Linux: http://linkat.xtec.net/portal/index.php

XARXES

d' Ordinadors-2

Eduard Romo
Professor d' Informàtica

La familia de protocolos TCP/IP

ÍNDICE

La familia de protocolos TCP/IP

1. Protocolo TCP-IP

1.1 Historia

En 1969 la agencia ARPA (Advanced Research Projects Agency) del Departamento de Defensa de los Estados Unidos inició un proyecto de interconexión de ordenadores mediante redes telefónicas. Al ser un proyecto desarrollado por militares en plena guerra fría, un principio básico de diseño era que la red debía poder resistir la destrucción de parte de su infraestructura (por ejemplo a causa de un ataque nuclear), de forma que dos nodos cualesquiera pudieran seguir comunicados siempre que hubiera alguna ruta que los uniera. Esto se consiguió en 1972 creando una red de conmutación de paquetes denominada ARPAnet, la primera de este tipo que operó en el mundo. La conmutación de paquetes unida al uso de topologías malladas mediante múltiples líneas punto a punto dio como resultado una red altamente fiable y robusta.

La ARPAnet fue creciendo paulatinamente, y pronto se hicieron experimentos utilizando otros medios de transmisión de datos, en particular enlaces por radio y vía satélite; los protocolos existentes tuvieron problemas para ínter operar con estas redes, por lo que se diseñó un nuevo conjunto o pila de protocolos, y con ellos una arquitectura. Este nuevo conjunto se denominó TCP/IP (Transmission Control Protocol/Internet Protocol) nombre que provenía de los dos protocolos más importantes que componían la pila; la nueva arquitectura se llamó sencillamente *modelo TCP/IP*, los nuevos protocolos fueron especificados por vez primera por Cerf y Kahn en un artículo publicado en 1974. A la nueva red, que se creó como consecuencia de la fusión de ARPAnet con las redes basadas en otras tecnologías de transmisión, se la denominó Internet.

1.2 Breve concepto de TCP/IP

Ya que dentro de un sistema TCP/IP los datos transmitidos se dividen en pequeños paquetes de unos 1500 bytes, éstos resaltan una serie de características.

- La tarea de IP es llevar los datos a granel (los paquetes) de un sitio a otro. Las computadoras que encuentran las vías para llevar los datos de una red a otra (denominadas enrutadores) utilizan el protocolo IP para trasladar los datos. En resumen *IP mueve los paquetes de datos a granel, mientras que el protocolo TCP se encarga del flujo y asegura que los datos estén correctos.*
- Las líneas de comunicación se pueden compartir entre varios usuarios. Cualquier tipo de paquete puede transmitirse al mismo tiempo, y se ordenará y combinará cuando llegue a su destino.
- Los datos no tienen que enviarse directamente entre dos computadoras. Cada paquete pasa de computadora en computadora hasta llegar a su destino. Éste, claro está, es el secreto de cómo se pueden enviar datos y mensajes entre dos computadoras aunque no estén conectadas directamente entre sí. Lo que realmente sorprende es que sólo se necesitan algunos segundos para enviar un archivo de buen tamaño de una máquina a otra, aunque estén separadas por miles de kilómetros y pese a que los datos tienen que pasar por múltiples computadoras. Una de las razones de la rapidez es que, **cuando algo anda mal**, **sólo es necesario volver a transmitir un paquete, no todo el mensaje**.
- Los paquetes no necesitan seguir la misma trayectoria. La red puede llevar cada paquete de un lugar a otro y usar la conexión más idónea que esté disponible en ese instante. No todos los paquetes de los mensajes

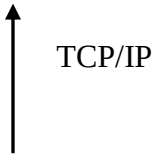
tienen que viajar, necesariamente, por la misma ruta, ni necesariamente tienen que llegar todos al mismo tiempo.

- La flexibilidad del sistema lo hace muy confiable. Si un enlace se pierde, el sistema usa otro. Cuando usted envía un mensaje, el TCP divide los datos en paquetes, ordena éstos en secuencia, agrega cierta información para control de errores y después los lanza hacia fuera, y los distribuye. En el otro extremo, el TCP recibe los paquetes, verifica si hay errores y los vuelve a combinar para convertirlos en los datos originales. De haber error en algún punto, el protocolo TCP destino envía un mensaje solicitando que se vuelvan a enviar determinados paquetes.

1.3 Las capas del modelo TCP/IP

En el modelo TCP/IP se pueden distinguir **cuatro capas**: en subrayado el nombre de las capas y dentro de los paréntesis y en mayúsculas el nombre de los protocolos utilizados en dicha capa. La capa física no se considera una capa del protocolo TCP/IP digamos que es el cable de la red, la ponemos para que se vean mas claro los niveles.

CAPA	PROTOCOLOS
Capa de APLICACIÓN	(HTTP, SMTP, POP, FTP, TELNET...)
Capa de TRANSPORTE	(UDP, TCP)
Capa de RED o de INTERNET	(IP)
Capa de ACCESO o de ENLA CE	(ETHERNET, TOKEN RING...)
Cable coaxial, par trenzado...	Es el nivel mas bajo, lo que llamamos el cable, el hierro, el Hw



El nivel más bajo es la capa física. Aquí nos referimos al medio físico por el cual se transmite la información. Generalmente será un cable aunque no se descarta cualquier otro medio de transmisión como ondas o enlaces vía satélite.

La capa de acceso determina la manera en que las estaciones (ordenadores) envían y reciben la información a través del soporte físico proporcionado por la capa anterior. Es decir, una vez que tenemos un cable, ¿cómo se transmite la información por ese cable? ¿Cuándo puede una estación transmitir? ¿Tiene que esperar algún turno o transmite sin más? ¿Cómo sabe una estación que un mensaje es para ella? Pues bien, son todas estas cuestiones las que resuelve esta capa.

Las dos capas anteriores quedan a un nivel inferior del protocolo TCP/IP, es decir, no forman parte de este protocolo. La capa de red define la forma en que un mensaje se transmite a través de distintos tipos de redes hasta llegar a su destino. El principal protocolo de esta capa es el IP aunque también se encuentran a este nivel los protocolos ARP, ICMP e IGMP. Esta capa proporciona el direccionamiento IP y determina la ruta óptima a través de los encaminadores (*routers*) que debe seguir un paquete desde el origen al destino.

La capa de transporte (protocolos TCP y UDP) ya no se preocupa de la ruta que siguen los mensajes hasta llegar a su destino. Sencillamente, considera que la comunicación extremo a extremo está establecida y la utiliza. **Además añade la noción de puertos**, como veremos más adelante.

Una vez que tenemos establecida la comunicación desde el origen al destino nos queda lo más importante, ¿qué podemos transmitir? La capa de aplicación nos proporciona los distintos servicios de Internet: correo electrónico, páginas Web, FTP, TELNET…

1.4 Capa de acceso o enlace

Es la capa inferior de la jerarquía de protocolos de TCP/IP. Hay muchos protocolos de acceso a la red (uno por cada estándar físico de red, el mas conocido es Ethernet). Encapsula Datagramas en Frames (es decir en tramas) y mapea direcciones IP a direcciones físicas. Ejemplos de RFC's que definen protocolos de la capa de acceso a red son: RFC826 y RFC894

Esta capa se construye con la tarjeta de red, los drivers y los programas asociados. Un ejemplo: el Sistema Ethernet. Ethernet es una tecnología de redes de área local (LAN) que transmite información entre computadores a una velocidad de 10 Mbps (Ethernet), 100 Mbps (Fast Ethernet) ó 1000 Mbps (Gigabit Ethernet).

1. Los medios que soporta 10 Mbps son coaxial grueso, coaxial delgado, par trenzado y fibra óptica.

1. Los medios que soporta 100 Mbps son par trenzado y fibra óptica

1. Los medios que soporta 1000 Mbps son par trenzado y fibra óptica

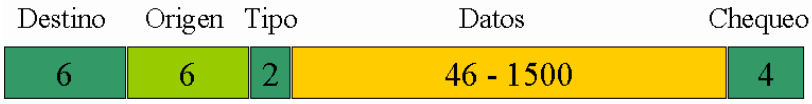
El frame Ethernet

1. El corazón del sistema Ethernet es el frame Ethernet utilizado para llevar datos entre computadores.

1. El "frame" consta de varios bits organizados en varios campos.

1. Estos campos incluyen la dirección física de las interfaces Ethernet, un campo variable de datos (entre 46 y 1500 bytes) y un campo de chequeo de error.

