REDES: Tema 1. Introducción

(Grado en Ingeniería Informática)

Francisco Argüello Pedreira y José Carlos Cabaleiro Domínguez
Departamento de Electrónica y Computación
Universidad de Santiago de Compostela

2 de septiembre de 2011

Índice general

Capítulo 1

Introducción

1.1. Introducción a las prácticas

En esta asignatura haremos prácticas de TCP/IP en lenguaje C basadas en el uso de sockets. A continuación veremos una pequeña introducción de como usar direcciones IP, nombres de hosts, puertos y sockets en lenguaje C.

1.1.1. Direcciones IP

Todo ordenador se identifica en Internet con una dirección IP, en la actualidad de tipo IPv4. Las direcciones IPv4 están compuestas por 4 bytes separados por puntos. En lenguaje C se pueden utilizar de dos formas:

- *Formato textual*. Es decir, como cadenas de caracteres ASCII, por ejemplo, `char *ip="193.110.128.200";` (16 caracteres).

- *Formato binario*. Es decir, como 4 bytes, por ejemplo[1], `unsigned char ip[]={193,110,128,200};` o, lo que es equivalente, como un entero de 32 bits sin signo, por ejemplo, `uint32_t ip=3363860161u` ($193 + 110 \times 2^8 + 128 \times 2^{16} + 200 \times 2^{24}$).

  Es usual que este entero sin signo se encapsule en una estructura de tipo `struct in_addr` que contiene una única componente `uint32_t`.

`struct in_addr`	
`uint32_t s_addr`	dirección IPv4

  Por ejemplo, `struct in_addr ip; ip.s_addr=3363860161u;`

En la librería C existen algunas funciones para realizar conversiones entre los dos formatos. Algunas de ellas las veremos en las prácticas.

[1] Suponiendo que el procesador es *little-endian*, como el x86.

1.1.2. Nombres de host

La forma alternativa de identificar un ordenador en Internet es mediante un nombre de host. En lenguaje C se utiliza una cadena de caracteres para almacenarlo, por ejemplo, `char *name="www.elmundo.es".`

Es usual utilizar la estructura `struct hostent` para almacenar los nombres y direcciones IP de un host. Esta estructura contiene las siguientes componentes:

`struct hostent`	
`char *h_name`	el nombre "oficial" del host.
`char **h_aliases`	lista de alias del host, acabada con un puntero nulo.
`int h_addrtype`	el tipo de dirección, `AF_INET` (o `AF_INET6` para IPv6).
`int h_length`	la longitud en bytes de cada dirección.
`char **h_addr_list`	lista de IPs del host, acabada con un puntero nulo.
`char *h_addr`	sinónimo para `h_addr_list[0]`, la primera IP del host.

Hay que tener en cuenta:

- Un nombre de host puede tener asignadas varias direcciones IP. Esto lo vemos, por ejemplo, con el comando `host www.elmundo.es`, que nos devuelve como respuestas 193.110.128.209 y 193.110.128.200.

  La razón de ello es por fiabilidad y balanceo de la carga. Los ordenadores con el mismo nombre suelen ser servidores espejo, es decir, que tienen los mismos contenidos. Cuando se solicita un acceso a un ordenador de este tipo, en primer lugar se hace la traducción del nombre a la dirección IP. En cada traducción se devuelve una de las direcciones IP del conjunto, mientras que en sucesivas traducciones se recorre el conjunto de forma cíclica.

- Un ordenador puede tener varios nombres (el nombre oficial o canónico y los alias). Uno de los motivos es la facilidad de administración: desde Internet varios ordenadores pueden identificarse con el mismo nombre si tienen los mismos contenidos, pero desde la red interna es más práctico identificarlos con nombres distintos. Por ejemplo, si tecleamos el comando `host 193.110.128.209` obtenemos `wwwb2.elmundo.es` (antes habíamos usado `www.elmundo.es`).

  Otra razón desde el punto de vista práctico es la posibilidad de alojar varios sitios web distintos en el mismo ordenador.

1.1.3. Puertos

Una vez que la transmisión llega al ordenador destino hay que entregar los datos a la aplicación (servicio) correcta. Cada servicio se identifica mediante un número de puerto. Es un entero de 16 bits, por ejemplo, `int puerto=8000` (también podría usarse `unsigned short int` o `uint16_t`).

Es usual almacenar la información de los servicios en una estructura del tipo `struct servent`, que tiene las siguientes componentes:

`struct servent`	
`char *s_name`	el nombre "oficial" del servicio.
`char **s_aliases`	lista de alias del servicio, acabada con un puntero nulo.
`int s_port`	el número de puerto (en el orden *big-endian*).
`char *s_proto`	el nombre del protocolo que usa este servicio.

Por ejemplo, el servicio *telnet* no tiene alias, usa el puerto 23 y el protocolo *tcp*. (En las arquitecturas *little-endian* como x86 hay que reordenar los dos bytes del puerto).

1.1.4. Sockets

El socket es la interfaz entre una aplicación y la capa de transporte. La aplicación solicita una transmisión de datos y la capa de transporte la realiza. Haciendo una analogía con el correo postal, las aplicaciones equivalen a las personas y la capa de transporte al sistema postal. La interfaz entre ambos son los buzones.

De la misma forma que los usuarios del correo postal no entregan ni reciben las cartas de los carteros, sino que usan los buzones, las aplicaciones interactúan con la capa de transporte a través de los sockets. Para la aplicación, el envío de un dato equivale a la escritura en un socket, mientras que la recepción equivale a una lectura.

Para que dos procesos situados en diferentes ordenadores puedan comunicarse entre sí, de cada proceso necesitamos saber:

- La dirección IP del ordenador en el cual se ejecuta el proceso.
- El puerto que tiene asignado el proceso dentro del ordenador.

Por tanto, los sockets se contruyen a partir de direcciones IP y puertos. En la analogía con el correo postal, las direcciones IP equivalen a la dirección de un edificio (ciudad, calle, número), mientras que los puertos indican la posición del buzón (piso y letra). Los sockets equivalen a los buzones.

En lenguaje C, los sockets se identifican por el número de socket, que es una variable de tipo `int` devuelta en el momento de la creación del socket. Las direcciones que usa el socket (para IPv4) se encapsulan en una estructura de tipo `struct sockaddr_in` que contiene las siguientes componentes:

`struct sockaddr_in`	
`sa_family_t sin_family`	el tipo de socket, `AF_INET` para IPv4
`struct in_addr sin_addr`	la dirección IPv4
`uint16_t sin_port`	el número de puerto

1.1.5. Servidores y clientes

Un *servidor* es aquel que proporciona un servicio, mientras que un *cliente* es aquel que solicita un servicio. Por ejemplo, un servidor web es aquel que sirve las páginas web y un cliente web el que las solicita.

Se denomina servidor tanto al programa que efectúa el servicio (por ejemplo el programa servidor web) como al ordenador en el que se ejecuta (ordenador servidor). Lo mismo para el cliente.

Hay servidores web, de correo, de ficheros, de nombres, etc.

