CAPITULO 1

Los protocolos TCP/IP

En este primer capítulo realizaremos un estudio profundo de los protocolos básicos que permiten el funcionamiento de Internet. La familia de protocolos TCP/IP es la piedra angular sobre la que se sustentan el resto de servicios (WWW, FTP, SSH...) que actualmente podemos encontrar en Internet, y por tanto su estudio es parte fundamental para el resto del trabajo.

Se introducirá el sistema de direccionamiento, el formato utilizado en sus cabeceras y el modo de funcionamiento de los protocolos más importantes de esta familia:

- El protocolo de control de flujo (**ICMP**).

- El protocolo no fiable de transmisión de datos sin conexión (**UDP**).

- El protocolo fiable de transmisión de datos con conexión (**TCP**).

Finalmente se explicará el proceso de rutado (*routing*) de los datagramas IP así como su transito por la red local hasta el ordenador destinatario.

Este estudio se centrará únicamente en la versión 4 del protocolo IP, ya que aunque actualmente se está trabajando en la versión 6 [Ver00] esta es aún experimental.

1.1 Redes IP

Definiremos las **redes IP** como aquellas redes que utilizan los protocolos TCP/IP para su funcionamiento. Internet es una red IP.

"Las familias de protocolos TCP/IP permiten la comunicación entre diferentes tipos de ordenadores con independencia del fabricante, red a la que se encuentren conectados y sistema operativo utilizado." [Ric98-1]

Las redes IP se caracterizan por haber sido construidas siguiendo un esquema de capas (*layers*). Cada capa es la responsable de cada una de las diferentes facetas de la comunicación. De esta forma, se puede definir la familia de protocolos TCP/IP como una combinación de cuatro capas (ver figura 1-1) según el modelo OSI [Ric98-1].

En este esquema, la capa superior accede únicamente a los servicios prestados por la capa situada justo en el nivel inferior a ella. De esta forma, independizamos una capa del resto de capas inferiores, lo que nos permite tener un esquema modular.

APLICACIONES (Applications)	WWW, SSH, FTP...
TRANSPORTE (Transport)	TCP, UDP, ICMP...
RED o Interconexión de redes (Network)	IP
ENLACE o Red real (Link)	IEEE 802.2, 802.3...

FIG. 1-1: Estructura de cuatro capas.

- La ***capa de enlace*** o **capa de red real** (*link layer*), también denominada la capa de datos (*data layer*) o capa de acceso a red (*network interface layer*), incluye los mecanismos que permiten al sistema operativo enviar y recibir información a través de la red a la que se encuentra físicamente conectado (Ethernet, RDSI...).

- La ***capa de red*** o **capa de interconexión de redes** (*network layer),* también denominada capa de Internet (*Internet layer*), es la encargada de mover los paquetes de información a través de las diferentes redes para llegar a su destino. En esta capa encontramos los protocolos de más bajo nivel, destacando el IP (*Internet Protocol*).

- La ***capa de transporte*** (*transport layer*), es la encargada de proporcionar un flujo de datos entre dos ordenadores. Este flujo de datos puede ser fiable (*Transmission Control Protocol, TCP*) o no fiable (*User Datagram Protocol, UDP*).

- La ***capa de aplicaciones*** (*applications layer*), es la encargada de manejar los detalles particulares relativos a las diferentes aplicaciones que utilizará el usuario (WWW, TELNET, FTP...).

Este sistema permite una independencia entre las diferentes capas y obliga a que la comunicación entre dos ordenadores se realice mediante una comunicación entre las capas del mismo nivel de los dos ordenadores (ver figura 1-2).

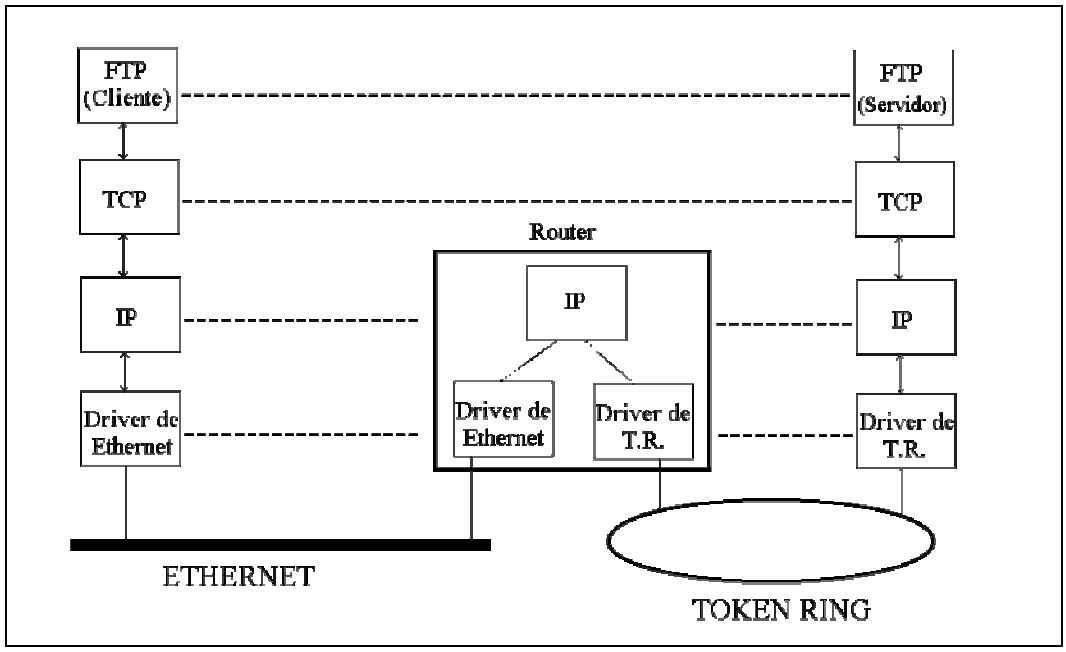

FIG. 1-2: Esquema de conexión de dos ordenadores en Internet.

La comunicación en Internet se produce mediante el intercambio de paquetes de información entre los distintos ordenadores. Estos paquetes de información (también denominados **datagramas**) viajan por los diferentes ordenadores que están conectados a Internet hasta que alcanzan su objetivo o son descartados por algún motivo.

De esta forma, en la comunicación de dos ordenadores por Internet podemos diferenciar dos tipos de funciones que pueden desempeñar los ordenadores por los cuales se transmiten los paquetes de información:

1. Ordenador ***emisor/receptor*** (*end-system o end-host*): Aquí se englobaría el ordenador origen o destinatario de la comunicación.

2. Ordenador ***intermedio*** *(intermediate-system, router o gateway*): Serían todos los ordenadores por los que van pasando los datagramas o paquetes de información hasta el ordenador destino de la comunicación o hasta el origen (en el caso de una respuesta).