El frame Ethernet Versión 2



1. **Destino**: 6 bytes, dirección física del nodo destino (MAC address)

1. **Origen**: 6 bytes, dirección del nodo origen (MAC address)

1. **Tipo**: 2 bytes, especifica el protocolo de la capa superior

1. **Datos**: entre 46 y 1500 bits, información de las capas superiores

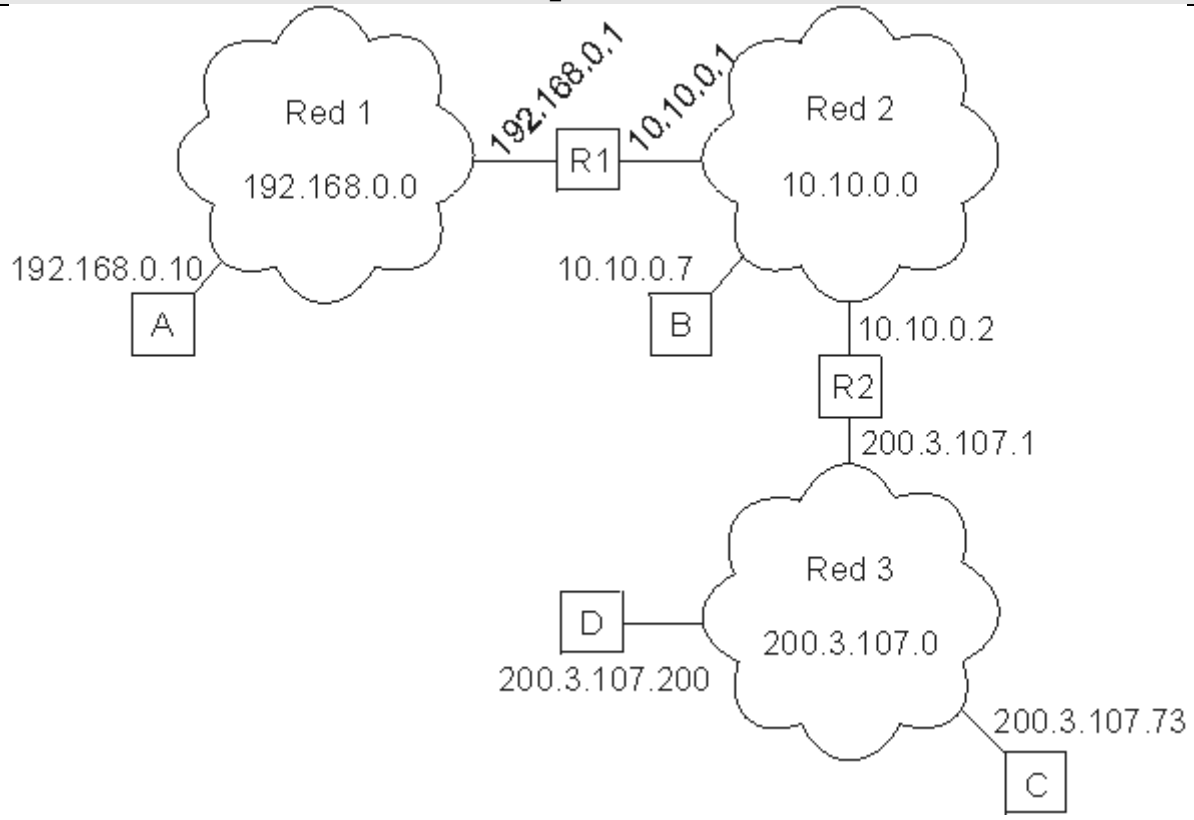
1. **Chequeo**: Secuencia de chequeo del frame.

Cuando un frame Ethernet es enviado al canal todas las interfaces revisan los primeros 6 bytes (48 bits). Si es su dirección MAC (o broadcast) reciben el paquete y lo entregarán al software de red, es decir a la capa superior, instalado en el computador.

1.5 Capa de red o Internet

La familia de protocolos TCP/IP fue diseñada para permitir la interconexión entre distintas redes. El mejor ejemplo de interconexión de redes es Internet: se trata de un conjunto de redes unidas mediante encaminadores o *routers*.A lo largo de este Curso aprenderemos a construir redes privadas que funcionen siguiendo el mismo esquema de Internet. En una red TCP/IP es posible tener, por ejemplo, servidores web y servidores de correo para uso interno. Obsérvese que todos los servicios de Internet se pueden configurar en pequeñas redes internas TCP/IP.A continuación veremos un ejemplo de interconexión de 3 redes. Cada *host* (ordenador) tiene una *dirección física* que viene determinada por su adaptador de red. Estas direcciones se corresponden con la *capa de acceso al medio* (Ethernet; token ring) y se utilizan para comunicar dos ordenadores que pertenecen a la misma red. Para identificar globalmente un ordenador dentro de un conjunto de redes TCP/IP se utilizan las direcciones IP (capa de red). Observando una dirección IP sabremos si pertenece a nuestra propia red o a una distinta (todas las direcciones IP de la misma red comienzan con los mismos números, según veremos más adelante).

Host	Dirección física	Dirección IP	Red
A	00-60-52-0B-B7-7D	192.168.0.10	Red 1
R1	00-E0-4C-AB-9A-FF	192.168.0.1	
	A3-BB-05-17-29-D0	10.10.0.1	Red 2
B	00-E0-4C-33-79-AF	10.10.0.7	
R2	B2-42-52-12-37-BE	10.10.0.2	
	00-E0-89-AB-12-92	200.3.107.1	Red 3
C	A3-BB-08-10-DA-DB	200.3.107.73	
D	B2-AB-31-07-12-93	200.3.107.200	



El concepto de **red** está relacionado con las direcciones IP que se configuren en cada ordenador, no con el cableado. Es decir, si tenemos varias redes dentro del mismo cableado solamente los ordenadores que permanezcan a una misma red podrán comunicarse entre sí. Para que los ordenadores de una red puedan comunicarse con los de otra red es necesario que existan routers que interconecten las redes. Un **router** o encaminador no es más que un ordenador con varias direcciones IP, una para cada red, que permita el tráfico de paquetes entre sus redes.

La capa de red se encarga de fragmentar cada mensaje en paquetes de datos llamados **datagramas IP** y de enviarlos de forma independiente a través de la red de redes. Cada datagrama IP incluye un campo con la dirección IP de destino. Esta información se utiliza para *enrutar* los datagramas a través de las redes necesarias que los hagan llegar hasta su destino.

Nota: *Cada vez que visitamos una página web o recibimos un correo electrónico es habitual atravesar un número de redes comprendido entre 10 y 20, dependiendo de la distancia de los hosts. El tiempo que tarda un datagrama en atravesar 20 redes (20 routers) suele ser inferior a 600 milisegundos.*

En el ejemplo anterior, supongamos que el ordenador 200.3.107.200 (D) envía un mensaje al ordenador con 200.3.107.73 (C). Como ambas direcciones comienzan con los mismos números, D sabrá que ese ordenador se encuentra dentro de su propia red y el mensaje se entregará de forma directa. Sin embargo, si el ordenador 200.3.107.200 (D) tuviese que comunicarse con 10.10.0.7 (B), D advertiría que el ordenador destino no pertenece a su propia red y enviaría el mensaje al router R2 (es el ordenador que le da salida a otras redes). El router entregaría el mensaje de forma directa porque B se encuentra dentro de una de sus redes (la Red 2).

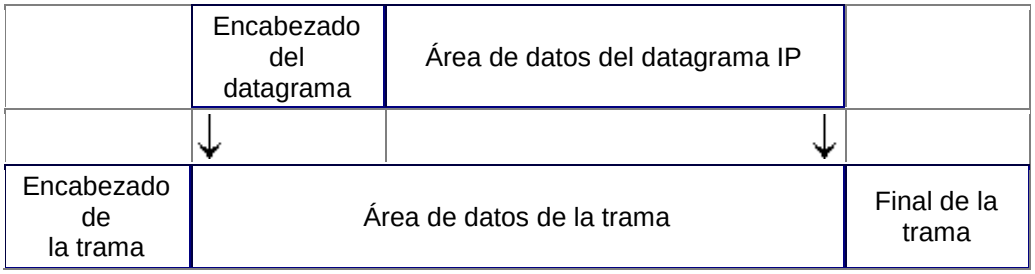
La familia de protocolos TCP/IP

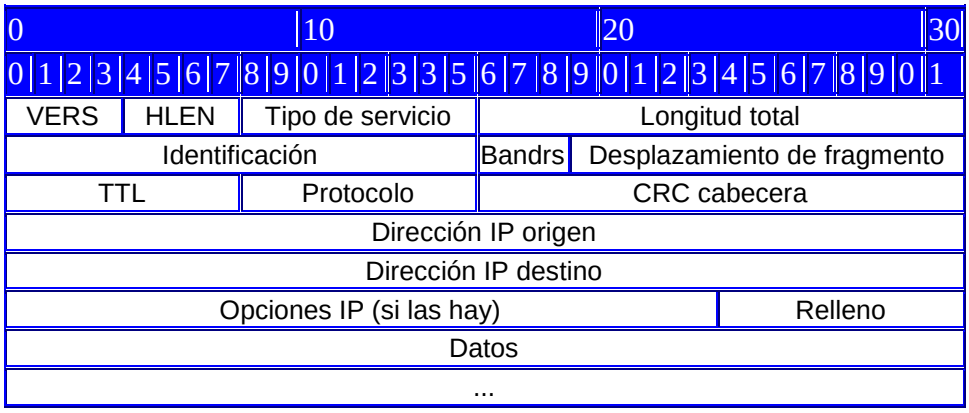
IP es el principal protocolo de la capa de red. Este protocolo define la unidad básica de transferencia de datos entre el origen y el destino, atravesando toda la red de redes. Además, el software IP es el encargado de elegir la ruta más adecuada por la que los datos serán enviados. Se trata de un sistema de entrega de paquetes (llamados *datagramas IP*) que tiene las siguientes características:

- Es no orientado a conexión debido a que cada uno de los paquetes puede seguir rutas distintas entre el origen y el destino. Entonces pueden llegar duplicados o desordenados.
- Es no fiable porque los paquetes pueden perderse, dañarse o llegar retrasados.

1.5.1 Formato de la trama IP

El datagrama IP es la unidad básica de transferencia de datos entre el origen y el destino. Viaja en el campo de datos de las tramas físicas (recuérdese la trama Ethernet) de las distintas redes que va atravesando. Cada vez que un datagrama tiene que atravesar un router, el datagrama *saldrá* de la trama física de la red que abandona y se *acomodará* en el campo de datos de una trama física de la siguiente red. Este mecanismo permite que un mismo datagrama IP pueda atravesar redes distintas: enlaces punto a punto, redes ATM, redes Ethernet, redes Token Ring, etc. El propio datagrama IP tiene también un campo de datos: será aquí donde viajen los paquetes de las capas superiores.