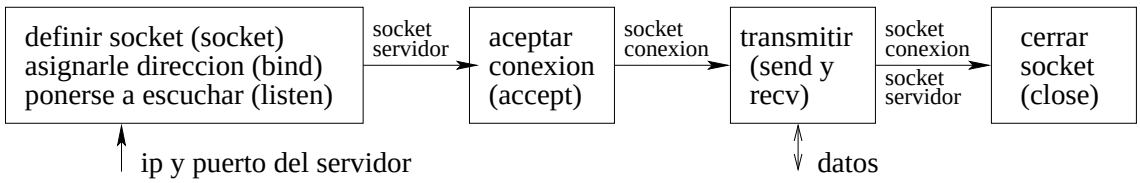
1.1.6. Sockets orientados a conexión

En los sockets orientados a conexión, antes de transmitir datos se solicita una conexión. El programa que solicita la conexión es el cliente, mientras que el que la acepta es el servidor. El uso de sockets orientados a conexión en lenguaje C implica 4 pasos básicos (cada uno de los cuales se implementa con una o más funciones):

1. Crear los sockets.
2. Solicitar una conexión (si es un cliente) o aceptarla (si es un servidor).
3. Enviar y recibir datos.
4. Cerrar los sockets.

Veamos más en detalle los lados de servidor y cliente.

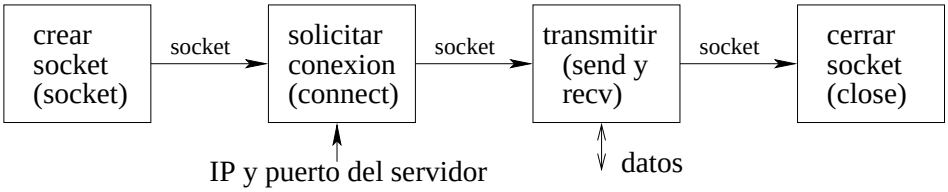
Lado del servidor:



1. Creamos un socket (con la función `socket`), le asignamos la *dirección IP y puerto del servidor* (con la función `bind`) y lo ponemos a la escucha, es decir, lo convertimos en *socket servidor* (con la función `listen`).
2. Cuando nos llega una solicitud de conexión por parte de un cliente la aceptamos (función `accept`). La función `accept` devuelve un segundo socket (el *socket de conexión*), que será el socket utilizado para la transmisión de datos.
3. El envío y recepción de datos se realizará escribiendo y leyendo en el *socket de conexión* (funciones `send` y `recv`).

4. Una vez acabada la transmisión debemos cerrar tanto el *socket de conexión* como el *socket servidor* (función `close`).

Lado del cliente:



1. Creamos el socket (función `socket`).

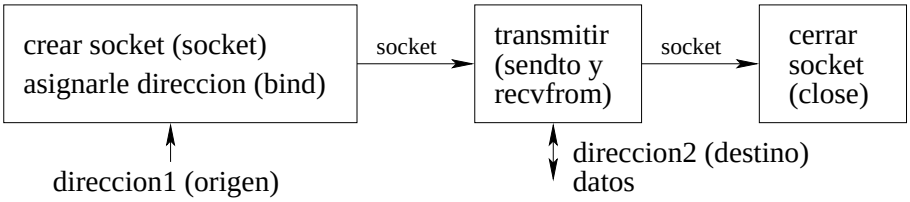
2. Solicitamos la conexión a un servidor (función `connect`). Para ello deberemos facilitar a la función la *dirección IP y puerto del servidor*.

3. El envío y recepción de datos la realizamos escribiendo y leyendo en el socket (funciones `send` y `recv`).

4. Una vez acabada la transmisión debemos cerrar el socket (función `close`).

1.1.7. Sockets sin conexión

Con este tipo de sockets los datos se pueden enviar en cualquier momento, sin necesidad de establecer una conexión previa. Desde el punto de vista de los sockets, ambos extremos funcionan de forma completamente simétrica, sin distinción entre servidor y cliente. Los pasos son los siguientes:



1. Creamos un socket (con la función `socket`) y le asignamos la dirección IP y puerto origen (*dirección1*) (con la función `bind`).

2. El envío y recepción de datos se realizará escribiendo y leyendo en el socket (funciones `sendto` y `recvfrom`). Para el envío debemos indicar la dirección IP y el puerto destino (*dirección2*), mientras que en la recepción los obtenemos del socket.

3. Una vez acabada la transmisión debemos cerrar el socket (función `close`).

1.2. Elementos de Internet

1.2.1. Punto de vista hardware

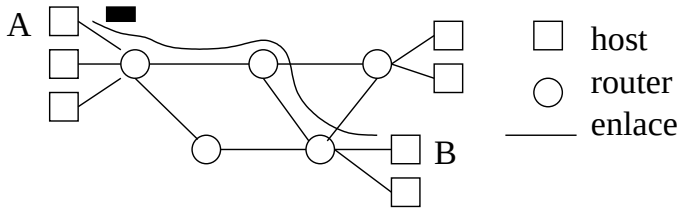
Desde el punto de vista hardware podemos distinguir los siguientes elementos:

1. **Hosts** (estaciones o hospedadores). Son los sistemas terminales, es decir, el origen o el destino de las transmisiones. Incluyen los ordenadores, pero también todo dispositivo que pueda conectarse a Internet como agendas, neveras, etc.

   Una impresora conectada por USB no es host, pero sí lo es si la impresora se conecta mediante una tarjeta de red.

2. **Enlaces**. Son los medios físicos por los que se realizan las transmisiones. Pueden ser cables o transmisores de radio.

3. **Routers** (rutadores o encaminadores). Son los dispositivos que interconectan los enlaces. Los datos llegan al router por uno de los enlaces y este los reenvía a través de otro. Así router a router se va generando la ruta (o camino) que siguen los datos desde el origen al destino.



1.2.2. Punto de vista software

Desde el punto de vista software, Internet funciona porque todos sus componentes ejecutan los mismos protocolos, es decir, que siguen las mismas reglas y el mismo formato para las comunicaciones (los estándares de Internet). Estos protocolos están descritos en unos documentos denominados RFC (solicitudes de comentarios). Se distinguen dos tipos de protocolos:

1. Protocolos básicos. Necesarios para que funcione Internet. Fundamentalmente TCP/IP (y algunos más como UDP, ICMP, etc).

2. Protocolos de aplicación. Necesarios para que funcionen ciertas aplicaciones. Como HTTP (para el web), SMTP (para el correo), etc.

1.2.3. Punto de vista comercial

Desde el punto de vista comercial, Internet está operado por numerosas compañías. En esto es similar a la red telefónica. En la red telefónica hay compañías que trabajan en el lazo local (telefónica, R), otras tienen líneas de larga distancia (telefónica, jazztel), otras operan en telefonía móvil (movistar, orange, vodafone), otras revenden servicios (tele2), etc. Una llamada de teléfono típicamente atraviesa la red de varios operadores (por ejemplo, lazo local + línea de larga distancia + telefonía móvil). La compañía con la que tenemos contrato es la que nos pasa la factura y luego una parte del dinero que recauda se emplea en pagar al resto de los operadores.

En el caso de Internet, podemos distinguir los siguientes *proveedores de servicios de Internet* (ISP):

1. Proveedores de baja escala (residenciales). Son los que proporcionan acceso a Internet a los usuarios. El usuario hace contrato con ellos y luego parte del dinero se distribuye entre los demás proveedores.

2. Proveedores de alta escala (nacionales o internacionales). Proporcionan las troncales que interconectan a los proveedores de baja escala y también las líneas de larga distancia.