El protocolo IP dispone de un sistema de numeración que permite diferenciar todos y cada uno de los ordenadores conectados. En la versión 4 de los protocolos TCP/IP[9], estas direcciones han de cumplir dos requisitos básicos (ver figura 1-3):

1. Deben ser únicas. No puede haber dos ordenadores con la misma dirección.

2. Las direcciones son números de 32 bits (4 bytes). Estas direcciones se representan mediante cuatro números decimales separados por un punto.

CLASE	RANGO		
A	0.0.0.0	Hasta	127.255.255.255
B	128.0.0.0	Hasta	191.255.255.255
C	192.0.0.0	Hasta	223.255.255.255
D	224.0.0.0	Hasta	239.255.255.255
E	240.0.0.0	Hasta	247.255.255.255

FIG. 1-3: Clases de direcciones IP en Internet.

Este tipo de direccionamiento, nos permite una gran flexibilidad a la hora de definir redes que posteriormente conectaremos a Internet. Así, una clase A sería ideal para redes muy grandes, ya que permite 128 redes (2^7) de 16.777.216 (2^24) ordenadores cada una. Mientras que una clase B permite 16.384 (2^14) redes con 65.535 ordenadores, y una clase C permite 2.097.152 (2^21) redes de 256 ordenadores.

Las clases D (*multicast*) y E (*reservada*) se utilizan para diferentes posibilidades como la de tener ordenadores en redes diferentes y que se vieran como si estuvieran en la misma (Ej. 2 ordenadores en la UAB, 1 en la UPC y 10 en la UB, y que todos recibieran la misma información, como en una multi-conferencia).

No obstante estas particularidades van más allá de los propósitos de este trabajo, y se recomienda la lectura de [Ric98-1] para más información.

[9] Aunque ya se llevan años trabajando en la versión 6 del protocolo IP [Ver00] la versión 4 es la que se está utilizando actualmente.

Una vez definido el direccionamiento de redes y ordenadores en Internet, mencionaremos la existencia de los servicios de DNS (*Domain Name Server*).

Debido a que es más fácil recordar un nombre (Centro de cálculo de la Universidad Autónoma de Barcelona, cc.uab.es) que una dirección numérica (158.109.0.4), se crearon los servidores de nombres (DNS), que son las máquinas encargadas de transformar un nombre en su dirección correspondiente.

1.2 El protocolo IP versión 4

El protocolo IP (*Internet Protocol*) es la pieza fundamental en la que se sustenta el sistema TCP/IP y por tanto todo el funcionamiento de Internet. Su especificación está recogida en **[RFC791]**.

La unidad de datos del protocolo IP es el *datagrama* (ver figura 1-4), cuyo tamaño máximo es de 65535 bytes (64K).

El protocolo IP facilita un sistema **sin conexión** (*connectionless*) y **no fiable** (*unreliable*) de entrega de datagramas entre dos ordenadores cualesquiera conectados a Internet.

IP da un servicio de entrega basado en el mejor intento (*best effort*). Esto implica que cuando hay algún funcionamiento anómalo de Internet, como podría ser un *router* colapsado, se contempla un sistema muy simple de tratamiento de errores. Este mecanismo de control de errores viene regulado por el protocolo ICMP (*Internet Control Message Protocol*).

En nuestro caso, el router colapsado descartaría el datagrama y enviaría un mensaje de error ICMP al ordenador de origen sin encargarse de la retransmisión del datagrama, lo que **no implica fiabilidad**.

Además, no mantiene ningún tipo de información referente al estado de las conexiones. Cada datagrama es encaminado de forma independiente, lo que le convierte en un **protocolo sin conexión**.

Debido a estas particulares características, puede pasar que se pierdan datagramas y/o que estos no lleguen en orden. De esta manera, cualquier fiabilidad que se necesite, deberá ser realizada por las capas superiores (TCP...).

La estructura de un datagrama IP está dividida en bloques de 32 bits (4 bytes). El datagrama IP se transmite enviando primero el bit 0, luego el bit 1, 2, 3... y así sucesivamente hasta finalizar el datagrama.

Este orden se denomina ***network byte order***. Es muy importante conocer este orden de transmisión de la información, puesto que los diferentes ordenadores tienen diferentes sistemas de almacenamiento de bits en memoria.

El formato *little endian*, consiste en almacenar los bits en orden inverso al *network byte order* (usando por ejemplo en los procesadores Intel), mientras que la otra posibilidad se denomina *big endian* (usado por ejemplo en los procesadores Motorola).

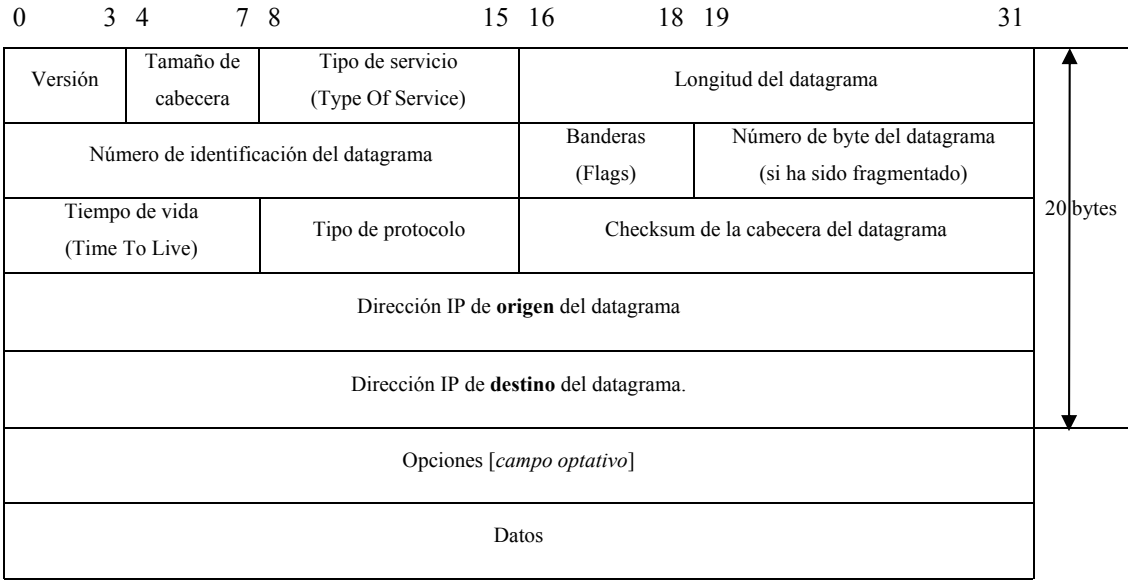

FIG. 1-4: Estructura de un datagrama IP v4.

La **versión** (4 bits), sirve para identificar a que versión específica (RFC) hace referencia el formato del datagrama. Esta información sólo es utilizada por los routers y capa IP de origen y final del datagrama. Esto permite la coexistencia de diferentes versiones del protocolo IP de una forma transparente al usuario. La versión actual es la 4 (conocida también cómo IPv4).