Campos del datagrama o trama IP:

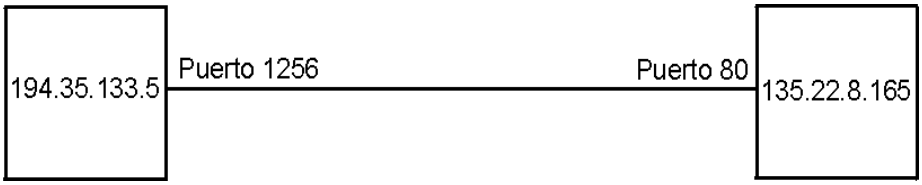
- **VERS** (4 bits). Indica la versión del protocolo IP que se utilizó para crear el datagrama. Actualmente se utiliza la versión 4 (IPv4) aunque ya se están preparando las especificaciones de la siguiente versión, la 6 (IPv6).
- **HLEN** (4 bits). Longitud de la cabecera expresada en múltiplos de 32 bits. El valor mínimo es 5, correspondiente a 160 bits = 20 bytes.
- **Tipo de servicio** *(Type Of Service)*. Los 8 bits de este campo se dividen a su vez en:
  - **Prioridad** (3 bits). Un valor de 0 indica baja prioridad y un valor de 7, prioridad máxima.
  - Los siguientes tres bits indican cómo se prefiere que se transmita el mensaje, es decir, son sugerencias a los encaminadores que se encuentren a su paso los cuales pueden tenerlas en cuenta o no.
  - **Bit D** *(Delay)*. Solicita retardos cortos (enviar rápido).
  - **Bit T** *(Throughput)*. Solicita un alto rendimiento (enviar mucho en el menor tiempo posible).

  - **Bit R** *(Reliability)*. Solicita que se minimice la probabilidad de que el datagrama se pierda o resulte dañado (enviar bien).
  - Los siguientes dos bits no tienen uso.
- **Longitud total** (16 bits). Indica la longitud total del datagrama expresada en bytes. Como el campo tiene 16 bits, la máxima longitud posible de un datagrama será de 65535 bytes.
- **Identificación** (16 bits). Número de secuencia que junto a la dirección origen, dirección destino y el protocolo utilizado identifica de manera única un datagrama en toda la red. Si se trata de un datagrama fragmentado, llevará la misma identificación que el resto de fragmentos.
- **Banderas** o indicadores (3 bits). Sólo 2 bits de los 3 bits disponibles están actualmente utilizados. El bit de *Más fragmentos* (**MF**) indica que no es el último datagrama. Y el bit de *No fragmentar* **(NF)** prohíbe la fragmentación del datagrama. Si este bit está activado y en una determinada red se requiere fragmentar el datagrama, éste no se podrá transmitir y se descartará.
- **Desplazamiento de fragmentación** (13 bits). Indica el lugar en el cual se insertará el fragmento actual dentro del datagrama completo, medido en unidades de 64 bits. Por esta razón los campos de datos de todos los fragmentos menos el último tienen una longitud múltiplo de 64 bits. Si el paquete no está fragmentado, este campo tiene el valor de cero.
- **Tiempo de vida** o TTL (8 bits). Número máximo de segundos que puede estar un datagrama en la red de redes. Cada vez que el datagrama atraviesa un router se resta 1 a este número. Cuando llegue a cero, el datagrama se descarta y se devuelve un mensaje ICMP de tipo "tiempo excedido" para informar al origen de la incidencia.
- **Protocolo** (8 bits). Indica el protocolo utilizado en el campo de datos: 1 para ICMP, 2 para IGMP, 6 para TCP y 17 para UDP.
- **CRC cabecera** (16 bits). Contiene la suma de comprobación de errores sólo para la cabecera del datagrama. La verificación de errores de los datos corresponde a las capas superiores.
- **Dirección origen** (32 bits). Contiene la dirección IP del origen.
- **Dirección destino** (32 bits). Contiene la dirección IP del destino.
- **Opciones IP**. Este campo no es obligatorio y especifica las distintas opciones solicitadas por el usuario que envía los datos (generalmente para pruebas de red y depuración).
- **Relleno.** Si las opciones IP (en caso de existir) no ocupan un múltiplo de 32 bits, se completa con bits adicionales hasta alcanzar el siguiente múltiplo de 32 bits (recuérdese que la longitud de la cabecera tiene que ser múltiplo de 32 bits).

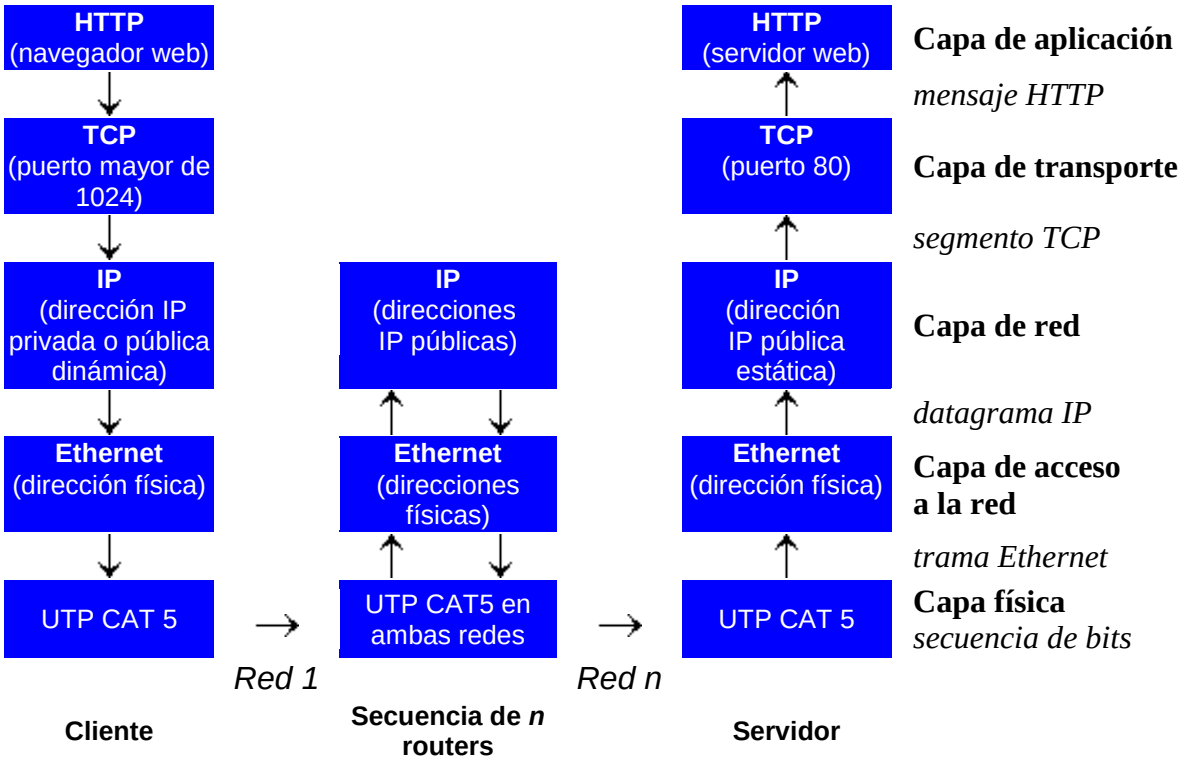
1.6 Capa de transporte

La capa de red transfiere datagramas entre dos ordenadores a través de la red utilizando como identificadores las direcciones IP. La capa de transporte añade la noción de *puerto* para distinguir entre los muchos destinos dentro de un mismo *host*. No es suficiente con indicar la dirección IP del destino, además hay que especificar la aplicación que recogerá el mensaje. Cada aplicación que esté esperando un mensaje utiliza un número de puerto distinto; más concretamente, la aplicación está a la espera de un mensaje en un puerto determinado (escuchando un puerto).

Pero no sólo se utilizan los puertos para la recepción de mensajes, también para el envío: todos los mensajes que envíe un ordenador debe hacerlo a través de uno de sus puertos. El siguiente diagrama representa una transmisión entre el ordenador 194.35.133.5 y el 135.22.8.165. El primero utiliza su puerto 1256 y el segundo, el 80.



La capa de transporte transmite mensajes entre las aplicaciones de dos ordenadores. Por ejemplo, entre nuestro navegador de páginas web y un servidor de páginas web, o entre nuestro programa de correo electrónico y un servidor de correo.



1.6.1 Puertos

Un ordenador puede estar conectado con distintos servidores a la vez; por ejemplo, con un servidor de noticias y un servidor de correo. Para distinguir las distintas conexiones dentro de un mismo ordenador se utilizan los puertos.

Un puerto es un número de 16 bits, por lo que existen 65536 puertos en cada ordenador. Las aplicaciones utilizan estos puertos para recibir y transmitir mensajes.

Los números de puerto de las aplicaciones cliente son asignados dinámicamente y generalmente son superiores al 1024. Cuando una aplicación cliente quiere comunicarse con un servidor, busca un número de puerto libre y lo utiliza.