1.3. Servicio orientado a conexión y sin conexión

TCP/IP, y en particular Internet, proporciona a las aplicaciones dos tipos de servicios: *servicio orientado a conexión* y *servicio sin conexión*. Cada aplicación, a través de la interfaz proporcionada por los sockets, usa el más adecuado para las funciones que realiza.

1.3.1. Servicio orientado a conexión

En Internet el servicio orientado a conexión es TCP (Protocolo de Control de la Transmisión). En general, un servicio orientado a conexión comprende tres fases. En la primera fase el cliente solicita al servidor una conexión. Esta es una fase de acuerdo en la que pueden fijarse ciertos parámetros y prepara a ambos extremos para la transmisión de datos. A continuación pueden transmitirse los datos. Por último, en la fase de desconexión ambos extremos se ponen de acuerdo en terminar la transmisión y liberan los recursos que han estado empleando.

Se denomina *servicio orientado a conexión* y no simplemente *servicio con conexión*, puesto que la "conexión" que se establece es software y no hardware. Sólo afecta a los hosts y consiste en establecer valores de variables y reserva de memoria. Los conmutadores de los routers no se ven afectados.

En Internet, el servicio orientado a conexión va unido a las siguientes características (otros servicios orientados a conexión no tienen porque tener estas características):

1. *Segmentación.* TCP va recogiendo los datos que la aplicación va escribiendo en el socket. Con ellos TCP va formando los paquetes de datos para transmitir. Estos paquetes tienen un tamaño máximo especificado. Si la aplicación genera mensajes más grandes, TCP los enviará en varios paquetes separados, esto es, segmentará el mensaje.

2. *Transferencia fiable.* La fiabilidad se consigue haciendo que el emisor retransmita los paquetes de datos que llegan con errores o no llegan. Para que el emisor sepa cuando esto ocurre, el receptor ha de enviar confirmaciones. Si el emisor no recibe alguna confirmación, supone que el paquete de datos ha llegado mal o se ha perdido y en consecuencia lo retransmite.

3. *Control de flujo.* Permite que el receptor controle la tasa de envío del emisor. El receptor puede ser lento en el procesamiento de los datos o disponer de poca memoria. TCP provee un mecanismo para que el receptor le diga al emisor la tasa de datos que está dispuesto a aceptar. Esta tasa de datos se escribe en un campo dentro de las confirmaciones.

4. *Control de congestión.* Permite que la tasa de envío del emisor se ajuste a las *capacidades de la red* para que no sature enlaces y routers. Un host considera que empieza a haber congestión cuando muchos paquetes de datos no llegan al destino. La respuesta del host es disminuir la tasa de envío.

1.3.2. Servicio sin conexión

En Internet el servicio sin conexión es UDP (Protocolo de Datagrama de Usuario). En un servicio sin conexión, el emisor envía paquetes de datos en el momento que quiere, sin necesidad de acuerdo con el receptor. No hay fase de conexión ni confirmaciones, por lo que el emisor no sabe si el paquete de datos llegó al destino. Tampoco hay control de flujo ni control de congestión. Al no tener estas características, la transmisión es más rápida, aunque menos fiable.

TCP se usa cuando la aplicación necesita fiabilidad, por ejemplo, telnet, ftp, smtp, http, etc. UDP se suele usar cuando es más importante la rapidez y no importa que algunos paquetes de datos se pierdan, por ejemplo, en telefonía IP y en videoconferencia.

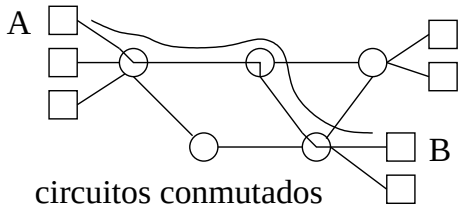
1.4. Tipos de redes

Desde el punto de vista hardware, podemos distinguir las siguientes redes:

conmutación	circuitos (sin multiplexado, FDM, TDM)
	paquetes (datagramas, circuitos virtuales)
difusión	

1.4.1. Redes de conmutación de circuitos (CC)

En una red de conmutación de circuitos, antes de la transmisión de datos hay una fase de conexión en la que se reservan los recursos hardware que se van a utilizar durante la fase de transmisión de datos. Estos recursos hardware quedan reservados y no pueden ser utilizados por otra transmisión. Además, durante la fase de conexión se establece la ruta que después van a seguir todos los datos. Los recursos hardware que se reservan son típicamente ranuras de tiempo o de frecuencia. La transmisión finaliza con una fase de desconexión en la cual se liberan todos los recursos.



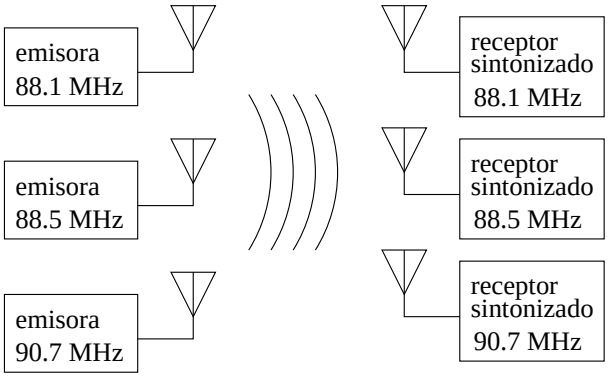
Podemos distinguir los siguientes tipos de redes de CC:

CC	sin multiplexado	
	con multiplexado	división en frecuencia (FDM)
		división en tiempo (TDM)

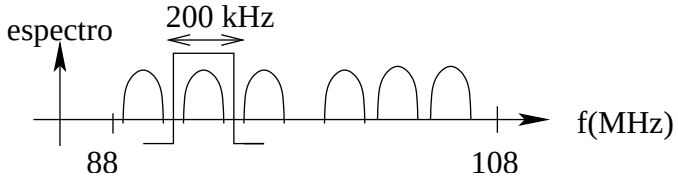
En las redes de CC sin multiplexado, por cada enlace sólo se puede realizar una transmisión de cada vez. En las redes de CC con multiplexado, la capacidad del enlace se reparte entre varias transmisiones dividiendo en varias ranuras el tiempo o la frecuencia. Eso sí, cada ranura queda reservada a una determinada transmisión y las demás no pueden usarla.

Multiplexado por división en frecuencia (FDM)

Por un mismo enlace se puede enviar varias transmisiones si modulamos a una frecuencia distinta cada una de ellas. Un ejemplo son las transmisiones por radio de las emisoras de radio y TV. En el caso de la radio FM:



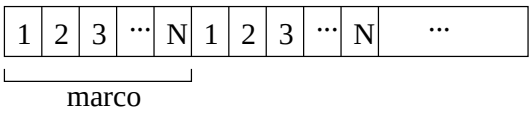
En los receptores, las distintas emisoras se pueden separar mediante filtros sintonizados a la frecuencia de la correspondiente emisora.



Otro ejemplo de división en frecuencia es la compartición de un cable coaxial entre varias transmisiones, usada por ejemplo en la TV por cable.

Multiplexado por división en tiempo (TDM)

Aquí se asigna periódicamente una ranura de tiempo a cada transmisión. Primero transmite 1, luego 2, etc, hasta N. Una vez completado un ciclo (marco) se comienza de nuevo con la transmisión 1.



Como ejemplo podemos poner el sistema telefónico.

