tipos de protección adicional, y difieren de los cables que permanecerán a la intemperie y de aquellos que irán por una cañería.

El cable que posee una malla protectora en cada par trenzado se denomina cable **STP** (*Shielded Twisted Pair*), como muestra la **Figura 4**. El cable que no posee esa protección se llama cable **UTP** (*Unshielded Twisted Pair*), como muestra la **Figura 5**.

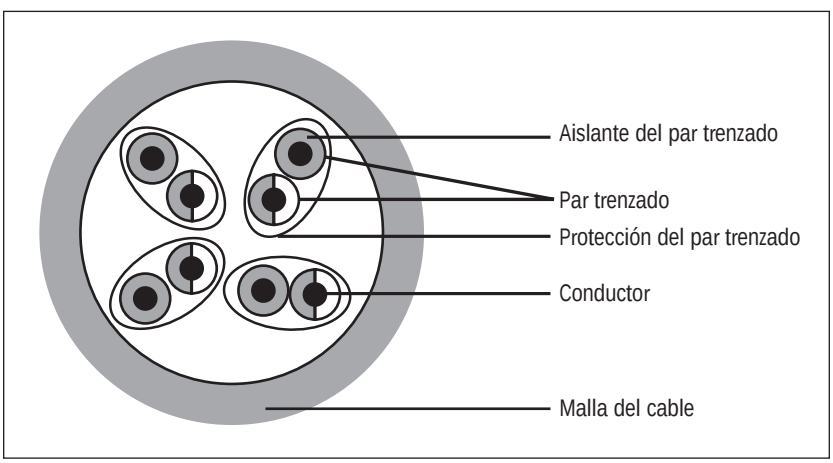

*Figura 4. Cable **STP**. Posee una malla protectora en cada par de cable trenzado.*

III CHECKSUM

Un **checksum** es un chequeo de redundancia cuyo fin es detectar errores de transferencia o almacenamiento y garantizar la integridad de la información. Podemos obtener un checksum de cualquier pieza de información, ya sea un paquete o un archivo (en este caso mediante el comando **md5sum** de GNU/Linux).

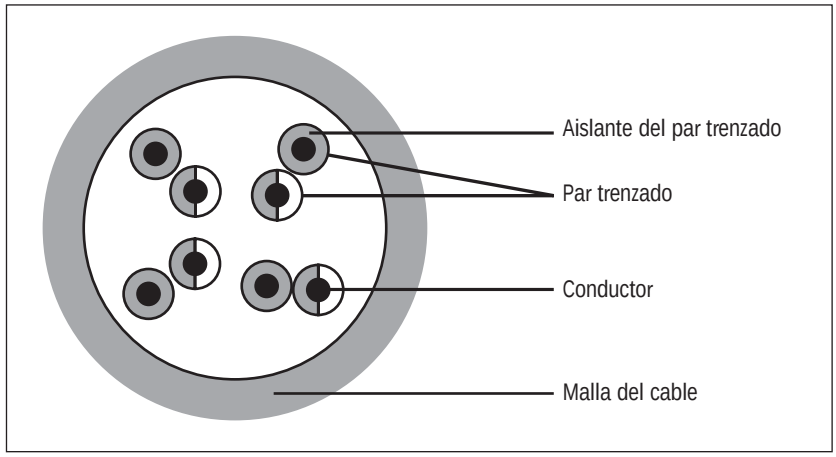

*Figura 5. El cable **UTP** carece de malla protectora en cada par de cable trenzado.*

Al tener medios físicos más aptos se minimizó la posibilidad de pérdida de paquetes. Sin embargo, el método todavía contaba con algunos problemas a ser resueltos por un **protocolo de control de transferencia**:

1) Reordenamiento de paquetes.
2) Pérdida o duplicado de paquetes.
3) Información errónea en paquetes transferidos correctamente.

El reordenamiento de paquetes a cargo del receptor fue un problema simple de resolver: numerándolos. Mientras el sistema reordena los paquetes puede notar la falta de algunos y el duplicado de otros. Si bien los paquetes faltantes sí son un problema, el único inconveniente que causan los paquetes duplicados es que malgastan ancho de banda. El protocolo de control de transferencia en el equipo receptor deberá notar la falta de un paquete y solicitarlo nuevamente al equipo transmisor. Luego, el único problema que nos queda es la transferencia de información errónea dentro de los paquetes transferidos en forma correcta. Aquí es cuando el protocolo de transferencia debe hacer uso del denominado **chequeo de redundancia cíclica** o **CRC** (*Cyclic Redundancy Check*). Antes de que el paquete se transfiera deberá contar con un campo que contenga un **checksum** o **hash**. Cuando el receptor obtenga el paquete, calculará el checksum y lo comparará con el incluido en lo que recibió. Si ambos checksum coinciden, el paquete sigue su camino. Si los checksum no coinciden, el sistema lo considerará como si fuera un paquete perdido y solicitará su retransferencia.
Otro problema surgió cuando distintas instituciones o empresas intentaban intercambiar información mediante equipos directamente conectados. Dado que no compartían una base electrónica, ni un mismo protocolo de intercambio de datos, surgió la necesidad de marcar un estándar: el **modelo de referencia OSI**. Mediante la descripción de este modelo formalizaremos algunos conceptos introducidos antes.