En cambio, las aplicaciones servidoras utilizan unos números de puerto prefijados: son los llamados puertos *well-known* (bien conocidos). Estos puertos están definidos en la *RFC 1700* y se pueden consultar en http://www.ietf.org/rfc/rfc1700.txt. A continuación se enumeran los puertos *well-known* más usuales:

Palabra clave	Puerto	Descripción
	0/tcp	Reserved
	0/udp	Reserved
tcpmux	1/tcp	TCP Port Service Multiplexer
rje	5/tcp	Remote Job Entry
echo	7/tcp/udp	Echo
discard	9/tcp/udp	Discard
systat	11/tcp/udp	Active Users
daytime	13/tcp/udp	Daytime
qotd	17/tcp/udp	Quote of the Day
chargen	19/tcp/udp	Character Generator
ftp-data	20/tcp	File Transfer [Default Data]
ftp	**21/tcp**	**File Transfer [Control]**
telnet	**23/tcp**	**Telnet**
smtp	**25/tcp**	**Simple Mail Transfer**
time	37/tcp/udp	Time
nameserver	42/tcp/udp	Host Name Server
nicname	43/tcp/udp	Who Is
domain	**53/tcp/udp**	**Domain Name Server**
bootps	67/udp/udp	Bootstrap Protocol Server
tftp	69/udp	Trivial File Transfer
gopher	70/tcp	Gopher
finger	79/tcp	Finger
www-http	**80/tcp**	**World Wide Web HTTP**
dcp	93/tcp	Device Control Protocol
supdup	95/tcp	SUPDUP
hostname	101/tcp	NIC Host Name Server
iso-tsap	102/tcp	ISO-TSAP
gppitnp	103/tcp	Genesis Point-to-Point Trans Net
rtelnet	107/tcp/udp	Remote Telnet Service
pop2	109/tcp	Post Office Protocol - Version 2
pop3	**110/tcp**	**Post Office Protocol - Version 3**

sunrpc	111/tcp/udp	SUN Remote Procedure Call
auth	113/tcp	Authentication Service
sftp	115/tcp/udp	Simple File Transfer Protocol
nntp	**119/tcp**	**Network News Transfer Protocol**
ntp	123/udp	Network Time Protocol
pwdgen	129/tcp	Password Generator Protocol
netbios-ns	137/tcp/udp	NETBIOS Name Service
netbios-dgm	138/tcp/udp	NETBIOS Datagram Service
netbios-ssn	**139/tcp/udp**	**NETBIOS Session Service**
snmp	161/udp	SNMP
snmptrap	162/udp	SNMPTRAP
irc	194/tcp	Internet Relay Chat Protocol
www-http	**https/tcp**	**www con seguridad**

Los puertos tienen una memoria intermedia *(buffer)* situada entre los programas de aplicación y la red, de tal forma que las aplicaciones transmiten la información a los puertos. Aquí se va almacenando hasta que pueda enviarse por la red. Una vez que pueda transmitirse, la información irá llegando al puerto destino donde se irá guardando hasta que la aplicación esté preparada para recibirla.

Los dos protocolos principales de la capa de transporte son UDP y TCP. El primero ofrece una transferencias de mensajes no fiable y no orientada a conexión y el segundo, una transferencia fiable y orientada a conexión.

1.6.2 Protocolo TCP en la capa de transporte

El protocolo TCP (*Transmission Control Protocol*, protocolo de control de transmisión) está basado en IP que es no fiable y no orientado a conexión, y sin embargo es:

- Orientado a conexión. Es necesario establecer una conexión previa entre las dos máquinas antes de poder transmitir ningún dato. A través de esta conexión los datos llegarán siempre a la aplicación destino de forma ordenada y sin duplicados. Finalmente, es necesario cerrar la conexión.
- Fiable. La información que envía el emisor llega de forma correcta al destino.

El protocolo TCP permite una comunicación fiable entre dos aplicaciones. De esta forma, las aplicaciones que lo utilicen no tienen que preocuparse de la integridad de la información: dan por hecho que todo lo que reciben es correcto.

El flujo de datos entre una aplicación y otra viajan por un *circuito virtual*. Sabemos que los datagramas IP pueden seguir rutas distintas, dependiendo del estado de los encaminadores intermedios, para llegar a un mismo sitio. Esto significa que los datagramas IP que transportan los mensajes siguen rutas diferentes aunque el protocolo TCP logré la ilusión de que existe un único circuito por el que viajan todos los bytes uno detrás de otro (algo así como una tubería entre el origen y el destino). Para que esta comunicación pueda ser posible es necesario abrir previamente una conexión. Esta conexión garantiza que todos los datos lleguen correctamente de forma ordenada y sin duplicados. La unidad de datos del protocolo es el *byte*, de tal forma que la aplicación origen envía bytes y la aplicación destino recibe estos bytes.

Sin embargo, cada byte no se envía inmediatamente después de ser generado por la aplicación, sino que se espera a que haya una cierta cantidad de bytes, se agrupan en un *segmento* y se envía el segmento completo. Para ello son necesarias unas *memorias intermedias* o *buffers*. Cada uno de estos segmentos viaja en el campo de datos de un datagrama IP. Si el segmento es muy grande será necesario fragmentar el datagrama, con la consiguiente pérdida de rendimiento; y si es muy pequeño, se estarán enviando más cabeceras que datos. Por consiguiente, es importante elegir el mayor tamaño de segmento posible que no provoque fragmentación.

El protocolo TCP envía un *flujo de información no estructurado*. Esto significa que los datos no tienen ningún formato, son únicamente los bytes que una aplicación envía a otra. Ambas aplicaciones deberán ponerse de acuerdo para comprender la información que se están enviando.

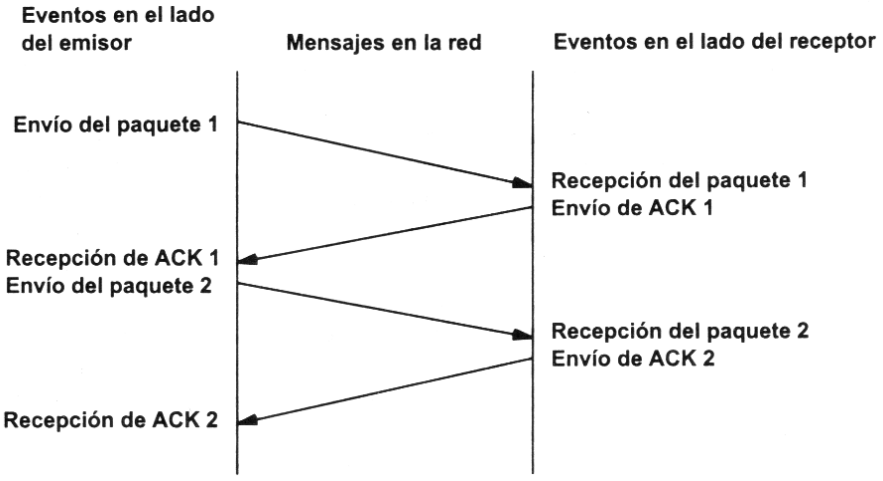
Cada vez que se abre una conexión, se crea un canal de comunicación bidireccional en el que ambas aplicaciones pueden enviar y recibir información, es decir la conexión es *full-dúplex*.

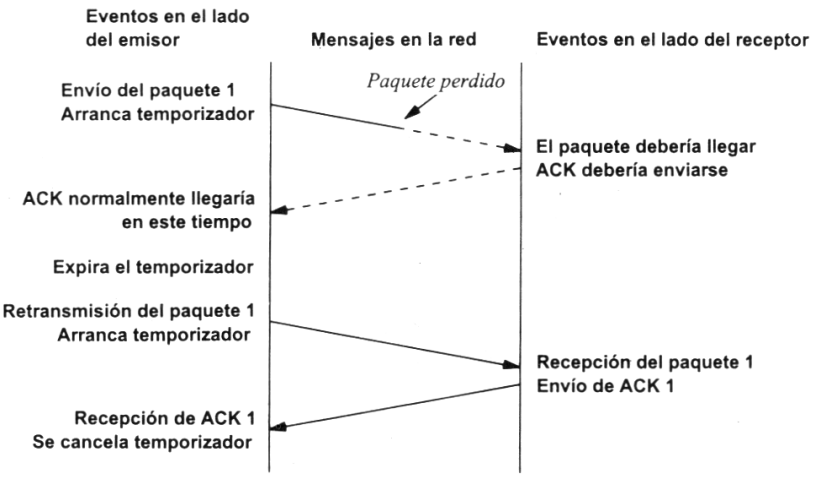
Fiabilidad

¿Cómo es posible enviar información fiable basándose en un protocolo no fiable? Es decir, si los datagramas que transportan los segmentos TCP se pueden perder, ¿cómo pueden llegar los datos de las aplicaciones de forma correcta al destino?

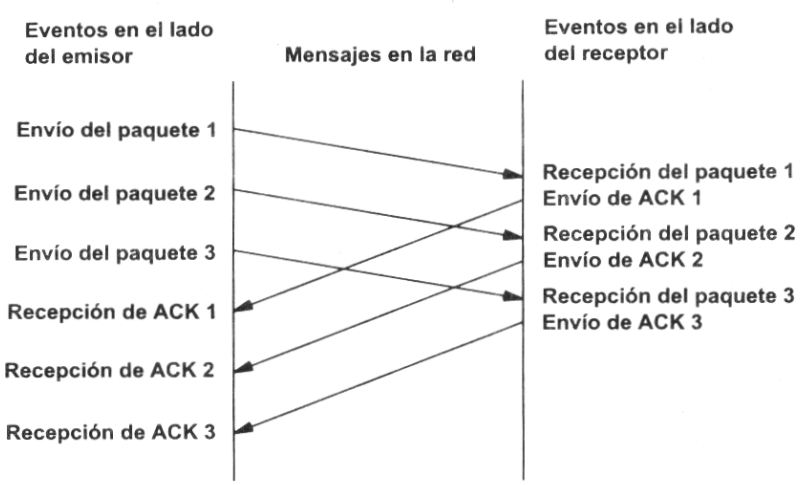
La respuesta a esta pregunta es sencilla: cada vez que llega un mensaje se devuelve una confirmación (*acknowledgement*) para que el emisor sepa que ha llegado correctamente. Si no le llega esta confirmación pasado un cierto tiempo, el emisor reenvía el mensaje.