Las redes de conmutación de circuitos (tanto FDM como TDM) son derrochadoras porque los recursos están reservados para una transmisión aunque ésta al final no los use. Por ejemplo, en las transmisiones telefónicas el usuario sólo transmite (habla) menos del 50 % del tiempo (en concreto 3/8 de tiempo total contando todos los períodos de silencio); el resto del tiempo escucha.

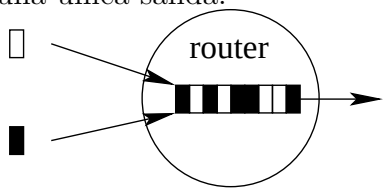
1.4.2. Redes de conmutación de paquetes (CP)

En este tipo de red no se reservan recursos para cada transmisión, sino que todos se comparten y asignan bajo demanda. Eso sí, pueden ser tanto sin conexión (datagramas) como con conexión (circuitos virtuales).

Estas redes trabajan con paquetes, que son bloques de datos de una longitud determinada. Si el mensaje a transmitir es más grande, se segmenta en varios paquetes. Cada paquete contiene, además de datos, una cabecera con información

de control para que el paquete pueda llegar al destino. También hay paquetes que llevan sólo información de control, como los paquetes de confirmaciones.

Los routers de Internet funcionan como conmutadores de paquetes, usualmente operando mediante almacenamiento y reenvío. Cuando un paquete llega al router, se procesa y se almacena en la cola de salida correspondiente hasta que le toque el turno para pasar al enlace (retransmisión al siguiente router). En el caso de que la cola esté llena, el paquete se descarta, pues el router no puede operar a más del 100 %. Por ejemplo, en un router en que llegan paquetes por dos entradas con destino una única salida:



El retardo producido por el router tiene, por tanto, dos componentes:

- *Retardo de almacenamiento y reenvío.* Es el tiempo necesario para escribir (o leer) el paquete en la memoria, que es proporcional al tamaño del paquete.
- *Retardo de espera en la cola.* Que es proporcional a lo cargada que esté la red.

Comparacion de CC con CP

Pregunta. Sea una red compartida por varios usuarios cada uno de los cuales transmite un 10 % del tiempo.

(a) Si la red funciona en modo de CC con 10 canales (ranuras de tiempo o frecuencias), ¿cuántos usuarios podrá haber simultáneamente?

(b) ¿Y si la red funciona en modo de CP?

Respuesta. En la red de CC hay 10 canales, por lo tanto sólo puede haber 10 usuarios simultáneamente. Se supone que la red tiene recursos para que cada uno de estos 10 usuarios transmita el 100 % del tiempo, aunque al final sólo lo haga el 10 % del mismo.

Si la misma red (con la misma capacidad de routers y enlaces) se pone a trabajar en modo de CP, teóricamente podría soportar $10 \times 10 = 100$ usuarios simultáneamente. Sin embargo, esto es en teoría, porque en la práctica los retardos serían muy altos. En la práctica (con cálculos de la teoría de colas) podría haber 35 usuarios con retardos prácticamente despreciables.

Segmentación

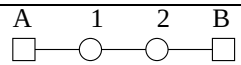
Las razones para la segmentación son varias:

- El tiempo de transmisión es más corto (ver a continuación).

- Una transmisión no satura la red con mensajes enormes, sino que da la oportunidad a otras transmisiones para que intercalen sus paquetes.

- En caso de errores en un paquete, sólo hay que transmitir el paquete con errores y no el mensaje completo.

Veamos mediante un ejemplo que la segmentación ayuda a reducir los retardos. Sea la siguiente transmisión:

Mensaje de 10^7 bits
2 routers intermedios 
Enlaces de 2 Mbps (millones de bits por segundo, para velocidades $M = 10^6$)
Obtener el tiempo de transmisión en dos casos: (a) sin segmentar (b) segmentado en 2500 paquetes de 4000 bits cada uno

Vamos tener en cuenta sólo el tiempo de almacenamiento y reenvío. Sabiendo que tenemos enlaces de 2 Mbps, los tiempos necesarios para almacenar y reenvíar el mensaje completo y uno de los paquetes, son respectivamente,

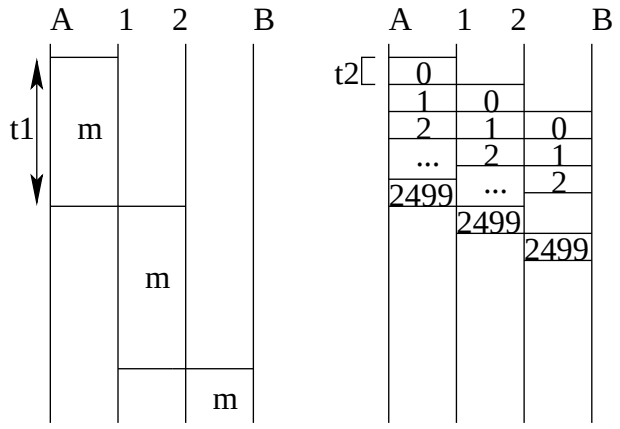
$$t_1 = \frac{10^7 \text{ bits}}{2 \text{ Mbps}} = 5 \text{ sg} \tag{1.1}$$

$$t_2 = \frac{4000 \text{ bits}}{2 \text{ Mbps}} = 2 \text{ msg} \tag{1.2}$$

Teniendo en cuenta el esquema de la siguiente figura, los retardos de transmisión en los dos casos, son:

$$t_{mensaje} = 3t_1 = 15 \text{ sg} \tag{1.3}$$

$$t_{paquetes} = 2500t_2 + 2t_2 = 5 + 0,004 = 5,004 \text{ sg} \tag{1.4}$$



Tipos de redes de conmutación de paquetes

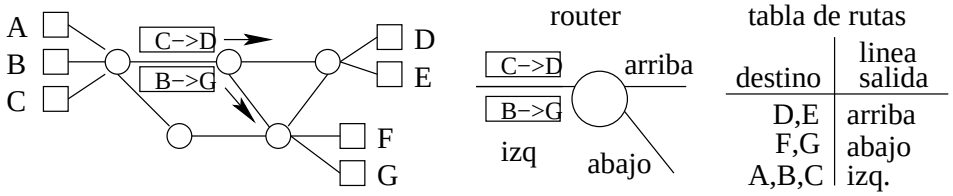
Podemos distinguir los siguientes tipos de redes de CP:

CP	datagramas
	circuitos virtuales

Las redes de datagramas son *sin conexión* y el encaminamiento de los paquetes se hace en función de la dirección del destino, mientras que las redes de circuitos virtuales son *con conexión* y el encaminamiento se hace en función del número de circuito virtual.

Redes de datagramas

Cada paquete tiene una cabecera que incluye entre otras cosas la dirección IP del destino. Cuando el paquete llega a un router, este le examina la cabecera y dependiendo del destino lo coloca en la cola de salida más apropiada. Esta operación realizada por el router se denomina encaminamiento. Para seleccionar la salida más adecuada en cada caso, el router utiliza la información almacenada en su *tabla de rutas*.



Los routers, por otra parte, no mantienen información de estado, es decir, no recuerdan los encaminamientos previos. Si una secuencia de paquetes de la misma transmisión llega al mismo router, cada uno de ellos se encamina independientemente. Incluso puede ocurrir que los paquetes sigan rutas diferentes, por ejemplo, si hay varias rutas válidas hacia el mismo destino y el router selecciona aleatoriamente cada una de ellas.