El **tamaño de la cabecera** (*Header Length*), son 4 bits (2^4 = 16 posiciones, 0...15) que indican el número de palabras de 32 bits que ocupa la cabecera. Estos 4 bits de tamaño máximo nos limitan a un tamaño de cabecera máximo de 60 bytes (15 * 32 bits = 60 bytes). No obstante, el valor usual de este campo es 5 (5 * 32 bits = 20 bytes).

El **campo del tipo de servicio** (*Type Of Service*), se compone de 8 bits. Los primeros 3 bits tienen una función obsoleta y no se contemplan actualmente. Los 4 bits siguientes definen el tipo de servicio (ver figura 2-12) y el último bit no se utiliza actualmente y debe tener valor 0. Solo 1 de los 4 bits del tipo de servicio puede estar activo a la vez. El tipo de servicio determina la política a seguir en el envío del datagrama por Internet. Las opciones posibles son:

1. minimizar el retraso (*minimize delay*)
2. maximizar el rendimiento (*maximize throughput*)
3. maximizar la fiabilidad del transporte (*maximize reliability*)
4. minimizar el coste económico del transporte (*minimize monetary cost*).

Tipo de aplicación	Minimizar retraso	Maximizar rendimiento	Maximizar fiabilidad	Minimizar coste	Valor en hexadecimal
TELNET	1	0	0	0	0x10
FTP	0	1	0	0	0x08
SMTP	0	1	0	0	0x08
DNS (UDP)	1	0	0	0	0x10
DNS (TCP)	0	0	0	0	0x00
ICMP	0	0	0	0	0x00
BOOTP	0	0	0	0	0x00

FIG. 1-5: Valores típicos del tipo se servicio según la aplicación.

La **longitud del datagrama** (*Total Length*), es un número de 16 bits (2^16 = 65536, 0...65535) que indica la longitud total del datagrama. Este valor es muy importante, ya que nos permite saber que tamaño de memoria debemos reservar para la recepción del datagrama. Además, nos indica el número de bytes a leer, lo que nos permite un simple control de error. De esta forma, si el valor es incorrecto, el número de bytes leídos será como máximo de 65535, acotando el error. Además nos limita el número de bytes a enviar en un datagrama (*Maximum Transfer Unit, MTU*) a 65535 – 20 (tamaño típico de la cabecera) = 65515 bytes.

Si el tamaño del datagrama, es mayor que el tamaño máximo del paquete de red (Ej. Datagrama de 32000 bytes enviado sobre una Ethernet, que tiene un tamaño máximo de paquete de 1500 bytes), éste se fragmenta en N trozos.

El **número de identificación del datagrama** (*Identificartion Field*), es un número de 16 bits que en caso de fragmentación de un datagrama nos indica su posición en el datagrama original. Esto nos permite recomponer el datagrama original en la máquina de destino. Este valor nos indica que un datagrama puede ser fragmentado en un máximo de 65535 fragmentos.

Las **banderas** (*Flags*) son 3 bits. El primero permiten señalar si el datagrama recibido es un fragmento de un datagrama mayor, bit M (*More*) activado. El segundo especifica si el datagrama no debe fragmentarse, bit DF (*Don't fragment*) activado y el tercero no se utiliza actualmente, asignándole el valor 0 [San99].

El **número de byte en el datagrama** (*Fragmentation Offset*), nos indica la posición en bytes que ocupan los datos en el datagrama original. Sólo tiene sentido si el datagrama forma parte de uno mayor que ha sido fragmentado. Este campo tiene un máximo de 13 bits (2^13 = 8192, como nos indica el desplazamiento en bytes 8192 * 8 bits = 65536). De esta forma, siempre podemos reconstruir el datagrama original con los fragmentos.

El **tiempo de vida** (*Time To Live*), es un campo de 8 bits que indica el tiempo máximo que el datagrama será válido y podrá ser transmitido por la red. Esto permite un mecanismo de control para evitar datagramas que circulen eternamente por la red (por ejemplo en el caso de bucles).

Este campo se inicializa en el ordenador de origen a un valor (máximo 2^8 = 256) y se va decrementando en una unidad cada vez que atraviesa un router. De esta forma, si se produce un bucle y/o no alcanza su destino en un máximo de 255 "saltos", es descartado. En este caso se envía un datagrama ICMP de error al ordenador de origen para avisar de su pérdida.

El **tipo de protocolo** (*Protocol*), es un valor que indica a que protocolo pertenece el datagrama (TCP, UDP, ICMP...). Es necesario debido a que todos los servicios de Internet utilizan IP como transporte, lo cual hace necesario un mecanismo de discriminación entre los diferentes protocolos.

El **checksum de la cabecera del datagrama** (*Header Checksum*), es una suma de comprobación que afecta sólo a la cabecera del datagrama IP. El resto de protocolos TCP, UDP, IGMP... tienen su propia cabecera y *checksum*. Su función es simplemente la de un mecanismo de control de errores. De esta forma, si se encuentra un error en el *checksum* de un datagrama IP, este es simplemente descartado y no se genera ningún mensaje de error. Esto implica que es deber de las capas superiores el control del flujo de los datagramas para asegurarse que estos lleguen correctamente al destino, ya sea utilizando un protocolo fiable (TCP) o implementando internamente algún tipo de control.

Tanto la **dirección IP de origen como la de destino** (*IP address*), están formadas por dos números de 32 bits. Estas direcciones se corresponden a una distribución según la figura 1-3.

1.3 El protocolo ICMP

El protocolo ICMP (*Internet Control Message Protocol*) es un protocolo simple que va encapsulado en datagramas IP (ver figura 1-6) y que tiene por función el control del flujo de la comunicación así como la comunicación de errores [RFC792].

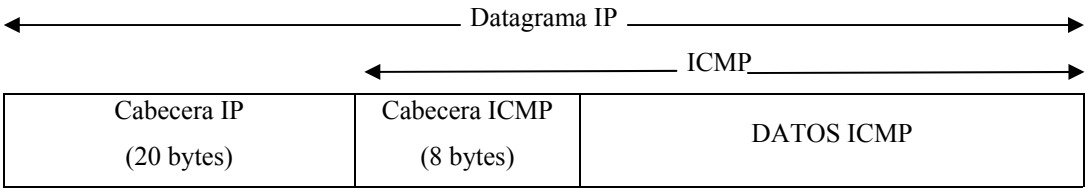

FIG. 1-6: Estructura de un mensaje ICMP.

La cabecera del protocolo ICMP tiene un tamaño de 8 bytes que contiene varios campos (ver figura 1-7) que permiten la identificación del mensaje.