Veamos a continuación la manera más sencilla (aunque ineficiente) de proporcionar una comunicación fiable. El emisor envía un dato, arranca su temporizador y espera su confirmación (ACK). Si recibe su ACK antes de agotar el temporizador, envía el siguiente dato. Si se agota el temporizador antes de recibir el ACK, reenvía el mensaje. Los siguientes esquemas representan este comportamiento:





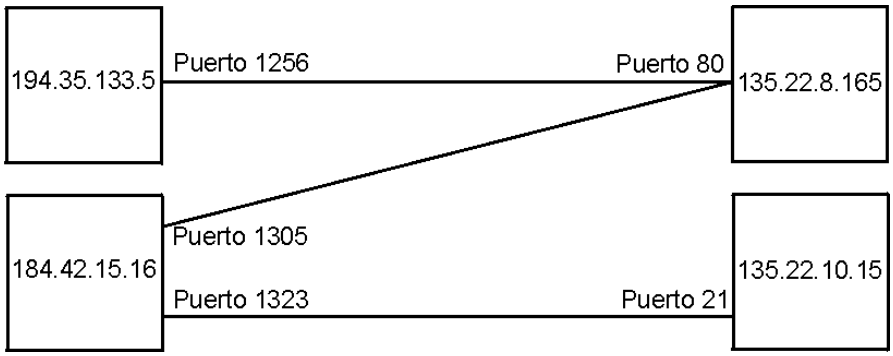
Este esquema es perfectamente válido aunque muy ineficiente debido a que sólo se utiliza un sentido de la comunicación a la vez y el canal está desaprovechado la mayor parte del tiempo. Para solucionar este problema se utiliza un *protocolo de ventana deslizante*, que se resume en el siguiente esquema. Los mensajes y las confirmaciones van numerados y el emisor puede enviar más de un mensaje antes de haber recibido todas las confirmaciones anteriores.



Conexiones

Una conexión son dos pares *dirección IP:puerto*. No puede haber dos conexiones iguales en un mismo instante en toda la Red. Aunque bien es posible que un mismo ordenador tenga dos conexiones distintas y simultáneas utilizando un mismo puerto. El protocolo TCP utiliza el concepto de conexión para identificar las transmisiones. En el siguiente ejemplo se han creado tres conexiones. Las dos primeras son al mismo servidor Web (puerto 80) y la tercera a un servidor de FTP (puerto 21).

Host 1	Host 2
194.35.133.5:1256	135.22.8.165:80
184.42.15.16:1305	135.22.8.165:80
184.42.15.16:1323	135.22.10.15:21

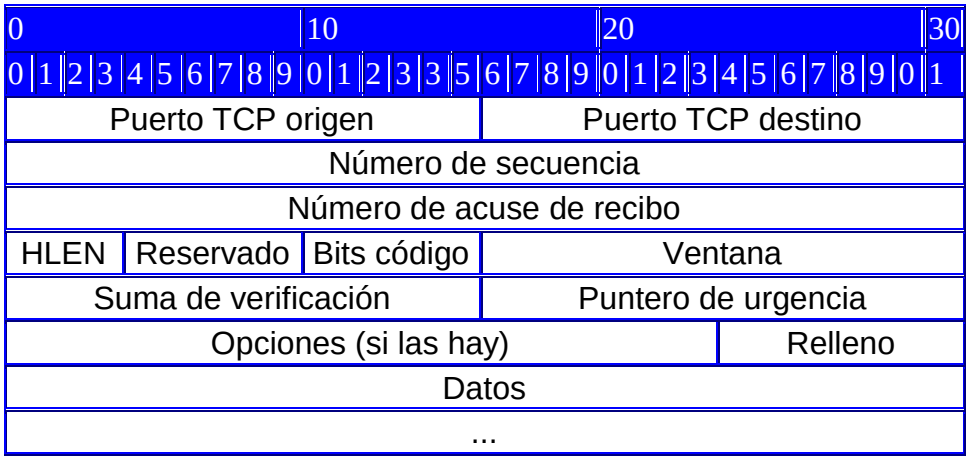


Para que se pueda crear una conexión, el extremo del servidor debe hacer una *apertura pasiva* del puerto (escuchar su puerto y quedar a la espera de conexiones) y el cliente, una *apertura activa* en el puerto del servidor (conectarse con el puerto de un determinado servidor).

Nota: *El comando* ***NetStat*** *muestra las conexiones abiertas en un ordenador, así como estadísticas de los distintos protocolos de Internet.*

1.6.3 Formato de la trama TCP

Ya hemos comentado que el flujo de bytes que produce una determinada aplicación se divide en uno o más segmentos TCP para su transmisión. Cada uno de estos segmentos viaja en el campo de datos de un datagrama IP. Para facilitar el control de flujo de la información los bytes de la aplicación se numeran. De esta manera, cada segmento indica en su cabecera el primer byte que transporta. Las confirmaciones o acuses de recibo (ACK) representan el siguiente byte que se espera recibir (y no el número de segmento recibido, ya que éste no existe).

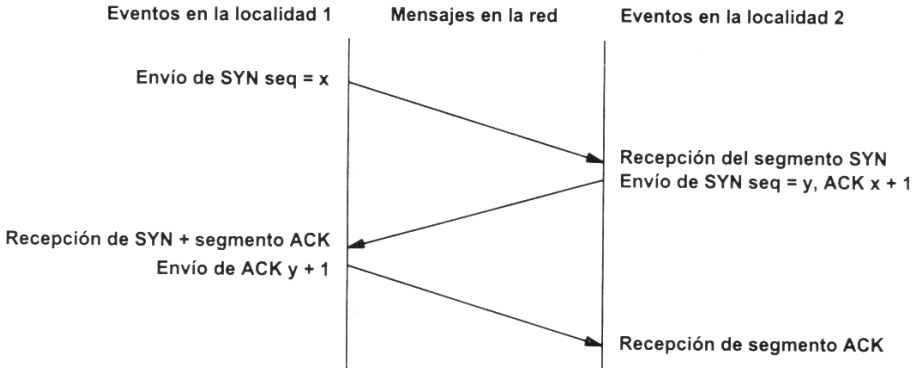


- **Puerto fuente** (16 bits). Puerto de la máquina origen. Al igual que el puerto destino es necesario para identificar la conexión actual.
- **Puerto destino** (16 bits). Puerto de la máquina destino.
- **Número de secuencia** (32 bits). Indica el número de secuencia del primer byte que trasporta el segmento.
- **Número de acuse de recibo** (32 bits). Indica el número de secuencia del siguiente byte que se espera recibir. Con este campo se indica al otro extremo de la conexión que los bytes anteriores se han recibido correctamente.
- **HLEN** (4 bits). Longitud de la cabecera medida en múltiplos de 32 bits (4 bytes). El valor mínimo de este campo es 5, que corresponde a un segmento sin datos (20 bytes).
- **Reservado** (6 bits). Bits reservados para un posible uso futuro.
- **Bits de código** o indicadores (6 bits). Los bits de código determinan el propósito y contenido del segmento. A continuación se explica el significado de cada uno de estos bits (mostrados de izquierda a derecha) si está a 1.
- **URG.** El campo *Puntero de urgencia* contiene información válida.
- **ACK.** El campo *Número de acuse de recibo* contiene información válida, es decir, el segmento actual lleva un ACK. Observemos que un mismo segmento puede transportar los datos de un sentido y las confirmaciones del otro sentido de la comunicación.
- **PSH.** La aplicación ha solicitado una operación *push* (enviar los datos existentes en la memoria temporal sin esperar a completar el segmento).
- **RST.** Interrupción de la conexión actual.

- **SYN.** Sincronización de los números de secuencia. Se utiliza al crear una conexión para indicar al otro extremo cual va a ser el primer número de secuencia con el que va a comenzar a transmitir (veremos que no tiene porqué ser el cero).
- **FIN.** Indica al otro extremo que la aplicación ya no tiene más datos para enviar. Se utiliza para solicitar el cierre de la conexión actual.
- **Ventana** (16 bits). Número de bytes que el emisor del segmento está dispuesto a aceptar por parte del destino.
- **Suma de verificación** (24 bits). Suma de comprobación de errores del segmento actual. Para su cálculo se utiliza una *pseudo-cabecera* que también incluye las direcciones IP origen y destino.
- **Puntero de urgencia** (8 bits). Se utiliza cuando se están enviando datos urgentes que tienen preferencia sobre todos los demás e indica el siguiente byte del campo *Datos* que sigue a los datos urgentes. Esto le permite al destino identificar donde terminan los datos urgentes. Nótese que un mismo segmento puede contener tanto datos urgentes (al principio) como normales (después de los urgentes).
- **Opciones** (variable). Si está presente únicamente se define una opción: el tamaño máximo de segmento que será aceptado.
- **Relleno.** Se utiliza para que la longitud de la cabecera sea múltiplo de 32 bits.
- **Datos.** Información que envía la aplicación.

1.6.4 Establecimiento de una conexión

Antes de transmitir cualquier información utilizando el protocolo TCP es necesario abrir una conexión. Un extremo hace una *apertura pasiva* y el otro, una *apertura activa*. El mecanismo utilizado para establecer una conexión consta de *tres vías*.



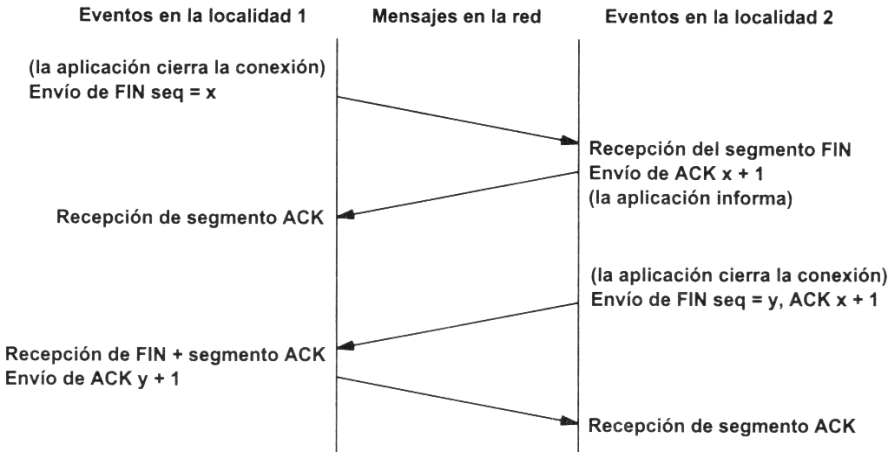
1. La máquina que quiere iniciar la conexión hace una apertura activa enviando al otro extremo un mensaje que tenga el bit SYN activado. Le informa además del primer número de secuencia que utilizará para enviar sus mensajes.
2. La máquina receptora (un servidor generalmente) recibe el segmento con el bit SYN activado y devuelve la correspondiente confirmación. Si desea abrir la conexión, activa el bit SYN del segmento e informa de su primer número de secuencia. Deja abierta la conexión por su extremo.
3. La primera máquina recibe el segmento y envía su confirmación. A partir de este momento puede enviar datos al otro extremo. Abre la conexión por su extremo.
4. La máquina receptora recibe la confirmación y entiende que el otro extremo ha abierto ya su conexión. A partir de este momento puede enviar ella también datos. La conexión ha quedado abierta en los dos sentidos.

Observamos que son necesarios 3 segmentos para que ambas máquinas abran sus conexiones y sepan que la otra también está preparada.

Números de secuencia.— *Se utilizan números de secuencia distintos para cada sentido de la comunicación. Como hemos visto el primer número para cada sentido se acuerda al establecer la comunicación. Cada extremo se inventa un número aleatorio y envía éste como inicio de secuencia. Observamos que los números de secuencia no comienzan entonces en el cero. ¿Por qué se procede así? Uno de los motivos es para evitar conflictos: supongamos que la conexión en un ordenador se interrumpe nada más empezar y se crea una nueva. Si ambas han empezado en el cero es posible que el receptor entienda que la segunda conexión es una continuación de la primera (si utilizan los mismos puertos).*

1.6.5 Cierre de una conexión

Cuando una aplicación ya no tiene más datos que transferir, el procedimiento normal es cerrar la conexión utilizando una variación del mecanismo de 3 vías explicado anteriormente.



El mecanismo de cierre es algo más complicado que el de establecimiento de conexión debido a que las conexiones son full-duplex y es necesario cerrar cada uno de los dos sentidos de forma independiente.

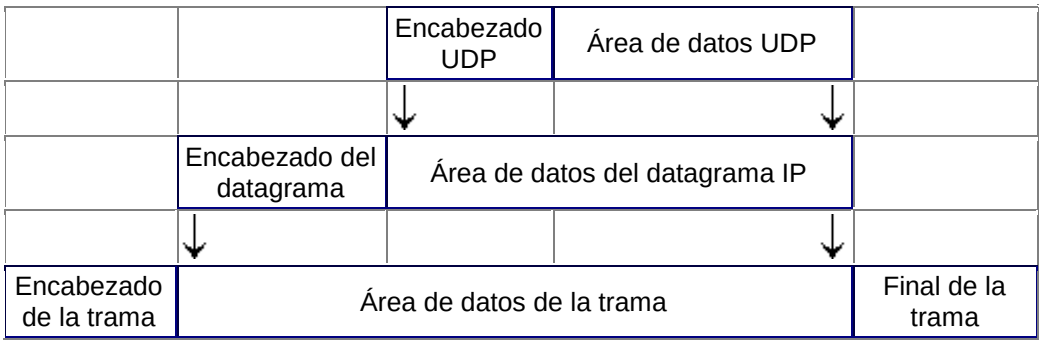
1. La máquina que ya no tiene más datos que transferir, envía un segmento con el bit FIN activado y cierra el sentido de envío. Sin embargo, el sentido de recepción de la conexión sigue todavía abierto.
2. La máquina receptora recibe el segmento con el bit FIN activado y devuelve la correspondiente confirmación. Pero no cierra inmediatamente el otro sentido de la conexión sino que informa a la aplicación de la petición de cierre. Aquí se produce un lapso de tiempo hasta que la aplicación decide cerrar el otro sentido de la conexión.
3. La primera máquina recibe el segmento ACK.
4. Cuando la máquina receptora toma la decisión de cerrar el otro sentido de la comunicación, envía un segmento con el bit FIN activado y cierra la conexión.
5. La primera máquina recibe el segmento FIN y envía el correspondiente ACK. Observemos que aunque haya cerrado su sentido de la conexión sigue devolviendo las confirmaciones.
6. La máquina receptora recibe el segmento ACK.

1.6.6 Protocolo UDP

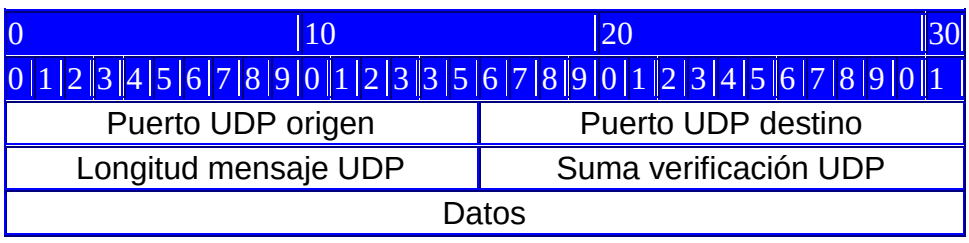
El protocolo UDP (*User Datagram Protocol*, protocolo de datagrama de usuario) proporciona una comunicación muy sencilla entre las aplicaciones de dos ordenadores. Al igual que el protocolo IP, UDP es:

- No orientado a conexión. No se establece una conexión previa con el otro extremo para transmitir un mensaje UDP. Los mensajes se envían sin más y éstos pueden duplicarse o llegar desordenados al destino.
- No fiable. Los mensajes UDP se pueden perder o llegar dañados.

UDP utiliza el protocolo IP para transportar sus mensajes. Como vemos, no añade ninguna mejora en la calidad de la transferencia; aunque sí incorpora los puertos origen y destino en su formato de mensaje. Las aplicaciones (y no el protocolo UDP) deberán programarse teniendo en cuenta que la información puede no llegar de forma correcta.



1.6.7 Formato de la trama UDP



...

- **Puerto UDP de origen** (16 bits, opcional). Número de puerto de la máquina origen.
- **Puerto UDP de destino** (16 bits). Número de puerto de la máquina destino.
- **Longitud del mensaje UDP** (16 bits). Especifica la longitud medida en bytes del mensaje UDP incluyendo la cabecera. La longitud mínima es de 8 bytes.
- **Suma de verificación UDP** (16 bits, opcional). Suma de comprobación de errores del mensaje. Para su cálculo se utiliza una *pseudo-cabecera* que también incluye las direcciones IP origen y destino. Para conocer estos datos, el protocolo UDP debe interactuar con el protocolo IP.
- **Datos**. Aquí viajan los datos que se envían las aplicaciones. Los mismos datos que envía la aplicación origen son recibidos por la aplicación destino después de atravesar toda la Red de redes.

1.7 Capa de aplicación

La capa de aplicación contiene todos los protocolos de alto nivel que se utilizan para ofrecer servicios a los usuarios. Entre estos podemos mencionar tanto los 'tradicionales', que existen desde que se creó el TCP/IP: terminal virtual (TelNet), transferencia de ficheros (FTP), correo electrónico (SMTP) y servidor de nombres (DNS), servicio de news (NNTP), la Web (HTTP), etc.

- **FTP (File Transfer Protocol).** Se utiliza para transferencia de archivos.
- **SMTP (Simple Mail Transfer Protocol)**. Es una aplicación para el envio de correo electrónico.
- **TELNET:** Permite la conexión a una aplicación remota desde un proceso o terminal.
- **POP .** Permite el recibo de correo.
- **HTTP (Hyper text transfer protocol).** Lo que conocemos como la web o ver páginas web.
- **NFS (Network File System).** Permite la utilización de archivos distribuidos por los programas de la red.

2. Nombres de dominio

Generalmente nosotros no trabajamos con direcciones IP sino con nombres de dominio del estilo de www.lluisacura.com o www.hiperbolsa.com Para que esto pueda ser posible es necesario un proceso previo de conversión de nombres de dominio a direcciones IP, ya que el protocolo IP requiere direcciones IP al enviar sus datagramas. Este proceso se conoce como *resolución de nombres*.