Podemos tomar como símil el correo postal. Cada vez que una carta llega a una oficina postal se encamina sólo en función de la dirección del destinatario.

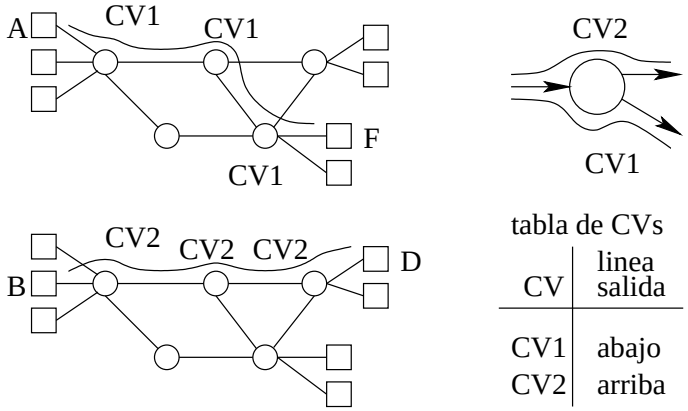
Redes de circuitos virtuales

En una red de circuitos virtuales (CV), antes de transmitir datos, el host origen realiza una solicitud de conexión a la red. La red responde planificando una ruta al destino y asignando a dicha ruta un número de circuito virtual. A cada uno de los paquetes se le escribe en la cabecera ese número, que es el que van a utilizar los routers para realizar el encaminamiento. Cuando termina la transmisión se realiza una desconexión en la que se borra la información asociada al circuito virtual.

Todos los routers por los que pasa el circuito virtual son informados en el momento de la conexión. Los routers han de mantener información de estado.

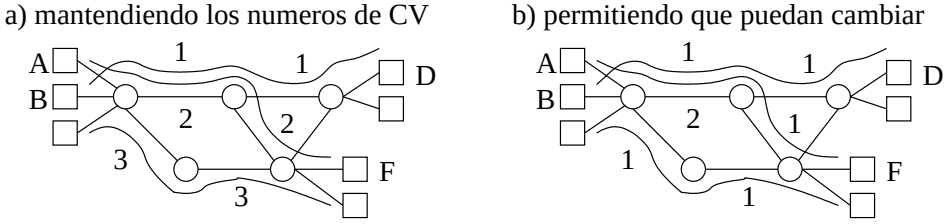
En concreto han de mantener una tabla con los circuitos virtuales que tienen abiertos. Esta tabla se conoce con el nombre de *tabla de circuitos virtuales*.

Podemos tomar como símil el de una empresa de mensajería que planifica todos los envíos. En el momento en que un cliente solicita un envío, la empresa le asigna un número, selecciona un ruta y avisa a todas las oficinas por las que va a pasar dicho envío.



Hay que dejar claro que sólo se planifica la ruta, no se reservan recursos. Los recursos (enlaces, memoria, etc) se siguen compartiendo entre todas las transmisiones. Estamos por tanto ante una red de conmutación de paquetes y no de conmutación de circuitos.

Veamos con más detalle los números de circuitos virtuales. Hasta ahora hemos supuesto que un circuito virtual mantiene el mismo número durante toda la ruta. Esto no es así, sino que a un mismo circuito virtual se le pueden asignar distintos números en los distintos enlaces. Son dos las razones:

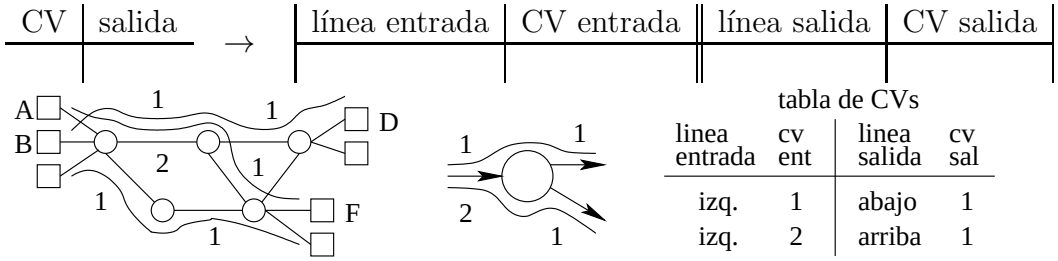


1. Así los números se pueden reusar en los distintos enlaces. Números en enlaces distintos son independientes y pueden repetirse. Esto implica que se pueden usar números más pequeños (campo de menos bits) con el consiguiente ahorro de espacio en los paquetes.

2. Ahora no se necesita una tabla global con los números de todos los CVs de la red pues no hay que garantizar que estos números sean únicos globlamente. Basta con garantizar que los números no se repitan en el mismo enlace, lo que puede hacerse con tablas locales.

Se introduce, sin embargo, una complicación: hay que almacenar en las tablas los cambios que se hacen en los números de CVs. Estas tablas las almacenan los

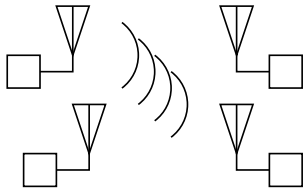
routers, que son también los encargados de cambiar los números de CVs en las cabeceras de los paquetes. La tabla de CVs de los routers pasa a tener la siguiente forma:



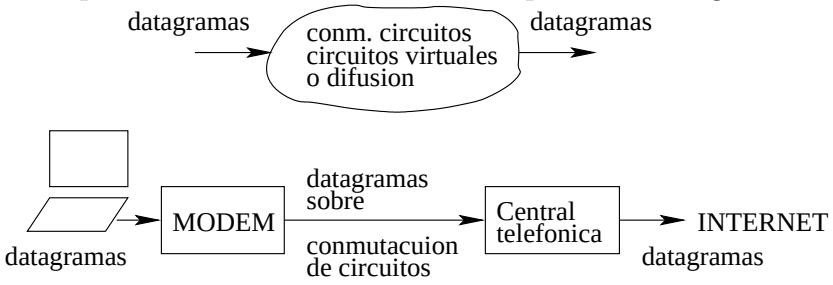
En general podemos decir que las redes de circuitos virtuales son ventajosas (más fiables y rápidas) en redes pequeñas y homogéneas, puesto que planifican con más cuidado las transmisiones. En redes grandes y heterogéneas como Internet es casi imposible usar circuitos virtuales por lo que usan datagramas (que son más flexibles, tolerantes a fallos y necesitan menos coordinación y recursos).

1.4.3. Redes de difusión

En una red de difusión todos los hosts reciben las transmisiones, pero sólo el host destinatario procesa la transmisión. Un ejemplo típico de red de difusión son las redes inalámbricas, Ethernet 802.11b/g o Wi-Fi.



Internet es una red de datagramas, pues así lo especifica IP (Protocolo Inter-red) de la arquitectura TCP/IP. Sin embargo, algunas partes de Internet pueden funcionar internamente (en la capa de enlace, que veremos al final del tema) con otro tipo de tecnología: conmutación de circuitos, circuitos virtuales o difusión. En estos casos conviven las dos tecnologías (por ejemplo, conmutación de circuitos por debajo y datagramas por encima), pero en el momento en que la transmisión abandona esta parte de la red sólo sobrevive la parte de datagrama.