MODELO DE REFERENCIA OSI

El modelo de referencia OSI (*Open Systems Interconnection Basic Reference Model*) es una descripción abstracta basada en capas para las comunicaciones y el diseño de protocolos de redes de computadoras. Este modelo consta de 7 capas (como muestra la **Figura 6**), que procederemos a caracterizar a continuación. Comenzaremos desde la última capa, la 7, hasta llegar a la primera, la capa física.

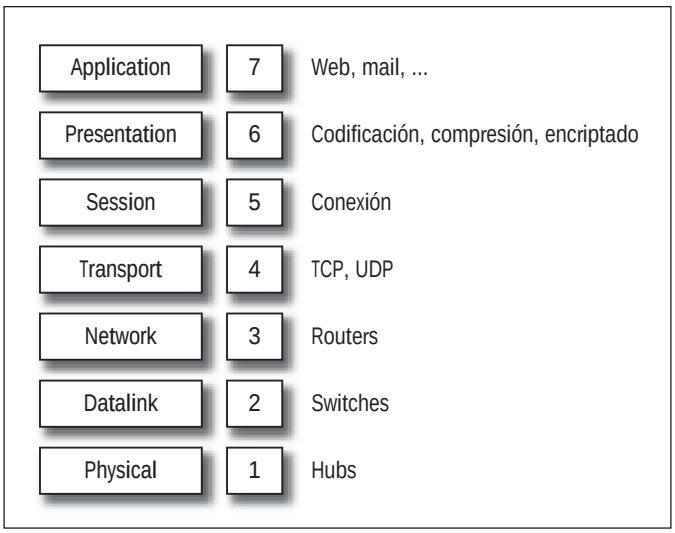

Figura 6. Representación del modelo de referencia OSI. Se pueden apreciar las 7 capas que lo componen y su orden.

Capa 7: capa de aplicación

La capa de aplicación del modelo de referencia OSI provee los medios necesarios para que el usuario pueda acceder a la información en la red mediante una aplicación. Consideramos aplicación a cualquier pieza de código que haga uso de información en la red. Podemos tomar como ejemplos de esta capa las aplicaciones que hagan uso de alguno de los siguientes protocolos:

MODELO DE REFERENCIA OSI

Si queremos profundizar más sobre el modelo de referencia OSI, es aconsejable leer la documentación provista por **Wikipedia** en el siguiente sitio: **http://es.wikipedia.org/wiki/modelo_OSI**. Nos ejemplifica cada capa de manera simple y clara. Asimismo, en este sitio figuran otros links donde podremos encontrar interesantes representaciones gráficas del modelo.

PROTOCOLO	DESCRIPCIÓN
SMTP (Simple Mail Transfer Protocol) e IMAP (Internet Message Access Protocol)	Clientes de correo: thunderbird, pico, nail.
FTP (File Transfer Protocol)	Clientes de ftp: mozilla firefox, mozilla, netscape, ftp (el comando).
HTTP (Hypertext Transfer Protocol)	Cualquier navegador http: mozilla thunderbird, netscape.
SSH (Secure Shell)	Clientes de SSH: putty, openssh.
IRC (Internet Relay Chat)	Clientes de IRC: irssi, BitchX.
SIP (Session Initiation Protocol) y H.323	Clientes de VoIP y videoconferencia: kphone.
DNS (Domain Name System)	Prácticamente cualquier aplicación que acceda a Internet.

Tabla 1. Ejemplos de protocolos de la capa 7.

Hay un ejemplo que a veces ocasiona confusión: **Telnet**. Esta palabra es un homónimo y ahí yace la confusión. Telnet es un protocolo de la capa 7 y a la vez una aplicación de la misma capa. Por lo tanto, se puede considerar como ejemplo al comando **telnet** de GNU/Linux ya que puede hacer uso del protocolo Telnet que pertenece a esta capa.

Capa 6: capa de presentación

La capa de presentación modifica la información enviada o recibida con el fin de establecer una interfaz estándar para la capa de aplicación. Esta interfaz se puede considerar como un **traductor**. Algunos de los procedimientos más importantes que se llevan acabo en esta capa son los de **encriptado**, **compresión** y **codificación**.

Cuando ingresamos a un sitio cuya URL comienza con **https://** estamos ingresando a un **sitio seguro**. Esto quiere decir que la información que ingresa y egresa está siendo **encriptada** por la capa de presentación. Cuando escuchamos alguna estación de radio por Internet, el audio está siendo **codificado** y **comprimido**. Se codifica para bajarle la calidad al audio y luego se comprime. Ambos procesos se realizan con el objetivo de disminuir el uso de ancho de banda. En ambos ejemplos, la capa de aplicación nunca se entera de que la información que ingresa está siendo encriptada, codificada o comprimida. La capa de presentación se encargó de traducirla o presentarla. De manera análoga, la capa de aplicación envía información para luego ser traducida o presentada por la capa de presentación.

Capa 5: capa de sesión

Esta capa es la encargada de controlar los diálogos y/o las conexiones (llamadas sesiones) entre computadoras. Establece, administra y finaliza las conexiones entre apliaciones locales y remotas.

La capa de sesión opera las conexiones, ya sea en **half-duplex** (un extremo transmite y el otro recibe información, por turnos) o en **full- duplex** (ambos transmiten o reciben información a la vez). Un procedimiento de ejemplo de esta capa es el llamado **cierre exitoso**, una característica de TCP.