3. Necesidad del DNS (Domain Name Server)

En los orígenes de Internet, cuando sólo había unos cientos de ordenadores conectados, la tabla con los nombres de dominio y direcciones IP se encontraba almacenada en un único ordenador con el nombre de HOSTS.TXT. El resto de ordenadores debían consultarle a éste cada vez que tenían que resolver un nombre. Este fichero contenía una estructura plana de nombres, tal como hemos visto en el ejemplo anterior y funcionaba bien ya que la lista sólo se actualizaba una o dos veces por semana.

Sin embargo, a medida que se fueron conectando más ordenadores a la red comenzaron los problemas: el fichero HOSTS.TXT comenzó a ser demasiado extenso, el mantenimiento se hizo difícil ya que requería más de una actualización diaria y el tráfico de la red hacia este ordenador llegó a saturarla.

Es por ello que fue necesario diseñar un nuevo sistema de resolución de nombres que distribuyese el trabajo entre distintos servidores. Se ideó un sistema jerárquico de resolución conocido como DNS (*Domain Name System*, sistema de resolución de nombres).

3.1 Componentes del DNS

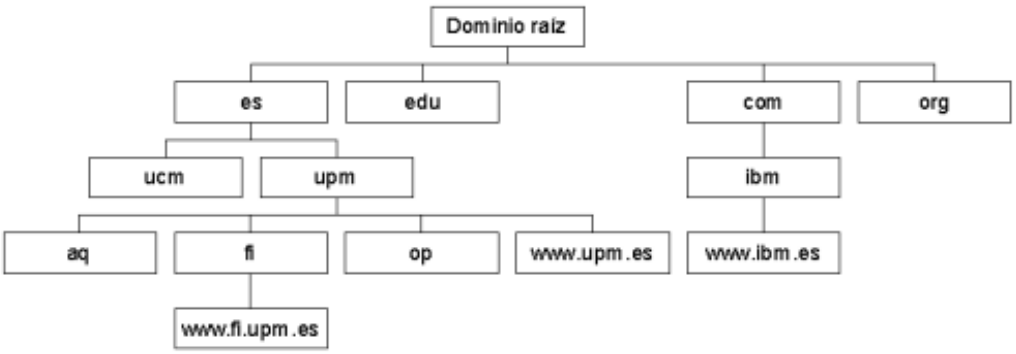
Para su funcionamiento, el DNS utiliza tres componentes principales:

- **Clientes DNS** *(resolvers)*. Los clientes DNS envían las peticiones de resolución de nombres a un servidor DNS. Las peticiones de nombres son preguntas de la forma: ¿Qué dirección IP le corresponde al nombre *nombre.dominio*?
- **Servidores DNS** *(name servers)*. Los servidores DNS contestan a las peticiones de los clientes consultando su base de datos. Si no disponen de la dirección solicitada pueden reenviar la petición a otro servidor.

- **Espacio de nombres de dominio** *(domain name space).* Se trata de una base de datos distribuida entre distintos servidores.

Espacio de nombres de dominio

El espacio de nombres de dominio es una estructura jerárquica con forma de árbol que clasifica los distintos dominios en niveles. A continuación se muestra una pequeña parte del espacio de nombres de dominio de Internet:



El punto más alto de la jerarquía es el dominio raíz. Los dominios de primer nivel (es, edu, com...) parten del dominio raíz y los dominios de segundo nivel (upm, ucm, microsoft...), de un dominio de primer nivel; y así sucesivamente. Cada uno de los dominios puede contener tanto *hosts* como más subdominios.

Un **nombre de dominio** es una secuencia de nombres separados por el carácter delimitador *punto*. Por ejemplo, www.fi.upm.es. Esta máquina pertenece al dominio *fi* (Facultad de Informática) que a su vez pertenece al dominio *upm* (Universidad Politécnica de Madrid) y éste a su vez, al dominio *es* (España).

Generalmente cada uno de los dominios es gestionado por un servidor distinto; es decir, tendremos un servidor para el dominio aq.upm.es (Arquitectura), otro para op.upm.es (Obras Públicas) y así sucesivamente.

Los dominios de primer nivel *(Top-Level Domains)* han sido clasificados tanto en función de su estructura organizativa como geográficamente. Ejemplos:

En función de su estructura organizativa:

Nombre de dominio	Significado
com	organizaciones comerciales
net	Redes
org	otras organizaciones
edu	instituciones educativas y universidades
gov	organizaciones gubernamentales
mil	organizaciones militares

Geográficamente:

Nombre de dominio	Significado
es	España
tw	Taiwán
fr	Francia
tv	Tuvalu
Cat	Catalunya

Zonas de autoridad

Una *zona de autoridad* es la porción del espacio de nombres de dominio de la que es responsable un determinado servidor DNS. La zona de autoridad de estos servidores abarca al menos un dominio y también pueden incluir subdominios; aunque generalmente los servidores de un dominio delegan sus subdominios en otros servidores. Por ejemplo el ordenador lluisacura.net, que es el Linux es el PC autoridad de su DNS.

Por ejemplo, en la escuela lluisacura, hay un ordenador que es el que controla todos los nombres DNS del centro, en el cual hay un sistema server 2003 instalado, en el que entre otras cosas tenemos un DNS (/etc/named si fuese Linux) en el que nosotros somos la autoridad. Nadie puede cambiar la configuración de nuestro DNS.

3.2 Tipos de servidores DNS

Dependiendo de la configuración del servidor, éste puede desempeñar distintos papeles:

- **Servidores primarios** *(primary name servers)*. Estos servidores almacenan la información de su zona en una base de datos local. Son los responsables de mantener la información actualizada y cualquier cambio debe ser notificado a este servidor.
- **Servidores secundarios** *(secundary name servers)*. Son aquellos que obtienen los datos de su zona desde otro servidor que tenga autoridad para esa zona. El proceso de copia de la información se denomina *transferencia de zona*.
- **Servidores maestros** *(master name servers)*. Los servidores maestros son los que transfieren las zonas a los servidores secundarios. Cuando un servidor secundario arranca busca un servidor maestro y realiza la transferencia de zona. Un servidor maestro para una zona puede ser a la vez un servidor primario o secundario de esa zona. Estos servidores extraen la información desde el servidor primario de la zona. Así se evita que los servidores secundarios sobrecargen al servidor primario con transferencias de zonas.
- **Servidores locales** *(caching-only servers)*. Los servidores locales no tienen autoridad sobre ningún dominio: se limitan a contactar con otros servidores para resolver las peticiones de los clientes DNS. Estos servidores mantienen una *memoria caché* con las últimas preguntas contestadas. Cada vez que un cliente DNS le formula una pregunta, primero consulta en su memoria caché. Si encuentra la dirección IP solicitada, se la devuelve al cliente; si no, consulta a otros servidores, apunta la respuesta en su memoria caché y le comunica la respuesta al cliente.

Los servidores secundarios son importantes por varios motivos. En primer lugar, *por seguridad* debido a que la información se mantiene de forma redundante en varios servidores a la vez. Si un servidor tiene problemas, la información se podrá recuperar desde otro. Y en segundo lugar, *por velocidad* porque evita la sobrecarga del servidor principal distribuyendo el trabajo entre distintos servidores situados estratégicamente (por zonas geográficas, por ejemplo).

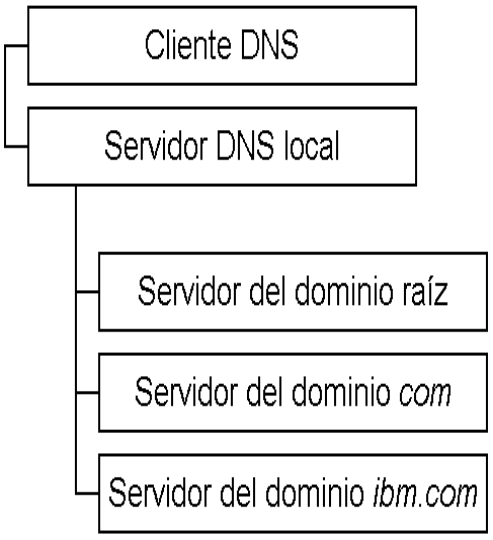
3.3 Resolución de nombres de dominio

La resolución de un nombre de dominio es la traducción del nombre a su correspondiente dirección IP. Para este proceso de traducción los *resolvers* pueden formular dos tipos de preguntas (recursivas e iterativas).

- **Preguntas recursivas**. Si un cliente formula una pregunta recursiva a un servidor DNS, éste debe intentar por todos los medios resolverla aunque para ello tenga que preguntar a otros servidores.
- **Preguntas iterativas**. Si, en cambio, el cliente formula una pregunta iterativa a un servidor DNS, este servidor devolverá o bien la dirección IP si la conoce o si no, la dirección de otro servidor que sea capaz de resolver el nombre.