1.5. Acceso a Internet

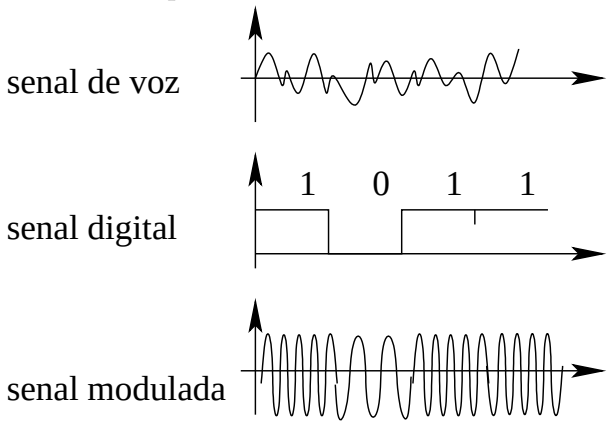
El acceso residencial a Internet se puede realizar principalmente de 3 formas:

1. Módem telefónico (56 kbps).
2. ADSL (línea digital asimétrica del suscriptor).
3. Cable HFC (híbrido fibra-coaxial).

Mientras que el acceso empresarial se suele realizar a través de redes locales.

1.5.1. Módem telefónico

El módem telefónico usa la línea telefónica como si la transmisión a Internet fuese una llamada de voz normal. Primero llama al número telefónico del ISP y una vez establecida la conexión, convierte la señal digital en una señal analógica modulada con características parecidas a la voz. Esto se denomina modulación.



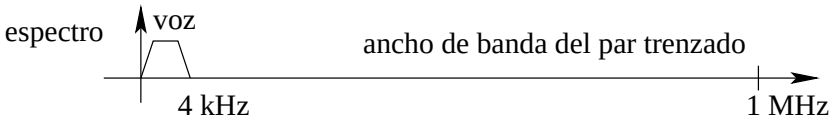
En el receptor se realiza la operación inversa (demodulación). De ahí el nombre de modem (modulador-demodulador).

El problema que tiene esto es que las señales de voz que utiliza el sistema telefónico tienen un ancho de banda de frecuencias muy estrecho, sólo hasta 4 kHz. En este ancho de banda tan pequeño sólo puede conseguirse, usando modulación, una velocidad de transmisión máxima de 56 kbps, y eso siempre que la línea telefónica esté en buenas condiciones.

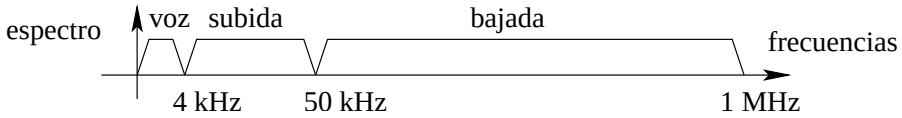
1.5.2. ADSL

ADSL significa línea digital asimétrica del suscriptor (es decir, del abonado mediante un contrato a la compañía telefónica). ADSL trata de aprovechar todo el ancho de banda en frecuencias del cable que conecta nuestra casa con la central telefónica. Hay que tener en cuenta que en las llamadas de voz normal la capacidad de este cable está desaprovechada puesto que sólo se utilizan los primeros 4 kHz

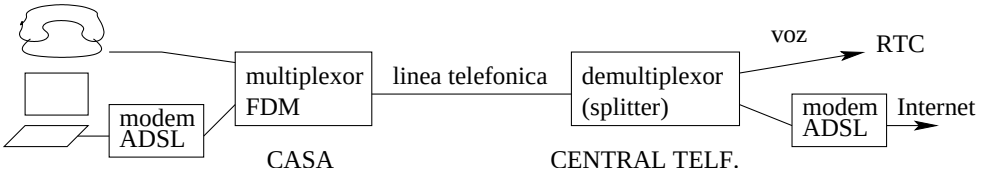
del ancho de banda aproximado de 1 MHz que tiene el cable telefónico (par trenzado).



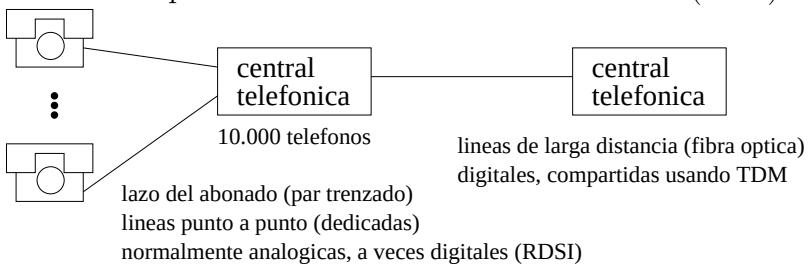
La idea del ADSL es el aprovechamiento de la totalidad del ancho de banda del cable para la transmisión en Internet. Así, usa multiplexión por división en frecuencia, dividiendo el ancho de banda del cable en tres canales independientes: el primero para la transmisión de voz telefónica normal, el segundo para el envío a Internet (canal de subida) y el tercero para recepción (el canal de bajada), siendo este último el que tiene la velocidad mayor. Las velocidades de subida y bajada son diferentes puesto que un usuario residencial usualmente descarga más de lo que sube, y de aquí viene el nombre de línea asimétrica.



Al tener un ancho de banda de frecuencias tan grande, de casi 1 MHz, se puede transmitir a alta velocidad, hasta 10 Mbps (ADSL2+ alcanza 24 Mbps), si bien los ISPs pueden ofrecer comercialmente velocidades menores. Además, como las señales de voz y datos van por canales separados, permite usar el teléfono y la conexión a Internet simultánemante. El esquema del ADSL es el siguiente:

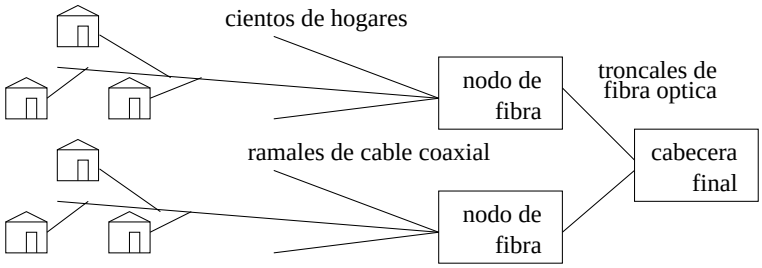


En el caso el caso de ADSL, en la central telefónica hay que modificar la circuitería para que pueda gestionar frecuencias mayores que 4 kHz y separar las distintas bandas, lo que no ocurría con el módem telefónico. Para completar esta sección veamos un esquema de la red telefónica conmutada (RTC).



1.5.3. Cable HFC (híbrido fibra y coaxial)

Un esquema de una red de acceso por cable (son posibles otras topologías) es:



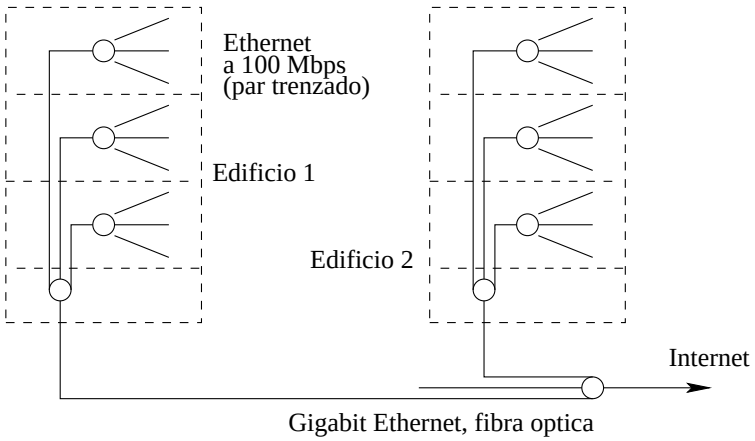
La cabecera final es equivalente a una central telefónica y centraliza todas las transmisiones de los abonados que desde aquí pasan a Internet a través de un router. A continuación, las líneas troncales de fibra óptica conectan la cabecera con los nodos de fibra. Y por último, desde los nodos de fibra parten ramales de cable coaxial para dar servicios de TV, teléfono e Internet a cientos de hogares.