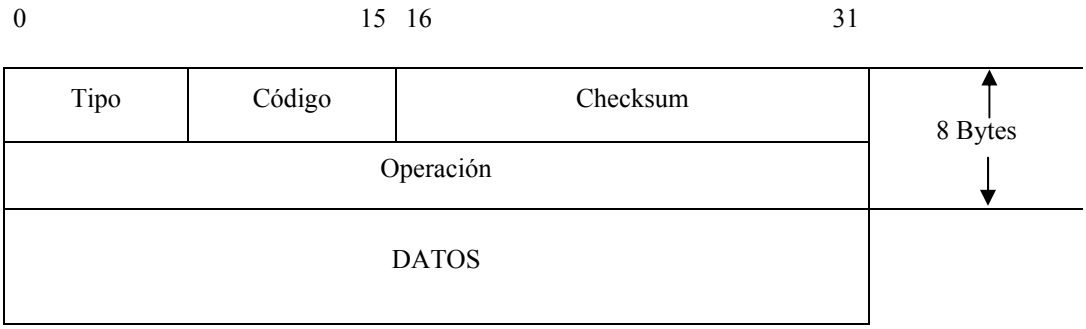

FIG. 1-7: Cabecera de un datagrama ICMP.

El **Tipo** (*Type*) indica el carácter del mensaje enviado, ya que el protocolo permite especificar una gran variedad de errores o mensajes de control de flujo de las comunicaciones (ver figura 1-8).

El campo de **Código** (*Code*) indica el código de error dentro del tipo de error indicado en el campo "tipo". Es decir, agrupa los mensajes en tipos y dentro de cada tipo especifica el código concreto al que se refiere (ver figura 1-8).

El **Checksum** simplemente permite verificar la integridad del mensaje enviado, lo que permite detectar posibles errores en el envío, transporte o recepción del mensaje de control ICMP.

El campo de **Operación** (*Operation*) depende directamente del contenido de los campos de tipo y código, permitiendo la inclusión de una información extra referida al código de error.

TIPO (*Type*)	**CÓDIGO** (*Code*)
0	Echo Reply Codes: 0 No Code
1	Unassigned
2	Unassigned
3	Destination Unreachable Codes: 0 Net Unreachable 1 Host Unreachable 2 Protocol Unreachable 3 Port Unreachable 4 Fragmentation Needed and Don't Fragment was Set 5 Source Route Failed 6 Destination Network Unknown 7 Destination Host Unknown 8 Source Host Isolated 9 Communication with network is Administratively Prohibited 10 Communication with Host Administratively Prohibited 11 Destination Network Unreachable for TOS 12 Destination Host Unreachable for TOS
4	Source Quench Codes: 0 No Code
5	Redirect Codes: 0 Redirect for the Network 1 Redirect Datagram for the Host 2 Redirect for the TOS and Network 3 Redirect for TOS and Oct
6	Alternate Host Address Codes: 0 Alternate Address for Oct
7	Unassigned
8	Echo Codes: 0 No Code
9	Router Advertisement Codes: 0 No Code
10	Router Selection Codes: 0 No Code
11	Time Exceded Codes: 0 Time to Live exceeded in Transit 1 Fragment Reassembly Time Exceded
12	Parameter Problem Codes: 0 Pointer indicates the error 1 Missing a Required Option 2 Bad Length
13	Timestamp Codes: 0 No Code
14	Timestamp Reply Codes: 0 No Code
15	Information Request Codes 0 No Code
16	Information Reply Codes 0 No Code
17	Address Mask Request Codes 0 No Code
18	Address Mask Reply Codes: 0 No Code
19	Reserved (for Security)
20-29	Reserved (for Robustness Experiment)
30	Traceroute
31	Datagram Conversion Error
32	Mobile Host Redirect
33	IPv6 Where-Are-You
34	IPv6 I-Am-Here
35	Mobile Registration Request
36	Mobile Registration Reply

FIG. 1-8: Tabla de tipos y códigos del protocolo ICMP.

1.4 El protocolo UDP

El protocolo UDP (*User Datagram Protocol*) se podría definir como un **protocolo simple** y **orientado a datagrama** [Ric98-1]. Su definición se recoge en **[RFC7680]** publicado por Postel en 1980.

De esta forma, cada envío de datos se corresponde con un único envío de un datagrama independiente del resto de datagramas y de la misma comunicación (ver figura 1-9).

De esta forma, siguiendo los preceptos de encaminamiento de datagramas por Internet, la entrega al destino no está asegurada por el propio protocolo. Es mas, ni tan siquiera se asegura que los datagramas lleguen en el orden en el que fueron enviados.

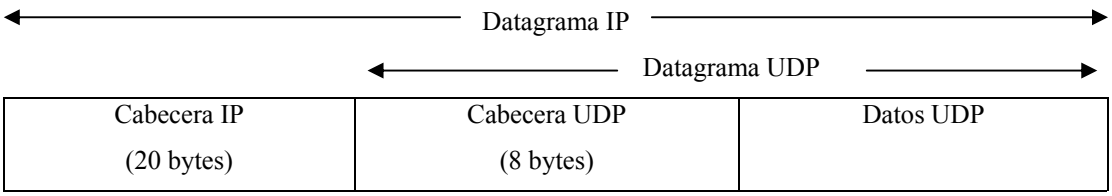

FIG. 1-9: Estructura de un datagrama UDP.

Debido a que UDP utiliza IP para su transporte por Internet, en caso de ser necesario (por diferentes tamaños de MTU, por ejemplo), este se fragmentará y ninguno de estos fragmentos (al igual que el datagrama original) proporcionara ningún tipo de seguridad o fiabilidad en la entrega.

Debido a la sencillez de este protocolo, el diseño de su cabecera (figura 1-10), es mucho más simple que el de IP.

El **número de puerto** (*Port*) se utiliza en la comunicación entre dos ordenadores para diferenciar las diferentes conexiones existentes. Si tenemos varias comunicaciones desde nuestro ordenador (por ejemplo un TELNET al *cc.uab.es* mirando el correo y un

FTP a *blues.uab.es* bajando un fichero), al recibir un datagrama IP debemos saber a cual de las conexiones pertenece.

Asignando un número de puerto a la comunicación podemos saber a que conexión pertenece. Al ser un número de 16 bits, podemos deducir que el número máximo de conexiones que un ordenador puede tener simultáneamente en uso es de 65535 (2^16).

En [WWW58] se puede obtener una lista completa de los servicios asociados a los puertos TCP y UDP.

La **longitud del datagrama** (*UDP Length*) hace referencia al tamaño del datagrama en bytes, y engloba la cabecera (8 bytes) más los datos que transporta. El mínimo valor de longitud es 8 bytes (por lo tanto, el protocolo permite enviar un datagrama UDP con 0 bytes).

Este campo es redundante, ya que utiliza IP para su transporte, y éste ya incorpora un campo para la longitud de los datos (ver figura 2-11) que sería la longitud del datagrama IP menos el tamaño de la cabecera.

El campo de ***checksum*** al igual que en IP, sirve como método de control de los datos, verificando que no han sido alterados. Este *checksum* cubre tanto la cabecera UDP como los datos enviados. Es necesario debido a que el *checksum* del protocolo IP tan sólo cubre la cabecera IP y no los datos que transporta. Si se detecta un error en el *checksum*, el datagrama es descartado sin ningún tipo de aviso.