Veamos un ejemplo: Estamos trabajando con Internet Explorer y escribimos en la barra de dirección: www.ibm.com. En primer lugar, el navegador tiene que resolver el nombre de dominio a una dirección IP. Después podrá comunicarse con la correspondiente dirección IP, abrir una conexión TCP con el servidor y mostrar en pantalla la página principal de IBM. La siguiente gráfica muestra el esquema de resolución:

1. Nuestro ordenador (cliente DNS) formula una **pregunta recursiva** a nuestro servidor DNS local (generalmente el proveedor de Internet).
2. El servidor local es el responsable de resolver la pregunta, aunque para ello tenga que reenviar la pregunta a otros servidores. Suponemos que no conoce la dirección IP asociada a www.ibm.com; entonces formulará una **pregunta iterativa** al servidor del dominio raíz.
3. El servidor del dominio raíz no conoce la dirección IP solicitada, pero devuelve la dirección del servidor del dominio *com*.
4. El servidor local reenvía la pregunta iterativa al servidor del dominio *com*.
5. El servidor del dominio *com* tampoco conoce la dirección IP preguntada, aunque sí conoce la dirección del servidor del dominio *ibm.com,* por lo que devuelve esta dirección.
6. El servidor local vuelve a reenvíar la pregunta iterativa al servidor del dominio *ibm.com*.
7. El servidor del dominio *ibm.com* conoce la dirección IP de www.ibm.com y devuelve esta dirección al servidor local.



8. El servidor local por fin ha encontrado la respuesta y se la reenvía a nuestro ordenador.

Preguntas inversas

Los clientes DNS también pueden formular **preguntas inversas**, esto es, conocer el nombre de dominio dada una dirección IP. Para evitar una búsqueda exhaustiva por todo el espacio de nombres de dominio, se ha creado un dominio especial llamado *in-addr.arpa*. Cuando un cliente DNS desea conocer el nombre de dominio asociado a la dirección IP *a.b.c.d*, formula una pregunta inversa a *d.c.b.a.in-addr.arpa*. La inversión de los bytes es necesaria debido a que los nombres de dominio son más genéricos por la derecha, al contrario que ocurre con las direcciones.

4. LA NUEVA VERSIÓN DE IP (IPng)

La nueva versión del protocolo IP recibe el nombre de IPv6, aunque es también conocido comúnmente como IPng (*Internet Protocol Next Generation*). El número de versión de este protocolo es el 6 (que es utilizada en forma mínima) frente a la antigua versión utilizada en forma mayoritaria. Los cambios que se introducen en esta nueva versión son muchos y de gran importancia, aunque la transición desde la versión antigua no debería ser problemática gracias a las características de compatibilidad que se han incluido en el protocolo. IPng se ha diseñado para solucionar todos los problemas que surgen con la versión anterior, y además ofrecer soporte a las nuevas redes de alto rendimiento (como ATM, Gigabit Ethernet, etc.)
Una de las características más llamativas es el nuevo sistema de direcciones, en el cual se pasa de los 32 a los 128 bit, eliminando todas las restricciones del sistema actual. Otro de los aspectos mejorados es la seguridad, que en la versión anterior constituía uno de los mayores problemas. Además, el nuevo formato de la cabecera se ha organizado de una manera más efectiva, permitiendo que las opciones se sitúen en extensiones separadas de la cabecera principal.

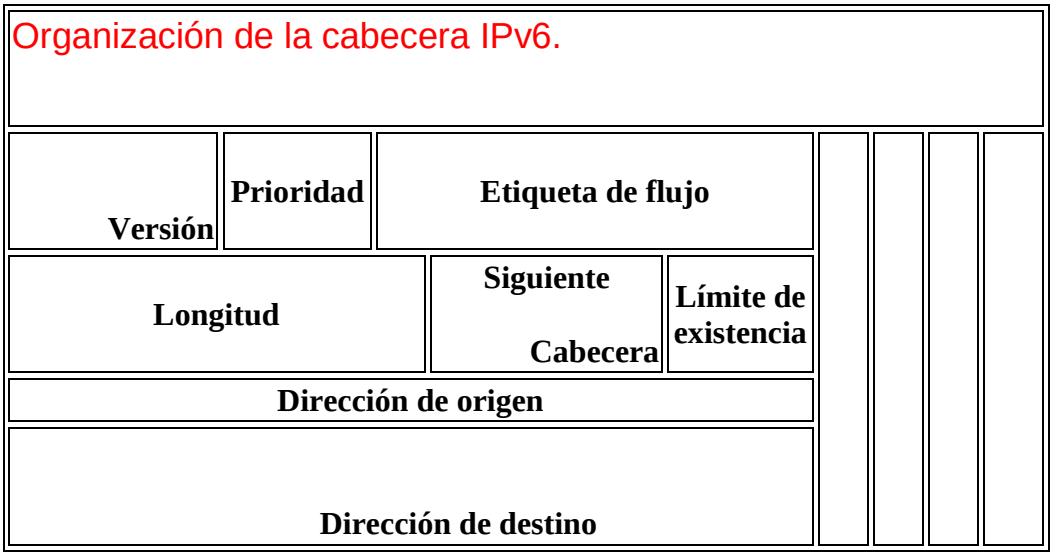
Formato de la cabecera.
El tamaño de la cabecera que el protocolo IPv6 añade a los datos es de 320 bit, el doble que en la versión antigua. Sin embargo, esta nueva cabecera se ha simplificado con respecto a la anterior. Algunos campos se han retirado de la misma, mientras que otros se han convertido en opcionales por medio de las extensiones. De esta manera los *routers* no tienen que procesar parte de la información de la cabecera, lo que permite aumentar de rendimiento en la transmisión. El formato completo de la cabecera sin las extensiones es el siguiente:

- **Versión:** Número de versión del protocolo IP, que en este caso contendrá el valor 6. *Tamaño: 4 bit.*
- **Prioridad:** Contiene el valor de la prioridad o importancia del paquete que se está enviando con respecto a otros paquetes provenientes de la misma fuente. *Tamaño: 4 bit.*
- **Etiqueta de flujo:** Campo que se utiliza para indicar que el paquete requiere un tratamiento especial por parte de los *routers* que lo soporten. *Tamaño: 24 bit.*
- **Longitud:** Es la longitud en bytes de los datos que se encuentran a continuación de la cabecera. *Tamaño: 16 bit.*
- **Siguiente cabecera:** Se utiliza para indicar el protocolo al que corresponde la cabecera que se sitúa a continuación de la actual. El valor de este campo es el mismo que el de protocolo en la versión 4 de IP. *Tamaño: 8 bit.*
- **Límite de existencia:** Tiene el mismo propósito que el campo de la versión 4, y es un valor que disminuye en una unidad cada vez que el paquete pasa por un nodo. *Tamaño:8 bit.*
- **Dirección de origen:** El número de dirección del *host* que envía el paquete. Su longitud es cuatro veces mayor que en la versión 4. *Tamaño: 128 bit.*

- **Dirección de destino:** Número de dirección de destino, aunque puede no coincidir con la dirección del *host* final en algunos casos. Su longitud es cuatro veces mayor que en la versión 4 del protocolo IP. *Tamaño: 128 bit.*

****

Organización de la cabecera IPv6.



Las extensiones que permite añadir esta versión del protocolo se sitúan inmediatamente después de la cabecera normal, y antes de la cabecera que incluye el protocolo de nivel de transporte. Los datos situados en cabeceras opcionales se procesan sólo cuando el mensaje llega a su destino final, lo que supone una mejora en el rendimiento. Otra ventaja adicional es que el tamaño de la cabecera no está limitado a un valor fijo de bytes como ocurría en la versión 4.
Por razones de eficiencia, las extensiones de la cabecera siempre tienen un tamaño múltiplo de 8 bytes. Actualmente se encuentran definidas extensiones para *routing* extendido, fragmentación y ensamblaje, seguridad, confidencialidad de datos, etc.

Direcciones en la versión 6.

El sistema de direcciones es uno de los cambios más importantes que afectan a la versión 6 del protocolo IP, donde se han pasado de los 32 a los 128 bit (cuatro veces mayor). Estas nuevas direcciones identifican a un interfaz o conjunto de interfaces y no a un nodo, aunque como cada interfaz pertenece a un nodo, es posible referirse a éstos a través de su interfaz.
El número de direcciones diferentes que pueden utilizarse con 128 bits es enorme. Teóricamente serían 2^{128} direcciones posibles, siempre que no apliquemos algún formato u organización a estas direcciones. Este número es extremadamente alto, pudiendo llegar a soportar más de 665.000 **trillones** de direcciones distintas por cada **metro cuadrado** de la superficie del planeta Tierra. Según diversas fuentes consultadas, estos números una vez organizados de forma práctica y jerárquica quedarían reducidos en el peor de los casos a 1.564 direcciones por cada metro cuadrado, y siendo optimistas se podrían alcanzar entre los tres y cuatro trillones.
Existen tres tipos básicos de direcciones IPng según se utilicen para identificar a un interfaz en concreto o a un grupo de interfaces. Los bits de mayor peso de los que componen la dirección IPng son los que permiten distinguir el tipo de dirección, empleándose un número variable de bits para cada caso. Estos tres tipos de

direcciones son:

Direcciones *unicast:* Son las direcciones dirigidas a un único interfaz de la red. Las direcciones *unicast* que se encuentran definidas actualmente están divididas en varios grupos. Dentro de este tipo de direcciones se encuentra también un formato especial que facilita la compatibilidad con las direcciones de la versión 4 del protocolo IP.

Direcciones *anycast:* Identifican a un conjunto de interfaces de la red. El paquete se enviará a un interfaz cualquiera de las que forman parte del conjunto. Estas direcciones son en realidad direcciones *unicast* que se encuentran asignadas a varios interfaces, los cuales necesitan ser configurados de manera especial. El formato es el mismo que el de las direcciones *unicast*.

Direcciones *multicast:* Este tipo de direcciones identifica a un conjunto de interfaces de la red, de manera que el paquete es enviado a cada una de ellos individualmente. Las direcciones de *broadcast* no están implementadas en esta versión del protocolo, debido a que esta misma función puede realizarse ahora mediante el uso de las direcciones *multicast*.