Las troncales están hechas de fibra óptica puesto que estas líneas necesitan la mayor capacidad posible. En el resto de las líneas se usa cable coaxial puesto que este tiene la ventaja de que su instalación es más fácil y barata. Todos los cables son compartidos (multiplexados) en TDM o FDM y pueden transmitir cientos de canales de TV, teléfono e Internet.

1.5.4. Acceso empresarial

El acceso de una empresa a Internet normalmente se realiza mediante una red local (LAN, local area network) usualmente de tipo Ethernet, conectada mediante un router y a un ISP con un enlace dedicado (aparte de la red telefónica).

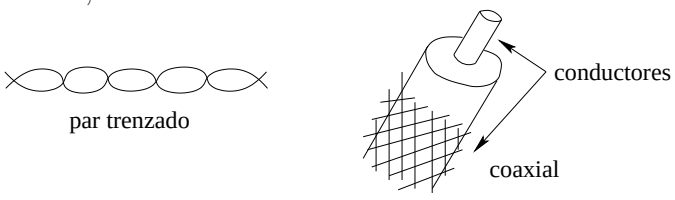
La Universidad de Santiago, por ejemplo, dispone de múltiples redes locales de tipo Ethernet de 100 Mbps unidas a una red troncal (backbone) Gigabit Ethernet que interconecta los edificios de la Universidad. El acceso a Internet se realiza mediante un router que enlaza con la red de RedIris. RedIris es el organismo que en España gestiona la Internet pública, esto es, la parte de Internet que interconecta los organismos del Estado, universidades, escuelas, hospitales, ministerios, etc.



1.6. Medios de transmisión (cables)

Conforme se va propagando la señal por un cable se va atenuando, esto es, va perdiendo energía, debido principalmente a dos factores: radiación y resistencia. En el primer caso parte de la señal se convierte en radiación y escapa a través del espacio; en el segundo caso parte de la señal se convierte en calor. Por consiguiente, para distancias largas hay que poner repetidores cada cierta longitud de cable. La atenuación también limita la velocidad a la que puede transmitirse.

Vamos a ver tres tipos de cable, que podemos ordenar de menor a mayor capacidad de la siguiente forma: par trenzado, cable coaxial y fibra óptica. La relación de capacidad de un tipo de cable al siguiente es un factor bastante grande, en números redondos, de unas centeneras de veces.



1.6.1. Par trenzado (TP)

El par trenzado (TP) es el cable de tipo telefónico. Consta de dos hilos de cobre trenzados, es decir, dispuestos en espiral y enrollados entre sí. El enrollado disminuye algo las pérdidas de energía por radiación y las interferencias que puede causar a otros pares. Dentro de un conducto de plástico pueden disponerse cientos de estos pares. En este tipo de cable podemos distinguir las siguientes versiones:

UTP (sin apantallar, unshielded)	categorías 1, 2, 3, 4, 5, 5e, 6
STP (apantallado, shielded)	

En el apantallado (STP), que es el de mayor calidad, el par está recubierto de una fina película de metal, que tiene como objeto disminuir algo las pérdidas de energía por radiación y las interferencias. El par sin apantallar (UTP) carece de este recubrimiento. Dependiendo de la calidad del cable UTP se distinguen varias categorías, de las cuales las categorías 1 y 2 son las de peor calidad y se usan en telefonía, mientras que la categoría 3 se usa en las redes locales de 10 Mbps y la categoría 5 en las de 100 Mbps.

1.6.2. Cable coaxial

El cable coaxial está constituido por dos conductores concéntricos: uno central sólido y otro exterior formado por una malla de hilos finos que recubre al primero. Entre ambos hay plástico. La ventaja de que los conductores estén dispuestos concéntricamente es que la radiación queda atrapada entre ambos y no escapa al espacio, evitando todas las pérdidas de energía por radiación.

Hay dos calidades de cable coaxial, el de 50 ohmios que se utiliza para transmitir señales digitales en banda base (es decir sin modular) en algunas redes locales. El de 75 ohmios es de mejor calidad, se utiliza en banda ancha (es decir con modulación) en las redes de cable CFH y permite cientos de canales de TV, teléfono e Internet.

1.6.3. Fibra óptica

La fibra óptica, en vez de transmitir señales eléctricas, transmite luz, por lo que evita las pérdidas por radiación. Las fibras están hechas de vídrio o plástico transparente y ultrapuro y tienen tan baja atenuación que pueden utilizarse hasta distancias de 100 km sin repetidores. Son tan finas como un cabello, por lo que dentro de un conducto pueden alojarse miles de ellas.

El problema que tienen las fibras ópticas es que son más difíciles de instalar y que el precio de los dispositivos ópticos (transmisores, receptores, multiplexos, etc.) es más elevado que los correspondientes eléctricos.

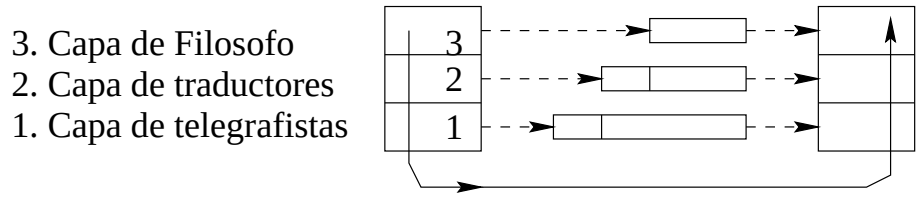
La capacidad de transmisión de la fibra óptica es la mayor de cualquier tipo de cable. Recordamos que la relación de capacidad de un tipo de cable al siguiente es de unas centenares de veces en este orden: *par trenzado* << *cable coaxial* << *fibra óptica*.

1.7. Arquitectura en capas

La arquitectura en capas se utiliza para facilitar el dise no de los protocolos de comunicación. Consiste simplemente en dividir la comunicación en tareas independientes y poner cada una de ellas en una capa. Las capas superiores utilizan los servicios de las inferiores. La arquitectura en capas permite conectar computadores y sistemas muy diferentes con la única condición de que respeten las especificaciones de cada capa.

1.7.1. Una analogía

La idea de la comunicación multicapa puede explicarse mediante la ayuda de la siguiente analogía. Supóngase por ejemplo, que hay dos filósofos, uno en Kenia y otro en Indonesia que desean comunicarse. Estos filósofos constituyen la capa 3 de la comunicación. Cada uno de ellos tiene que hacer uso de un traductor. Supongamos que disponemos de traductores del keniata al inglés y del inglés al indonesio. Estos traductores constituyen la capa 2. Cada uno de los traductores, a su vez, contacta con un telegrafista para que le transmita el mensaje. Los telegrafistas constituyen la capa 1.