Capa 4: capa de transporte

La capa de transporte provee una transferencia transparente de información entre los usuarios finales, y así libera a las capas superiores de cualquier preocupación y otorga un método de transferencia confiable de la información. Esta capa es la encargada de ofrecer estabilidad sobre un enlace determinado mediante el control de flujo, la fragmentación, la defragmentación y el control de errores.
La capa de transporte es la encargada de convertir mensajes en: fragmentos TCP (*Transfer Control Protocol*) o UDP (*User Datagram Protocol*). El mejor ejemplo de esta capa es TCP o **protocolo de control de transferencia**, tal como lo describimos al comienzo del capítulo.

Capa 3: capa de red

La capa 3 es una de las capas más interesantes que componen el modelo de referencia OSI, dado que aquí es donde aparece el concepto de **ruteo** y donde opera el protocolo de Internet o IP (*Internet Protocol*).
La capa de red provee los medios funcionales y procedurales de transferencia de secuencias de información de longitud variable. Garantiza realizarlo de una fuente a un destino a través de más de una red, y manteniendo la calidad y confiabilidad provista por la capa de transporte.

Como mencionamos previamente, en esta capa es donde operan los routers. Estos equipos de red ofrecen la posibilidad de crear extensas redes y mantener al mismo tiempo la calidad de transmisión. Al mismo tiempo, esto también posibilita la interconexión entre estas extensas redes, de modo que se genera así una red interconectada, como la que podemos observar en la **Figura 7**.

III CÓDECS

Códec (*Codificator-Decodificator*) es una pieza de software (o circuito electrónico) encargada de transformar un flujo de datos. Esto se puede utilizar para la transmisión, el almacenaje o el cifrado. Actualmente, los protocolos de la capa 7 utilizados para la telefonía en Internet (**H.323** o **SIP**) hacen uso de codecs de voz y video para disminuir la cantidad de información a transmitir.

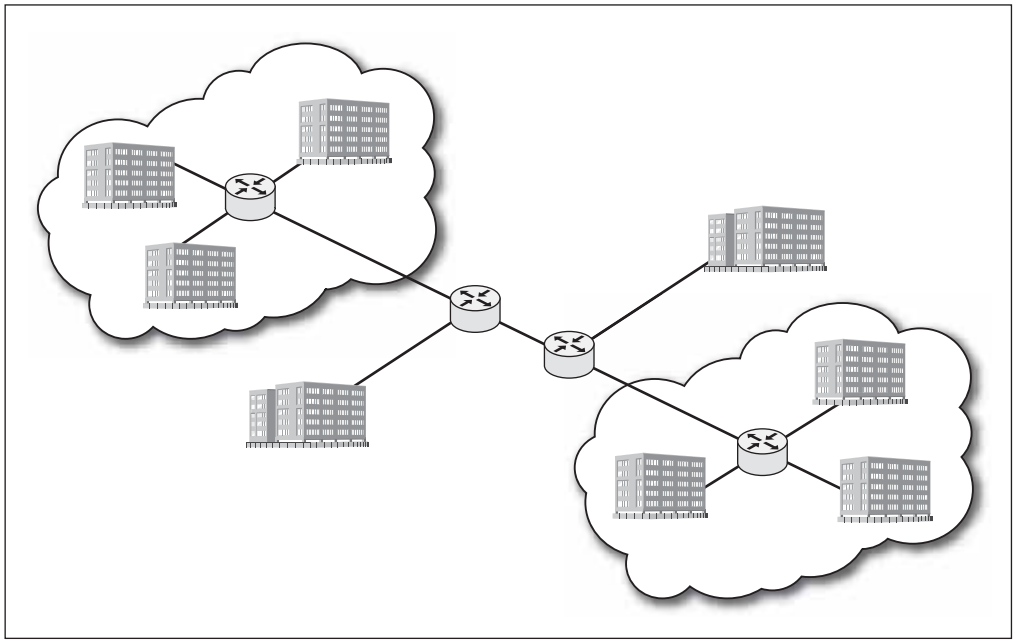

Figura 7. Diagrama de redes interconectadas mediante el uso de routers. Se puede apreciar la escalabilidad que permite el uso de estos equipos de red.

El equipamiento de red **Cisco** es uno de los más utilizados en todo el mundo. Esta empresa provee de equipamiento tanto para redes de pequeña escala así como también para redes core de grandes ISP, como vemos en las imágenes a continuación.

*Figura 8. Routers **Cisco** de la serie 2800. Estos equipos pueden comportarse como router y a su vez pueden utilizarse para la telefonía por Internet (VoIP).*

*Figura 9. Routers **Cisco** de la serie 7600. Estos equipos conforman la última tecnología en routers actuales y están orientados a redes de gran escala. Son muy caros y generalmente operan en el core o corazón de las grandes empresas de telecomunicaciones.*

Capa 2: capa de enlace de datos

Igual que la capa de red, la capa de enlace de datos provee los medios funcionales y procedurales para transferir información entre entidades de red y para detectar y posiblemente corregir errores que pudiesen ocurrir en la capa física. Es la encargada de ordenar las piezas de información de la capa física a piezas de información lógica, conocidas como **frames** o marcos.

Claros ejemplos de esta capa son los estándares **IEEE 802**, dentro de los cuales están **Ethernet**, **Token Ring**, **Wi-fi**, **Bluetooth**, etcétera. En estos casos es posible dividir esta capa en dos distintas:

1) Capa de Control de Acceso al Medio (**MAC***: Media Access Control*).
2) Capa de Enlace de Control Lógico (**LLC**: *Logical Link Control).*

TÉRMINO DEPRECADO

El término inglés *deprecated* se utiliza para caracterizar alguna aplicación o pieza de código que ha caído en desuso o se recomienda su reemplazo por alguna otra aplicación o pieza de código. Sin embargo, en español su significado es completamente distinto. Un ejemplo en GNU/Linux es el reemplazo del deprecado **OSS** (*Open Sound System*) por **ALSA** (*Advanced Linux Sound Architecture*).

En esta capa operan los **bridges** y los **switches**.

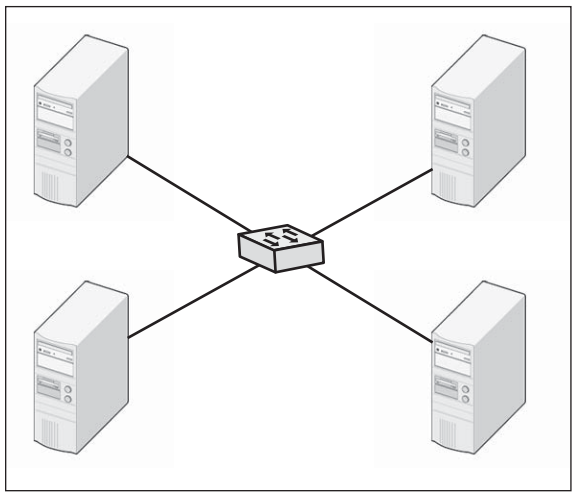

Figura 10. Diagrama de equipos interconectados mediante el uso de un switch. La cantidad de equipos a conectar depende de las bocas disponibles en este equipamiento de red.

Nuevamente **Cisco** es el principal proveedor de switches para redes de media a gran escala. La serie 1900 de **Cisco** fue muy utilizada.

*Figura 11. Switches **Cisco** de la serie Catalyst Express 500. Estos equipos cuentan con 8 a 24 bocas. También difieren en soporte de la fibra óptica.*

*Figura 12. Switches **Cisco** de la serie Catalyst 6500. Estos equipos de red se consideran multicapa dado que con el agregado de hardware especial pueden operar en las capas 2 y 3. Se utilizan para redes de gran escala.*

Capa 1: capa física

La capa física del modelo de referencia OSI es, como su nombre lo indica, la que define las características físicas de la conexión de una computadora a una red de datos. Este conjunto de especificaciones físicas describe el medio de transferencia de información. A su vez, el medio puede ser guiado (principalmente cables), o no guiado (mediante radiofrecuencia).

Otra de las características de la capa física es que se encarga de transmitir los bits de información mediante transferencia. La capa 1 describe características eléctricas como la velocidad de transmisión, los tipos de conexión, como ya mencionamos, es decir si es half-duplex o full-duplex, y la manera en la que la información se transmite. Un conocido equipo de red que pertenece a la capa 1 del modelo de referencia OSI es el deprecado hub.

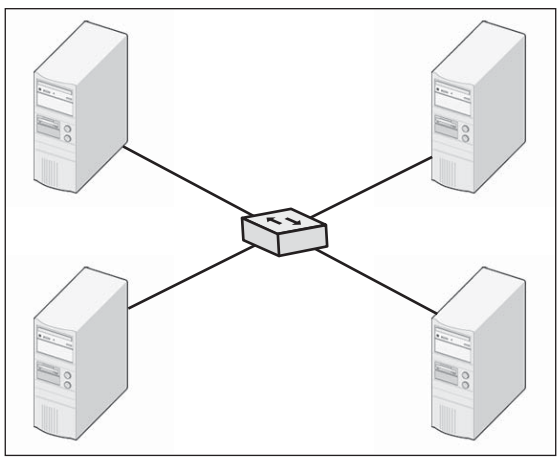

Figura 13. Diagrama de equipos interconectados mediante el uso de un hub. La similitud de su topología con la de un switch es muy notoria. Sin embargo, el hub es mucho más deficiente.

Con switches y routers ya podemos conformar grandes redes que se caracterizan por la escalabilidad que nos provee la utilización de routers.

¿SWITCH O HUB?

Se dice que un hub es un dispositivo *dumb* o tonto, ya que lo único que hace es recibir paquetes y retransmitirlos por todas sus bocas. En cambio, un switch es un hub inteligente. La diferencia está en que un switch guarda una tabla que asocia direcciones MAC con sus bocas. Esto evita la colisión de paquetes, el mal uso del ancho de banda e incrementa la seguridad.

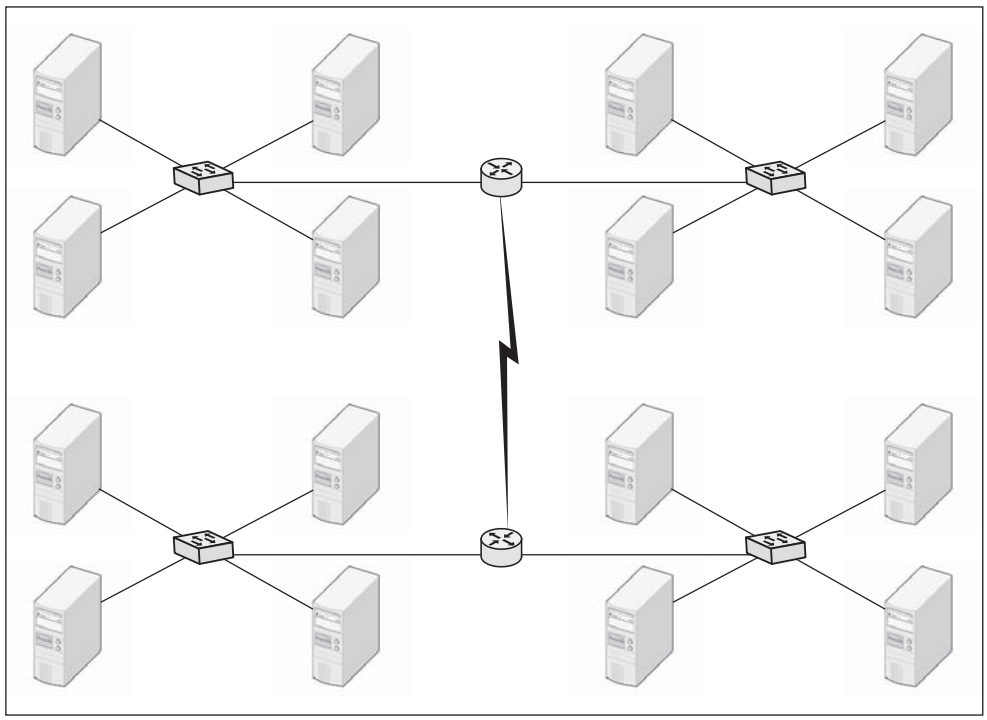

Figura 14. Diagrama de una red conformada por switches y routers. Nuevamente se aprecia la escalabilidad que nos ofrecen los routers.

PROTOCOLO DE INTERNET (IP)

Actualmente, la mayoría de las redes (para no decir todas), utiliza el **Protocolo de Internet** (**IP**) que pertenece a la capa 3 del modelo de referencia OSI. En este tipo de redes, cada equipo que interactúa se identifica de manera única mediante una dirección IP. Una dirección IP es un número binario de 32 bits (un bit es un 0 ó un 1) conformado por 4 octetos de 8 bits cada uno. A diferencia de la dirección MAC, que es un número hexadecimal fijo asignado por el fabricante a cada dispositivo de red, la dirección IP puede cambiarse y hasta asignarse de manera automática mediante el uso de determinados servicios (como **DHCP**). Un ejemplo sería la dirección IP 127.0.0.1, que en su formato binario sería 01111111.00000000.00000000.0000001. Al ser binaria, la dirección IP más pequeña disponible sería 0.0.0.1 y la más grande 255.255.255.255. Hay que comprender que las direcciones IP se utilizan en formato decimal para que los humanos puedan leerlas, pero en realidad son números binarios. Por ello mismo, la dirección 999.999.999.999 no existe, dado que el número 999 en binario es 1111100111 y son 10 bits, en lugar de 8 (octeto). Este formato de direcciones IP corresponde a la versión 4 de IP, llamado comunmente **IPv4**.

Algunos años atrás, ante el vertiginoso crecimiento de los equipos de red en Internet, surgió la necesidad de extender la cantidad de direcciones IP a un número muy grande, el cual nos permite a cada ser humano del planeta asignarle una dirección IP hasta a nuestra heladera. Esta idea fue originalmente plasmada en la versión 6 del Protocolo de Internet (**IPv6**). IPv6 es una idea ambiciosa que sin dudas tendrá éxito. IPv4 permite una cantidad de 2^{32} posibles direcciones únicas de Internet. En contraste, IPv6 ofrece 2^{128} direcciones únicas. Éstos sólo son números y no podemos imaginarnos la diferencia, así que hagamos una analogía que nos la ilustre.
IPv4 cuenta con alrededor de 4.3 miles de millones de direcciones IP, lo cual no permitiría que cada ser humano en la Tierra pueda tener una IP en su teléfono celular. Sin embargo, con IPv6, cada una de las 6 mil millones de personas en este planeta podría tener 5×10^{28} direcciones IP. Para hacer efectiva la impresión, abandonemos la notación exponencial y escribamos todos los ceros. Cada persona podría tener 50000000000000000000000000000 direcciones IP disponibles para asignarle a cualquier cosa (lavarropas, heladera, televisión, celular, etcétera).

Las direcciones IP se dividen en tres clases: **A**, **B** y **C**. También existen las clases **D** y **E**. La primera es la llamada **multicast** y la última se reserva para el desarrollo de protocolos nuevos. Una dirección IP puede dividirse en dos y obtenerse así una serie de bits asociada a la red y otra al host. Veamos algunos ejemplos:

10.0.0.1	**0**0001010.00000000.00000000.00000001
172.16.0.1	**10**101100.00010000.00000000.00000001
192.168.0.1	**110**00000.10101000.00000000.00000001

Es posible detectar con claridad un patrón binario creciente en los primeros bits de cada dirección IP: 0, 10 y 110, que en decimal sería 0, 3 y 6 respectivamente. Estos primeros bits en las direcciones IP determinan la división de las clases de redes. También podemos definir la **máscara de subred** como un código numérico que forma parte de la dirección IP. Tiene el mismo formato que una IP pero afecta un solo un segmento particular de la red. Se utiliza para dividir redes de gran tamaño en redes de menor tamaño, y esto facilita la administración. En la **Tabla 2** figuran las cuatro clases de redes con sus respectivas características.

CLASE	BITS INICIALES	IP INICIAL	IP FINAL	MÁSCARA DE SUBRED POR DEFECTO
Clase A	0	0.0.0.0	126.255.255.255	255.0.0.0 ó /8
Clase B	10	128.0.0.0	191.255.255.255	255.255.0.0 ó /16
Clase C	110	192.0.0.0	223.255.255.255	255.255.255.0 ó /24
Clase D (multicast)	1110	224.0.0.0	239.255.255.255	255.255.255.255 ó /32
Clase E (reservada)	1111	240.0.0.0	255.255.255.255	255.255.255.255 ó /32

Tabla 2. Clases de redes.

Una notación alternativa para nombrar más cómodamente una IP con su respectiva máscara de subred es contar la cantidad de bits que pertenecen a la sección de red de una dirección IP. Por ejemplo: la máscara 255.255.255.0 en su formato binario sería 11111111.11111111.1111111.0. Si contamos los bits, obtenemos que hay 24 (una máscara de subred de una dirección IP de clase C). Por lo tanto, podemos nombrar cualquier dirección IP que pertenece a esta red como IP/24.

En conclusión, si tenemos la red 200.123.155.0 con máscara de subred 255.255.255.0 (clase C) y tomamos, por ejemplo, la primera IP de esta red, ésta se puede expresar como 200.123.155.1/24, así como también 200.123.155.1/255.255.255.0.

SERVICIOS DE RED

En los capítulos siguientes hablaremos mucho sobre los servicios de red y cómo configurarlos en GNU/Linux. Sin embargo, primero necesitamos definir qué es un servicio de red, o al menos qué es un servicio.

Comencemos con un ejemplo. Cuando abrimos un explorador de Internet e ingresamos una URL (*Uniform Resource Locator*) accedemos a un servicio de red. La URL especifica, entre otras cosas, la dirección remota del equipo que contiene la información a la que queremos acceder. Esta dirección remota (bajo una red TCP/IP) se denomina dirección IP. Sería una gran complicación tener que acordarnos la dirección IP de cada uno de los sitios a los que queramos acceder. Es por ello que existe un servicio de red llamado **servicio de nombre de dominio** o **DNS** (*Domain Name System*) que nos traduce un nombre (llamado **host**) a una dirección IP. Asimismo, un servidor de DNS puede traducirnos una dirección IP a un nombre y se suelen llamar **nombres reversos**. Veremos este tema en profundidad en el **Capítulo 9**.
Además de la IP o host (de la forma **vhost.dominio.tld**), una URL también nos especifica un protocolo. Si tomamos la URL **http://www.google.com.** (nótese el punto final) podemos dividirla en 4 partes:

TÉRMINO ESCUCHAR

Cuando alguna aplicación, ya sea que corre en nuestra consola o en background como **daemon**, utiliza conexiones de red con rol de servidor puede abrir algún puerto TCP o UDP y esperar algún cliente. Este estado se denomina **LISTEN**, el término inglés que significa **escuchar**.

1	http://	Protocolo del servicio, en este caso HTTP
2	www.	VirtualHost o vhost
3	google.	Dominio
4	com.	TLD (Top Level Domain)

Tabla 3. Partes de una URL.

El protocolo del servicio, que pertenece a la capa 7 o de **Aplicación** del modelo de referencia OSI, establece las pautas sobre la comunicación entre la aplicación cliente y servidor. Hay una característica muy importante y es que la **IANA** (*Internet Assigned Numbers Authority*) le asignó a cada protocolo de comunicación un puerto de manera predeterminada. En la **Tabla 3** figuran algunos de los protocolos más conocidos y sus puertos TCP predeterminados.

SERVICIO	PUERTO TCP PREDETERMINADO
HTTP	80
SMTP	25
POP3	110
IMAP	143
HTTPS	443
TELNET	23
FTP	21
SSH	22
DNS	53
IRC	¿6667 ó 194?

Tabla 4: Protocolos y puertos TCP.

Para obtener un listado completo de todos los protocolos y sus puertos TCP y UDP predeterminados basta hacer lo siguiente desde cualquier consola:

```
inwx@shellfire ~ $ cat /etc/services
```

La selección del número de puerto para cada servicio no tiene importancia. Esto es así porque cada servicio, excepto quizá SMTP, puede correr en cualquier puerto TCP. Existen 1111111111111111 puertos TCP y UDP, o sea 16 bits, lo cual en decimal quiere decir que existen 65535 puertos TCP.
Para representar un puerto distinto al predeterminado en una URL se utiliza la siguiente notación: **protocolo://vhost.dominio.tld:puerto**. Por ejemplo, si **Google** corriera su sitio en el puerto 8080 podríamos acceder a éste mediante la URL **http://www.google.com:8080**.

Una característica muy importante es que todos los servicios conocidos se encuentran por debajo del puerto 1024 inclusive, o sea el número en binario del puerto no debe ser mayor que 10000000000. De hecho, en cualquier sistema operativo basado en Unix, un usuario común no puede correr una aplicación que quiera escuchar en alguno de estos puertos dado que son los denominados **puertos reservados**. Una anécdota interesante de este hecho es lo que ocurrió con el protocolo **IRC** (*Internet Relay Chat*). Si buscamos este protocolo en **/etc/services** veremos que aparecen dos puertos: el 194 y el 6667 (hay un tercero que corresponde a IRC con SSL, pero dejemos ese caso de lado). Como muchos sabrán, el puerto más famoso pero no oficial del servicio IRC es el 6667, entonces ¿por qué aparece el 194? Este puerto fue el oficialmente asignado al servicio IRC por la IANA pero debido a que correr este servicio en el puerto 194 requiere de permisos de root (al estar por debajo del 1024, que es bastante inseguro), surgió el puerto 6667 y otros cercanos a éste como una alternativa para correr un **IRCd** (daemon de IRC) con permisos de usuario común.

Otra cuestión a tener en cuenta es que el cliente se conecta a un puerto remoto desde un puerto local. El puerto que abre el cliente para establecer la conexión al equipo remoto siempre se encuentra por sobre el puerto 1024 y se elige de manera aleatoria dependiendo de los puertos disponibles en el momento de establecer dicha conexión.

MODELO DE REFERENCIA OSI EN GNU/LINUX

Veamos el proceso de conexión de un navegador web (**Firefox**, por ejemplo) al buscador web más utilizado del momento: **Google**. Vamos a hacer uso del comando **tcpdump** para capturar todos los paquetes que ingresen o egresen de nuestro sistema con destino a **google.com**. Este comando lo utilizaremos bastante para poder comprender las comunicaciones entre los equipos de red.
Primero vamos a abrir una terminal, ejecutar el comando y aguardar a que éste comience a capturar los paquetes. Luego, abrimos el explorador e ingresamos la

III DAEMONS

En GNU/Linux o en cualquier otro derivado de Unix, existen aplicaciones llamadas **daemons** que corren en background en vez de estar en contacto directo con el usuario. Generalmente, las aplicaciones daemon tienen la letra **d** al final de su nombre. Ejemplos: **syslogd**, **crond**, **smbd**, **nfsd**, **ircd**.

dirección **google.com**, de manera tal que el programa ya esté capturando los paquetes una vez que comienza la conexión al buscador. Si hicimos todo correctamente vamos a obtener algo como esto:

```
shellfire ~ # tcpdump host google.com
tcpdump: verbose output suppresed, use -v or -vv for full protocol decode

listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes

22:36:39.042036 IP shellfire.unirc.com.ar.58392 > py-in-f99.google.
     com.http: S 2881448842:2881448842(0) win 5840 <mss 1460,sackOK,times
     tamp 104511641 0,nop,wscale 2>

22:36:39.054978 IP py-in-f99.google.com.http > shellfire.unirc.
     com.ar.58392: S 3196303845:3196303845(0) ack 2881448843 win 8712 <mss
     1460,nop,nop,sackOK,nop,wscale 0,nop,nop,timestamp 1900080 104511641>

22:36:39.055030 IP shellfire.unirc.com.ar.58392 > py-in-f99.google.
     com.http: . ack 1 win 1460 <nop,nop,timestamp 104511644 1900080>

22:36:39.055242 IP shellfire.unirc.com.ar.58392 > py-in-f99.google.
     com.http: P 1:690(689) ack 1 win 1460 <nop,nop,timestamp 104511645 1900080>

22:36:39.097732 IP py-in-f99.google.com.http > shellfire.unirc.com.
     ar.58392: . ack 690 win 15695 <nop,nop,timestamp 1900085 104511645>

22:36:39.457231 IP py-in-f99.google.com.http > shellfire.unirc.com.
     ar.58392: P 405:425(20) ack 690 win 16384 <nop,nop,timestamp 1900120
     104511645>

22:36:39.457278 IP shellfire.unirc.com.ar.58392 > py-in-f99.google.
```

COMANDOS Y RESPUESTAS

Para comprender mejor la ejecución de comandos que realizaremos a lo largo de este libro, identificaremos el texto con fondo gris y sin negrita como **comandos ingresados por teclado**, mientras que el texto con fondo gris y con negrita se referirá a las **respuestas** que obtendremos **del sistema**.

```
     com.http: . ack 1 win 1460 <nop,nop,timestamp 104511745 1900085,nop,
     nop,sack 1 {405:425}>

22:36:39.461264 IP py-in-f99.google.com.http > shellfire.unirc.com.
     ar.58392: P 1:405(404) ack 690 win 16384 <nop,nop,timestamp 1900120
     104511645>

22:36:39.461278 IP shellfire.unirc.com.ar.58392 > py-in-f99.google.
com.http: . ack 425 win 1728 <nop,nop,timestamp 104511746 1900120>
```

Aquí podemos ver cada paquete de información transmitido con todos sus datos, en particular host y puerto cliente; y host y puerto servidor.

En este ejemplo, **shellfire.unirc.com.ar.58392** y **py-in-f99.google.com.http** son una composición de la forma **vhost.dominio.tld.puerto**. Podemos hacer dos observaciones interesantes: la primera es que el puerto cliente es mayor que el puerto 1024 y la segunda es la traducción de hostnames y puertos (en el string del servidor aparece **http** en vez de 80).

Cuando una aplicación intenta traducir una IP a un nombre o un puerto a un protocolo, ésta hace uso de la función **gethostbyaddr()** o de **getprotobynumber()**, respectivamente. Ambas funciones pertenecen a la librería **netdb.h**. Estas funciones hacen uso del archivo de sistema **/etc/nsswitch.conf** para saber dónde buscar la información de traducción. De manera predeterminada, el archivo **nsswitch.conf** tiene la siguiente configuración (nótese que cualquier usuario puede leer este archivo):

```
inwx@shellfire ~ $ cat /etc/nsswitch.conf
# /etc/nsswitch.conf:
# $Header: /var/cvsroot/gentoo-x86/sys-libs/glibc/files/nsswitch.conf,v 1.1 2005
/05/17 00:52:41 vapier Exp $
```

III CLIENTES SSH

Cuando nos encontremos en algún sistema operativo que no sea GNU/Linux o derivado de Unix, vamos a necesitar otro cliente ssh que el provisto por **openssh**. Existen dos clientes muy utilizados para **Microsoft Windows**: **Secure CRT** y **Putty**. El primero es pago y el segundo es gratuito.

```
passwd:      compat
shadow:      compat
group:       compat

# passwd:    db files nis
# shadow:    db files nis

# group:     db files nis

hosts:       files dns
networks:    files dns

services:    db files
protocols:   db files
rpc:         db files
ethers:      db files

netmasks:    files
netgroup:    files
bootparams:  files

automount:   files
aliases:     files
inwx@shellfire ~ $
```

Haremos referencia a este archivo muchas veces, dado que nos va a ayudar a entender cómo nuestro sistema obtiene determinada información.

Aquí podemos ver que cuando el sistema necesite traducir hosts de la forma **vhost.domain.tld** a una dirección IP, primero buscará una entrada que coincida en los archivos de sistema (**/etc/hosts**). Si esta búsqueda no resulta exitosa, el sistema pro-

EL VHOST WWW

El subdominio o vhost **www** se utiliza al principo de un sitio web. Muchas organizaciones decidieron agregar, por convención, el servicio ofrecido como subdominio. Sin embargo, el uso de este subdominio no es requerido por ningún estándar técnico. De hecho, existen muchos sitios web a los que podemos acceder sin anteponer el **www**.

cederá a buscar mediante el método **dns**: leerá el archivo **/etc/resolv.conf** y consultará a algún servidor DNS definido en éste. En caso de que el sistema quiera traducir protocolos (de puerto a protocolo o viceversa), primero buscará mediante **db** y si la búsqueda no es exitosa, intentará buscar los archivos del sistema, en este caso **/etc/protocols**. En resumen, en una IP podemos correr tantas aplicaciones como queramos, que se pueden escuchar en cualquiera de los 65535 puertos TCP, si tenemos los privilegios necesarios. Cada aplicación, en caso de ser cliente, hará uso de un puerto más grande que el 1024.

En los capítulos siguientes vamos a explicar el funcionamiento de los servicios de red más importantes y cómo instalarlos, configurarlos y optimizarlos.

RESUMEN

A lo largo de este capítulo hicimos una introducción teórica al mundo de networking. Describimos el modelo de referencia OSI en su totalidad, pasamos por sus 7 capas y ejemplificamos cada una de ellas. Luego describimos exhaustivamente el protocolo IP con sus distintas clases de redes, concluimos con una detallada introducción a los servicios de red y describimos puertos locales, puertos remotos y métodos de capturación de paquetes mediante el uso de la consola de GNU/Linux.

ACTIVIDADES

TEST DE AUTOEVALUACIÓN

1 Mencione tres protocolos que pertenezcan a la capa 7 del modelo de referencia OSI.

2 ¿A qué capa del modelo de referencia OSI pertenece el protocolo IP?

3 ¿Cuántos puertos TCP hay disponibles para uso como cliente?

4 ¿A qué clase de red pertenece la IP 15.2.1.2?

5 ¿Cuál es la IP más grande de la red 172.16.20.0/16?

6 ¿Cuál es la máscara de subred en notación reducida de una red de clase B?

7 ¿Qué herramienta de GNU/Linux se puede utilizar para monitorear el intercambio de información con un equipo remoto?

8 ¿Qué archivo de sistema guarda la información para la traducción de servicios?

9 ¿Cuál es el archivo de sistema cuya configuración rige la traducción de servicios y protocolos?

10 ¿Cuál es el archivo de sistema que contienen los servidores DNS a los cuales consultar en el caso de que la búsqueda en los archivos del sistema falle?

PRÁCTICAS

Repita el proceso de capturado de paquetes para una conexión a un servidor de correo saliente y confirme la traducción del puerto.

Procedimiento:

1) Considere el servidor de correo entrante (POP3) mail.yahoo.com.
2) Abra una consola y ejecute tcpdump host mail.yahoo.com
3) En otra consola ejecute telnet mail.yahoo.com 110

Podrá ver la información obtenida en la consola que corre el tcpdump. Verifique la traducción de 110 a pop3.

Aclaración: Es muy probable que no pueda conectarse a este servidor dado que Yahoo estableció filtros para las direcciones IP dinámicas. En ese caso, pruebe con el servidor de correo saliente de su proveedor de Internet.