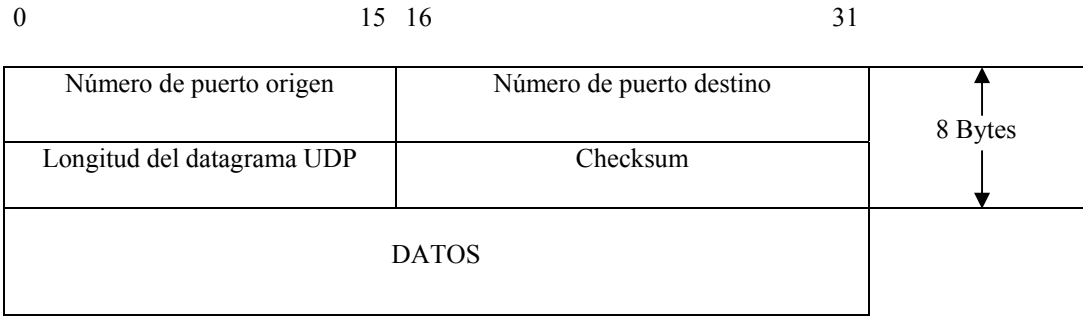

FIG. 1-10: Cabecera de un datagrama UDP.

1.5 El protocolo TCP

El protocolo TCP (*Transmission Control Protocol*) se podría definir como un protocolo **orientado a conexión**, **fiable** y **orientado a un flujo de bytes** [Ric98-1]. Su definición se recoge en **[RFC793]** publicado por Postel en 1981.

Aunque el protocolo TCP al igual que UDP utiliza los servicios de IP para su transporte por Internet (ver figura 1-11), es un protocolo **orientado a conexión**. Esto significa que las dos aplicaciones envueltas en la comunicación (usualmente un cliente y un servidor), deben establecer previamente una comunicación antes de poder intercambiar datos.

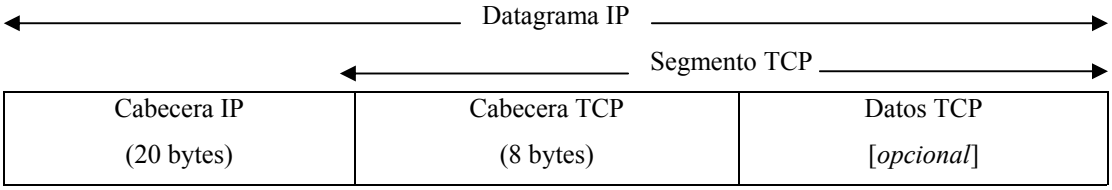

FIG. 1-11: Estructura de un segmento TCP.

TCP es también un protocolo **fiable**. La fiabilidad proporcionada por este protocolo viene dada principalmente por los siguientes aspectos:

1. Los datos a enviar son reagrupados por el protocolo en porciones denominadas *segmentos* [Ric98-1]. El tamaño de estos segmentos lo asigna el propio protocolo. Esto lo diferencia claramente de UDP, dónde cada porción de datos generada corresponde a un datagrama.

2. Cuando en una conexión TCP se recibe un segmento completo, el receptor envía una respuesta de confirmación (*Acknowledge*) al emisor confirmando el número de bytes correctos recibidos. De esta forma, el emisor da por correctos los bytes enviados y puede seguir enviando nuevos bytes.

3. Cuando se envía un segmento se inicializa un temporizador (*timer*). De esta forma, si en un determinado plazo de tiempo no se recibe una confirmación (*Acknowledge*) de los datos enviados, estos se retransmiten.

4. TCP incorpora un *checksum* para comprobar la validez de los datos recibidos. Si se recibe un segmento erróneo (fallo de *checksum* por ejemplo), no se envía una confirmación. De esta forma, el emisor retransmite los datos (bytes) otra vez.

5. Como IP no garantiza el orden de llegada de los datagramas, el protocolo TCP utiliza unos números de secuencia para asegurar la recepción en orden, evitando cambios de orden y/o duplicidades de los bytes recibidos.

6. TCP es un protocolo que implementa un control de flujo de datos. De esta forma, en el envío de datos se puede ajustar la cantidad de datos enviada en cada segmento, evitando colapsar al receptor. Este colapso sería posible si el emisor enviara datos sin esperar la confirmación de los bytes ya enviados.

La cabecera del segmento TCP (figura 1-12), es bastante más compleja que la de UDP debido a que la comunicación es más elaborada y debe proporcionar fiabilidad. Esto implica una serie de información adicional que debe mantenerse para poder conocer el estado de la comunicación en cualquier momento.

El número de **puerto origen** y número de **puerto destino**, sirven para diferenciar una comunicación en un ordenador de las demás. Cumple la misma función que en el datagrama UDP.

La tupla formada por la dirección IP y el número de puerto se denomina *socket*. Este término se utilizó luego en la especificación de la interface de programación de Berkeley (API).

Análogamente, la agrupación de las dos tuplas que definen una conexión entre dos ordenadores se denomina *socket pair*. En [WWW58] se puede obtener una lista completa de los servicios asociados a los puertos TCP y UDP.

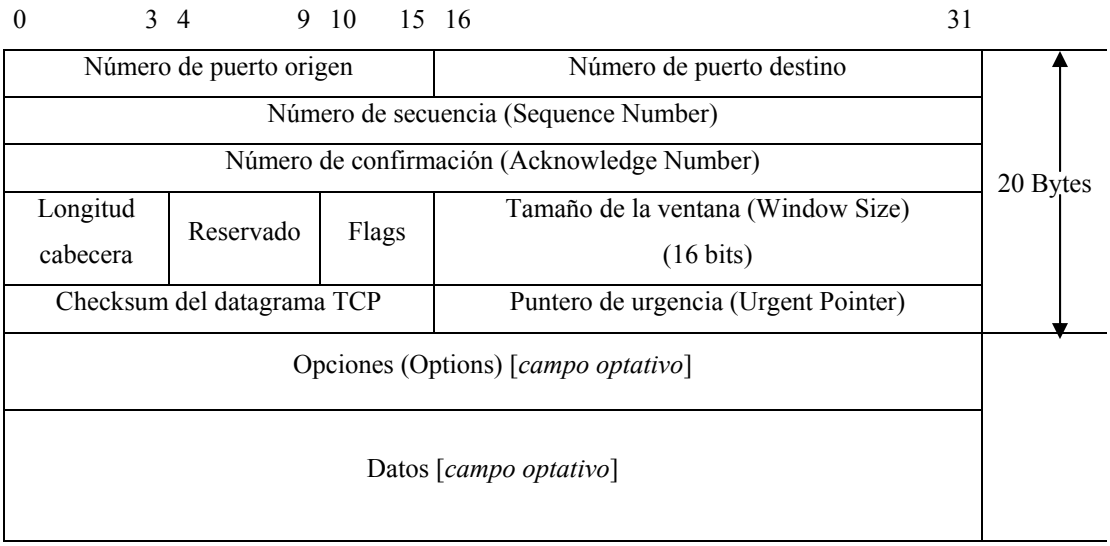

FIG. 1-12: Cabecera de un segmento TCP.

El **número de secuencia** (*Sequence Number*), identifica el byte concreto del flujo de datos que actualmente se envía del emisor al receptor. De esta forma, TCP numera los bytes de la comunicación de una forma consecutiva a partir del número de secuencia inicial. Cuando se establece una comunicación, emisor y receptor eligen un número de secuencia común, lo que nos permite implementar mecanismos de control como asegurar que los datos lleguen en el orden adecuado. Es un número de 32 bits, con lo que podemos enviar 2^32-1 bytes antes de que reinicie el ciclo.

El **número de confirmación** (*Acknowledge Number*), es el número de secuencia más uno. De este modo se especifica al emisor que los datos enviados hasta este número de secuencia menos uno son correctos. De aquí la importancia de la elección al principio de la comunicación de un número de secuencia común.

La **longitud de la cabecera** (*header Length*), especifica en palabras de 32 bits (4 bytes) el tamaño de la cabecera del segmento TCP incluyendo las posibles opciones. De esta forma el tamaño máximo es 15 * 4 = 60 bytes. No obstante, lo usual es tener un tamaño de 20 bytes (cabecera dónde no se incluyen opciones).

Las **banderas** (*Flags*), son las encargadas de especificar los diferentes estados de la comunicación. Así mismo, también validan los valores de los distintos campos de la cabecera de control. Puede haber simultáneamente varios *flags* activados. En la figura 1-13 podemos ver los distintos *flags* existentes y su significado.

FLAG	**Significado**
URG	Si es válido el puntero de urgencia.
ACK	El valor situado en el campo de confirmación (acknowledge) es válido.
PSH	El receptor debe pasar los datos a la aplicación o antes posible.
RST	RESET de la conexión
SYN	Inicio de comunicación. Búsqueda de un número de secuencia común.
FIN	El emisor finaliza el envío de datos.

FIG. 1-13: Flags del segmento TCP.

El **tamaño de la ventana** (*Window Size*), es el número de bytes desde el número especificado en el campo de confirmación, que el receptor está dispuesto a aceptar. El tamaño máximo es de (2^16) 65535 bytes. De esta forma, el protocolo TCP permite la regulación del flujo de datos entre el emisor y el receptor.

El **checksum del segmento** TCP, al igual que el del UDP o IP, tiene la función de controlar los posibles errores que se produzcan en la transmisión. Este *checksum* engloba la cabecera TCP y los datos. En caso de error, el datagrama/segmento queda descartado y el propio protocolo es el encargado de asegurar la retransmisión de los segmentos erróneos y/o perdidos.

El **puntero de urgencia** (*Urgent Poiner*), es válido sólo si el *flag* de **URG** se encuentra activado. Consiste en un valor positivo que se debe sumar al número de secuencia especificando una posición adelantada dónde podemos enviar datos urgentes.

Las **opciones** (*Options*), nos permiten especificar de forma opcional características extras a la comunicación. Un ejemplo de las opciones es el MSS (*Maximum Segment Size*), que especifica el tamaño máximo de datos que el emisor desea recibir [Ric98-1]. Esta opción se indica al inicio de la comunicación (*flag* SYN activado).

Los **datos** (*Data*) son opcionales. Esto significa que podemos enviar simplemente cabeceras TCP con diferentes opciones. Esta característica se utiliza por ejemplo al iniciar la comunicación o en el envío de confirmaciones. De esta manera, minimizamos el *overhead* ya que tan sólo enviamos/recibimos lo necesario para establecer o confirmar la comunicación.

1.5.1 Establecimiento de conexión TCP

TCP es un protocolo orientado a conexión. Esto implica que se ha de realizar un paso previo antes de poder intercambiar datos. Este paso es el de establecimiento de conexión.

Este paso es fundamental para poder garantizar la fiabilidad del protocolo, ya que es en este paso previo dónde se obtienen los números de secuencia que permitirán gestionar cualquier intercambio entre lo dos extremos de la comunicación. El método escogido para establecer la conexión se denomina **protocolo de 3 pasos** (*three way handshake,* ver figura 1-14).

En este esquema, podemos diferenciar un cliente activo que inicia la conexión (*active open*) y un servidor pasivo que tan sólo se limita a contestar (*passive open*) a las peticiones de conexión. Los tres pasos desarrollados en la petición de conexión son:

1. El cliente (ordenador que desea iniciar la comunicación) selecciona un número aleatorio de secuencia (***x*** en la figura 1-14). A continuación activa el *flag* de SYN en el campo de opciones. Finalmente envía un segmento TCP al ordenador destino.

2. El servidor (ordenador con el que se quiere establecer una comunicación) recibe la petición y almacena el número de secuencia ***x***. Elige un número aleatorio (***y*** en la figura 1-14) que utilizará como número de secuencia y activa los *flags* SYN y ACK.

   Finalmente envía un segmento con el número de secuencia elegido y con una confirmación del valor recibido mas uno (***ACK x+1*** en la figura 1-14).

3. El cliente almacena el número de secuencia (***y*** en la figura 1-14). Activa el *flag* de ACK y finalmente envía una confirmación del número recibido mas uno ***(ACK y+1*** en la figura 1-14).

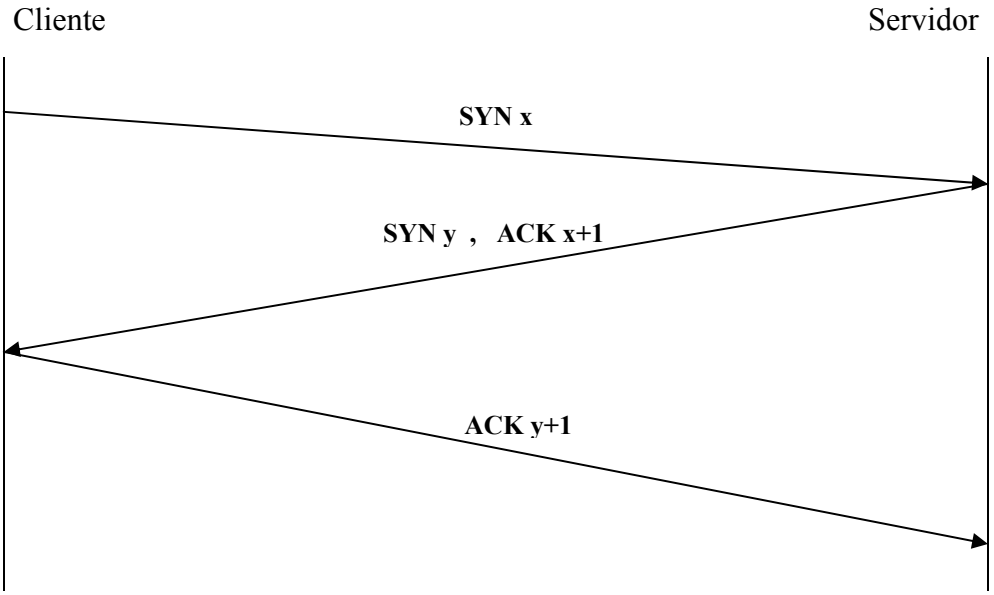

FIG. 1-14: Establecimiento de una conexión TCP (*three way handshake*).

En el caso de que este tercer paso no se realice, después de un cierto tiempo (entre 70 y 130 segundos dependiendo del sistema operativo) la conexión se liberará.

Esta característica se produce porque cuando se inicia una conexión TCP también se inicializa un temporizador (*timer*), que al finalizar (*time-out*) nos permite liberar la conexión y los recursos asociados.

1.5.2 Finalización de conexión TCP

Tal y como hemos visto en el punto anterior, se necesitan tres acciones para iniciar una conexión TCP. Para finalizarla se necesitan cuatro acciones debido a que la comunicación es *full duplex* (los datos pueden ser enviados y/o recibidos independientemente y en cualquier dirección de la comunicación).

Esto obliga a que cada sentido de la comunicación deba ser finalizado independientemente (ver figura 1-15).

Debido a la posibilidad de que cualquiera de los dos extremos implicados en la comunicación pueda enviar y/o recibir datos, tenemos la posibilidad de que cualquiera de los dos extremos finalice la comunicación (enviando una señal ***FIN***) hacia su sentido una vez finalizado el proceso de enviar/recibir datos. Esto nos obliga a realizar dos medias finalizaciones (*half-close*) de conexión, una hacia cada sentido.

TCP proporciona la capacidad de que cualquiera de los dos extremos de la conexión pueda finalizar su salida (*output*) de datos permitiéndole todavía recibir datos del otro extremo. Esta capacidad se denomina ***half-close***.

En el caso de finalizar sólo un sentido de la comunicación (de cliente a servidor por ejemplo), el cliente puede seguir recibiendo datos del servidor enviando únicamente reconocimiento de datos (acknowledge, ***ACK***). De esta forma cuando el servidor envíe una señal de ***FIN*** la conexión TCP finalizará totalmente.

El envío se una señal ***FIN*** tan solo indica que no se producirá mas intercambio de datos en ese sentido. De esta forma es posible que en una conexión TCP se continúen enviando datos después de recibir una señal ***FIN***. Esta posibilidad es muy poco aprovechada en las aplicaciones que utilizan TCP actualmente.

En un proceso de *half-close* podemos diferenciar de forma clara un extremo activo (*active close*) y un extremo pasivo (*passive close*).

El extremo activo es el que envía la señal de FIN para finalizar ese sentido de la comunicación, mientras que el extremo pasivo se limita a aceptar la petición y enviar un reconocimiento (*acknowledge*) de final.

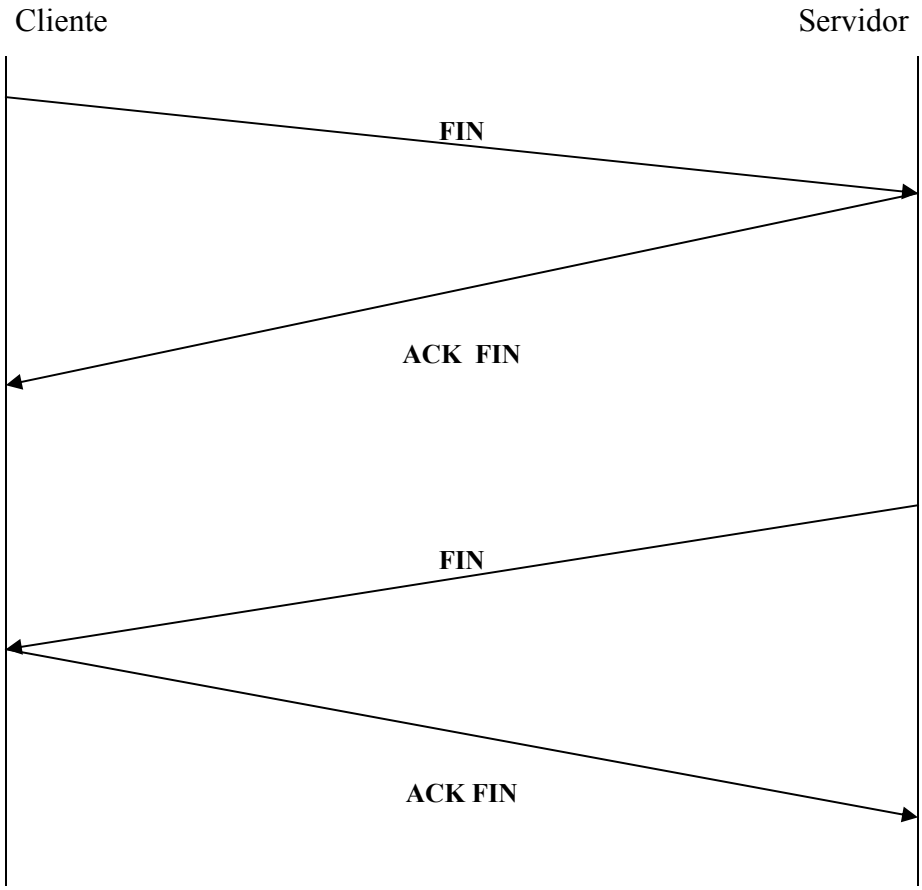

FIG. 1-15: Finalización de una conexión TCP.

De esta forma tenemos que para finalizar una conexión TCP debemos finalizarla en cada uno de los dos sentidos. Esto obliga a que alternativamente el cliente y el servidor adopten los papeles activo y pasivo en un proceso que consta de 4 fases:

1. (El cliente adopta el papel activo) El cliente decide finalizar la comunicación en su sentido. Enviando al servidor una señal de finalización (***FIN)*** con un número de secuencia.

2. (El servidor adopta el papel pasivo) El servidor recibe esta señal y responde con un reconocimiento (*acknowledge, ACK*) de señal. Enviando el número de secuencia recibido más uno.

   Cabe señalar que la finalización (FIN) al igual que la señal de petición de conexión (***SYN***) consume un número de secuencia. Finalización de la primera *half-close*.

3. (El servidor adopta un papel activo) El servidor decide finalizar la conexión en su sentido y envía una señal de finalización (***FIN***) de conexión al cliente.

4. (El cliente adopta un papel pasivo) El cliente acepta la petición de finalizar la conexión respondiendo con un ***ACK*** y enviando el número de secuencia recibido mas uno. Finalización de la segunda *half-close*. Finalización de la conexión TCP.

1.6 Encaminamiento de datagramas

El proceso de rutado o encaminamiento (*routing*) hace referencia a cómo los distintos datagramas IP enviados circulan y seleccionan el camino hacia su destino.

Debido a la construcción de Internet para que fuera tolerante a fallos, estos caminos no son fijos o estáticos, sino que existen diferentes vías que se actualizan dinámicamente para que un paquete alcance su destino sin necesidad de usar siempre la misma ruta. De esta forma, tenemos que cada paquete se encamina de forma independiente hacia su destino.

Los ordenadores encargados de recibir y distribuir los datagramas hasta su siguiente paso hacia el destino se denominan *routers*. Estos routers se encargan de leer la dirección destino del paquete y seleccionar su vía de salida.

En la figura 1-16 podemos ver el proceso de encaminamiento, dónde un router recibe un datagrama IP y lo conduce hasta el siguiente router.

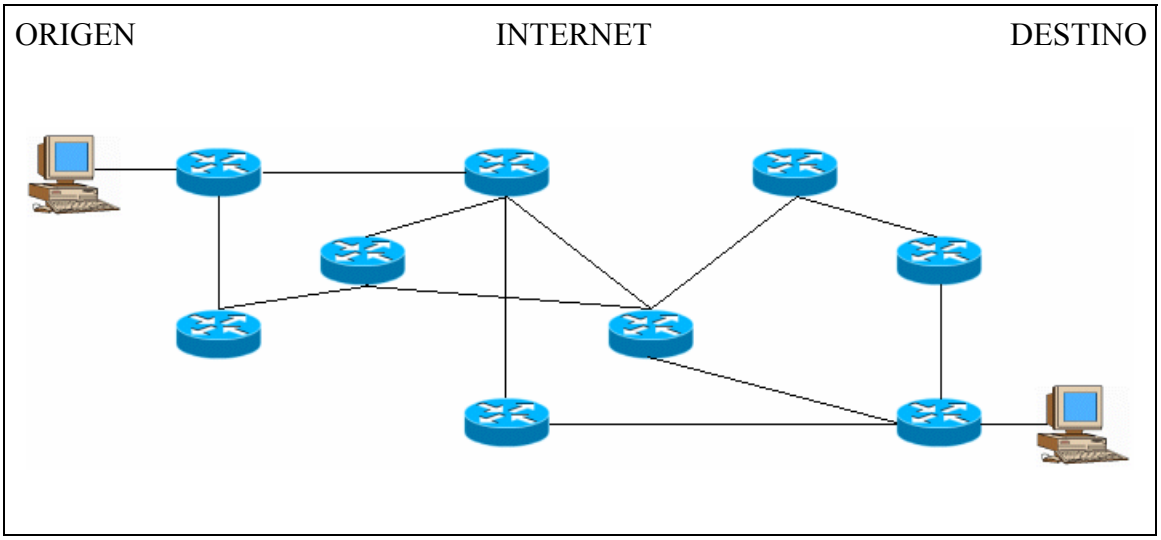

FIG. 1-16: Rutado de paquetes en Internet.

Podemos ver pues que esta comunicación se produce mediante saltos unitarios (*hops*) de router a router hasta que alcanza su destino o el paquete es descartado (ya sea porque no exista una ruta al destino de la información, o porque se ha consumido el tiempo de vida del datagrama).

Una vez que el datagrama llega hasta el router final, este se propaga por la red local hasta su destino. A diferencia que de lo que ocurre en Internet, en las redes locales (LAN) el direccionamiento de los ordenadores conectados se produce mediante una dirección hardware única denominada dirección **MAC** (*Media Access Control*) [WWW25][WWW26][WWW32].

La dirección MAC es única para cada dispositivo de red y viene serigrafiada en el hardware de la tarjeta, consiste en un número de 48 bits que indican el fabricante y su número de serie (ver figura 1-17).

Esta dirección suele expresarse como 12 dígitos hexadecimales (0-F) que se dividen en 24 bits (6 dígitos hexadecimales) para el número de organización OUI (*Organization Unique Identifier*) y 24 bits para la dirección de número de serie.

Una vez que el router final recibe un paquete IP para un ordenador que está conectado en su red local (LAN), realiza una petición a toda la red local para que se identifique el ordenador destinatario y así poder enviarle el datagrama IP.

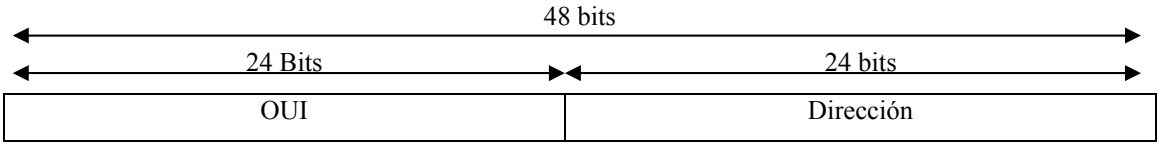

FIG. 1-17: Dirección MAC.

Para facilitar este proceso, existen dos protocolos para la obtención de la dirección MAC de una dirección IP dada (***ARP***) y otro para la obtención de la dirección IP de una dirección MAC (***RARP***) [WWW12][WWW13][RFC903].

- **ARP** (*Address Resolution Protocol*) [WWW27]: Obtención de la dirección MAC de una dirección IP. Al recibir un paquete IP para X.Y.Z.T, el router pregunta cual es la dirección MAC de esta dirección IP para enviarla por la red local.

- **RARP** (*Reverse ARP*) [WWW28]: Obtención de la dirección IP correspondiente a una dirección MAC dada. Usado por ejemplo en las máquinas que usan DCHP para la obtención de una dirección IP automática.

El uso de los protocolos ARP y RARP propaga mensajes a todos los ordenadores conectados en la red local, ya que el destinatario debe identificarse de entre todos los posibles candidatos. Los ordenadores que no responden ignoran estas peticiones.

En el protocolo IP también existe una forma de identificar la red a la que pertenece la dirección IP, para ello simplemente debemos sustituir los bits de la máscara de red por ceros (ver capítulo de redes IP).

Análogamente, IP también tiene un sistema de radiodifusión (*broadcast*) [RFC919] que permite la comunicación simultánea con **todos** los ordenadores de la misma red, simplemente debemos sustituir los bits de la máscara de red por unos.

1.7 Resumen

En este capítulo se ha realizado una presentación de la familia de protocolos TCP/IP versión 4 que abarca tanto sus definiciones formales como sus aspectos funcionales.

IP (*Internet Protocol*), que es un protocolo **no fiable** y **no orientado a conexión** que se encarga del transporte de los datos por Internet.

UDP (*User Datagram Protocol*), protocolo **simple** y **orientado a datagrama** que se encarga de enviar datagramas usando los servicios del protocolo IP.

ICMP (*Internet Control Message Protocol*), protocolo encapsulado en datagramas IP que se encarga del control del flujo de la comunicación y de los errores que puedan ocurrir.

TCP (*Transmission Control Protocol*) que es un protocolo **fiable**, **orientado a conexión** y **orientado a byte** que utiliza los servicios del protocolo IP. Este protocolo establece una conexión previa con el destino (mediante el *three way handshake*) que le permite regular el flujo de bytes enviados y su correcta recepción, retransmitiendo los bytes recibidos incorrectamente o perdidos. Finalmente realiza una fase de desconexión del receptor (mediante el sistema de *half close*) antes de dar por concluida la comunicación.

También se ha comentado el proceso de encaminamiento o rutado (routing) de datagramas por Internet así como su tránsito final hacia la red local (LAN) dónde alcanza al ordenador de destino mediante el uso de ARP y RARP.