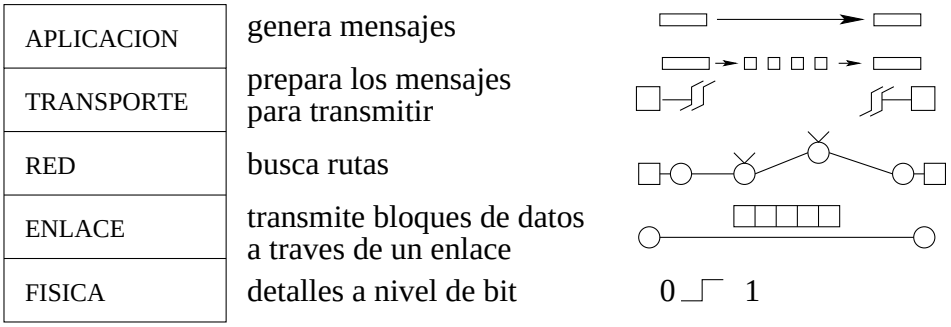
Las ventajas de este modelo son las siguientes:

1. Cada capa es independiente de los demás. Los traductores pueden conmutar del inglés al francés si ambos están de acuerdo y ello no afectará a las capas 1 y 3. También puede cambiarse los telegrafistas por telefonistas sin que se modifiquen las capas 2 y 3.

2. Aunque las transmisiones son en vertical, los protocolos pueden dise narse como si fuesen horizontales. La capa de traductores, por ejemplo puede ignorar el contenido de los datos que recibe, mientras que en una *cabecera* pone las instrucciones que ha de seguir el traductor del destino. De igual forma, la capa de telegrafistas ignora el contenido de lo que transmite la capa superior, mientras que puede a nadir instrucciones para el telegrafista del destino. El proceso de dise no se simplifica.

1.7.2. La arquitectura TCP/IP

Se ilustra esquemáticamente en la siguiente figura.

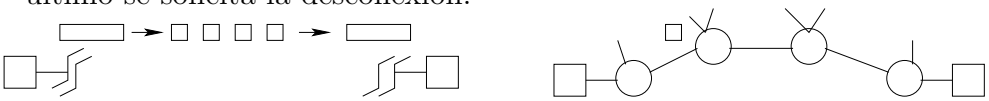


- En la capa **capa de aplicación** se localizan los procesos que se comunican entre sí mediante mensajes. Supóngase que queremos comunicar dos de esos procesos, por ejemplo, el proceso de grabación en fichero con el proceso que proporciona la hora, un servidor con un cliente de impresión, etc. Para que pueda realizarse la comunicación tenemos que acordar un protocolo: formato para los mensajes (campos obligatorios, opciones posibles) y unas reglas (quién inicia la comunicación, si se necesitará autentificación, qué hacer en caso de errores, etc).

- Supongamos que los procesos a comunicar están situados en distintos computadores y que existe algún tipo de conexión entre estos. La **capa de trans-**

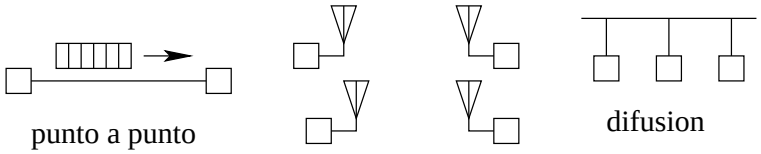
porte prepara los mensajes para que puedan transmitirse fuera del computador.

La siguiente figura (mitad izquierda) muestra el caso particular del protocolo de transporte TCP. Antes de la transmisión de datos, el origen solicita una conexión al destino. A continuación, en el origen, TCP va recogiendo los datos que el proceso escribe en el socket y forma con ellos paquetes adecuados para transmitir, mientras que en el destino TCP comprueba que todos los paquetes llegan sin errores y reensambla el mensaje original. Por último se solicita la desconexión.



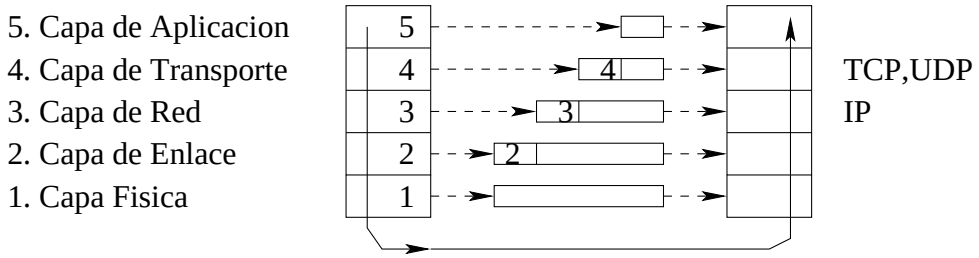
- Si los computadores no están directamente conectados, sino que forman parte de una red, se introduce la **capa de red**. Esta capa es la encargada de buscar las rutas y de llevar a los paquetes desde el host origen hasta el host destino. Se muestra en forma de esquema en la parte derecha de la figura anterior.

- La **capa de enlace** se encarga de los detalles de bajo nivel de la transmisión de cada bloque de datos (paquete) de un extremo a otro de un enlace. En el enlace se puede utilizar cualquier tipo de tecnología de red: conmutación de circuitos, datagramas, circuitos virtuales, difusión, etc. El enlace puede ser de cable o de ondas de radio y puede ser punto a punto (un emisor y un receptor) o de difusión (múltiples transmisores). Se encarga de comenzar la transmisión cuando el enlace está libre, de ir almacenado los datos en la memoria cuando van llegando, de detectar colisiones en enlaces de difusión, etc.



- La **capa física** trabaja a nivel de bits: convierte los bits 0 y 1 en pulsos, es decir, en señales eléctricas. Conforma el pulso dándole la correspondiente forma, tensión (amplitud) y duración. También define las características mecánicas y físicas del medio de transmisión (calidad del cable, número de hilos, su longitud, etc).

1.7.3. Algunos detalles



Cabeceras. Aparte de los datos que se transmiten, cada una de las capas debe transmitir instrucciones. Estas se a naden a los mensajes como cabeceras. Cada cabecera contiene las intrucciones necesarias para que el origen y destino de cada una de las capas se comuniquen. En el origen, se ponen las cabeceras (instrucciones) y en el destino se leen y se interpretan.

Protocolos especificados en TCP/IP. De las cinco capas que constituyen el modelo TCP/IP, sólo es obligatorio usar determinados protocolos en las capas de transporte y red. Los protocolos especificados en la capa de transporte son TCP y UDP (también se especifican algunos otros, como ICMP) y en la capa de red, IP. Con sólo respetar los protocolos de estas dos capas podemos transmitir sin problemas en Internet.

La capa superior, la de aplicación, no tiene protocolos obligatorios. Sin embargo, hay unas aplicaciones que son prácticamente estándar, por ejemplo *telnet* (para sesiones remotas), *ftp* (para transferencia de ficheros), *http* (para transferencia de páginas web) y *smtp* (para transferencias de correos).

Las capas inferiores, enlace y física afectan sólo a la red local (a nuestros enlaces), por lo que podemos usar los que queramos. Una vez que la transmisión sale de nuestra red local, se usarán las capas de enlace y física de las conexiones exteriores.

El esquema de los paquetes TCP/IP se muestra en la siguiente figura: