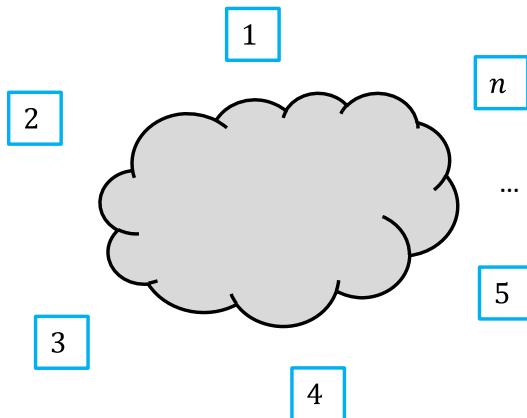


## 6.1. REDES DE COMPUTADORES: DEFINICIÓN Y TIPOS

Una **red de computadores** es un sistema capaz de interconectar un conjunto de computadores geográficamente dispersos. El problema que resuelve se ilustra en la **Figura 6.1**, donde se ha supuesto que  $n$  es el número de computadores a interconectar. La solución debe posibilitar que cualquier pareja de ellos puedan entablar comunicación.



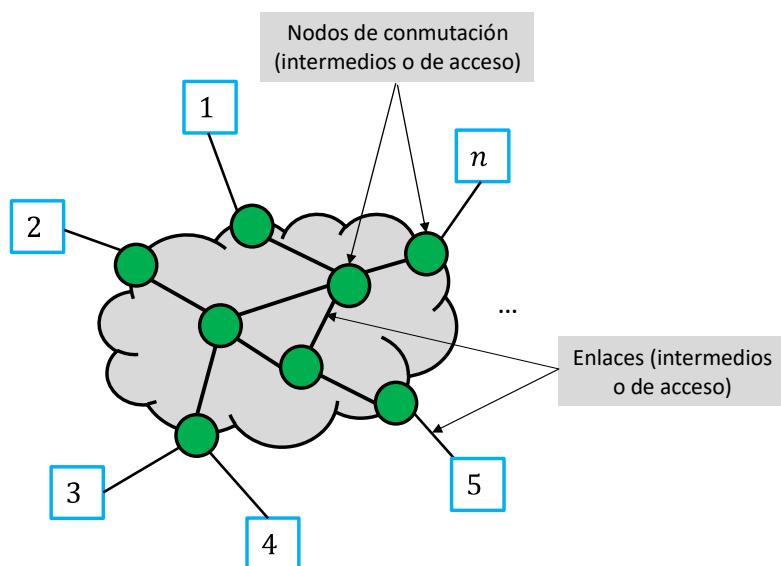
**Figura 6.1.** Una red de computadores va a ser una solución compleja a un problema de formulación muy sencilla.

Una forma de garantizar la conectividad de cada computador con el resto consiste en enlazar “todos con todos”, es decir, establecer un enlace bidireccional entre cada posible pareja. En tal caso, el número de enlaces resultante es  $\frac{n(n-1)}{2} \sim n^2$ , lo que indica que esta solución es inviable en términos de coste económico, no solo porque el número de computadores que se conectan a una red tiende a aumentar, sino también porque su dispersión geográfica, y por tanto la longitud requerida para tales enlaces, podría ser muy grande. Por otro lado, cada computador necesitaría  $n - 1$  interfaces, y cuando las tuviera todas ocupadas y se añadieran más computadores a la red, no habría forma de conectarse con ellos.

Desde finales del siglo XIX, la investigación ha ido proponiendo soluciones igualmente eficaces, pero mucho menos costosas. Así, en la actualidad existe una amplia variedad de estándares de redes y de redes comerciales, pero todas las opciones se ubican en una de estas dos grandes categorías: redes commutadas y redes de medio compartido. Históricamente, se desarrollaron primero las redes commutadas.

Como se ilustra en la **Figura 6.2**, una **red commutada** presenta una estructura mallada que resulta de la combinación de enlaces de transmisión y nodos de conmutación. Un **nodo de conmutación** es esencialmente un dispositivo de red cuya función es redistribuir los flujos de tráfico en función de los destinatarios de cada uno. El símil con una red de carreteras es evidente: los tramos de carretera equivalen a los enlaces de transmisión bidireccionales (suponiendo que dichos tramos constan de carriles en ambas direcciones), y las rotundas equivalen a los nodos de conmutación. Como veremos

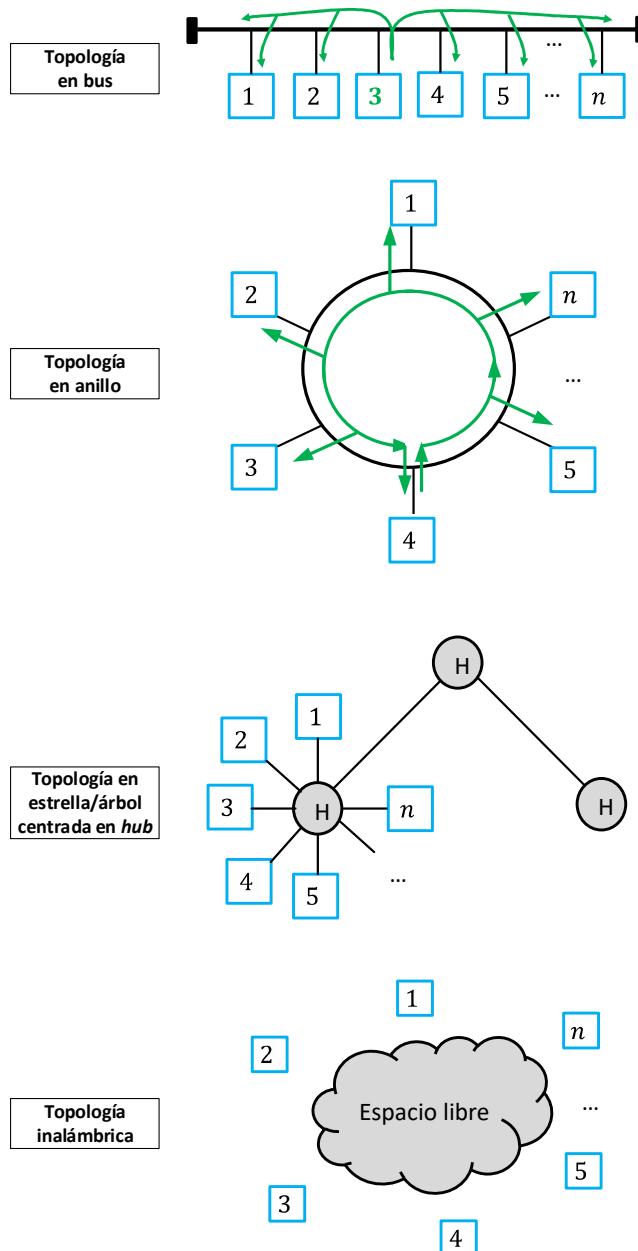
más adelante, hay dos tipos de nodos de commutación, que dan lugar a sendos tipos de redes commutadas. Así, podremos distinguir entre nodos y redes de commutación de circuitos, por un lado, y nodos y redes de commutación de paquetes, por otro. Aunque ambos tipos de nodos actúan como redistribuidores de tráfico, su tecnología es totalmente distinta, siendo la de commutación de paquetes la más utilizada. En la figura también se establece la distinción entre **nodos de acceso**, situados en la periferia de la red, y **nodos intermedios**, así como entre **enlaces de acceso** y **enlaces intermedios**. Los computadores se conectan a la red a través de los nodos de acceso y mediante los enlaces de acceso. Como podemos observar, en comparación con la solución “todos con todos”, la red commutada requiere menos enlaces de transmisión, además de menor longitud, y a cada computador le basta con tener una sola interfaz de conexión, independientemente del número de computadores restantes.



**Figura 6.2.** Las redes commutadas presentan una estructura en forma de grafo, en el que los arcos son los enlaces de transmisión, y los vértices los nodos de commutación.

Otra estrategia de interconexión la constituyen las redes de medio compartido. En una **red de medio compartido**, todos los computadores se conectan al mismo medio de transmisión, de forma que la señal enviada por uno de ellos llega a todos los demás y, en particular, al destinatario deseado. Se trata de una solución conceptualmente muy simple, que además elimina la necesidad de implementar una función de encaminamiento, como ocurre con las redes commutadas: en éstas, el correcto funcionamiento de la red exige que los nodos sean capaces de determinar una ruta para los datos correspondientes a cada pareja de computadores interconectados, mientras que en una red de medio compartido, este problema queda inmediatamente solucionado por el simple hecho de que el medio de transmisión es común a todos ellos. No obstante, en las redes de medio compartido surge un problema que no se daba en las commutadas: si dos o más computadores transmiten al mismo tiempo, sus señales se superponen, de forma que los destinatarios reciben la señal superpuesta o señal suma, de la que no es posible extraer las señales individuales por

separado. Este fenómeno es lo que se denomina una **colisión**. Para abordar ese nuevo problema, existen dos grandes mecanismos de compartición de canal: **mecanismos de prevención de colisiones**, consistentes en eliminar total o casi totalmente el riesgo de colisión, y **mecanismos de reacción a colisiones**, cuya estrategia es dejar que las colisiones se produzcan, pero contrarrestándolas de forma eficaz y eficiente.



**Figura 6.3.** Topologías de redes de medio compartido.

La **Figura 6.3** muestra las cuatro topologías básicas sobre medio compartido. Aunque físicamente son distintas, desde el punto de vista lógico son equivalentes, pues en todas ellas la señal generada

por un dispositivo llega a todos los demás. Históricamente, la primera versión de red sobre medio compartido presentaba una **topología en bus**, consistente en que todos los computadores se conectaban a un mismo bus de comunicaciones, típicamente un tramo de cable coaxial. Esta topología se encuentra hoy en día totalmente obsoleta.

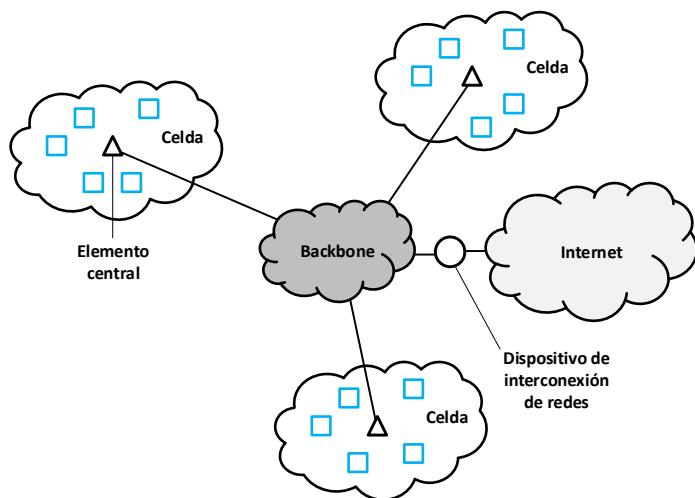
Más tarde se desarrolló la **topología en anillo**, en la que, a diferencia del caso anterior, los computadores se conectan a un bus cerrado. La señal circula solamente en un sentido del anillo, y es eliminada del mismo por el mismo computador que la ha emitido, después de dar una vuelta entera y visitar el resto de computadores. Los estándares en esta topología hacían uso del par trenzado o la fibra óptica. Actualmente el primero está obsoleto y el segundo en declive.

La **topología en estrella centrada en hub** introduce un nuevo dispositivo de red, el **hub**, que no es más que un repetidor multipuerto, es decir, un dispositivo que repite la señal que le llega por un puerto de entrada hacia todos los demás puertos de salida (recuérdese que los enlaces son bidireccionales, y por tanto cada puerto de un dispositivo final o intermedio es de entrada y salida). Así pues, el **hub** no procesa la señal que recibe, simplemente la repite. La **topología en árbol centrada en hub** no es más que una extensión de la topología anterior para el caso en que se utilizan múltiples **hubs**. En tal caso los **hubs** forman un grafo en forma de árbol, evitando así que la señal realice ciclos indefinidamente. La topología en estrella/árbol centrada en **hub** hace uso del par trenzado como medio de transmisión, pero actualmente se encuentra en declive pues el **hub** ha sido substituido por un conmutador. El **conmutador** es un dispositivo de red más “inteligente” que el **hub**, pues a diferencia de éste, procesa la señal y la reenvía sólo por el puerto de salida que la conduce al destinatario. Hablamos en este caso de una **topología en estrella/árbol centrada en conmutador**, pero ya no es una red de medio compartido, sino un caso particular de red conmutada. De hecho, es el resultado de la migración de la tecnología de conmutación (de paquetes) al ámbito de las redes centradas en **hub**.

Finalmente, una solución tecnológicamente de plena actualidad y en continuo desarrollo es la **topología inalámbrica**, pues permite la movilidad de los dispositivos interconectados. En este caso, cada uno dispone de una antena isotrópica, de forma que la señal electromagnética emitida es recibida por todos los que se encuentran a su alcance. Esta topología es la base de todas las redes inalámbricas comerciales, pero representa solamente el primer segmento de las mismas. En efecto, la **Figura 6.4** muestra que las redes inalámbricas constan generalmente de dos segmentos o tramos, uno inalámbrico en forma de **celda** alrededor de un elemento central (punto de acceso en las redes wifi, estación base en las redes celulares de telefonía móvil o repetidor o puerta de enlace en las redes WiMAX), y otro, el **backbone**, que suele ser cableado y permite la conexión entre dispositivos pertenecientes a diferentes celdas o entre dispositivos de la red inalámbrica y el exterior. ¿Dónde están, pues, las diferencias entre unas redes inalámbricas y otras? En general, en aspectos tales como la velocidad de transmisión, la banda de frecuencias de trabajo (con/sin licencia), el alcance, el mecanismo de compartición de canal en el tramo inalámbrico, el mecanismo de conmutación en el **backbone**, etc.

Es importante observar que, si bien en las redes de medio compartido no hay nodos de conmutación, y en particular no hay nodos de conmutación de paquetes, esto no significa que no podamos hablar de paquetes en dichas redes. Como veremos más adelante, el paquete es la unidad básica de información que circula por una red, tanto si es de conmutación de paquetes, como si es de medio compartido. Es simplemente el resultado de un proceso de fragmentación de la información, que veremos detalladamente al estudiar los modelos de capas.

La **Tabla 6.1** muestra las características de los diversos tipos de redes introducidos, así como los estándares más representativos de cada uno. Uno de los elementos de caracterización hace referencia al ámbito geográfico. Desde este punto de vista, podemos distinguir las siguientes categorías, en orden creciente de su cobertura geográfica:



**Figura 6.4.** Estructura típica de las redes inalámbricas. El *backbone* interconecta los elementos centrales entre sí y con el exterior (Internet pública).

- **Red de área personal o PAN (personal area network).** Una red de área personal permite interconectar dispositivos del ámbito personal, como por ejemplo un computador con sus periféricos. Las distancias entre los dispositivos que se comunican es del orden de 1 metro y la cobertura de estas redes se sitúa, por tanto, alrededor del metro cuadrado.
- **Red de área local o LAN (local area network).** Su cobertura alcanza un edificio o, a lo sumo, un conjunto de edificios o un campus universitario. La distancia entre los dispositivos interconectados puede variar entre 10 m y 1 Km grosso modo.
- **Red de área metropolitana o MAN (metropolitan area network).** Como su nombre indica, cubre un área metropolitana (ciudad), con un radio medio de unos 10 Km.
- **Red de área amplia o WAN (wide area network).** Cubre una extensa región geográfica, arbitrariamente grande, desde un país hasta un continente (distancias entre cientos y miles de kilómetros). En el límite superior de las redes de área amplia se encuentran las de **escala planetaria**. El ejemplo por excelencia es la red **Internet**, aunque, como veremos, se trata más

bien de una red de redes<sup>1</sup>. Puesto que las redes telefónicas de todos los países están interconectadas y hacen uso de la misma tecnología (comutación de circuitos), la red telefónica mundial puede verse también como una red de escala planetaria. Finalmente, en un futuro quizás no muy lejano existirá también una [Internet interplanetaria](#).

**Tabla 6.1.** Tipos de redes, características principales y ejemplos y/o estándares más representativos. En el caso de las redes inalámbricas, las diferencias de escala se reflejan en el tamaño de la celda básica y, proporcionalmente, en el tamaño de la red completa.

Tipo de red	Mecanismo/Topología	Medio de transmisión	Escala	Ejemplos/Estándares	Estado
Commutada	Comutación de circuitos	Guiado, radioenlace punto a punto	WAN	PSTN <sup>2</sup> , ISDN <sup>3</sup>	Vigente
	Comutación de paquetes	Guiado, radioenlace punto a punto	WAN	frame relay	En declive
		WAN	ATM <sup>4</sup>	Vigente	
		Planetaria	Internet	Vigente	
	Par trenzado, fibra óptica	LAN, MAN, WAN	Ethernet (sobre topología en estrella/árbol centrada en conmutador)	Vigente	
Medio compartido	Topología en bus	Coaxial	LAN	Ethernet, token-bus	Obsoleto
	Topología en anillo	Par trenzado	LAN	Token-ring	Obsoleto
		Fibra óptica	MAN	FDDI <sup>5</sup>	En declive
	Topología en estrella/árbol centrada en hub	Par trenzado	LAN	Ethernet	En declive
	Topología inalámbrica	Espacio libre (primer tramo)	PAN	Bluetooth	Vigente
			LAN	Wifi	Vigente
			MAN	WiMAX	Vigente
			WAN	Redes celulares (1G – 3G)	Obsoleto
				Redes celulares (4G – 5G)	Vigente

A parte de la cobertura, hay diferencias de otra índole entre las categorías mencionadas. Por ejemplo, las redes de área local suelen ser propiedad de la misma entidad propietaria de los dispositivos interconectados. Esto significa que es dicha entidad la que asume el coste económico de adquisición y mantenimiento de la red, así como la responsabilidad en la gestión de la misma.

<sup>1</sup> Cualquier interconexión entre redes de distinta naturaleza se denomina [internet](#) (en minúscula). La Internet (en mayúscula) es, con diferencia, la internet más popular y la de mayor cobertura.

<sup>2</sup> PSTN es el acrónimo de *public switched telephone network* (red telefónica comutada o RTC, típicamente una en cada país).

<sup>3</sup> ISDN es el acrónimo de *integrated services digital network* (red digital de servicios integrados o RDSI, surgida como extensión de la RTC digitalizada para transportar datos además de voz).

<sup>4</sup> ATM es el acrónimo de *asynchronous transfer mode*.

<sup>5</sup> FDDI es el acrónimo de *fiber distributed data interface*.

Por el contrario, en el caso de las redes de área amplia, los usuarios suelen ser dueños únicamente de los dispositivos finales conectados a la red, mientras que los recursos de la misma suelen ser propiedad de un operador de telecomunicaciones. Por tanto, en este caso el usuario queda liberado de las tareas de gestión y mantenimiento de la red, y en términos económicos el coste asumido se “reduce” a un contrato de alquiler con el operador por los servicios de conexión a la red.

La **Tabla 6.1** revela que una de las tecnologías de red más populares es la constituida por las redes Ethernet, especialmente en su variante estrella/árbol centrada en conmutador. Como podemos observar, esta variante es utilizable en todos los ámbitos (LAN, MAN, WAN), lo que no hace más que confirmar que la escala geográfica ha dejado de ser el primer criterio de clasificación a la hora de catalogar las redes, para convertirse en una característica más.

## 6.2. TERMINOLOGÍA DE REDES

---

Para completar este capítulo introductorio, conviene revisar algunos términos propios del ámbito de las redes de computadores. En comunicación de datos, el escenario de referencia era un transmisor conectado a un receptor mediante un tramo de medio de transmisión, que llamábamos enlace. En realidad, éste era solamente un tipo de enlace, el **enlace punto a punto**, denominado así porque interconecta únicamente dos dispositivos (en general, dotados de transmisor y receptor, pues las comunicaciones suelen ser bidireccionales). El otro tipo de enlace es el llamado **enlace broadcast** (o de difusión), que corresponde al caso en que el medio de transmisión es compartido por más de dos dispositivos.

Por otro lado, la comunicación (o conexión) a través de un enlace, tanto si es punto a punto como si es *broadcast*, puede ser de tres tipos:

- **Símplex**: La comunicación es unidireccional, es decir, sólo puede tener lugar en una dirección, ya sea entre un transmisor y un receptor o entre un transmisor y múltiples receptores.
- **Half-dúplex**: La comunicación es bidireccional, pero no admite simultaneidad, es decir, puede tener lugar en las dos direcciones, pero nunca a la vez.
- **Full-dúplex**: La comunicación es bidireccional y puede darse en las dos direcciones simultáneamente. Es el caso más frecuente, y el que supondremos salvo que se especifique lo contrario.

Estos términos fueron acuñados por el ANSI. La UIT-T, por su parte, sólo diferencia entre enlaces **símplex** y **dúplex**, correspondientes, respectivamente, a los enlaces *half-dúplex* y *full-dúplex* de la terminología ANSI. Salvo que se especifique lo contrario, adoptaremos las definiciones ANSI.

De acuerdo con las tipologías establecidas, podemos afirmar que cualquier conexión a través de una red conmutada se sustenta en una sucesión de enlaces punto a punto, generalmente *full-dúplex*. Recordemos que tales enlaces podían ser de acceso o intermedios. Esta diferenciación es

significativa, ya que mientras los enlaces de acceso son **enlaces dedicados**, pues su ancho de banda<sup>6</sup> está disponible para un solo usuario, los enlaces intermedios son compartidos por múltiples conexiones, es decir, múltiples usuarios, los cuales se reparten el ancho de banda total. Se dice que son **enlaces multiplexados**. Por ejemplo, si el ancho de banda de un enlace intermedio, generalmente mayor que el de un enlace de acceso, es de 10 Gbps, las múltiples conexiones se reparten 10 Gbps en una dirección del enlace y 10 Gbps en la dirección opuesta, es decir, se reparten una **velocidad de transferencia** total de 20 Gbps. En cambio, si un enlace de acceso es de 100 Mbps, el único usuario del mismo dispone de una velocidad de transferencia de 200 Mbps.

Por el contrario, en una red de medio compartido, el enlace es *broadcast* y sólo permite la comunicación *half-dúplex*. Si la comunicación *full-dúplex* fuera posible, significaría que un dispositivo cualquiera podría estar transmitiendo y recibiendo a la vez, pero esto implicaría que hay dos dispositivos transmitiendo simultáneamente y, por tanto, provocando una colisión. Un ejemplo sencillo puede terminar de aclararlo. Supongamos que tres dispositivos, A, B, y C, están conectados a través de una red de medio compartido, y A transmite a B a la vez que recibe de C; entonces B recibe las señales de A y C superpuestas, es decir, colisionadas. El reparto de ancho de banda en las redes de medio compartido tiene también características específicas. Si se trata de una red de medio compartido con mecanismo de resolución de colisiones reactivo, una parte del ancho de banda ofrecido por el enlace *broadcast* se dedica precisamente a dicho mecanismo, quedando el resto disponible para el conjunto de usuarios. Esta parte disponible para los usuarios nos la proporciona una medida de eficiencia. Por ejemplo, si el enlace ofrece un ancho de banda de 100 Mbps, y la eficiencia del mecanismo de resolución de colisiones es del 85%, el ancho de banda disponible para el conjunto de usuarios es de 85 Mbps, y la velocidad de transferencia disponible es también de 85 Mbps, pues el enlace es *half-dúplex*. Si el mecanismo de resolución de colisiones hubiera sido preventivo, los usuarios se repartirían directamente los 100 Mbps, y ésta sería igualmente la velocidad de transferencia.

Por último, otra cuestión terminológica es la referida al modo de direccionamiento, según el cual podemos distinguir los siguientes tres tipos de conexiones según la población de destinatarios:

- **Unicast**: La conexión iniciada por un dispositivo tiene un solo destinatario.
- **Multicast**: La conexión se establece con un grupo específico de destinatarios.
- **Broadcast**: La conexión se dirige a todos los destinatarios posibles.

Estos tres modos son aplicables a cualquier tipo de red, conmutada o de medio compartido. Con respecto a este último caso, no debe confundirse el término *broadcast* referido al tipo de enlace con dicho término referido al modo de direccionamiento. El primero es un *broadcast* físico, mientras que el segundo es un *broadcast* lógico.

---

<sup>6</sup> Aunque el término “ancho de banda” se refiere estrictamente a la anchura de la respuesta frecuencial del medio de transmisión utilizado, dada su relación directa con la velocidad de transmisión (fórmula de Shannon), en la práctica se utiliza como sinónimo de ésta. Desde esta perspectiva, el ancho de banda de un medio de transmisión es su **velocidad nominal**.

## 7.1. DESARROLLO DE UNA ARQUITECTURA DE CAPAS GENÉRICA

---

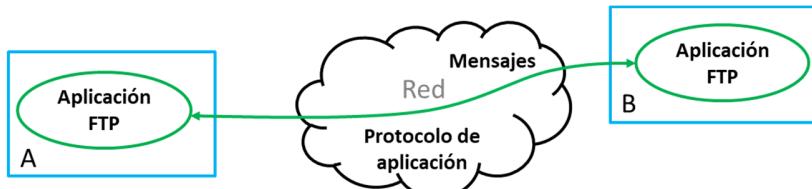
La **arquitectura de capas** de las redes de computadores es el resultado de la descomposición del problema de interconectar dos computadores en varios sub-problemas a diferentes niveles. Lógicamente, no existe una forma única de llevar a cabo esta descomposición, y de hecho son múltiples los modelos de capas que han sido desarrollados tanto para redes propietarias como públicas. En la actualidad, los dos modelos más importantes son OSI (*Open Systems Interconnection*) y TCP/IP (*Transmission Control Protocol/Internet Protocol*). El primero fue desarrollado por el organismo de estandarización ISO (*International Organization for Standardization*), mientras que el segundo fue formalizado por el IETF (*Internet Engineering Task Force*). El modelo OSI es un estándar oficial (*de jure*), pero tan solo vigente como referencia teórica o conceptual, pues no ha llegado a implantarse, salvo de forma parcial en casos muy concretos; en cambio, el modelo TCP/IP es el estándar de uso (*de facto*), pues todas las comunicaciones a través de Internet encajan con este modelo. Por lo tanto, el interés de este capítulo se centra sobre todo en la arquitectura TCP/IP, aunque llegaremos a ella desarrollando un modelo de capas genérico sobre la base de una aplicación concreta, la transferencia de ficheros.

Supongamos que dos computadores, A y B, están conectados a través de una red, y que A quiere enviar un fichero a B. Esto implica que una aplicación de transferencia de ficheros residente en A debe establecer una conversación con otra aplicación de transferencia de ficheros residente en B. Utilicemos para ambas aplicaciones la denominación FTP (*file transfer process*) en un sentido general, siendo FTP-A la que reside en A y FTP-B la que reside en B. La conversación establecida conlleva el intercambio de dos tipos de mensajes:

- Mensajes de control: peticiones, respuestas, confirmaciones, etc.
- Mensaje de datos: esencialmente sería el fichero, pero acompañado de información de control, como las direcciones origen y destino y las etiquetas que identifican la aplicación dentro de cada computador. Esta información de control también debe estar presente en los mensajes de control. Durante el envío del fichero, las direcciones origen y destino son las que corresponden, respectivamente, a los computadores A y B. Estas direcciones son necesarias para que la red pueda realizar su función de encaminamiento. También durante el envío del fichero, la etiqueta origen es la etiqueta de la aplicación en A y la de destino es la etiqueta de la aplicación en B. Estas etiquetas son necesarias para que el fichero se entregue a la aplicación de transferencia de ficheros de B, de entre las múltiples aplicaciones que puedan estar ejecutándose en dicho computador, y a la vez para que la aplicación de transferencia de ficheros en B reconozca que el envío fue generado por la aplicación de transferencia de ficheros en A. Posibilitan, pues, el encaminamiento interno dentro de cada computador.

El intercambio de mensajes entre las dos aplicaciones de transferencia de ficheros se realiza de acuerdo con unas reglas, es decir, un protocolo. Formalmente, el **protocolo** es el conjunto de reglas que gobiernan el intercambio de mensajes entre dos entidades (diálogo), así como el formato (sintaxis) y el significado (semántica) de dichos mensajes. En definitiva, el protocolo es el idioma común que han de manejar las dos entidades (aplicaciones de transferencia de ficheros en nuestro

ejemplo) para que se entiendan. Puesto que ambas entidades son aplicaciones, ese idioma común es, de manera específica, un **protocolo de aplicación**, y el **mensaje** es precisamente la unidad básica de información intercambiada a nivel de protocolo de aplicación. La **Figura 7.1** ilustra estos conceptos. Es importante observar que en ningún momento se ha impuesto que las dos aplicaciones tengan que ser producto de un mismo fabricante o desarrollador de software, que tengan que estar escritas en un mismo lenguaje, o que tengan que ejecutarse en el mismo entorno o sistema operativo. Lo único importante es que “hablen el mismo idioma”.



**Figura 7.1.** Las aplicaciones dialogan intercambiándose mensajes según las reglas de un protocolo de aplicación. En este caso, las aplicaciones son de transferencia de ficheros.

Por otro lado, también es importante que la transferencia del fichero se realice de forma eficiente y fiable. Estas condiciones conllevan la implementación de nuevas funcionalidades en el proceso de comunicación:

- Dado que, por un lado, la red no es infalible e introduce errores de transmisión, y, por otro, el tamaño del dato a transferir (fichero en nuestro ejemplo) puede ser muy grande, resulta más eficiente fragmentarlo y transferirlo por partes. De lo contrario, si fuera enviado como un todo, cualquier error de bit conllevaría la retransmisión de ese todo, las veces que fuera necesario, y el tiempo invertido en ello sería inaceptable. Con la fragmentación, en cambio, sólo hay que retransmitir el fragmento o los fragmentos erróneos, pudiendo asegurar en el destinatario los que ya han sido recibidos correctamente. La contrapartida es que a cada fragmento hay que añadirle la información de control que era necesaria para el fichero completo (direcciones y etiquetas). Así, el conjunto formado por esa información de control y el fragmento del dato (fichero) se denomina **segmento**, y la función encargada de crear los segmentos a partir del mensaje original es la **segmentación**.
- Además de eficiente, la transmisión tiene que ser fiable, lo cual implica garantizar que el destinatario reciba una copia exacta del fichero original. Para ello, es necesario que todos los segmentos se reciban sin pérdidas ni duplicidades, y en el orden correcto, objetivos que se alcanzan con la simple enumeración de los mismos. En esto consiste la función de **secuenciación**. No obstante, esta funcionalidad es tan solo necesaria, pero no suficiente, pues los segmentos pueden llegar sin pérdidas ni duplicidades y en el orden correcto, pero erróneos. La función que se encarga precisamente de gestionar la ocurrencia de esos posibles errores de transmisión es el **control de errores**. Finalmente, también puede ocurrir que la entidad destinataria reciba los segmentos a una velocidad mayor a la velocidad con que puede procesarlos. Un *buffer* de entrada en dicha entidad puede acomodar parcialmente esa diferencia de velocidades, pero no es la solución definitiva, pues este *buffer* tiene una capacidad

limitada. Si, cuando se alcanza dicha capacidad, siguen llegando segmentos, estos se pierden. La función de **control de flujo** es la encargada de gestionar este problema, y consiste básicamente en que la entidad receptora de los segmentos va notificando a la entidad emisora la cantidad de segmentos para los que dispone de espacio en el *buffer*. Es pues, la entidad que recibe la que regula el flujo de la entidad que transmite. Tanto la función de control de flujo como la de control de errores se apoyan en la enumeración de los segmentos introducida por la función de secuenciación. Además, la función de control de errores requiere la inserción de un código de detección o corrección de errores, calculado mediante un algoritmo a partir de los bits a proteger. Por tanto, números de secuencia y códigos de protección contra errores son nuevas informaciones de control que se añaden a las direcciones de los computadores y las etiquetas de las aplicaciones.

Resumiendo, segmentación, secuenciación, control de flujo y control de errores son funcionalidades que contribuyen a la transferencia eficiente y fiable del fichero. No obstante, eficiencia y fiabilidad son prestaciones exigibles a cualquier aplicación, no solamente a la transferencia de ficheros. Por consiguiente, en lugar de incluir esas funciones en el diseño de las aplicaciones, resulta mucho más práctico implementarlas en una entidad aparte, que denominamos **módulo de transporte**, al que puede invocar cualquier aplicación para transferir mensajes de forma eficiente y fiable. De este modo, se obtienen dos ventajas:

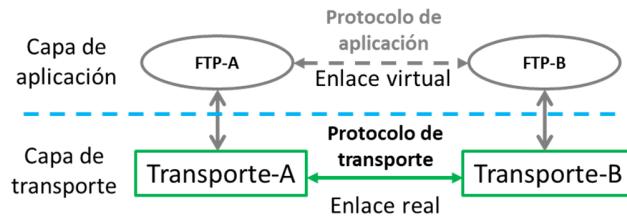
- Evitamos el replicado de estas funcionalidades en todas las aplicaciones.
- Liberamos a los desarrolladores de aplicaciones de la necesidad de conocer e implementar los mecanismos que permiten satisfacer las exigencias de eficiencia y fiabilidad.

La **Figura 7.2** muestra el nuevo escenario, en el que las dos aplicaciones ya no se comunican directamente a través de la red, sino a través de sendos módulos de transporte. Si FTP-A y FTP-B son, respectivamente, las aplicaciones de transferencia de ficheros residentes en los computadores A y B, y, análogamente, Transporte-A y Transporte-B son los módulos de transporte en A y B, el envío del fichero del computador A al computador B tiene lugar de la siguiente manera:

1. FTP-A envía el fichero a Transporte-A.
2. Transporte-A ejecuta las funciones de segmentación, secuenciación y control de errores, y envía los segmentos resultantes a Transporte-B, el cual ejerce el control de flujo y participa también en el control de errores.
3. Transporte-B, una vez ha recibido todos los segmentos correctamente, extrae de ellos los fragmentos de datos y los ensambla para reconstruir el fichero original y entregarlo a la aplicación FTP-B.

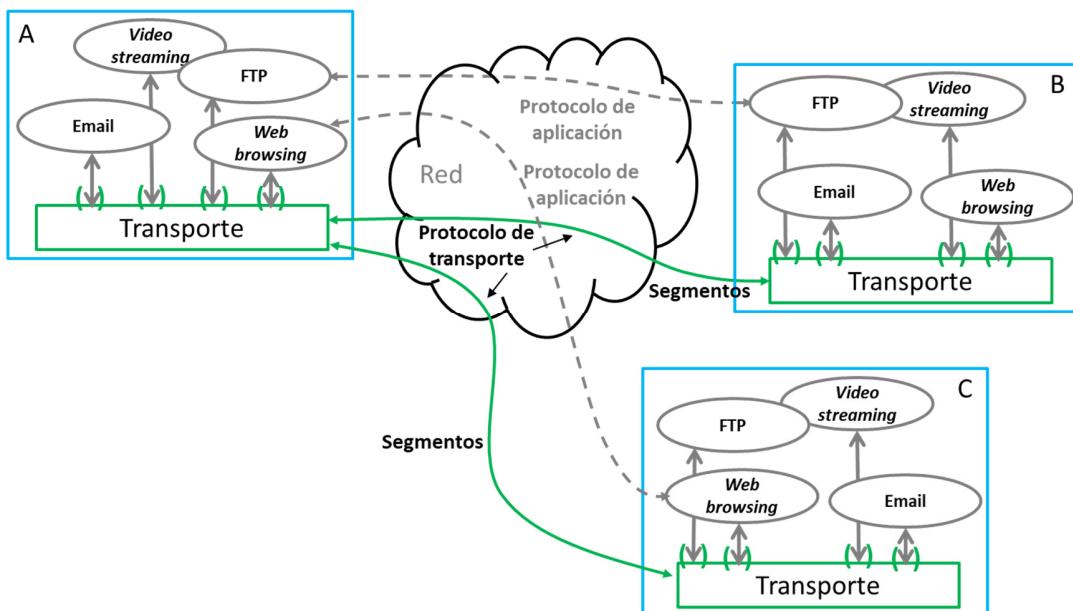
FTP-B recibe el fichero como si se lo hubiera enviado directamente FTP-A, pero a través de un enlace que ya no es real, sino virtual, puesto que el enlace real a través de la red tiene lugar entre las entidades de transporte. El resultado es, pues, una arquitectura de dos capas, una concerniente a las aplicaciones, o **capa de aplicación**, y la otra concerniente al transporte eficiente y fiable de los mensajes generados por las aplicaciones, o **capa de transporte**. Al igual que en la capa de aplicación,

los dos módulos de la capa de transporte tienen que dialogar según las reglas de un protocolo común, que ahora es un **protocolo de transporte**. Este protocolo gobierna el intercambio, la sintaxis y la semántica de las unidades de información básicas en la capa de transporte, es decir, los segmentos. Y como ya ocurría con los mensajes, los segmentos pueden ser de control o de datos.



**Figura 7.2.** Arquitectura de dos capas: capa de aplicación y capa de transporte.

La **Figura 7.3** muestra como cualquier computador puede tener múltiples sesiones de aplicación establecidas a través del único módulo de transporte. Hablamos entonces del multiplexado<sup>1</sup> de conexiones en la capa de transporte, lo cual es posible gracias, precisamente, al etiquetado de las aplicaciones. En dicha figura, cada etiqueta está representada como una puerta de acceso que conecta el módulo de transporte con una aplicación concreta, de forma que el módulo de transporte conoce siempre de qué aplicación provienen los mensajes (cuando transmite) o a qué aplicación entregar los mensajes (cuando recibe).



**Figura 7.3.** Multiplexado de conexiones a través de la capa de transporte.

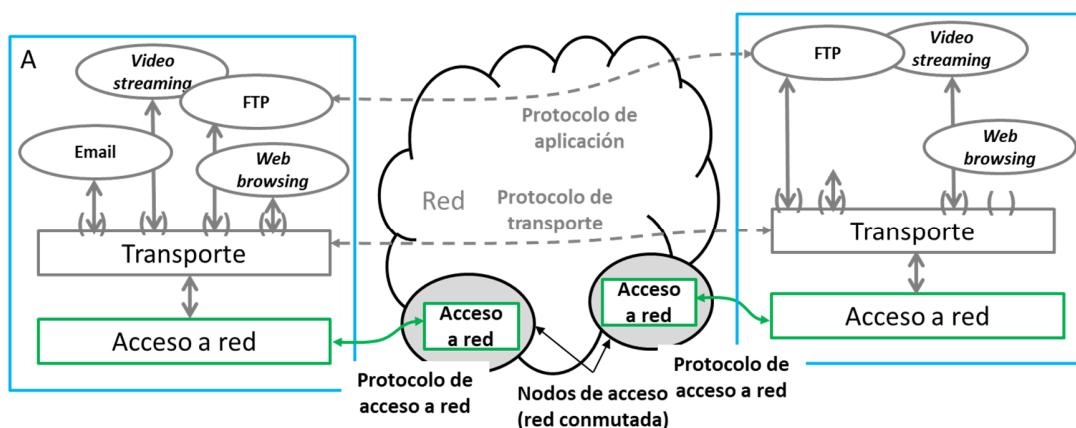
<sup>1</sup> El término **multiplexado** ya salió al referirnos a las múltiples conexiones encaminadas a través de un mismo enlace intermedio en una red conmutada. De hecho, es un término versátil, aplicable cuando varias conexiones comparten, sin conflicto o competición, un mismo recurso.

La **Figura 7.3** también pone de manifiesto que el segmento es la unidad básica de información en la capa de transporte, de la misma manera que el mensaje lo era en la capa de aplicación. Y también podemos distinguir los siguientes tipos de segmentos:

- Segmentos de control: peticiones, respuestas, confirmaciones, etc.
- Segmentos de datos: cada segmento de datos contiene un fragmento del dato original, juntamente con información de control (también incluida en los segmentos de control), que por el momento engloba las direcciones de los computadores, las etiquetas de las aplicaciones, el número de secuencia y el código de protección contra errores.

Hasta el momento hemos considerado la red que conecta los computadores A y B como si fuera un elemento único, reconocido directamente por ambos computadores. Sin embargo, existen muchas redes en el mercado, y cada una define sus propias reglas de acceso, es decir, su **protocolo de acceso a red** (o, simplemente, protocolo de red). Nuevamente se presenta la disyuntiva acerca de dónde implementar ese protocolo. Podríamos introducirlo como una funcionalidad más del módulo de transporte, pero resulta más ventajoso incluirlo en un módulo aparte, por las siguientes razones:

- Evitamos tener que diseñar un módulo de transporte para cada tipo de red, o un módulo de transporte más “grueso” para varios tipos de redes.
- Los requisitos de eficiencia y fiabilidad, de los que es responsable la capa de transporte, son exigibles independientemente de la red que conecte los dos extremos, por lo que no tiene mucho sentido fusionar ambos aspectos en un mismo diseño. Dicho de otro modo, hacerlo implicaría replicar innecesariamente las funciones de la capa de transporte.
- Liberamos a los desarrolladores de software de transporte de la necesidad de conocer las especificidades de las múltiples redes existentes.



**Figura 7.4.** Arquitectura de tres capas: capa de aplicación, capa de transporte y capa de acceso a red. Se ha supuesto que la red es comutada.

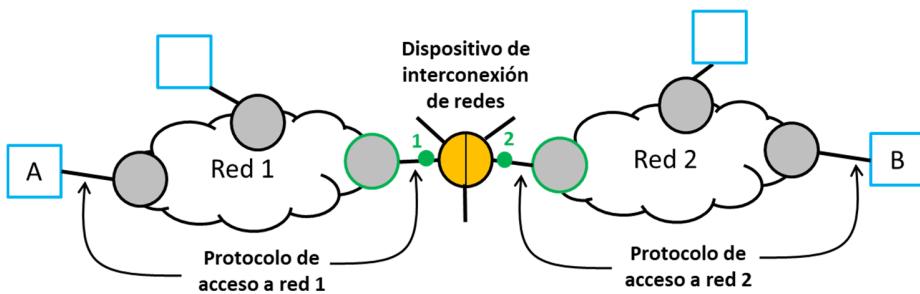
Con el nuevo módulo, que denominamos **módulo de acceso a red**, resulta la arquitectura de tres capas mostrada en la **Figura 7.4**. En ella, el enlace entre los módulos de transporte ha pasado a ser

virtual, pues el enlace real tiene lugar entre cada computador y su nodo de acceso a red (como ocurre realmente).

Esta figura también pone de manifiesto una diferencia significativa entre los protocolos de aplicación y transporte, por un lado, y el protocolo de acceso a red, por otro. Mientras los primeros gobiernan el diálogo extremo a extremo entre entidades residentes en los dispositivos finales (computadores A y B), el protocolo de acceso a red gobierna el diálogo entre entidades residentes en dispositivos conectados directamente (cada computador y su nodo de acceso). Se dice que los protocolos de aplicación y transporte son *end-to-end* o *host-to-host*, y que el protocolo de acceso a red es *host-to-network*.

De entre todas las informaciones de control que en la arquitectura de dos capas manejaba la capa de transporte (direcciones de los computadores, etiquetas de las aplicaciones, números de secuencia y códigos de protección contra errores), en la nueva arquitectura hay que transferir la gestión de las direcciones de los computadores a la capa de acceso a red, pues constituyen informaciones relevantes para la función de enrutamiento que debe encontrar un camino que conecte los nodos de acceso mostrados en la **Figura 7.4**. Serán, pues, las unidades básicas de información a nivel de capa de acceso a red las que contengan las direcciones de los computadores origen y destino, además de otras posibles informaciones de control.

Por otro lado, la **Figura 7.4** ignora el hecho de que, generalmente, las conexiones entre dispositivos finales requieren cruzar varias redes en lugar de sólo una. La **Figura 7.5** plantea este escenario para el caso de dos redes intermedias y distintas entre A y B, la Red 1 y la Red 2. Es obvio que la solución a la interconexión de estas dos redes no puede consistir en enlazar directamente los dos nodos de acceso señalados, puesto que manejan protocolos de acceso a red distintos. Debe existir un dispositivo intermedio capaz de realizar la “traducción”. En general, los dispositivos intermedios disponen de múltiples puertos de entrada-salida para interconectar múltiples redes entre sí. En el caso de la **Figura 7.5**, son los puertos 1 y 2 del dispositivo intermedio los que se conectan respectivamente a los nodos de acceso de Red 1 y Red 2. Por el puerto 1, el dispositivo intermedio utiliza el protocolo de acceso a red 1 (protocolo de acceso a red definido por Red 1), mientras que por el puerto 2 utiliza el protocolo de acceso a red 2 (protocolo de acceso a red definido por Red 2). Resumiendo, el puerto 1 equivale a un dispositivo más conectado a Red 1, y el puerto 2 a un dispositivo más conectado a Red 2.



**Figura 7.5.** Conexión entre A y B a través de redes distintas.

¿Cómo opera, sin embargo, la traducción que debe tener lugar en el dispositivo intermedio? Observemos primeramente que, a diferencia del caso en que los computadores A y B estaban conectados a la misma red, en el escenario de la **Figura 7.5** las direcciones de ambos computadores ya no tienen por qué obedecer al mismo formato, pues éste queda definido por el protocolo de acceso impuesto por cada red. Entonces, ¿cómo puede el computador A especificar el destinatario del fichero? Debe existir un nivel de direccionamiento global, por encima de las especificidades de cada red, que permita identificar cualquier dispositivo final independientemente de la red concreta a la que esté conectado directamente. En consecuencia, el nuevo escenario de interconexión a través de múltiples redes contempla dos niveles de direccionamiento:

- **Global.** Permite cruzar múltiples redes, al ser reconocido por los dispositivos de interconexión.
- **Local.** Permite acceder a la propia red.

Correspondientemente, una **dirección global** permite identificar únicamente un dispositivo a escala global o mundial, mientras que una **dirección local** permite identificar únicamente un dispositivo sólo dentro de la red a la que está conectado directamente. Asumiendo que los dispositivos finales gestionan ambos niveles de direccionamiento, el computador A puede especificar la dirección global de B, y el dispositivo intermedio puede determinar el puerto de salida que conduce a esa dirección global, en nuestro caso el puerto 2 (**Figura 7.5**). Para ello, el dispositivo intermedio dispone de una **tabla de encaminamiento** que va actualizando regularmente, y que esencialmente es una lista de correspondencia entre direcciones globales de destino y puertos de salida. El uso de direcciones locales y globales en el envío de una unidad básica de información del computador A al computador B tiene lugar tal como se especifica en la **Tabla 7.1**.

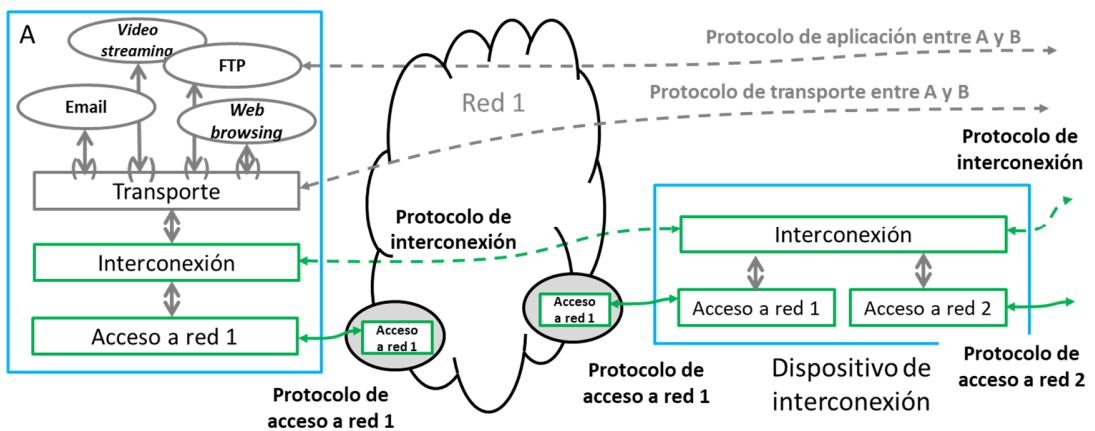
**Tabla 7.1.** Direcciones origen y destino en cada tramo de conexión a través de dos redes intermedias.

Tramo	Direcciones origen	Direcciones destino
A → Puerto 1	Local A Global A	Local Puerto 1 Global B
Puerto 1 → Puerto 2 (tramo interno)	A partir de Global B decide Puerto 2	
Puerto 2 → B	Local Puerto 2 Global A	Local B Global B

Basándonos en la **Tabla 7.1**, podemos subrayar algunos hechos:

- Los dispositivos de interconexión son los que deciden el encaminamiento a través de múltiples redes.
- Las direcciones globales origen y destino no cambian de un tramo a otro.
- En el dispositivo de interconexión, un módulo único, común a todos los puertos, y por encima de las especificidades de cada uno de ellos, es el que decide el siguiente salto (puerto de salida) a partir de la dirección global de destino.

El módulo común al que acabamos de referirnos, es el responsable de gestionar las direcciones globales, y lógicamente no está presente sólo en los dispositivos intermedios para la interconexión de redes, sino también en los dispositivos finales. Surge así una nueva capa, la **capa de interconexión**, ubicada entre la capa de transporte y la capa de acceso a red de la arquitectura de tres capas que habíamos alcanzado. La arquitectura resultante de cuatro capas se ilustra en la **Figura 7.6** sobre la base del escenario de la **Figura 7.5**, para el segmento de red entre el computador A y el dispositivo de interconexión (la figura se reproduce simétricamente entre el dispositivo de interconexión y el computador B). Obsérvese que un nuevo protocolo, el **protocolo de interconexión**, gobierna el diálogo entre sucesivos módulos de interconexión, por lo que implica una interacción que de forma general podemos denotar como **host-to-router-to-router-to-host**. El término **router** corresponde precisamente a la denominación del dispositivo de interconexión en el ámbito de la red Internet.



**Figura 7.6.** Arquitectura de cuatro capas. En el dispositivo de interconexión, sólo se implementan las dos capas inferiores.

La arquitectura de cuatro capas que acabamos de desarrollar ya es prácticamente la arquitectura TCP/IP en la que se basa la red Internet. En las dos secciones siguientes se describen con detalle esta arquitectura y la arquitectura OSI. Ambas son arquitecturas abiertas, es decir, creadas para facilitar la interconexión entre equipos de distintos fabricantes<sup>2</sup>, aunque su proceso de desarrollo fue muy diferente. El modelo OSI fue creado por ISO como marco de referencia para el diseño de futuros protocolos, con carácter preceptivo. En cambio, el modelo TCP/IP fue un producto de la comunidad investigadora, después de años de trabajo dedicados al desarrollo de protocolos que posibilitaran la interconexión entre computadores a través de diversas redes, lo que constituyó la base de la Internet actual. La arquitectura TCP/IP surgió, pues, como modelo que encajaba con la estructuración de estos protocolos, de ahí su carácter descriptivo, no preceptivo. Visto el auge de

<sup>2</sup> A diferencia de otras **arquitecturas propietarias**, desarrolladas como modelo para la interconexión con equipos de un determinado fabricante. Algunos ejemplos son la arquitectura SNA (*Systems Network Architecture*) desarrollada por IBM en 1974 y la arquitectura DECnet desarrollada por Digital Equipment Corporation en 1975.

los protocolos TCP/IP, los proponentes del modelo OSI intentaron ajustarlo para garantizar su vigencia, pero el modelo OSI adolecía de un defecto importante: fue diseñado para la conexión a través de una sola red, no a través de múltiples redes. En efecto, como veremos en la Sección 7.3, en la arquitectura OSI no hay una capa de interconexión de redes, sino simplemente una capa de red. El éxito y fracaso de uno y otro modelo se puede resumir con una simple, pero contundente afirmación (más o menos literalmente)<sup>3</sup>:

*A diferencia del modelo OSI, que fue definido por comités antes de que los protocolos fueran implementados, el modelo de referencia TCP/IP fue formalizado cuando los protocolos ya habían sido diseñados y probados.*

Podemos concluir la presente sección señalando las ventajas inherentes al desarrollo de una arquitectura de capas abierta y recopilando la terminología utilizada en este ámbito. Las ventajas son las siguientes:

- Toda arquitectura de capas abierta define un marco para el desarrollo de estándares de protocolos, los cuales facilitan la interoperabilidad entre módulos y dispositivos, aunque provengan de fabricantes diferentes. Por ejemplo, veremos a continuación que uno de los protocolos de transporte más importantes de la arquitectura TCP/IP es precisamente TCP, de manera que, cuando una aplicación hace uso de tal protocolo, el módulo de aplicación correspondiente invoca el servicio de un módulo de transporte que llamamos TCP. Normalmente este módulo está implementado en software, sin embargo, TCP no es una pieza de software estándar, sino un protocolo estándar. Es decir, “módulo TCP” significa que se trata de un módulo de transporte capaz de comunicarse según las reglas del protocolo TCP, pero puede haber sido implementado por cualquier desarrollador de software. Esta implementación es irrelevante desde el punto de vista de la arquitectura TCP/IP, y de hecho dos módulos TCP pueden comunicarse mientras utilicen correctamente el “idioma TCP”, independientemente de quienes hayan sido sus desarrolladores.
- La arquitectura acelera el proceso de incorporación de los avances tecnológicos en las redes, ya que estos suelen ir asociados a alguna de las capas, de forma que no es necesario rediseñar las otras.
- La arquitectura tiene también un aspecto pedagógico, ya que favorece la comprensión de los mecanismos que operan en las redes, al presentar el problema de comunicación entre computadores como un conjunto de problemas menores y más fáciles de abordar.
- Finalmente, la arquitectura proporciona también un esquema para la ejecución de pruebas de diagnóstico de fallos en las redes. Cuando se produce un fallo, en principio es atribuible a cualquiera de las capas. Por lo tanto, una forma sistemática y organizada de proceder consiste en explorar las posibles anomalías capa por capa, empezando por la capa inferior, que es donde se producen habitualmente los fallos más simples.

---

<sup>3</sup> Douglas S. Comer: *Internetworking with TCP/IP. Volume One*. Pearson Education Limited, 2014. Sixth Edition.

Para terminar, la **Tabla 7.2** recoge la nomenclatura básica utilizada en el ámbito de la arquitectura de capas de las redes de computadores.

**Tabla 7.2.** Terminología básica de arquitectura de redes, aplicable a cualquier modelo de capas.

Término	Definición
Arquitectura	Desglose vertical del proceso de comunicación entre dos sistemas mediante subprocessos más sencillos de distinta jerarquía.
Capa	Cada uno de los niveles que corresponden a los subprocessos en los que se ha desglosado el proceso de comunicación entre dos sistemas.
Módulo o entidad	Subsistema capaz de generar y recibir información, dentro de un sistema físicamente diferenciado como es un computador o un nodo.
Entidades homólogas ( <i>peers</i> )	Entidades situadas en la misma capa de sistemas distintos.
Protocolo	Conjunto de reglas que gobiernan el intercambio de información entre dos entidades homólogas. En la especificación de un protocolo intervienen tres aspectos: formato (aspecto sintáctico), significado (aspecto semántico) y reglas de intercambio de las unidades básicas de información que maneja el protocolo.
PDU ( <i>protocol data unit</i> )	Unidad básica de información de un protocolo. Formalmente, la unidad básica de información de un protocolo de capa X se puede expresar con la notación X-PDU, donde X es la inicial de la denominación de la capa. Por ejemplo, la unidad básica de información en la capa de transporte puede llamarse segmento o T-PDU.
Interfaz	Ámbito en el que tiene lugar el diálogo entre capas adyacentes dentro de un mismo sistema. No se estandariza, salvo que tales capas estén físicamente instaladas en dispositivos distintos, como ocurre en el caso de la llamada <b>interfaz física</b> .

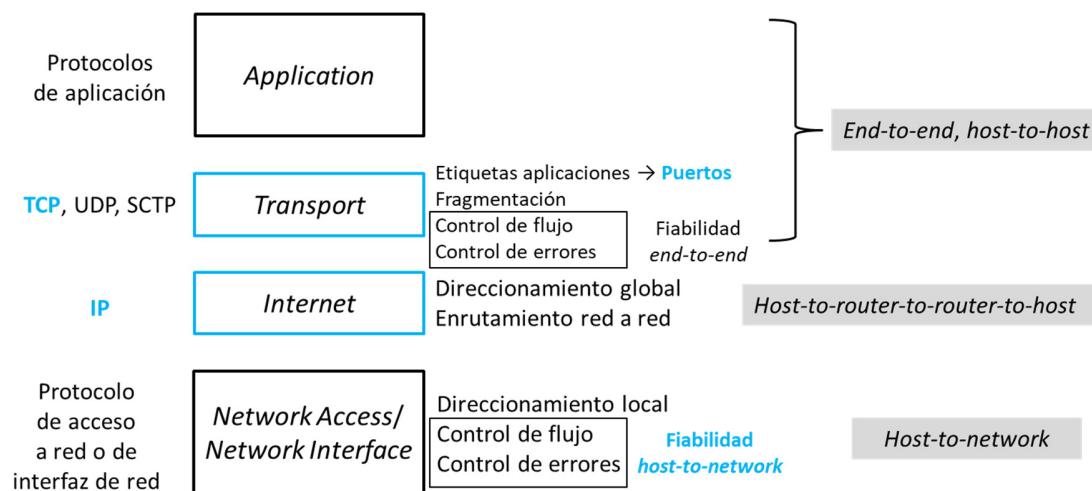
## 7.2. ARQUITECTURA TCP/IP

El germen de la red Internet fue la red experimental ARPANET, desarrollada como parte del proyecto DARPA del DoD (*Department of Defense*) de EEUU, iniciado en 1969. Con esta red se pretendía demostrar la viabilidad de la interconexión entre computadores de redes distintas, en este caso pertenecientes a diversas instituciones académicas y estatales. El proyecto culminó en 1983 con la migración de ARPANET a la nueva pila de protocolos estándar TCP/IP. A partir de entonces, ARPANET se fue expandiendo para convertirse en la red Internet. En 1989, la arquitectura TCP/IP fue formalizada por el IETF a través del documento RFC 1122.

No existe un modelo único de arquitectura TCP/IP, sino que, aparte de las versiones introducidas por el IETF, los principales autores han querido reflejar su propia visión. En general, todos los modelos se diferencian por el número de capas (cuatro o cinco) y la denominación de las mismas. Concretamente, la arquitectura propuesta por la multinacional Cisco se ajusta bastante bien a la mayoría de escenarios reales, tanto por el número de capas como por sus denominaciones. En la **Figura 7.7** se muestra dicha arquitectura y se recuerdan las funciones principales de cada capa, los protocolos más relevantes de algunas de ellas, y el tipo de interacción que conllevan. Obsérvese que coincide esencialmente con la arquitectura genérica de cuatro capas desarrollada en la sección

anterior. Más adelante se describirán las otras versiones de la arquitectura TCP/IP, y en particular la significación de usar un modelo de cinco capas en lugar de cuatro.

Como se subraya en la [Figura 7.7](#), el núcleo de la arquitectura lo constituyen las dos capas intermedias, es decir, la capa de transporte y la capa *internetwork*. Los principales protocolos en la capa de transporte son TCP (*Transport Control Protocol*) y UDP (*User Datagram Protocol*), y entre ellos TCP. El protocolo de transporte SCTP (*Stream Control Transmission Protocol*) fue desarrollado más tarde de manera específica para aplicaciones de *streaming*. Básicamente reúne las prestaciones de TCP y UDP que mejor se adaptan a los requisitos de tiempo real de estas aplicaciones. Para la capa *internetwork* sólo se ha desarrollado un protocolo, el protocolo IP (*Internet Protocol*), si bien en dos versiones, la versión IPv4 y la versión posterior IPv6. Actualmente ambas coexisten, pero IPv4 sigue siendo dominante. El nombre de la arquitectura proviene precisamente de los dos protocolos más importantes del núcleo de la misma, e incluso es frecuente observar que las denominaciones de las capas de ese núcleo son substituidas por los nombres de tales protocolos.



**Figura 7.7.** Arquitectura TCP/IP de cuatro capas (modelo de Cisco).

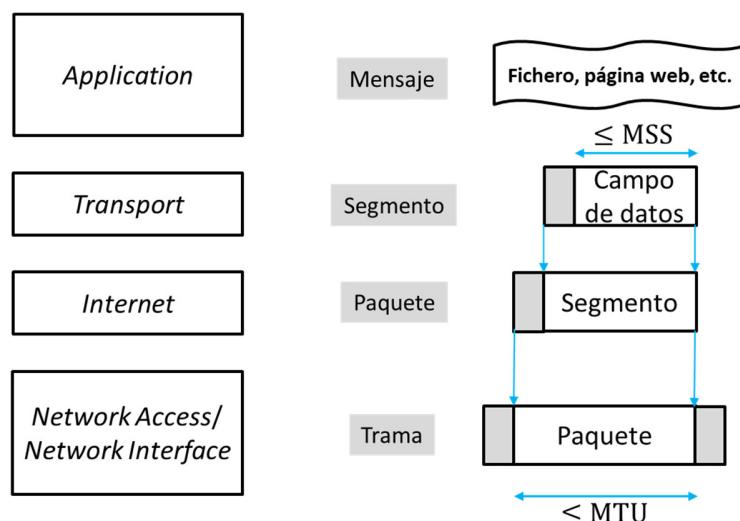
La [Figura 7.7](#) merece dos observaciones adicionales:

- El uso del término **puerto** en la arquitectura TCP/IP para referirse a lo que en el desarrollo de la arquitectura genérica hemos denominado etiqueta de aplicación.
- La función de enrutamiento (encaminamiento) de la capa *Internet* (protocolo IP) es un enrutamiento red a red, es decir, busca un camino a nivel de múltiples redes (o múltiples *routers*, o múltiples saltos), pero no a nivel interno en esas redes. Éste es un aspecto de implementación de cada red de paso, irrelevante en lo que a la arquitectura TCP/IP se refiere.
- El replicado de las funciones de control de flujo y errores en la capa de acceso a red. Por lo que respecta a la función de control de flujo, el planteamiento es el mismo, pero sobre las unidades básicas de información de la capa de acceso a red. A ese nivel, también hay una entidad que transmite y otra que recibe, pudiendo la primera sobrecargar la segunda. Por su parte, el control de errores en la capa de acceso a red contribuye a reducir los errores detectados en la capa de

transporte, donde las retransmisiones resultan mucho más costosas en consumo de tiempo. Resumiendo, mientras el control de flujo y el control de errores en la capa de transporte proporcionan fiabilidad entre extremos de la conexión, es decir, **fiabilidad end-to-end**, el control de flujo y el control de errores en la capa de acceso a red proporcionan **fiabilidad host-to-network**.

Otro aspecto importante es el que se refiere a la implementación de las capas en los dispositivos. Generalmente, los módulos de aplicación, transporte e *internetwork* se implementan en software, mientras que es bastante común que la implementación del módulo de acceso a red sea en hardware, en forma de **tarjeta de red** o **NIC** (*network interface card*). Es precisamente en estos casos que el modelo TCP/IP de Cisco encaja perfectamente. No obstante, en otros casos considerados más adelante, el escenario real se ve mejor representado por un modelo de cinco capas.

Finalmente, una cuestión que tan solo hemos abordado parcialmente, es la que se refiere al proceso de construcción de las unidades básicas de información (PDU) de cada capa. La **Figura 7.8** muestra este proceso con relación a las capas de la arquitectura TCP/IP, si bien aquí no hay diferencias significativas con otros modelos TCP/IP y la arquitectura OSI que trataremos en la siguiente sección. La figura también especifica la denominación de las diversas unidades básicas de información según la capa considerada.



**Figura 7.8.** Unidades básicas de información en cada capa, y proceso de encapsulado.

La construcción de las unidades básicas de información conlleva dos aspectos. Por un lado, cómo se distribuyen y ubican las informaciones de control, y, por otro, cómo se relacionan las unidades básicas de información entre sí. Recordemos que el segmento contenía las siguientes informaciones de control relacionadas con las funciones de la capa de transporte: etiquetas de las aplicaciones (origen y destino), es decir, lo que ahora denominamos puertos (origen y destino), número de secuencia (funciones de secuenciación, control de flujo y control de errores) y código de protección contra errores (función de control de errores). Las direcciones de los computadores A y B fueron

relegadas a la capa de acceso a red antes de introducir la interconexión a través de múltiples de redes. La interconexión de redes y el necesario desdoblamiento de las direcciones en locales y globales, obliga a una reubicación, casi evidente una vez descritas las funciones de la capa *internetwork* y la capa de acceso a red: las direcciones globales origen y destino son informaciones de control que debe manejar la capa *internetwork*, mientras que las direcciones locales origen y destino son informaciones de control que debe manejar la capa de acceso a red.

Cualesquiera que sean las informaciones de control que competen a cada capa, el procedimiento adoptado consiste en ubicarlas en forma de **cabecera** de la correspondiente PDU. Esta cabecera queda antepuesta al **campo de datos** de dicha PDU, que a su vez contiene la PDU de la capa inmediatamente superior, lo que se conoce como **encapsulado**. El proceso desde la capa superior a la inferior es, por tanto, el siguiente:

1. El dato a nivel de capa de aplicación (mensaje de datos) es fragmentado en la capa de transporte<sup>4</sup>, que añade una cabecera a cada fragmento para formar un segmento.
2. El segmento se encapsula en la PDU de la capa *internetwork*, conocida como **paquete**.
3. El paquete se encapsula en la PDU de la capa de acceso a red, conocida como **trama**. Este es el único encapsulado que ubica la PDU de la capa superior entre una cabecera y un tráiler. El tráiler contiene el campo de chequeo de errores que implementa la función de control de errores en la capa de acceso a red. Se ubica al final de la trama por una cuestión de eficiencia: el cómputo del código de protección contra errores se realiza habitualmente a velocidad de hardware (cuando el módulo de acceso a red es una tarjeta de red), a partir de los bits a proteger; para no aprovechar esa alta velocidad, dicho cómputo puede realizarse al mismo tiempo que la trama se transmite, con sólo uno o dos bits de retraso, y acoplar el código de protección apenas termina de enviarse el último bit del campo de datos.

Obsérvese que la relación entre segmentos, paquetes y tramas es de uno a uno, es decir, cada trama contiene un paquete, y cada paquete contiene un segmento. De hecho, es frecuente utilizar el término “paquete” para referirse a las tramas e incluso a los segmentos. Además, es el término que da lugar a la denominación “comutación de paquetes” como técnica de red comutada, técnica utilizada prácticamente por todas las redes de computadores, excepto en casos muy concretos que veremos más adelante.

La **Figura 7.8** también pone de manifiesto dos magnitudes relevantes, la *MTU (maximum transfer unit)* y la *MSS (maximum segment size)*. La primera se refiere al máximo tamaño del campo de datos de las tramas, y es un valor impuesto por la red. Por ejemplo, en el caso de las redes Ethernet, este valor es de 1500 B. La segunda magnitud indica el máximo tamaño del campo de datos de los segmentos (y no el máximo tamaño de los segmentos, como sugiere su denominación). Dado el valor de *MTU*, y especificados los protocolos de las capas *internet* y de transporte, y por tanto

---

<sup>4</sup> Este proceso es algo más complejo, y se analiza con detalle sobre el ejemplo de una sesión de navegación web.

especificados los tamaños de las cabeceras respectivas,  $IPH$  y  $TPH$ , se puede obtener el valor de  $MSS$  del siguiente modo:

$$MSS = MTU - IPH - TPH \quad (7.1)$$

Los valores habituales de las cabeceras *internet* y de transporte se muestran en la **Tabla 7.2**. Es importante notar que es el valor de  $MTU$  el que condiciona el valor de  $MSS$  según la ecuación (7.1), y no al revés. Y es el valor de  $MSS$  el que a su vez condiciona el proceso de fragmentación que tiene lugar en la capa de transporte. A modo de ejemplo, si la aplicación considerada hace uso del protocolo TCP, el protocolo de *internet* es IPv4 y la red local es Ethernet, aplicando la ecuación (7.1) se tiene:  $MSS = 1500 - 20 - 20 = 1460$  B. Muchas capturas Wireshark confirman este valor.

**Tabla 7.2.** Tamaños del formato normal de las cabeceras de las capas *internet* y de transporte.

Cabecera		Tamaño (B)	
Capa <i>internet</i>	IPv4	$IPH =$	20
	IPv6		40
Capa de transporte	TCP	$TPH =$	20
	UDP		8
	SCTP		12

En los siguientes apartados, completamos la descripción del ámbito TCP/IP centrándonos en tres aspectos: otros modelos TCP/IP, el *socket* y la estructura básica de un *router*.

### Otros modelos TCP/IP

A diferencia del modelo OSI descrito en la siguiente sección, el modelo TCP/IP se presenta en varias versiones, precisamente porque no responde a una especificación prestablecida desde un organismo oficial, como ocurre con el primero. Las diferentes versiones se recogen en la **Tabla 7.3**. Conceptualmente son muy similares, siendo en el número de capas donde reside la diferencia más importante.

**Tabla 7.3.** Versiones más importantes de la arquitectura TCP/IP. Para las dos capas superiores, no hay diferencia alguna entre los modelos, ni siquiera en términos de nomenclatura. El modelo de Tannenbaum/Kurose es el que más se parece al modelo OSI descrito en la Sección 7.3.

Modelo IETF (RFC 1122)	Modelo Tannebaum/Kurose	Modelo Cisco	Modelo Forouzan	Modelo Stallings	Modelo Comer
<i>Application</i>	<i>Application</i>	<i>Application</i>	<i>Application</i>	<i>Application</i>	<i>Application</i>
<i>Transport</i>	<i>Transport</i>	<i>Transport</i>	<i>Transport</i>	<i>Transport</i>	<i>Transport</i>
<i>Internet</i>	<i>Network</i>	<i>Internet</i>	<i>Network</i>	<i>Internet</i>	<i>Internet</i>
<i>Link</i>	<i>Link</i>	<i>Network Access/ Network Interface</i>	<i>Data Link</i>	<i>Network Access/ Data Link</i>	<i>Network Interface</i>
	<i>Physical</i>		<i>Physical</i>	<i>Physical</i>	<i>Physical hardware</i>

Los modelos de 5 capas pueden verse como el resultado de desdoblar la capa más baja de los modelos de 4 capas. El desdoblamiento consiste en separar los aspectos lógicos y funcionales (diálogo a nivel de tramas, sintaxis y semántica de las mismas) de aquellos puramente físicos (velocidad de transmisión, técnica de codificación de línea o modulación digital, banda de frecuencias de trabajo, tipo de cableado, potencia transmitida, alcance, etc.). El número de capas que mejor se ajusta depende del escenario real considerado. Por ejemplo, ya hemos visto que las tarjetas de red Ethernet y wifi se corresponden con la implementación hardware de todos los aspectos señalados, tanto lógicos y funcionales como físicos, por lo que estos escenarios quedan mejor representados por un modelo de 4 capas. Por otro lado, hay protocolos de la capa más baja que están implementados como una entidad software que puede ejecutarse sobre distintos tipos de enlaces físicos. Por ejemplo, es habitual utilizar el protocolo HDLC (descrito en el Capítulo 8) para enlaces punto a punto de larga distancia entre *routers*, sobre la variedad de líneas serie de las jerarquías *E-carrier* o *T-carrier* (consideradas en el Capítulo 9). Para éste y otros casos similares se ajusta mejor una arquitectura de 5 capas. Salvo que se especifique lo contrario, asumiremos un modelo de 4 capas, en cuyo caso aceptaremos cualquiera de las denominaciones para la capa más baja: enlace (IETF), acceso a red (Cisco) o interfaz de red (Cisco).

### El *socket*

El **socket** es una estructura de datos del tipo fichero (`file`) de Unix<sup>5</sup>, que hace el papel de interfaz entre una entidad de aplicación y una entidad de transporte en la arquitectura TCP/IP. A través del *socket* se lleva a cabo el intercambio de bytes entre ambas entidades. La fragmentación que tiene lugar en la capa de transporte (concretamente, en el módulo TCP) consiste básicamente en leer los bytes depositados por la entidad de aplicación en el *socket*, hasta alcanzar la cantidad de *MSS* bytes (o hasta alcanzar el último byte de un conjunto residual de tamaño inferior a *MSS*). Entonces, la entidad de transporte la añade la cabecera necesaria para formar un segmento, y lo envía a la capa *internet*.

La importancia del *socket* deriva de su papel fundamental en el diseño de las aplicaciones distribuidas. Por tanto, veamos en primer lugar en qué consisten éstas. Una **aplicación distribuida** es aquélla que está constituida por dos o más procesos residentes en dispositivos geográficamente dispersos. La ejecución de la aplicación requiere que dichos procesos se comuniquen, por lo que los dispositivos que los albergan tienen que estar conectados a través de una red. Así pues, el concepto de aplicación distribuida va asociado inherentemente a la presencia de una red de computadores. En otras palabras, las aplicaciones distribuidas son aquellas que generan tráfico de red. La **programación de sockets** es precisamente la parte del diseño de una aplicación distribuida que se ocupa de la comunicación entre los procesos participantes a través de las interfaces definidas por los *sockets*. Esta comunicación obedece a uno de los dos modelos siguientes:

- **Modelo cliente-servidor.** En este modelo, los procesos participantes se dividen en dos jerarquías. En el nivel más alto está el **proceso servidor**, único, activo permanentemente y

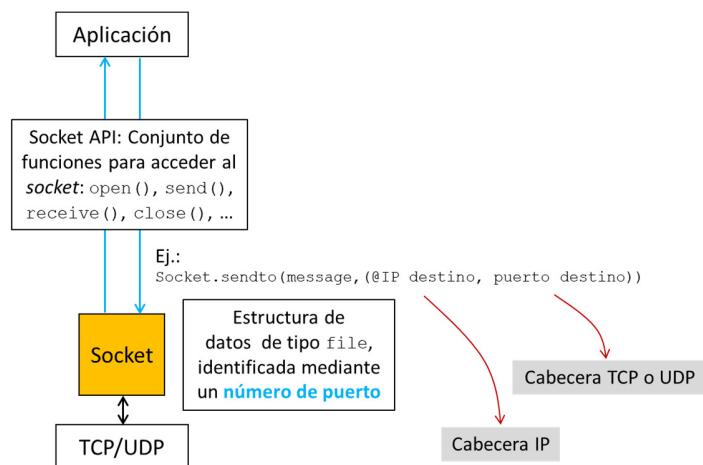
---

<sup>5</sup> La arquitectura TCP/IP fue desarrollada inicialmente sobre máquinas Unix.

responsable de la mayor carga de la aplicación. Reside en un computador que puede requerir prestaciones especiales, al que llamamos **servidor**. Por otro lado, los **procesos cliente** residen en los dispositivos de usuario y requieren el proceso servidor para completar la ejecución de la aplicación. En toda aplicación cliente-servidor, es el cliente quien inicia siempre la conexión con el servidor. La mayor parte de aplicaciones distribuidas obedecen a este modelo.

- **Modelo peer-to-peer.** En este caso, todos los procesos participantes tienen la misma jerarquía y residen en los dispositivos de usuario, sin requerir, por tanto, el apoyo de un servidor.

Cualquiera que sea el modelo de aplicación distribuida, la programación de *sockets* consiste en el manejo de las funciones que permiten acceder al *socket*<sup>6</sup>. El **Socket API** es precisamente el conjunto de todas estas funciones. Las más básicas, es decir, las que permiten abrir, leer, escribir o cerrar el *socket* como fichero Unix que es, se muestran en la **Figura 7.9**. Esta figura también revela que un puerto en la arquitectura TCP/IP es precisamente el identificador de un *socket* dentro de un dispositivo final, y que la identificación de ese mismo *socket* dentro de la red Internet comprende el número de puerto y la dirección IP del dispositivo.



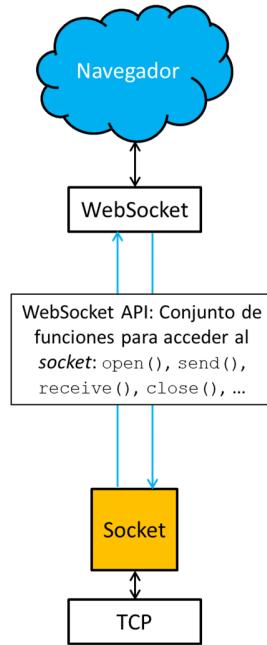
**Figura 7.9.** El Socket API.

Actualmente muchas aplicaciones cliente-servidor se diseñan en formato web, es decir, accediendo a un servidor web mediante un navegador. Dicho de otro modo, son aplicaciones escritas en el lenguaje que entienden los navegadores, de modo que son estos los que las ejecutan en el lado cliente. Dado que estas aplicaciones suelen mover datos multimedia con requisitos de tiempo real, exigen una velocidad de transferencia muy alta. Por ello, hacen uso del **protocolo de aplicación WebSocket**, una variante más rápida del protocolo HTTP orientada a satisfacer los requisitos indicados. WebSocket también utiliza los servicios del protocolo de transporte TCP, es compatible con HTTP y, de hecho, tiene asociado los mismos puertos bien conocidos (80, 443). Correspondientemente, el **WebSocket API** es el conjunto de funciones que posibilitan la

---

<sup>6</sup> En la actualidad hay muchas herramientas de programación de aplicaciones distribuidas que ofrecen una interfaz al programador, la cual le libera de manejar las funciones de bajo nivel que acceden directamente al *socket*.

comunicación entre el módulo de aplicación WebSocket y el protocolo TCP, tal como se describe en la [Figura 7.10](#). A día de hoy, WebSocket está implementado en la mayor parte de navegadores.



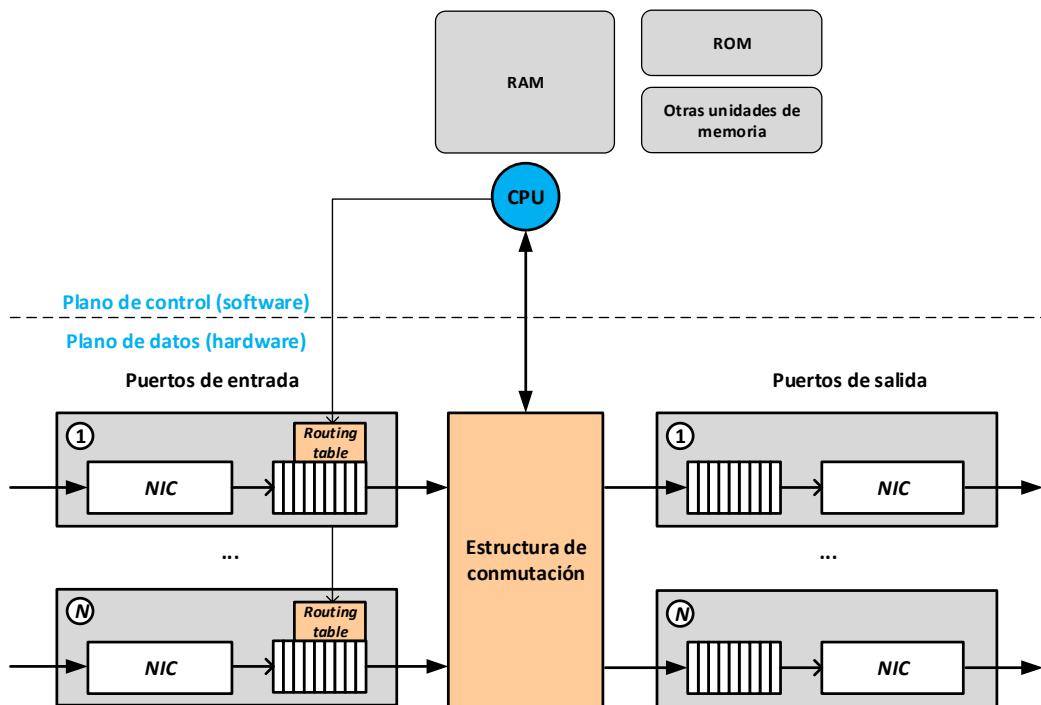
[Figura 7.10](#). El WebSocket API.

#### Estructura básica de un *router*

El *router* es el dispositivo intermedio por excelencia de la red Internet. Su función principal es procesar la cabecera de cada paquete que recibe por un puerto de entrada, determinar su dirección de destino global y reenviarlo a través del puerto de salida indicado en la tabla de encaminamiento (*routing table*). Básicamente, la tabla de encaminamiento es un listado de correspondencias entre direcciones de destino globales (direcciones IP) y puertos de salida. Por lo tanto, la arquitectura de capas de un *router* alcanza hasta la capa *internet*. Su estructura fundamental se muestra en la [Figura 7.11](#), donde el total de  $N$  puertos de entrada-salida aparecen desdoblados y separados por la [estructura de conmutación](#). Esta estructura es el núcleo de la llamada función de reenvío (*forwarding function*) del *router*, y su diseño puede obedecer a tres estrategias diferentes: conmutación vía memoria, conmutación vía bus y conmutación vía red de interconexión. La otra función importante del *router* es la función de encaminamiento (*routing function*). Podríamos resumir el papel de ambas funciones de la siguiente manera:

- **Función de encaminamiento.** Está función consiste en la ejecución distribuida de un [algoritmo de encaminamiento](#) basado en la Teoría de Grafos, para lo cual el *router* debe intercambiar información de control con otros *routers* según las reglas de un [protocolo de encaminamiento](#). El resultado final de la función de encaminamiento es la tabla de encaminamiento. En general, esta tabla es dinámica, ya que se actualiza regularmente en función del estado de la red.

- **Función de reenvío.** Esta función actúa a partir de la tabla de encaminamiento resultante de la función anterior. Consiste en leer la dirección global de destino en la cabecera de cada paquete, determinar el puerto de salida que le corresponde según la tabla de encaminamiento vigente, y enviar el paquete a ese puerto de salida a través de la estructura de conmutación.



**Figura 7.11.** Estructura básica de un *router* moderno. En general, la NIC admite operación full-dúplex, des-encapsulando los paquetes de las tramas entrantes y volviéndolos a encapsular en las tramas salientes. Los buffers de entrada y salida contienen, por tanto, paquetes.

La tabla de encaminamiento es, por tanto, el elemento que conecta las dos funciones principales del *router*. En la **Figura 7.12** se ilustra esta idea. La **Figura 7.11** también pone de manifiesto la existencia de dos planos de operación del *router*:



**Figura 7.12.** Conexión entre las funciones de encaminamiento y reenvío del *router*.

- El **plano de control**. En este plano, el *router* funciona como si de un computador se tratara, aunque con un propósito muy específico (la función de encaminamiento). Las operaciones previstas en este plano son ejecutadas por una CPU y gobernadas por el sistema operativo de

red<sup>7</sup> instalado en el *router*. Ejemplos de tales operaciones son las siguientes: arranque del *router*, todas las relacionadas con la función de encaminamiento (ejecución de algoritmos y protocolos de encaminamiento), administración de los recursos hardware, diálogo con el administrador (lenguaje de comandos y respuestas), gestión de la seguridad, ejecución del protocolo ARP y gestión de la tabla ARP (se verán más adelante) y actualización y copia de la tabla de encaminamiento.

- El **plano de datos**. En este plano se ejecuta la función de reenvío. En los primeros *routers*, esta función también era responsabilidad de la CPU, pero se transfirió al plano de datos con el objetivo de reducir el consumo de tiempo de este recurso. En el plano de datos, el procesado se realiza íntegramente en hardware, y por tanto a una velocidad muy alta, en consonancia con las altas velocidades de transmisión en los puertos de entrada-salida. El hardware del plano de datos comprende una tarjeta de línea (*line card*) en cada puerto de entrada-salida y la estructura de conmutación. Cada tarjeta de línea contiene, a su vez, una NIC (módulo de acceso a red o módulo de enlace), un *buffer* de entrada, un *buffer* de salida y una copia de la tabla de encaminamiento vigente. Esta copia es guardada por la CPU del *router* en una memoria de acceso muy rápida, cada vez que actualiza la tabla de encaminamiento. El procesado de las cabeceras de los paquetes y la búsqueda de las entradas en la copia de la tabla de encaminamiento ya no son, sin embargo, tareas de la CPU, sino operaciones ejecutadas en la propia tarjeta de línea. Dada la altísima velocidad con que pueden llegar las tramas por un puerto de entrada, para la localización de la entrada adecuada en la copia de la tabla de encaminamiento, también se utilizan algoritmos de búsqueda optimizados.

Los paquetes con origen y destino en sendos dispositivos finales tan solo cruzan el plano de datos de cada *router*. En cambio, los paquetes asociados a las funciones del plano de control (paquetes de los protocolos de encaminamiento, principalmente), bien entran por el plano de datos y se dirigen al plano de control, bien salen del plano de control para ser enviados a través del plano de datos. De ahí el doble sentido de la comunicación entre la CPU del *router* y la estructura de conmutación.

### 7.3. ARQUITECTURA OSI

En el año 1977, la organización ISO comenzó a trabajar en una nueva arquitectura que favoreciera la interoperabilidad universal entre entidades software y dispositivos de red, neutral con respecto a los fabricantes y sin intereses comerciales. Con ello se pretendía contrarrestar las tendencias monopolistas que habían propiciado el desarrollo de arquitecturas cerradas, que no favorecían para nada la interconexión de equipos que no estuvieran bajo el paraguas de los creadores y propietarios de las mismas. La nueva arquitectura, llamada **arquitectura OSI** (*Open Systems Interconnection*) fue publicada en el año 1984, después de un largo proceso en el que participaron los agentes más destacados de las industrias de las telecomunicaciones y el computador. No obstante, el desarrollo de protocolos en base a esta arquitectura fracasó como consecuencia de la complejidad de la

---

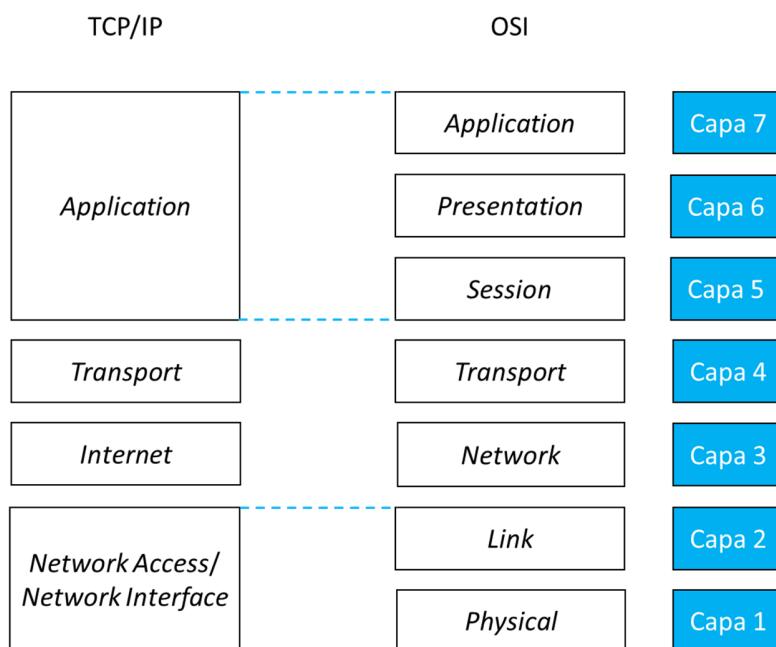
<sup>7</sup> Un **sistema operativo de red** es un sistema operativo instalado en un dispositivo de red (básicamente conmutador o *router*). Un ejemplo es el sistema operativo Cisco IOS.

misma, la gran cantidad de socios implicados, la lentitud y escasa flexibilidad del proceso de creación de estándares por parte de los comités de ISO, y, sobretodo, el auge de la arquitectura TCP/IP. A pesar de ello, la arquitectura OSI persiste en la actualidad como modelo conceptual de referencia.

Aunque el desarrollo de la arquitectura OSI no se basó para nada en la arquitectura TCP/IP, se puede ver la primera como una modificación de cualquier versión de cinco capas de la segunda en los siguientes términos:

- El desglose de la capa de aplicación en las tres capas siguientes, de mayor a menor jerarquía: una **capa de aplicación** propiamente dicha, una **capa de presentación** encargada de todos los aspectos relacionados con la sintaxis de la información (códigos alfanuméricos, formatos de compresión, etc.), y una **capa de sesión** ocupada en controlar el diálogo entre los procesos de aplicación, y de recuperarlo en caso de interrupción.
- La substitución de la capa de interconexión de redes, denominada capa *internet* en casi todas las versiones de la arquitectura TCP/IP, por una **capa de red** (*network*) que corresponde a la única red a la que se suponen conectados todos los dispositivos.

En ambos aspectos, la arquitectura TCP/IP se acerca más a la realidad. Por un lado, la mayor parte de las aplicaciones distribuidas vienen como una única componente software que engloba las tres funciones artificialmente diferenciadas en la arquitectura OSI; por otro lado, la mayor parte de las conexiones a través de Internet cruzan varias redes. En la **Figura 7.13** se muestra la arquitectura OSI. Como podemos observar, consta de un total de siete capas, que también se reconocen por su numeración. Las cuatro capas superiores llevan interacción entre extremos (*end-to-end*), mientras que en las tres capas inferiores la interacción es nodo a nodo (*node-to-node*).

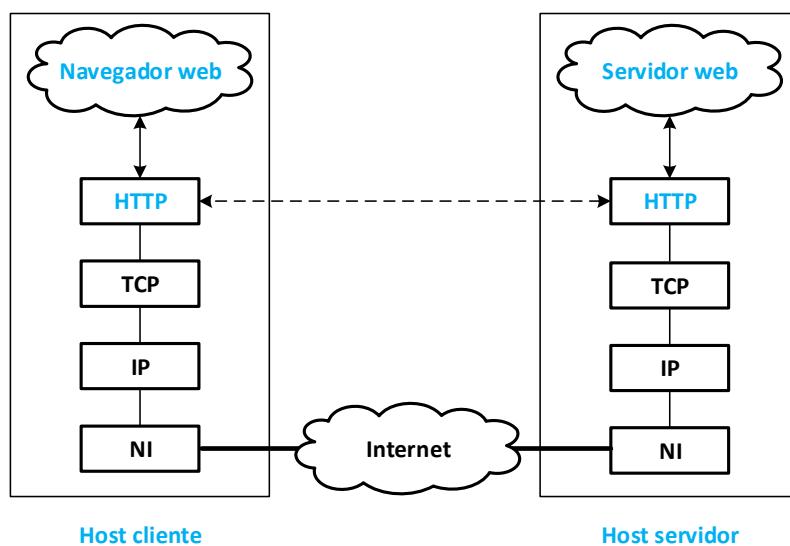


**Figura 7.13.** Arquitectura OSI y comparativa con la arquitectura TCP/IP (modelo Cisco).

## 7.4. ANATOMÍA DE UNA SESIÓN WEB

La **World Wide Web** (WWW) es un sistema de distribución de documentos de hipertexto a través de Internet. Estos documentos, conocidos como **páginas web**, son accesibles desde servidores especializados, los **servidores web**, y están escritos en **lenguaje HTML** (*HyperText Markup Language*) o su sucesor XHTML (*eXtensible HyperText Markup Language*). Son lenguajes que permiten dotar una página web de cualquier contenido, no sólo texto, imágenes, vídeo u otros objetos multimedia, sino también enlaces (**hiperenlaces**) a otros recursos (otras páginas web u otros objetos incrustados) disponibles en el mismo servidor o en otros.

La **navegación web** es una aplicación cliente-servidor estructurada tal como se muestra en la **Figura 7.14**. El usuario hace uso de un **navegador**, que no es más que el software del lado cliente, para acceder a las páginas web gestionadas por los servidores web. Un **servidor web** no es más que el software del lado servidor, el cual se encarga de almacenar, procesar y distribuir páginas web. Ejemplos de navegadores web son Mozilla Firefox, Internet Explorer, Google Chrome y otros. Ejemplos de servidores web son Apache HTTP Server y Nginx<sup>8</sup>. El diálogo entre un navegador y un servidor web hace uso del **protocolo de aplicación HTTP** (*HyperText Transfer Protocol*), el cual invoca los servicios del protocolo de transporte TCP para garantizar la fiabilidad de la transferencia de las páginas web. Está recogido principalmente en el RFC 2616.



**Figura 7.14.** Componentes de la navegación web en el contexto de la arquitectura TCP/IP.

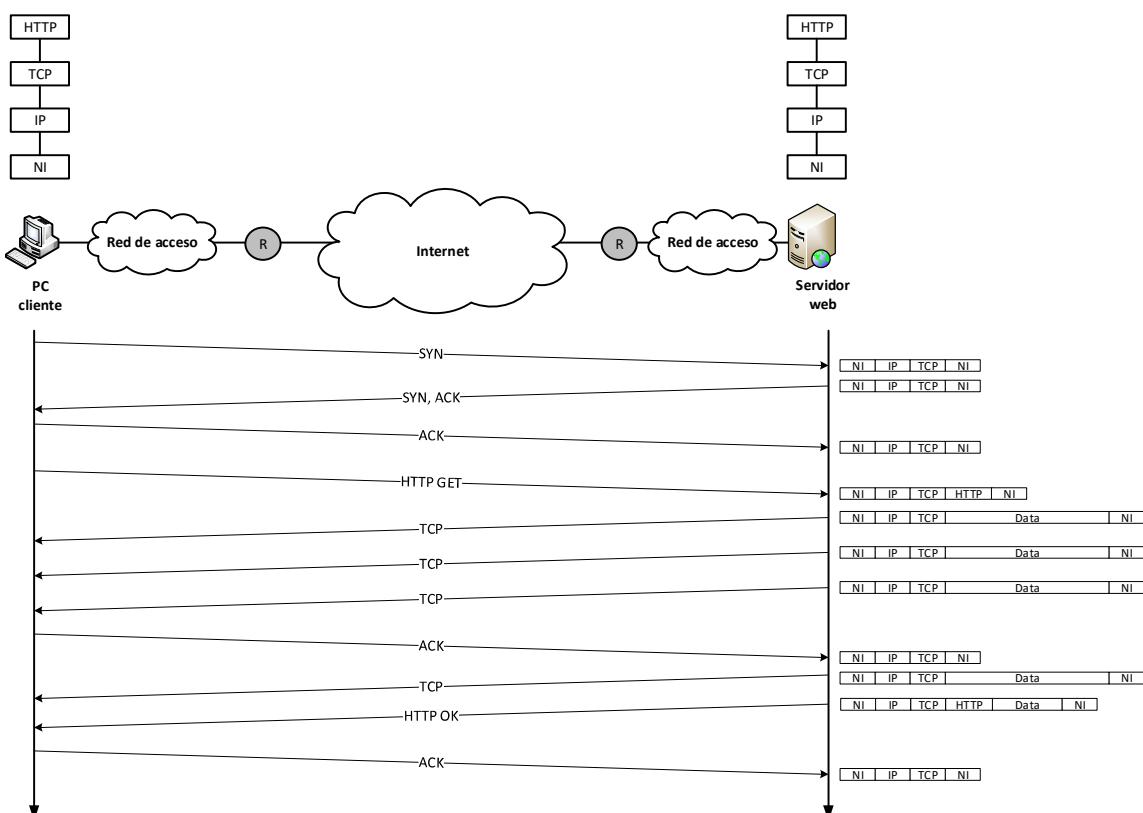
Actualmente, casi un 25% del tráfico de Internet es generado por el protocolo de aplicación HTTP, ocupando el primer puesto en el ranking de aplicaciones<sup>9</sup>. Esta posición prominente se debe no sólo al tráfico de descarga de páginas web, sino también, y principalmente, al hecho de que muchas aplicaciones y servicios de transferencia de ficheros y *streaming* adoptan hoy en día el formato web.

<sup>8</sup> Los servidores web suelen incluir las funcionalidades del protocolo HTTP.

<sup>9</sup> Sandvine: *The Global Internet Phenomena Report*. September 2019. <https://www.sandvine.com/resources>

Merece la pena, pues, diseccionar una sesión web para descubrir el intercambio básico de mensajes. Este análisis también nos permitirá descubrir la interacción entre HTTP y TCP. Recordemos que la función de la capa de transporte es proporcionar fiabilidad a la transferencia de los mensajes de aplicación, además de fragmentarlos cuando son demasiado grandes. Concretamente, el protocolo de transporte que implementa todas estas funcionalidades hasta el último término es TCP, ya que los otros protocolos de transporte (UDP, SCTP) tan solo las implementan parcialmente. Por lo tanto, cuando se trata de garantizar que una entidad de aplicación reciba una copia exacta de un mensaje de datos enviado por otra entidad de aplicación, TCP es el protocolo de transporte seleccionado.

El diagrama temporal de la **Figura 7.15** muestra el intercambio básico de mensajes HTTP y segmentos TCP durante la descarga del contenido HTML de una página web. El contenido HTML es el primer paso del proceso de descarga de una página web, y consiste en texto, objetos multimedia e hiperenlaces a otros recursos que deberán ser descargados a continuación (si los hay). En la figura se ha supuesto el escenario típico de cada extremo conectado a Internet a través de una red de acceso y un *router*, en consonancia con el encapsulamiento de los paquetes en tramas mostrado en la parte derecha de la misma. Además, el diálogo mostrado merece algunas consideraciones:

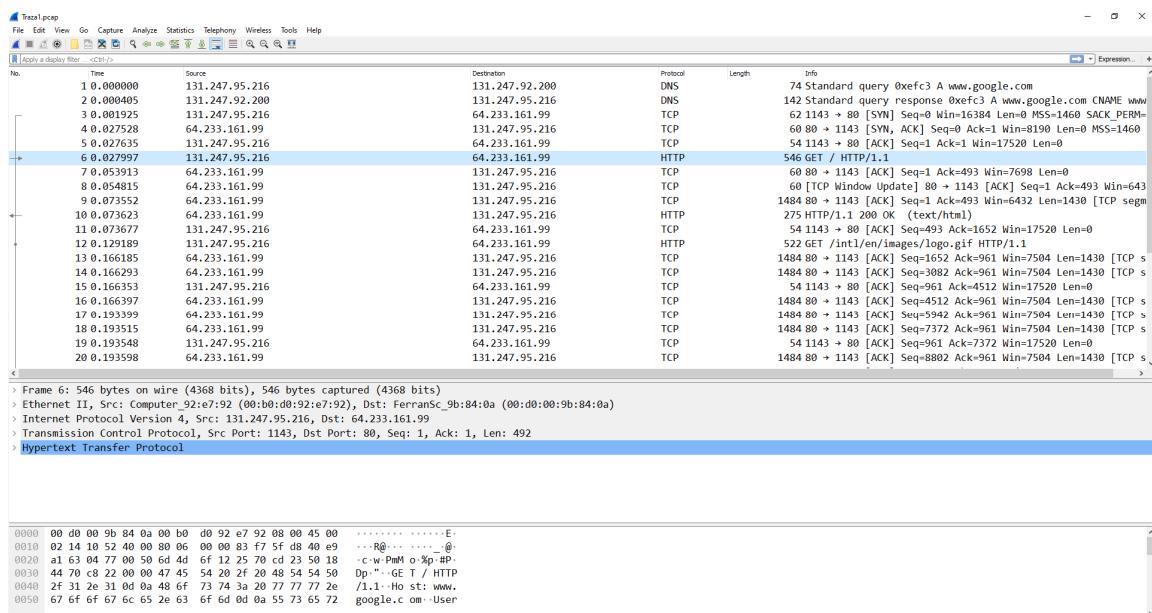


**Figura 7.15.** Intercambio de mensajes HTTP y segmentos TCP durante la descarga del contenido HTML de una página web, al principio de una sesión web con un determinado servidor. No se muestran las posibles descargas subsecuentes de otros objetos imbricados en la página.

- TCP es un **protocolo orientado a conexión**, lo cual significa que, antes de que tenga lugar la transferencia de datos TCP (mensajes HTTP en este caso), las dos entidades TCP tienen que notificarse y/o negociar ciertos parámetros para la correcta implementación de las funciones de fragmentación, control de flujo y control de errores. Esto tiene lugar durante la llamada **fase de establecimiento de la conexión TCP**. Por ejemplo, durante esta fase, las entidades TCP se notifican mutuamente los números de secuencia iniciales de los segmentos que van a enviar. También se notifican los valores de la variable *MSS*, calculados a partir de los valores *MTU* de sus respectivas redes de acceso, como parte inicial de un procedimiento que puede desembocar o no en el uso de un valor *MSS* común. Estas notificaciones y otras tienen lugar mediante un intercambio de tres segmentos de control que configuran el llamado **acuerdo a tres vías** (*three-way handshake*). Se trata de los siguientes segmentos (parte inicial del diagrama de tiempos):
  - Un **segmento SYN** por parte de la entidad que inicia la conexión. Mediante este segmento, dicha entidad especifica el número de secuencia inicial que va a utilizar (puede ser cualquiera, elegido aleatoriamente) y su valor *MSS*, entre otros parámetros.
  - La respuesta en forma de **segmento SYN, ACK** por parte de la entidad que recibe la petición de conexión. Este segmento tiene la doble función, **SYN** para anunciar las mismas variables que el segmento anterior, y **ACK** para dar acuse de recibo de la petición de conexión y los parámetros propuestos.
  - La confirmación por parte de la entidad que inició la conexión mediante un **segmento ACK**, que confirma la recepción del segmento **SYN** y los parámetros de la otra entidad.
- Una vez establecida la conexión TCP, la entidad solicitante envía un segmento con campo de datos no nulo, que en este caso consiste en el mensaje **HTTP GET** indicando la solicitud de una determinada página web.
- En el protocolo HTTP, la respuesta positiva a un mensaje **HTTP GET** es un mensaje **HTTP OK** formado por una cabecera **HTTP** y el contenido solicitado, que en nuestro caso es el contenido HTML de cierta página web (porque es la primera descarga después de haberse establecido la conexión TCP). El módulo **HTTP** del servidor deposita este voluminoso mensaje **HTTP OK** en el **buffer** de salida del socket por el que accede a los servicios de su módulo TCP. Este módulo va leyendo el contenido de dicho **buffer** en grupos de *MSS* bytes (así es como tiene lugar realmente la fragmentación), añadiendo a cada grupo una cabecera para formar un segmento TCP. El depósito de la información en el **buffer** se realiza de tal manera que el último segmento TCP enviado por el servidor contiene la cabecera **HTTP OK** junto con una fracción residual de contenido HTML (o del objeto que se estuviera descargando). En el diagrama de tiempos mostrado, la descarga tiene lugar a través de 5 segmentos TCP, el último de los cuales contiene la cabecera **HTTP OK** y el residuo de contenido HTML.
- De vez en cuando, la entidad que recibe la descarga envía algunos acuses de recibo. Si observamos las tramas en las que viajan estos acuses de recibo, el protocolo de capa más alta contenido en ellas es TCP, lo que significa que son acuses de recibo a nivel TCP, es decir, segmentos **ACK** que confirman positivamente la recepción de los datos (fragmentos del mensaje **HTTP OK**) enviados por el servidor hasta el momento. En particular, el último segmento **ACK** confirma positivamente la recepción del mensaje **HTTP OK** y el residuo de contenido HTML, pero no forma parte del diálogo a nivel HTTP.

- Para cada segmento TCP se muestra su encapsulamiento en un paquete IP y, a su vez, el encapsulamiento del paquete IP en una trama (cabecera y tráiler NI – *network interface*). No obstante, es importante constatar que esta cabecera y tráiler NI no son fijos, sino que cambian según la red de paso considerada, y particularmente según el extremo considerado. Por ejemplo, en el lado cliente, obedecen al formato de trama de la red de acceso del cliente, mientras que, en el lado servidor, obedecen al formato de trama de la red de acceso del servidor.

A modo de ilustración, la **Figura 7.16** muestra la ventana de Wireshark correspondiente a la captura de una sesión de navegación web. En el panel superior se ha resaltado la trama que encapsula el mensaje `HTTP GET`, que está precedido por la fase de establecimiento de la conexión TCP. Previamente se ejecuta una petición a un servidor DNS (*Domain Name System*) con el fin de averiguar la dirección o direcciones IP que corresponden al dominio al cual se quiere acceder. Obsérvese también que después del `HTTP OK` que pone fin a la descarga del contenido HTML de la página web solicitada, viene otro `HTTP GET`, pero éste ya no requiere una nueva petición al DNS ni una nueva fase de establecimiento de la conexión TCP. La razón es que el nuevo `HTTP GET` invoca la descarga de un objeto GIF que está en el mismo servidor que el contenido HTML.



**Figura 7.16.** Ventana Wireshark que describe parte de la captura de una sesión web. La trama seleccionada encapsula el primer mensaje HTTP GET enviado.

## 10.1. PLANO DE DATOS Y PLANO DE CONTROL

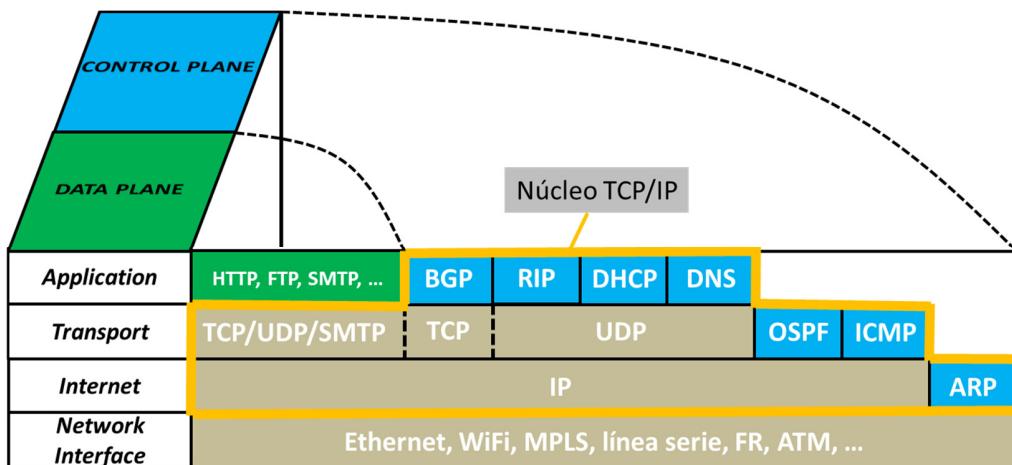
---

Internet no es sólo una amalgama de redes, sino también de tráficos. Monitorizando cualquier segmento de la misma, podemos descubrir flujos de paquetes generados por una gran variedad de protocolos. En un primer intento de clasificación, podemos distinguir entre los flujos derivados de la ejecución de las aplicaciones que manejan los usuarios desde sus dispositivos finales, y aquellos que provienen de la ejecución de protocolos que tienen una función de control. Estos últimos flujos suelen tener lugar entre dispositivos intermedios (fundamentalmente, *routers*), o entre dispositivos intermedios y dispositivos finales, y aunque nada tienen que ver con el tráfico asociado a las aplicaciones, son necesarios para que la gran red funcione. Unos y otros flujos comparten la arquitectura de capas TCP/IP y, en definitiva, la infraestructura hardware y software de Internet. Más formalmente, podemos afirmar que el tráfico de Internet se desglosa en dos planos:

- **Plano de datos.** Este plano comprende fundamentalmente los mensajes generados por las aplicaciones, es decir, los llamados **mensajes de usuario**. No obstante, esto no significa que solamente formen parte de este plano aquellos paquetes cuya cabecera de capa más alta corresponda a un protocolo de aplicación. Por ejemplo, la descarga de una página web se realiza a través de múltiples fragmentos, que salvo el último están directamente precedidos por una cabecera TCP. Estos segmentos TCP, así como los segmentos TCP ACK que dan acuse de recibo de los primeros, así como cualquier PDU vinculada al proceso de descarga de la página web, pertenecen también al plano de datos.
- **Plano de control.** Este plano contiene todo el tráfico de **mensajes de control** generados por ciertos protocolos cuya función conjunta es garantizar la conectividad de las sesiones establecidas por los usuarios. Dichos protocolos se pueden agrupar en dos categorías: los protocolos de encaminamiento y los protocolos auxiliares o satélites de IP. Un **protocolo de encaminamiento** define la sintaxis, la semántica y las reglas de intercambio de mensajes entre *routers* para la realización de la función de encaminamiento. Estos mensajes contienen la información topológica requerida por el **algoritmo de encaminamiento** que se ejecuta en cada uno de dichos *routers*, típicamente un algoritmo procedente de la Teoría de Grafos. Así pues, algoritmo y protocolo de encaminamiento son los dos elementos que conforman la función de encaminamiento en cada *router*, y de hecho suelen estar integrados en la misma componente software (aunque es referida únicamente como protocolo de encaminamiento). Ejemplos de protocolos de encaminamiento son RIP (*Routing Information Protocol*), OSPF (*Open Shortest Path First*), BGP (*Border Gateway Protocol*), y otros. Los **protocolos auxiliares** son más heterogéneos, pues cada uno tiene una función muy específica que nada tiene que ver con las funciones de los demás, salvo el hecho de que todos proporcionan un apoyo indispensable a la capa IP. Entre ellos, los más importantes son DHCP, ARP, ICMP y DNS. En la Sección 10.3 se describen más detalladamente estos protocolos.

La **Figura 10.1** muestra el desglose de la arquitectura TCP/IP en los planos de datos y de control, así como los protocolos más importantes en cada ámbito. Observamos que, efectivamente, los protocolos de transporte, el protocolo IP y los protocolos que gobiernan el acceso a los diferentes

tipos de redes son compartidos por ambos planos. La línea amarilla define el **núcleo TCP/IP**, es decir, el conjunto formado por los protocolos de transporte, el protocolo IP y los protocolos específicos del plano de control. Estos últimos aparecen en diferentes niveles, según el protocolo previsto en cada caso como proveedor del servicio de conexión. Es decir, los protocolos BGP, RIP, DHCP y DNS fueron diseñados para utilizar los servicios de la capa de transporte, y por tanto sus mensajes<sup>1</sup> son encapsulados en segmentos de transporte, TCP en el primer caso, y UDP en los restantes; lo mismo puede afirmarse sobre OSPF e ICMP con respecto a IP, y sobre ARP con respecto a la capa de acceso a red (en realidad, ARP funciona solamente para redes Ethernet y WiFi). No obstante, no puede afirmarse que, por ejemplo, DHCP, sea un protocolo de aplicación como HTTP; simplemente, están al mismo nivel, uno en cada plano, porque el proceso de encapsulado es el mismo para ambos.



**Figura 10.1.** Plano de datos y plano de control de la arquitectura TCP/IP. Existen más protocolos en ambos planos, pero los mostrados son los más importantes.

Probablemente, el protocolo más importante del núcleo, y clave del éxito de Internet, es el protocolo IP. A continuación, se describe la versión IPv4 (RFC 791) todavía hoy más extendida que la nueva versión IPv6 (RFC 2460).

## 10.2. EL PROTOCOLO IPv4

El **protocolo IP** es el principal responsable de que cualquier pareja de dispositivos puedan conectarse entre sí a través de Internet, independientemente de su ubicación geográfica y de la red a la que cada uno está directamente conectado. A diferencia de TCP, considerado más adelante, el protocolo IP ofrece un **servicio no orientado a la conexión**, lo que significa que no existe ninguna fase de establecimiento de la conexión antes de que tenga lugar el intercambio de paquetes IP entre los dos extremos de la comunicación. En el caso de IP, esta fase podría existir, por ejemplo, para que la red encontrara una ruta (cadena de *routers*) para dicha conexión, a través de la cual se

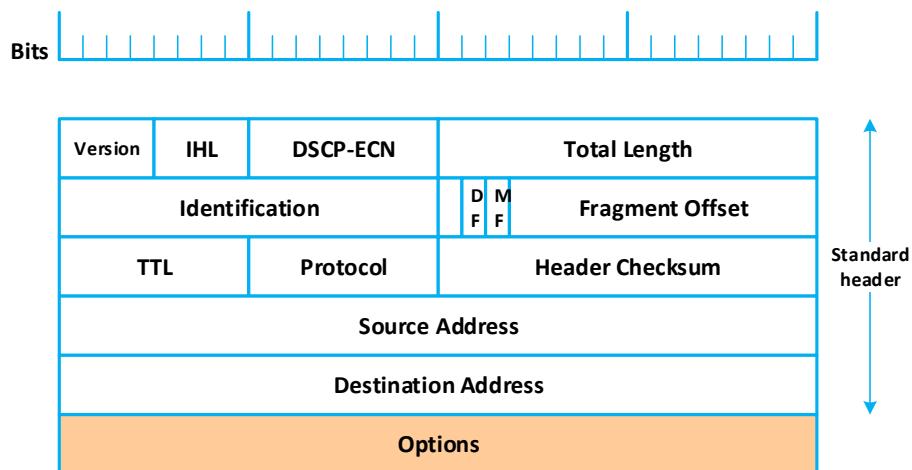
<sup>1</sup> El término mensaje designa la PDU de cualquier protocolo de aplicación y también de cualquier protocolo del plano de control.

encaminaran a continuación todos los paquetes de la misma. Sin embargo, IP fue diseñado de manera que cualquier dispositivo final pudiera enviar inmediatamente cada flujo de paquetes IP a su destino, sin incurrir en el retardo y el consumo de recursos que las múltiples fases de establecimiento de la conexión conllevarían. Por consiguiente, cada *router* decide el encaminamiento de cada paquete de manera individual e independiente al resto de paquetes de la misma conexión. La decisión de encaminamiento consiste en consultar la dirección IP de destino del paquete, buscar la entrada correspondiente en la tabla de encaminamiento, que es básicamente una lista de correspondencias entre destinos y puertos de salida, y reenviar el paquete al puerto de salida indicado en dicha tabla. El puerto de salida conduce al siguiente *router* y, por tanto, define el **siguiente salto** en la ruta hacia el destinatario. Puesto que el estado de la red es cambiante, y las tablas de encaminamiento de los *routers* son dinámicas, los paquetes de una misma conexión no tienen por qué ser encaminados necesariamente por la misma ruta, y de hecho pueden alcanzar el destinatario por rutas distintas e incluso llegar desordenados. Es el precio que se paga por evitar el retardo de una fase de establecimiento de la conexión, que por otra parte permitiría vincular cada paquete a una ruta y no a un destino. Por el hecho de que puede ser encaminado individualmente a partir de su dirección de destino, el paquete IP se denomina también **datagrama IP**.

A continuación, vemos dos aspectos relevantes del protocolo IPv4, como son el formato de los datagramas y la función de fragmentación.

#### Formato de la cabecera IPv4

El formato de la cabecera IPv4 se muestra en la **Figura 10.2**. Como puede observarse, su tamaño estándar es de 5 palabras de 32 bits. El tamaño y significado de los campos se describe a continuación:



**Figura 10.2.** Formato de la cabecera IPv4. Esta forma de mostrarlo, mediante palabras de 32 dispuestas verticalmente, es característica de todos los protocolos del núcleo TCP/IP. La regleta superior ayuda a determinar el tamaño de los diferentes campos.

- **Version** (4 bits). Indica la versión del protocolo IP a la que pertenece el datagrama. Este campo facilita el reconocimiento de la versión con la que ha sido elaborado el datagrama al software IP de los equipos de red y los equipos finales. Además, el tamaño de este campo posibilita que el protocolo IP pueda evolucionar a largo plazo.
- **IHL (Internet Header Length)** (4 bits). Este campo contiene la longitud total de la cabecera en palabras de 32 bits. Permite que el módulo IP que tenga que des-encapsular el campo de datos del paquete pueda determinar dónde termina la cabecera y empieza dicho campo, ya que la longitud de la cabecera puede variar dependiendo de la presencia de campos opcionales. De todos modos, el valor habitual y mínimo de este campo es 5, correspondiendo al tamaño estándar de 20 B de las cabeceras IPv4. El tamaño máximo de una cabecera es  $15 \cdot 32 = 480$  bits, es decir, 60 B.
- **DSCP (Differentiated Services Code Point) - ECN (Explicit Congestion Notification)** (6 + 2 bits). Este campo ha experimentado hasta 5 revisiones a lo largo de los años, recogidas en sus correspondientes documentos RFC. Está relacionado con los requisitos de calidad de servicio (*throughput*, retardo) de la aplicación que generó el datagrama. Fue diseñado especialmente para ofrecer un tratamiento prioritario a los paquetes correspondientes a servicios de *streaming* y/o de carácter interactivo, como por ejemplo los de voz sobre IP (VoIP). En la versión actual, los 6 primeros bits indican la clase de servicio requerida por la aplicación (subcampo DSCP), mientras que los 2 bits restantes contienen información relativa a la función de control de congestión (subcampo ECN), ya que se utilizan para indicar si el datagrama ha experimentado congestión en su paso a través de la red. Los detalles de este campo y de la función de control de congestión en Internet se estudian más adelante.
- **Total Length** (16 bits). Contiene la longitud total del datagrama (cabecera y campo de datos) en bytes. Permite que los datagramas puedan tener una longitud variable, hasta un máximo de 65535 bytes.
- Los campos correspondientes a la segunda palabra de 32 bits de la cabecera IP están todos relacionados con la función de fragmentación que se describe más adelante. Concretamente, son los campos **Identification** (16 bits), **Flags** (3 bits) y **Fragment Offset** (13 bits). El campo **Identification** se utiliza para identificar los diversos datagramas provenientes de la fragmentación de un mismo datagrama original. El campo **Flags** contiene 3 bits, de los cuales el primer bit está en desuso, el segundo bit es **DF (Don't Fragment)**, que cuando está activado indica a los *routers* que el datagrama no puede ser fragmentado, y el tercer bit es **MF (More Fragments)**, cuya activación especifica que el datagrama proviene de un fragmento que no es el último de la secuencia de fragmentos que hay que ensamblar para reconstituir el datagrama original en el destinatario. Cuando este bit aparezca desactivado después de una cadena de datagramas en la que estaba activado, el destinatario sabe que ha llegado el último fragmento. Finalmente, el campo **Fragment Offset** contiene información relativa a la posición del fragmento actual dentro del datagrama original (antes de ser fragmentado).
- **TTL (Time To Live)** (8 bits). Este campo sirve para evitar que los datagramas circulen permanentemente dentro de Internet, sin alcanzar su destinatario, como consecuencia de la formación de bucles de encaminamiento. Estos bucles constituyen una anomalía que puede

obedecer a diversas causas. Por ejemplo, condiciones de tráfico muy cambiantes, que provocan una continua actualización de las tablas de encaminamiento de los *routers*, y una rápida redefinición de los destinos inmediatos de los datagramas que los lleva a moverse formando círculos. Otra posible causa es un error de configuración de una ruta estática en algún tramo de la conexión. Cuando un paquete es emitido por primera vez, el valor del campo TTL corresponde a una cierta potencia de 2, y cada vez que llega a un *router*, éste lo reduce en una unidad antes de reenviarlo. Si un *router* recibe un datagrama con el campo TTL igual a 1, lo reduce a 0 y lo elimina de la red, enviando una notificación al dispositivo que lo generó (mediante un mensaje de control del protocolo ICMP que se describe más adelante). En consecuencia, el valor TTL adquiere la significación del máximo número de saltos que puede experimentar un datagrama en Internet, entendiendo por salto el paso por un *router*<sup>2</sup>.

- **Protocol** (8 bits). Este campo sirve para identificar el protocolo de capa superior encapsulado en el datagrama. De esta manera, el módulo IP que reciba el datagrama y tenga que desencapsular su campo de datos, sabe a qué protocolo entregarlo. Los identificadores utilizados son globalmente reconocidos en Internet. Inicialmente se hallaban recogidos en el documento RFC 1700, pero actualmente se puede encontrar la lista de todos los valores asignados en la base de datos del IANA (*Internet Assigned Numbers Authority*)<sup>3</sup>. Ejemplos de valores asignados son el 6 para el protocolo TCP y el 17 para UDP.
- **Header Checksum** (16 bits). En este campo se ubica un código de protección contra errores exclusivamente de la cabecera, dada la repercusión relativamente importante que puede tener un bit erróneo dentro de la misma. Nótese que forzosamente hay que recalcular este campo en cada *router*, puesto que como mínimo el valor del campo TTL habrá cambiado. En la cabecera IP no hay números de secuencia, por lo que la función de control de errores consiste simplemente en que el módulo IP receptor descarta el paquete cuando detecta que su cabecera es errónea.
- Finalmente vienen las direcciones de los computadores origen (*Source Address*) y destino (*Destination Address*) del datagrama. Son las llamadas direcciones IP, cuya longitud es de 32 bits (4 bytes) en IPv4.

#### Fragmentación de datagramas IPv4

Uno de los aspectos característicos de Internet es la heterogeneidad de redes que la conforman. Hay muchos tipos de redes a las que un computador puede conectarse para acceder a Internet, y hay muchos tipos de redes por las que pueden transitar los datagramas en su camino de origen a destino. Cada tipo de red tiene su valor *MTU* (Sección 7.2), el cuál determina el datagrama IP más largo que puede ser encapsulado en su estructura de trama. A su vez, este valor determina el tamaño máximo del campo de datos de los segmentos de transporte, típicamente segmentos TCP. Ese tamaño máximo es el valor *MSS*, relacionado con *MTU* a través de la ecuación (7.1). Durante la fase de establecimiento de una conexión TCP, por ejemplo, para iniciar una sesión web (Sección

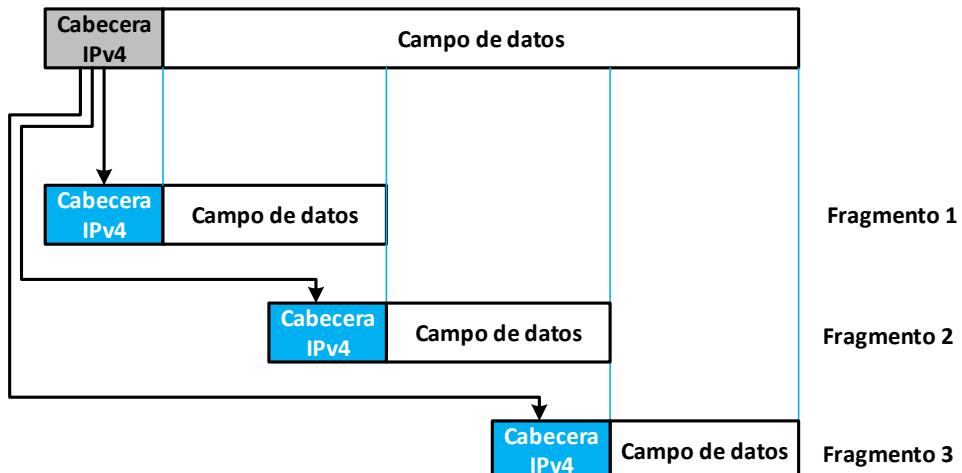
---

<sup>2</sup> El valor TTL inicial (fijado por el computador origen) depende de la implementación IP, pero suele ser 64 o superior.

<sup>3</sup> [http://www.iana.org/assignments/protocol\\_numbers/](http://www.iana.org/assignments/protocol_numbers/)

7.4), los dos extremos pueden acordar el valor *MSS* más restrictivo de los dos, o simplemente cada uno utilizar el suyo. En cualquier caso, es posible que el datagrama tenga que ser reenviado por un *router* intermedio a través de una red cuyo valor de *MTU* es lo suficientemente pequeño como para que el datagrama no pueda ser encapsulado. En este caso, el *router* tiene que fragmentar el datagrama (esta opción se eliminó al desarrollar IPv6), de manera que los múltiples datagramas resultantes tengan un tamaño compatible con la *MTU* de la red de siguiente salto. El proceso de fragmentación está diseñado de manera que el módulo IPv4 del host destinatario pueda reconstruir (ensamblar) el datagrama original a partir de todos los fragmentos, y des-encapsular a continuación el segmento TCP como si la fragmentación no hubiera tenido lugar. Ningún *router* puede realizar el ensamblado (para mantener la simplicidad del proceso de fragmentación y ensamblado, y evitar además que los *routers* incurran en retardos demasiado grandes).

La fragmentación de datagramas se lleva a cabo tal como se muestra en la **Figura 10.3**. No debe confundirse esta fragmentación con la segmentación que realiza el módulo TCP sobre los mensajes de aplicación en el host origen. Como puede observarse, el campo de datos del datagrama original se divide en varios fragmentos iguales, excepto el último que puede ser más pequeño, y a cada fragmento se le añade una cabecera IPv4, con lo que se forma un nuevo datagrama.



**Figura 10.3.** Fragmentación de datagramas IPv4.

La cabecera de cada fragmento es bastante parecida a la cabecera del datagrama original, pero presenta algunas modificaciones que conviene señalar. Estas modificaciones están orientadas a facilitar el proceso de ensamblado de los fragmentos para la reconstrucción del datagrama original. Es aquí donde entran en juego los campos de la segunda palabra de 32 bits de la cabecera IPv4. Concretamente, el campo `Identification` contiene un número de identificación que es fijado por el computador origen al crear el datagrama original (típicamente dicho computador incrementa en una unidad el valor de este campo por cada datagrama enviado). Cuando un *router* fragmenta un datagrama, la cabecera de cada fragmento contiene las mismas direcciones IP origen y destino, y el mismo valor del campo `Identification`, que el datagrama original. Se establece así el vínculo entre los datagramas descendientes y el datagrama del cual proceden. Entonces, el computador

destinatario utiliza el valor `Identification` para reconocer todos los fragmentos de un mismo datagrama y ensamblarlos. El *flag MF* vale 1 en todos los fragmentos excepto el último; cuando el computador destinatario recibe un fragmento con el flag `MF` a 0, sabe que con ese fragmento se completa el proceso de ensamblado. Puesto que los fragmentos pueden llegar desordenados (como datagramas que son, pueden seguir caminos distintos dentro de Internet), o incluso algunos pueden perderse, el campo `Fragment Offset` indica la posición del fragmento dentro del datagrama original, en unidades de 8 B. Esto implica que el tamaño del campo de datos de cada fragmento, excepto el último, tiene que ser un múltiplo de 8 B.

Para entender mejor el funcionamiento de estos campos, consideremos un ejemplo numérico en el que un datagrama se divide en 3 fragmentos, como en la [Figura 10.3](#). Supongamos que el campo de datos del datagrama tiene una longitud de 3500 bytes, y que este datagrama llega a un *router* que debe reenviarlo por un enlace Ethernet, cuya *MTU* es de 1500 bytes. En la [Tabla 10.1](#) se muestra la especificación de los campos `Identification`, `MF` y `Fragment Offset` del datagrama correspondiente a cada fragmento. Se ha supuesto que el computador origen asignó el valor 212 al campo `Identification` del datagrama original. El valor 148 del campo `Fragment Offset` del segundo fragmento proviene de la división de 1184 entre 8 (recuérdese que se especifica en múltiplos de 8 B), y el valor 296 en el tercer fragmento indica el desplazamiento equivalente a dos grupos de 1184 bytes ( $296 = 148 \cdot 2$ ).

**Tabla 10.1.** Especificación de los campos de fragmentación en la cabecera de los datagramas descendientes de un datagrama IPv4 original.

Datagrama	Tamaño (B)	Identification	MF	Fragment Offset
1	1184 + 20 (cabecera IPv4)	212	1	0
2	1184 + 20 (cabecera IPv4)	212	1	148
3	1132 + 20 (cabecera IPv4)	212	0	296

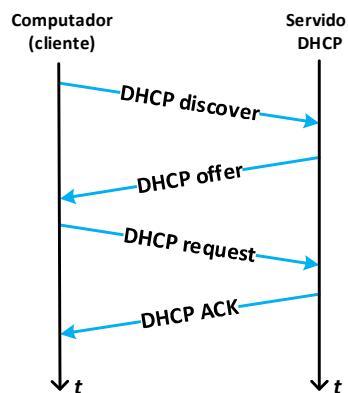
Si bien los tres campos relacionados con la fragmentación facilitan el correcto ensamblado de los datagramas en el módulo IP destinatario, no lo garantizan. En caso de que uno o varios datagramas descendientes se pierdan, el datagrama original no puede ensamblarse y se elimina, con lo que ya no se entrega ningún segmento al módulo de transporte. No obstante, este módulo de transporte, si es TCP, implementa el mecanismo de control de errores necesario para requerir la retransmisión del segmento, el cual volverá a ser encapsulado en un nuevo datagrama original.

### 10.3. PROTOCOLOS AUXILIARES

Los **protocolos auxiliares**, juntamente con los protocolos de encaminamiento, tienen la función de garantizar la conectividad entre cualquier conjunto de usuarios a través de Internet. Los protocolos auxiliares ejecutan funciones muy específicas, independientes entre sí, sin otro denominador común que contribuir a esa conectividad. Los más importantes son DHCP, ARP, ICMP y DNS. A continuación, se describen en este orden.

## El protocolo DHCP

El **protocolo DHCP** (*Dynamic Host Configuration Protocol*), recogido en el RFC 2131, se utiliza para la configuración dinámica de los parámetros IP básicos de un computador, que no son más que la dirección IP, el prefijo (máscara), la dirección IP de la puerta de enlace predeterminada (*router* de primer salto o *gateway* por defecto) y la dirección IP de un servidor DNS. Con estos parámetros, el computador adquiere presencia en Internet y es operativo tanto para establecer conexiones como para recibirlas. DHCP es una pieza más del sistema operativo de cualquier dispositivo final, y por defecto está configurado para que se ejecute cada vez que dicho dispositivo se conecta a una red. Siguiendo el modelo cliente-servidor, el dispositivo solicita la configuración IP a un servidor DHCP de forma automatizada, es decir, sin intervención del usuario ni del administrador de red. La alternativa a esta estrategia consistiría en que el administrador o el propio usuario asignara manualmente y de forma estática los parámetros IP al computador; sin embargo, esto supondría el consumo de tantas direcciones IP como computadores y dispositivos finales en general, cuando normalmente no van a estar todos conectados al mismo tiempo. La asignación de **direcciones IP estáticas** suele reservarse para aquellos dispositivos finales fijos y permanentemente conectados a la red de la organización, como es el caso de servidores e impresoras. También suelen asignarse direcciones IP estáticas a las interfaces o puertos de los *routers*. Para los dispositivos de los usuarios, que se conectan y desconectan frecuentemente, la mejor estrategia es la configuración dinámica.



**Figura 10.4.** Diálogo cliente-servidor en la ejecución del protocolo DHCP.

El protocolo DHCP es básicamente el procedimiento de cuatro etapas mostrado en el diagrama temporal de la **Figura 10.4**. Cada una se corresponde con el envío, en un sentido u otro, de un mensaje DHCP. A modo de ejemplo, en la **Tabla 10.2** se exponen los contenidos más relevantes de las capas implicadas en el intercambio de mensajes DHCP entre el computador A y el servidor DHCP del escenario de la **Figura 10.5**. Tomando como referencia la **Figura 10.4**, la **Figura 10.5** y la **Tabla 10.2**, podemos describir las cuatro etapas como sigue:

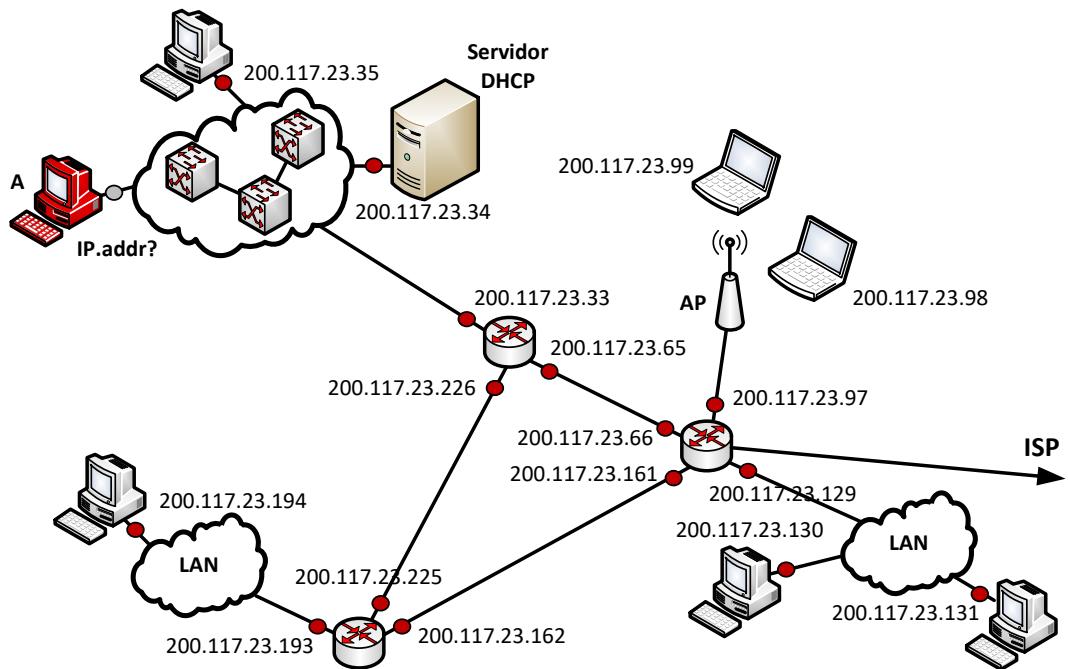
1. Etapa de descubrimiento de servidores (envío del mensaje `DHCP discover`). El protocolo DHCP está concebido bajo la idea de que el computador que lo ejecuta puede acceder a varios servidores DHCP. Esta primera etapa tiene pues, como objetivo, descubrir todos los servidores

disponibles. Evidentemente, el computador desconoce las direcciones IP de los servidores DHCP, y por supuesto carece de dirección IP propia (el objetivo del protocolo es precisamente obtener una). Por ello, utiliza la dirección IP origen 0.0.0.0, que es la dirección especial reservada para que un *host* sin dirección IP pueda auto-identificarse, y la dirección IP de destino 255.255.255.255, que es la dirección de *broadcast*<sup>4</sup>. En cuanto al protocolo UDP, el computador utiliza como puertos origen y destino los valores 68 y 67 respectivamente (son los números de puerto UDP asignados al protocolo DHCP). Finalmente, el mensaje DHCP contiene algunos parámetros significativos, como son su tipología (DHCPDISCOVER en esta primera etapa), y un identificador de transacción, contenido en el campo `xid`, cuyo valor elige aleatoriamente el computador (721 según la tabla). En caso de que haya varios clientes que invocan el servicio DHCP, este identificador servirá para emparejar correctamente cada mensaje de respuesta con su correspondiente mensaje previo de petición. Es importante que la selección del identificador de transacción se realice de manera que queda minimizada la probabilidad de coincidencia con el valor seleccionado por otro computador; por ello, a partir de un primer valor elegido aleatoriamente, los siguientes valores son seleccionados secuencialmente (obsérvese que, en el siguiente mensaje generado por el computador, éste utiliza el valor 722).

**Tabla 10.2.** Contenidos más relevantes de las cabeceras de los protocolos implicados (UDP, IP) en el intercambio de mensajes DHCP entre el computador A y el servidor DHCP de la [Figura 10.5](#).

Mensaje	Direcciones origen	Direcciones destino	Contenido DHCP
DHCP discover	Puerto = 68 @IPv4 = 0.0.0.0	Puerto = 67 @IPv4 = 255.255.255.255	DHCP message type = DHCPDISCOVER yiaddr = 0.0.0.0 xid = 721
DHCP offer	Puerto = 67 @IPv4 = 200.117.23.34	Puerto = 68 @IPv4 = 255.255.255.255	DHCP message type = DHCPOFFER yiaddr = 200.117.23.36 xid = 721 server ID = 200.117.23.34 IP address lease time = subnet mask = 255.255.255.224 local DNS server = default router = 200.117.23.33
DHCP request	Puerto = 68 @IPv4 = 0.0.0.0	Puerto = 67 @IPv4 = 255.255.255.255	DHCP message type = DHCPREQUEST yiaddr = 200.117.23.36 xid = 722 server ID = 200.117.23.34 IP address lease time = subnet mask = 255.255.255.224 local DNS server = default router = 200.117.23.33
DHCP ACK	Puerto = 67 @IPv4 = 200.117.23.34	Puerto = 68 @IPv4 = 255.255.255.255	DHCP message type = DHCPACK yiaddr = 200.117.23.36 xid = 722 server ID = 200.117.23.34 IP address lease time = subnet mask = 255.255.255.224 local DNS server = default router = 200.117.23.33

<sup>4</sup> Aunque sea un *broadcast* en el contexto de la red Internet, como mucho el paquete alcanza todos los dispositivos finales de la red de la organización, pero nunca es reenviado a la Internet pública. Es lo que se denomina un *broadcast limitado*.



**Figura 10.5.** Escenario de referencia para la descripción del protocolo DHCP. Se señalan las direcciones IPv4 de diversos dispositivos finales y de las interfaces de los *routers* de esta red corporativa.

2. Recepción de ofertas de servidores (recepción de mensajes *DHCP offer*). Cada servidor que recibe un mensaje *DHCP discover*, puede responder con un mensaje *DHCP offer* que contiene la dirección IP propuesta para el computador, a través del campo *yiaddr (your IP address)*, y otros parámetros de configuración como el tiempo de vida o de validez de la dirección IP asignada (*IP address lease time*)<sup>5</sup>, la máscara de subred (*subnet mask*), la dirección IP del servidor DNS local (*local DNS server*) y la dirección IP del *router* por defecto o puerta de enlace predeterminada (*default router*). La **puerta de enlace predeterminada** es la primera interfaz de *router* que debe cruzar un paquete cuando es enviado por un *host* a la Internet pública. La dirección IP de la puerta de enlace predeterminada es la dirección IP de dicha interfaz (cada interfaz de *router* tiene su propia dirección IP). La identidad del servidor (su dirección IP) también forma parte del contenido del mensaje *DHCP offer* y de los mensajes subsiguientes a través del campo *server ID*.

<sup>5</sup> Este campo es necesario para aprovechar direcciones IP que vuelven a estar disponibles, ya que fueron asignadas a computadores que posteriormente se desconectaron de la red. Su valor se expresa en segundos, y suele establecerse en varias horas o incluso días. Un computador puede renovar su dirección IP antes de que expire su tiempo de caducidad, mediante un nuevo mensaje *DHCP request*, y también puede liberar una dirección IP antes de desconectarse de la red, mediante otro tipo de mensaje denominado *DHCP release*.

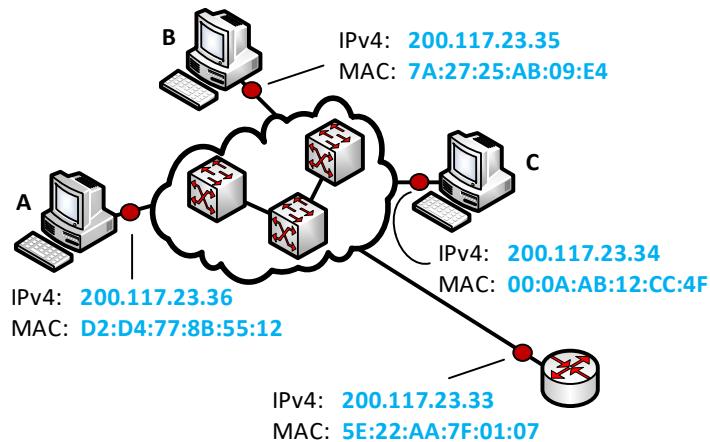
3. Petición DHCP (envío del mensaje `DHCP request`). Tras haber recibido una o varias ofertas de servidores DHCP, el computador selecciona una de ellas y envía un mensaje de petición al servidor que la ha generado, reproduciendo los parámetros de configuración.
4. Confirmación DHCP (envío del mensaje `DHCP ACK`). El servidor seleccionado contesta a la petición DHCP confirmando los parámetros de configuración. En este momento, los parámetros IP básicos del computador quedan configurados.

Por último, cuando un computador se conecta a una red que carece de servidor DHCP propio, los *routers* de la organización pueden actuar como intermediarios (*relay agents*) entre el computador y un servidor DHCP externo.

### El protocolo ARP

Hemos visto que cada computador conectado a Internet tiene dos niveles de direccionamiento: uno global (dirección IP), que lo identifica sin ambigüedad dentro de toda la red Internet, y uno local (dirección física o MAC), que lo identifica sin ambigüedad dentro de la red a la que está conectado directamente. La [Figura 10.6](#) muestra un ejemplo de escenario en el que se indican ambas direcciones para cada interfaz de *router*, puesto que ésta equivale a un dispositivo más en la correspondiente subred. El **protocolo ARP** (*Address Resolution Protocol*), descrito en el RFC 826, permite que un dispositivo conozca la dirección MAC de otro dispositivo conectado a la misma red, suponiendo que ya conoce su dirección IPv4. Por ejemplo, supongamos que el computador A de la [Figura 10.6](#) quiere enviar un paquete a una dirección IPv4 de la misma red Ethernet. Evidentemente A necesita conocer también la dirección MAC asociada a esa dirección IPv4, puesto que el paquete deberá ser encapsulado en una trama Ethernet, la cual contiene en su cabecera los campos de direcciones MAC origen y destino, que deberá llenar. Para ello, el computador A dispone de un módulo ARP que gestiona una tabla de correspondencias entre direcciones IPv4 y direcciones MAC de interfaces de la misma red. Es la **tabla ARP** o *ARP cache*. La [Figura 10.7](#) muestra una posible tabla ARP para el computador A. Obsérvese que cada entrada de la tabla va acompañada de un valor temporal llamado TTL (*Time To Live*), que indica el momento en que dicha entrada será eliminada de la tabla. Esta fecha de caducidad es necesaria dada la dinámica de conexiones y desconexiones que tienen lugar en cualquier red. Suele establecerse en 20 minutos desde el momento en que la entrada es insertada en la tabla, o desde la última renovación (cada vez que se hace uso de una entrada, se actualiza su fecha de caducidad). Si el paquete de A tiene como destino el computador B, cuya dirección IPv4 es 200.117.23.35, el módulo ARP sólo tiene que realizar una lectura de la tabla ARP para determinar la dirección MAC de B. Entonces ya puede encapsular el paquete y enviarlo. Pero, ¿qué ocurre si A quiere enviar un paquete al computador C, para el que no dispone de ninguna entrada en su tabla ARP, lo que equivale a decir que desconoce su dirección MAC? En este caso, el módulo ARP de A envía un mensaje de petición ARP (`ARP request`) y espera recibir un mensaje de respuesta ARP (`ARP response`) con la información solicitada. El protocolo ARP consiste en este intercambio básico de mensajes. Ambos mensajes tienen el mismo formato y se encapsulan en una trama de la capa de acceso a red (Ethernet o WiFi). En la [Tabla 10.3](#) se muestran los campos más relevantes de estos mensajes, así como las direcciones MAC origen y destino que aparecen en las tramas donde son encapsulados. Obsérvese que la petición ARP se envía en modo

*broadcast* (la dirección MAC de destino es FF:FF:FF:FF:FF:FF), lo cual es lógico pues se trata de interrogar a todos los dispositivos de la red con el objeto de averiguar qué dirección MAC corresponde a una dirección IPv4 dada (es como cuando alguien pregunta algo en voz alta). Además, el hecho de que se envíe en modo *broadcast* puede ser aprovechado por el resto de dispositivos conectados a la red, ya que pueden actualizar su tabla ARP con una entrada que contenga la asociación entre direcciones MAC e IPv4 del dispositivo que ha generado la petición ARP.



**Figura 10.6.** Escenario de referencia para la descripción del protocolo ARP. Se señalan las direcciones IPv4 y MAC de diversos dispositivos finales y de la interfaz del *router* que conecta la red a otras redes de la organización o a la Internet pública. El protocolo ARP se ejecuta exclusivamente en el ámbito de esa red, de manera que un mensaje ARP nunca cruza un *router*.

Tabla ARP		
Dirección IPv4	Dirección MAC	TTL
200.117.23.35	7A:27:25:AB:09:E4	22:10:00
200.117.23.33	5E:22:AA:7F:01:07	22:14:30

**Figura 10.7.** Tabla ARP del computador A. Se construye automáticamente, sin necesidad de intervención del administrador de redes. Se dice por ello que ARP es un protocolo *plug-and-play*.

**Tabla 10.2.** Contenidos más relevantes de los mensajes ARP y de los campos de direcciones MAC origen y destino de las tramas que los encapsulan.

Mensaje	Direcciones MAC	Contenido ARP
ARP request	Origen = D2:D4:77:8B:55:12 Destino = FF:FF:FF:FF:FF:FF	Sender Hardware Address (SHA) = D2:D4:77:8B:55:12 Sender Protocol Address (SPA) = 200.117.23.36 Target Protocol Address (TPA) = 200.117.23.34
ARP response	Origen: 00:0A:AB:12:CC:4F Destino: D2:D4:77:8B:55:12	Sender Hardware Address (SHA) = D2:D4:77:8B:55:12 Sender Protocol Address (SPA) = 200.117.23.36 Target Hardware Address (THA) = 00:0A:AB:12:CC:4F Target Protocol Address (TPA) = 200.117.23.34

Con respecto a la **Tabla 10.2**, obsérvese que la dirección MAC solicitada aparece dos veces, como dirección origen del mensaje ARP response, y en el campo THA de dicho mensaje. Esto no es una redundancia, sino una consecuencia de la estructura basada en capas de las comunicaciones. La dirección MAC solicitada tiene que aparecer como dirección origen de la trama que encapsula el mensaje, y como parte de dicho mensaje a fin de que el módulo ARP que generó la petición obtenga la información solicitada en la misma. Este módulo no puede obtener la dirección MAC solicitada a partir de la cabecera de la trama, puesto que recibe el mensaje ARP response una vez des-encapsulado de dicha trama.

Para terminar esta discusión sobre ARP, podemos preguntarnos qué ocurre si el computador A de la **Figura 10.6** quiere enviar un paquete a un computador fuera de la red a la que está conectado, es decir:

- ¿Necesita A conocer la dirección MAC de este otro computador?
- ¿Cuál debe ser la dirección MAC de destino de la trama enviada por A?

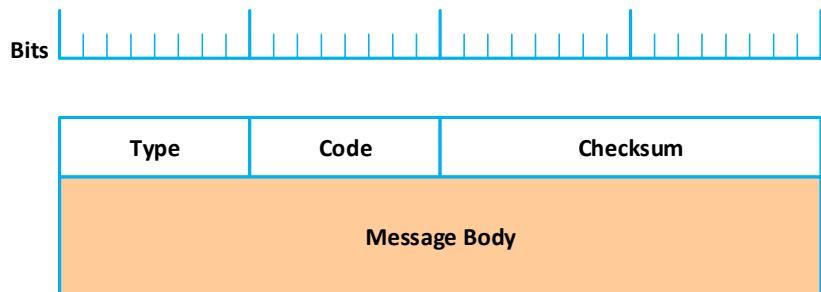
La respuesta a la primera pregunta es afortunadamente negativa, pues la dirección MAC del computador de destino sólo deberá ser conocida por el *router* local de la red a la que esté conectado dicho computador. Por otro lado, puesto que el paquete generado por A deberá cruzar el *router* mostrado en la **Figura 10.6**, la dirección MAC de destino de la trama enviada por A deberá ser la que corresponde a la interfaz con dirección IPv4 200.117.23.33 de ese *router*. Ésta es precisamente la dirección de la puerta de enlace predeterminada para A, la cual es conocida por A, pues forma parte de su configuración IP. Si todavía no conoce la dirección MAC de su puerta de enlace, A puede enviar una petición ARP para averiguarla.

#### El protocolo ICMPv4

El **protocolo ICMPv4** (*Internet Control Message Protocol*, versión para IPv4), definido en el RFC 792, es el protocolo utilizado por los computadores y, sobretodo, por los *routers*, para intercambiar información de control relativa a la capa Internet. Como se deduce de la **Figura 10.1**, los mensajes ICMPv4 se encapsulan en datagramas IPv4, en cuyo caso el campo *Protocol* de tales datagramas vale 1. El catálogo de mensajes ICMPv4 es bastante amplio. Una primera clasificación permite agruparlos en las dos categorías siguientes:

- **Mensajes de error** o alertas, en la mayoría de los casos enviados por un *router* a un *host*. En este caso, el *router* proporciona información acerca de alguna incidencia en un paquete originado en el host.
- **Mensajes de diagnóstico**. Son mensajes utilizados para obtener alguna información sencilla acerca de un tramo de Internet.

En la **Figura 10.8** se muestra el formato de un mensaje ICMPv4, cuyos tres primeros campos son fijos:



**Figura 10.8.** Formato de la cabecera de los mensajes ICMPv4.

- Campos `Type` (8 bits) y `Code` (8 bits). Conjuntamente, estos dos campos codifican el tipo de mensaje ICMPv4, es decir, su significado. Para un mismo valor del campo `Type`, los diversos valores del campo `Code` proporcionan una información más específica.
- `Checksum` (16 bits). Este campo para el chequeo de errores protege el mensaje ICMPv4 completo. Obedece al mismo algoritmo utilizado en IPv4.

El resto del mensaje es el `Message Body`, cuyo tamaño y contenido depende del tipo de mensaje ICMPv4. En el caso de los mensajes de error, este contenido consiste en una copia de la cabecera del paquete IPv4 que provocó la incidencia, y los primeros 64 bits del campo de datos de dicho paquete.

En la **Tabla 10.3** se describen los principales tipos de mensajes ICMPv4. Como podemos observar, el campo `Code` representa efectivamente un nivel de especificación más preciso que el campo `Type`; por ejemplo, el campo `Type` igual a 3 agrupa los mensajes ICMPv4 que pueden generarse cuando un datagrama no ha podido llegar a su destino, y en este caso el campo `Code` especifica la causa por la cual esto ha ocurrido.

Entre los mensajes de diagnóstico, los más utilizados son `echo request` y `echo reply`, a través del comando `ping` que viene instalado en los sistemas operativos. Con este comando, un administrador de redes puede probar la conectividad de la red de una organización, ejecutándolo sucesivas veces desde un computador a los diversos *routers* de la misma y a otros computadores. Concretamente, el comando `ping` provoca el envío de varios mensajes `echo request` al dominio o dirección IP especificados como argumento. El dispositivo examinado (puerto de *router* o *host*, en general), si está operativo y es accesible, devuelve un mensaje `echo reply`<sup>6</sup>. De esta manera, el usuario o administrador puede determinar si una determinada dirección IP es alcanzable o no. En la **Figura 10.9** se muestra un ejemplo de ejecución del comando `ping`.

<sup>6</sup> Aunque muchos computadores vienen ya con una configuración de la arquitectura TCP/IP que permite deshabilitar el envío de mensajes `echo reply`, a raíz de problemas de seguridad que han afectado a la propia implementación de dicha arquitectura.

**Tabla 10.3.** Principales mensajes ICMPv4<sup>7</sup>.

Campo Type	Campo Code	Mensaje	Descripción
0	0	echo reply	Utilizado por el programa ping de supervisión de red.
3	0	destination network unreachable	El mensaje ha sido generado por un <i>router</i> que carece de ruta para alcanzar la red del destinatario.
3	1	destination host unreachable	El mensaje ha sido generado por un <i>router</i> directamente conectado a la red del computador destinatario, pero por alguna razón no se puede conectar con dicho computador.
3	2	destination protocol unreachable	El mensaje ha sido generado por un computador destinatario que ha recibido un datagrama, pero no soporta el protocolo de transporte especificado en el campo Protocol de dicho datagrama.
3	3	destination port unreachable	El mensaje ha sido generado por un computador destinatario que ha recibido un datagrama que encapsula una PDU destinada a un número de puerto que no corresponde a ningún proceso ejecutado en el computador.
3	4	fragmentation needed and DF set	El mensaje ha sido generado por un <i>router</i> que recibió un datagrama más largo que la MTU de la interfaz de salida, pero con el bit DF ( <i>Don't Fragment</i> ) igual a 1.
3	6	destination network unknown	Red destinataria desconocida.
3	7	destination host unknown	Computador destinatario desconocido.
4	0	source quench	Típicamente este mensaje es generado por un <i>router</i> cuando recibe un datagrama que excede o está a punto de exceder la capacidad del <i>buffer</i> de salida al siguiente salto. El computador que recibe este mensaje tiene que reducir la velocidad a la que está enviando los datagramas. Se trata pues de un mensaje con funciones de control de congestión. Este mensaje también puede ser enviado por el computador destinatario, cuando recibe datagramas de un computador origen a velocidad excesiva. De todos modos, el uso de este tipo de mensaje quedó depreciado con el desarrollo de la función de control de congestión en TCP.
5	1	redirect for host	Típicamente este mensaje tiene lugar en escenarios donde dos o más <i>routers</i> están conectados a la misma LAN de un computador. Supongamos el caso de dos <i>routers</i> , R1 y R2, que interconectan la LAN del computador a sendas redes N1 y N2. Si el computador envía un datagrama destinado a la red N2 a través de R1, entonces R1 envía un mensaje ICMP de redirección al computador, indicándole que envíe todo el futuro tráfico a N2 a través de R2. Por tanto, este mensaje contribuye a corregir u optimizar la tabla de encaminamiento del computador.
8	0	echo request	Utilizado por el programa ping de supervisión de red.
9	0	router advertisement	Estos mensajes se utilizan para facilitar que los computadores puedan encontrar <i>routers</i> cercanos. Un computador debe conocer al menos la dirección de un <i>router</i> para poder enviar datagramas fuera de su red local.
10	0	router discovery	
11	0	TTL exceeded in transit	El mensaje ha sido generado por un <i>router</i> que ha descartado un datagrama porque su tiempo de vida ha expirado (el campo TTL de dicho datagrama ha alcanzado el valor 0).
11	1	fragment reassembly time exceeded	El mensaje ha sido generado por un computador de destino que no ha podido ensamblar todos los fragmentos de un datagrama antes de que expirara su temporizador de ensamblado.
12	0	parameter problem	El mensaje ha sido generado por un <i>router</i> o un computador que ha recibido un datagrama con una cabecera IP no válida (por ejemplo, con una opción incorrecta).
13	0	timestamp	Este mensaje es similar al mensaje echo request, pero con el campo Originate Timestamp de 32 bits, que indica el instante en milisegundos <sup>8</sup> en que el mensaje es enviado.
14	0	timestamp reply	Este es el mensaje de respuesta al anterior (type=13 code=0). Además de reproducir el campo Originate Timestamp, contiene dos campos adicionales, Receive Timestamp y Transmit Timestamp, que indican respectivamente los instantes en que el mensaje anterior (type=13 code=0) fue recibido y el instante en que el presente mensaje es transmitido.

<sup>7</sup> La lista *online* completa se puede consultar en [www.iana.org/assignments/icmp-parameters](http://www.iana.org/assignments/icmp-parameters).

<sup>8</sup> Referidos a la medianoche UT (*Universal Time*) en el formato estándar.

```
C:\Users > ping www.youtube.com

Haciendo ping a youtube-ui.l.google.com [216.58.211.206] con 32 bytes de datos:
Respuesta desde 216.58.211.206: bytes=32 tiempo=45ms TTL=52
Respuesta desde 216.58.211.206: bytes=32 tiempo=43ms TTL=52
Respuesta desde 216.58.211.206: bytes=32 tiempo=69ms TTL=52
Respuesta desde 216.58.211.206: bytes=32 tiempo=52ms TTL=52

Estadísticas de ping para 216.58.211.206:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
                (0% perdidos),
Tiempos aproximados de ida y vuelta en milisegundos:
    Mínimo = 43ms, Máximo = 69ms, Media = 52ms
```

**Figura 10.9.** Ejemplo de ejecución del comando `ping` en Windows. Los 32 bytes constituyen el tamaño por defecto del cuerpo del mensaje ICMPv4 (Message Body), sea `echo request` o `echo reply`. Son bytes de relleno arbitrarios e irrelevantes a efectos del comando. Su número es configurable.

La **Figura 10.9** refleja el resultado de la ejecución de un `ping` en el modo comandos de un ordenador Windows. En general, la configuración específica de la utilidad `ping`, así como las opciones disponibles y el formato de presentación en pantalla, varían de una implementación a otra; no obstante, todas coinciden en lo fundamental, que es el envío de una serie de mensajes `echo request` y la recepción (o no) de sus correspondientes `echo reply`, con el objeto de constatar la conectividad con un dominio o dirección IP, además de obtener diversas estimaciones relacionadas: el tiempo de ida y vuelta y el número de saltos o *routers* entre los dos extremos de la comunicación. Esta última estimación se obtiene a partir del valor del campo `TTL` del paquete `echo reply`, suponiendo que se conoce el valor inicial de dicho campo (siempre una potencia de 2, típicamente 64). Todas las implementaciones de `ping` generan también una breve estadística acerca de los envíos realizados.

Otra utilidad de diagnóstico interesante es `traceroute` (Unix, Linux, macOS y algunos otros) o `tracert` (Windows), ya que permite trazar la ruta a través de Internet entre dos dispositivos. El comando `traceroute` provoca la generación de mensajes ICMP con el objeto de extraer información de encaminamiento relevante. Su funcionamiento es bastante ingenioso y consiste en varias iteraciones iniciadas en el computador donde se ejecuta (computador origen). En todas las iteraciones, el `traceroute` estándar envía una ráfaga de 3 datagramas IP ordinarios, los cuales tienen como destino el computador con el que se va a trazar la ruta y encapsulan un segmento UDP con un puerto de destino poco probable y un campo de datos irrelevante. En la primera iteración, el campo `TTL` de los datagramas se fija a 1, de manera que el primer *router* de la ruta los elimine al reducir en una unidad el valor de dicho campo y dejarlo a 0. Al enviar al computador origen los correspondientes mensajes ICMP del tipo `TTL exceeded in transit (type = 11 code = 0)`, el *router* se identifica mediante el nombre de dominio al que pertenece y su dirección IP. De esta forma, el computador origen averigua la identidad del primer *router* en la ruta hacia el destinatario.

El computador origen repite este procedimiento aumentando en cada iteración el campo TTL en una unidad, lo que le permite ir averiguando los sucesivos *routers* de la ruta hacia el destinatario. Cuando la última ráfaga de datagramas alcanza el computador de destino, éste devuelve sendos mensajes ICMPv4 del tipo destination port unreachable (type = 3 code = 3), ya que no reconoce el puerto de destino de tales datagramas. Con la recepción de estos mensajes ICMPv4, el computador origen determina que ya no es necesario enviar más datagramas de prueba, y termina la ejecución del comando completando la presentación de resultados por pantalla. El comando tracert es parecido, pero en lugar de encapsular segmentos UDP, los datagramas IPv4 encapsulan mensajes ICMPv4 del tipo echo request. En la **Figura 10.10** se muestra un ejemplo de ejecución del comando tracert. Como podemos observar, el resultado del comando consiste en una lista ordenada de todos los saltos detectados, para cada uno de los cuales muestra su identidad y las 3 medidas de tiempo de ida y vuelta (el computador origen activa un temporizador cada vez que transmite un datagrama de prueba y lo detiene al recibir el mensaje de error).

---

```
C:\Users\UIB>tracert www.columbia.edu

Traza a la dirección www.columbia.akadns.net [128.59.105.24]
sobre un máximo de 30 saltos:

 1 <1 ms <1 ms < 1 ms 172.22.0.7
 2 * * * Tiempo de espera agotado para esta solicitud
 3 1 ms <1 ms <1 ms GE2-0-0.uib.rt1.bal.red.rediris.es [130.206.197.17]
 4 5 ms 5 ms 5 ms UIB.SO5-0-0.uv.rt1.val.red.rediris.es [130.206.245.26]
 5 11 ms 11 ms 11 ms UV.AE5.telmad.rt4.mad.red.rediris.es [130.206.245.90]
 6 11 ms 11 ms 11 ms rediris.mx1.mad.es.geant.net [62.40.124.73]
 7 32 ms 31 ms 31 ms ae7.mx1.gen.ch.geant.net [62.40.98.67]
 8 40 ms 39 ms 40 ms ae1.mx1.fra.de.geant.net [62.40.98.109]
 9 47 ms 46 ms 46 ms ae1.mx1.ams.nl.geant.net [62.40.98.129]
10 121 ms 121 ms 121 ms xe-0-3-0.102.rtr.newy32aoa.net.internet2.edu [198.71.45.236]
11 121 ms 121 ms 121 ms nyc-7600-internet2-newy.nysernet.net [199.109.4.153]
12 121 ms 121 ms 121 ms columbia.nyc-7600.nysernet.net [199.109.4.14]
13 121 ms 121 ms 121 ms cc-core-1-x-nyser32-gw-1.net.columbia.edu [128.59.255.5]
14 122 ms 122 ms 121 ms cc-conc-1-x-cc-core-1.net.columbia.edu [128.59.255.210]
15 121 ms 121 ms 121 ms fathom.com [128.59.105.24]

Traza completa.
```

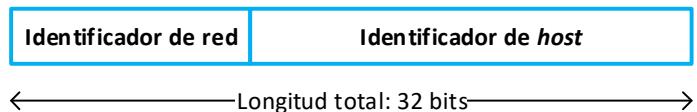
---

**Figura 10.10.** Ejemplo de ejecución del comando tracert en Windows. La Universidad de Columbia está en Nueva York, y por la evolución de los tiempos de ida y vuelta, podemos concluir que el cruce del Atlántico tiene lugar entre los saltos 9 y 10 (el comando se ejecutó desde un computador en Europa).

## 10.4. ADMINISTRACIÓN DE DIRECCIONES IPv4

Las direcciones IPv4 tienen una longitud de 32 bits y se especifican agrupándolos en 4 bytes, representando cada uno de ellos por su valor decimal y separándolos mediante un “.”. Un ejemplo es 175.35.212.7. Evidentemente, el valor decimal correspondiente a cada byte es siempre un número entre 0 y 255. Además, cada dirección IPv4 tiene una estructura jerárquica de 2 niveles:

- **Identificador de red.** Es la parte inicial de una dirección IPv4, y su longitud puede variar. Las direcciones IPv4 de todos los computadores conectados a una misma red tienen una parte común, que es el identificador de red.
- **Identificador de host.** Es la parte restante de una dirección IPv4, y permite identificar sin ambigüedad los diversos computadores conectados a una misma red.



**Figura 10.11.** Formato de las direcciones IPv4.

En la **Figura 10.11** se ilustra la estructura de una dirección IPv4. Cada organización obtiene su identificador de red del ISP con el que contrata los servicios de conexión a Internet. Cuanto más largo es el identificador de red, más corto es el identificador de host, y por tanto menos direcciones IPv4 individuales podrán distribuirse entre los computadores de la organización. Así pues, con la asignación del identificador de red, el ISP está asignando un bloque de direcciones de cierto tamaño, que deberá ser suficiente para cubrir las necesidades de conexión de la organización. En este punto, son varios los criterios que puede aplicar el administrador de redes:

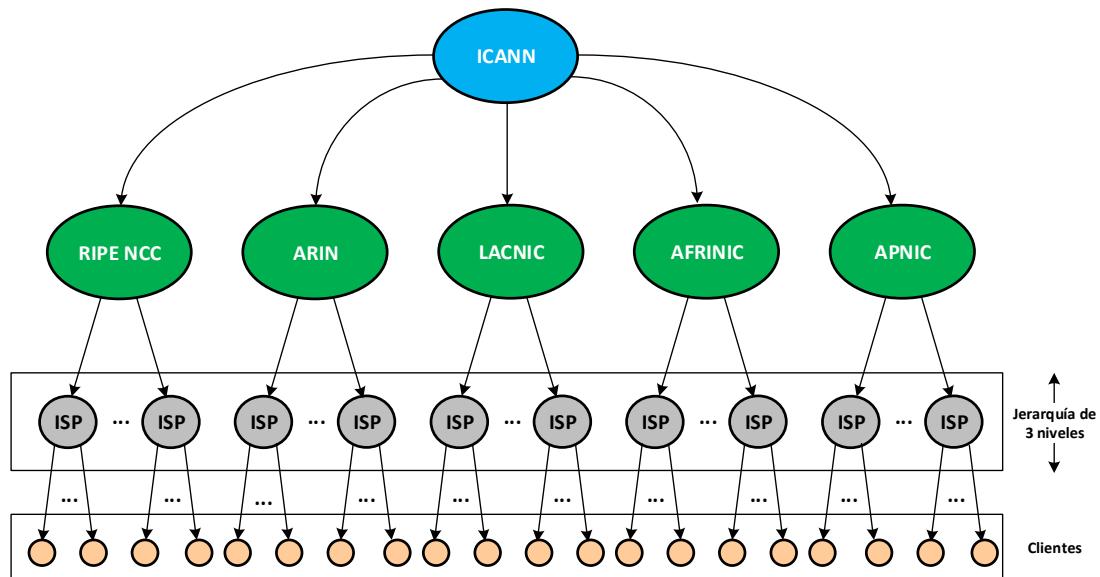
- Hacer uso de direcciones privadas (método de mapeado NAT descrito más adelante).
- Contratar un bloque de direcciones sólo para aquellos dispositivos que van a tener una dirección IPv4 estática, y hacer uso de direcciones privadas para el resto de dispositivos.
- Contratar un bloque mayor de direcciones IPv4 en función de la previsión de conexiones simultáneas.
- Contratar un bloque máximo, con tantas direcciones como dispositivos.

Dado que, desde principios de la década de los 90, las direcciones IPv4 empezaron a agotarse, y todavía no se ha extendido el uso de direcciones IPv6, el mapeado NAT se ha convertido en una estrategia de uso común, como mínimo en todas las redes domésticas (SOHO).

El Registro IP Regional o RIR (*Regional IP Registry*) es el encargado administrar y distribuir grandes bloques de direcciones IP en una región de escala continental. La jerarquía fundamental se muestra en la **Figura 10.12**. Como podemos observar, en la cabeza de la misma está la ICANN (*Internet Corporation for Assigned Names and Numbers*), una organización sin ánimo de lucro cuyos principios básicos de actuación están recogidos en el RFC 2050. ICANN es la máxima autoridad mundial en la asignación de bloques de direcciones IP, y también se encarga de la gestión de los servidores centrales de nombres de dominio (*DNS root servers*), de la asignación de tales nombres y de la resolución de los conflictos que ello conlleva. La organización ICANN delega sus funciones en los 5 RIR mostrados en la figura, cuyos ámbitos de actuación son los siguientes:

- RIPE NCC (*Réseaux IP Européens Network Coordination Centre*): Europa, Rusia, Oriente Medio y Asia Central.

- ARIN (*American Registry for Internet Numbers*): Estados Unidos, Canadá, parte de la región del Caribe y la Antártida.
- LACNIC (*Latin America and Caribbean Network Information Centre*): América Latina y partes de la región del Caribe.
- AFRINIC (*African Network Information Centre*): África.
- APNIC (*Asia-Pacific Network Information Centre*): Asia, Australia, Nueva Zelanda y países vecinos.



**Figura 10.12.** Jerarquía completa en la distribución de bloques de direcciones IPv4.

En la **Figura 10.13** se muestra un mapa con la cobertura geográfica de cada RIR. El proceso de distribución de bloques de direcciones procede de lo general a lo particular, según la jerarquía de la **Figura 10.12**. La ICANN adjudica grandes bloques de direcciones a los RIRs, los cuales se encargan de distribuir partes de estos bloques dentro de sus regiones de actuación. Esto se lleva a cabo a través de la jerarquía ISP de 3 niveles descrita en la Sección 9.1, hasta llegar a los clientes finales.

A continuación, se describen diversos aspectos relacionados con la especificación de los bloques IPv4 y la administración de estos bloques en el seno de una organización.

#### Especificación de bloques de direcciones IPv4

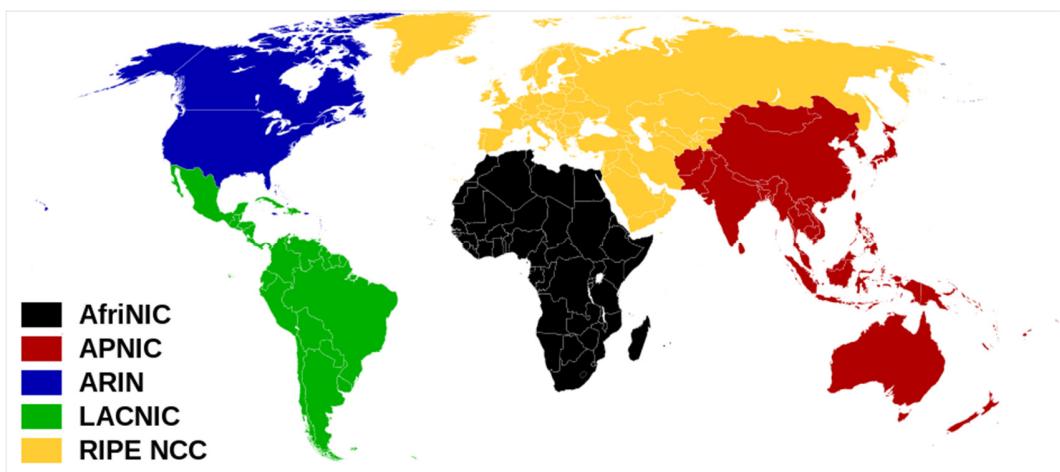
En la **Figura 10.14** se describe la forma de especificar un bloque de direcciones IPv4. Conlleva dos pasos:

1. En primer lugar, se fijan los bits del identificador de host a 0. El identificador de host constituye la parte libre con la que cuenta un administrador de redes para distribuir direcciones IPv4 en el seno de su organización. Estos bits están indicados mediante una cruz en la **Figura 10.14**, para

un identificador de red elegido arbitrariamente. Los 32 bits resultantes se representan como si fuera una dirección IPv4 cualquiera.

- En segundo lugar, hay que constatar la **longitud del prefijo**, que no es más que el número de bits contenidos en el identificador de red. Esta longitud se especifica a continuación del resultado obtenido en el paso anterior, separada mediante una barra inclinada.

Esta es la llamada **notación de prefijo**<sup>9</sup>. Implica que la combinación consistente en todos los bits iguales a 0 para el identificador de *host*, no podrá utilizarse como dirección IPv4 individual de un dispositivo. La combinación que consiste en igualar todos los bits del identificador de *host* a 1, está también reservada, en este caso para especificar la **dirección de broadcast** de la red.



**Figura 10.13.** Cobertura geográfica de cada RIR.

Identificador de red (parte fija)																Identificador de host (parte libre)									
0	1	1	1	1	0	1	0	1	0	1	0	0	0	1	1	0	1	1	1	0	X	X	X	X	
0	1	1	1	1	0	1	0	1	0	1	0	0	0	1	1	0	1	1	1	1	0	0	0	0	
0	1	1	1	1	0	1	0	1	0	1	0	0	0	1	1	0	1	1	1	1	0	0	0	0	
117																70					220				0
117.70.220.0/23																									

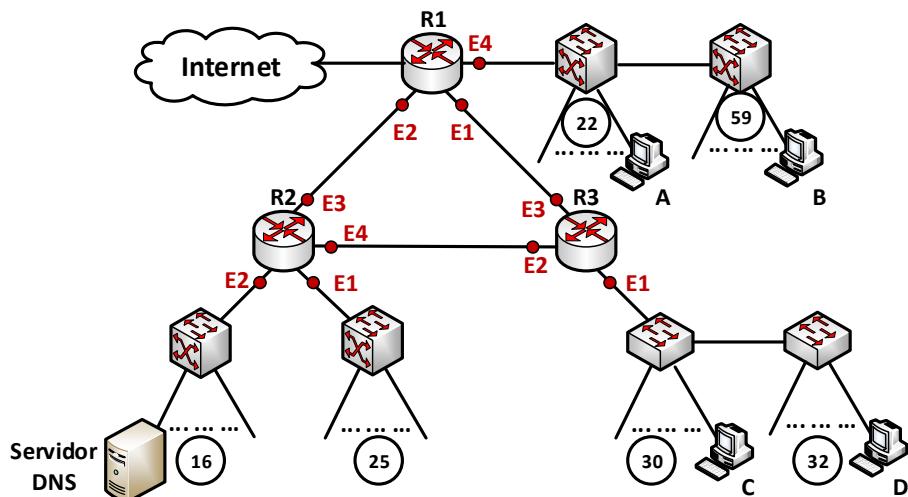
**Figura 10.14.** Elaboración del formato de un bloque de direcciones IPv4 según la notación de prefijo.

### Subnetting

A partir del bloque de direcciones IPv4 asignado a una organización, el administrador de redes debe distribuirlas entre los dispositivos que forman parte de la misma (manualmente y/o configurando adecuadamente el servidor DHCP). Existen dos formas de llevar a cabo este proceso:

<sup>9</sup> Esta notación substituye a la antigua basada en el uso de máscaras. La **máscara de red** se obtiene al fijar todos los bits del identificador de red a 1 y todos los bits del identificador de host a 0, y especificando la secuencia de bits resultante como una dirección IPv4. La introducción de la notación de prefijo tuvo lugar con IPv4 y en el caso de IPv6 es la única opción aceptada.

- Arbitriariamente, es decir, ignorando la subdivisión de la red completa en las redes departamentales que caracterizan toda organización mínimamente grande. Esta estrategia, aunque funcionalmente correcta, conlleva desventajas importantes, incluso para el propio administrador. Por un lado, dificulta la tarea de localizar físicamente un dispositivo a partir de su dirección IPv4 (por ejemplo, porque se ha detectado que tal dirección IPv4 genera un tráfico anómalo, o por cualquier otra razón). Por otro lado, contribuye a engrandecer el tamaño de las tablas de encaminamiento de los *routers*, pues prácticamente cada una contendrá tantas entradas como direcciones IPv4 individuales; a su vez, esto contribuye a aumentar los tiempos de búsqueda de dichas entradas.
- Mediante la técnica de ***subnetting***. Esta técnica consiste en fragmentar el bloque completo de direcciones de la organización en sub-bloques de tamaño diverso, en consonancia con el número de dispositivos de cada red departamental. De esta forma, ubicar físicamente un dispositivo a partir de su dirección IPv4 resulta mucho más sencillo, a la vez que permite reducir el tamaño de las tablas de encaminamiento y los tiempos de búsqueda en las mismas. El motivo es que las entradas de esas tablas pueden consistir simplemente en las especificaciones de los diversos sub-bloques.



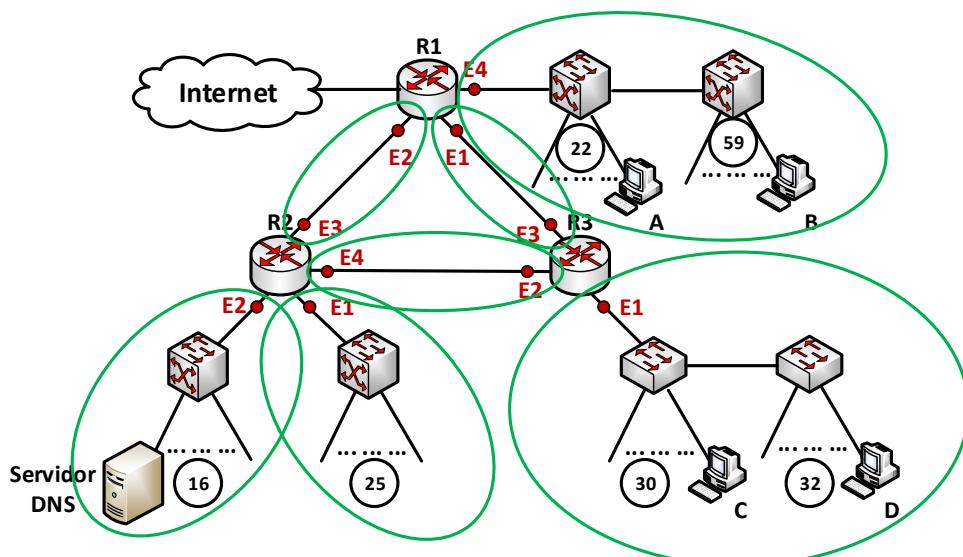
**Figura 10.15.** Ejemplo de red de una organización. Se especifican las denominaciones de los *routers* y sus interfaces, así como el número de dispositivos finales conectados a los comutadores y *hubs*.

A continuación, se describe la técnica de ***subnetting*** suponiendo que el administrador de redes asigna direcciones IPv4 a todos los dispositivos conectados a la organización, como si todos fueran a estar conectados simultáneamente. El método aplicado consiste en la asignación de **bloques contiguos y direcciones IPv4 crecientes** (evidentemente, podría llevarse a cabo por direcciones IPv4 decrecientes). Para mayor claridad, la descripción se realiza sobre un ejemplo concreto, consistente en el escenario mostrado en la **Figura 10.15**. La aplicación de la técnica consta de dos fases:

1. La primera fase consiste en identificar las subredes IP de la organización, puesto que la asignación de bloques se va a realizar por subredes IP (que normalmente coinciden con redes

departamentales). Una **subred IP** es la subred completa conectada a un puerto de *router*. Además, el concepto de subred IP coincide con el de dominio de broadcast. En la **Figura 10.16** se muestran las subredes IP dentro de la red de la **Figura 10.15**. Como podemos observar, los enlaces punto a punto entre *routers* también constituyen subredes IP. Y en particular, la subred IP formada por *hubs* es también un dominio de colisión.

2. A continuación, se lleva a cabo la asignación de bloques, después de ordenar las subredes IP de mayor a menor tamaño. El procedimiento se muestra en la **Figura 10.17**, en la que cada subred IP que contiene dispositivos finales es designada mediante el formato **[router]-[interfaz]**, y cada subred IP entre *routers* mediante el formato **[router]-[router]** (de todos modos, cualquier formato que permita la identificación de cada subred IP sin ambigüedad es válido). La descripción de las columnas es como sigue:



**Figura 10.16.** Especificación de las subredes IP de una organización.

Subred IP	Total de dispositivos (finales + intermedios)	Tamaño total requerido	Número de bits	Identificador de red + identificador de subred	Especificación del sub-bloque
R1-E4	$22 + 59 + 1 = 82$	$82 + 2 = 84$	7	117.70.1101110 <b>0.0</b> xxxxxx	117.70.220.0/25
R3-E1	$30 + 32 + 1 = 63$	$63 + 2 = 65$	7	117.70.1101110 <b>0.1</b> xxxxxx	117.70.220.128/25
R2-E1	$25 + 1 = 26$	$26 + 2 = 28$	5	117.70.1101110 <b>1.000</b> xxxx	117.70.221.0/27
R2-E2	$16 + 1 = 17$	$17 + 2 = 19$	5	117.70.1101110 <b>1.001</b> xxxx	117.70.221.32/27
R1-R2	$0 + 2 = 2$	$2 + 2 = 4$	2	117.70.1101110 <b>1.01000</b> xx	117.70.221.64/30
R1-R3	$0 + 2 = 2$	$2 + 2 = 4$	2	117.70.1101110 <b>1.010001</b> xx	117.70.221.68/30
R2-R3	$0 + 2 = 2$	$2 + 2 = 4$	2	117.70.1101110 <b>1.010010</b> xx	117.70.221.72/30

**Figura 10.17.** Proceso de asignación de bloques contiguos y direcciones crecientes a las subredes IP de una organización.

- Primera columna: Listado de subredes por orden decreciente de su tamaño.
- Segunda columna: Número total de dispositivos, desglosado entre número de dispositivos finales y número de dispositivos intermedios (interfaces de *router*, ya que cada interfaz de *router* equivale a un dispositivo adicional conectado a la subred).

- Tercera columna: Tamaño definitivo del sub-bloque. Se suma 2 a la columna anterior para tener en cuenta las combinaciones reservadas (todos los bits a 0 para especificar el sub-bloque y todos los bits a 1 para la dirección de *broadcast* de la subred IP).
- Cuarta columna: Número de bits requeridos para representar todas las direcciones IPv4 del bloque.
- Quinta columna: Especificación de los identificadores de subred de cada sub-bloque (el identificador de red es el mismo para todos). Obsérvese que la técnica de *subnetting* puede verse como la sustracción de bits al identificador de *host* para extender el identificador de red.
- Sexta columna: Especificación de los sub-bloques.

En base a las especificaciones obtenidas para los sub-bloques, posibles direcciones IPv4 de los computadores A, B, C y D y del servidor DNS de la [Figura 10.15](#) podrían ser 117.70.220.2 (A), 117.70.220.3 (B), 117.70.220.130 (C), 117.70.220.131 (D) y 117.70.221.34 (DNS) (elegidas arbitrariamente). Es práctica habitual de los administradores de redes asignar la primera dirección individual de cada subred a la interfaz de *router* (en el caso de los enlaces punto a punto entre routers, se asignan la primera y la segunda direcciones IPv4 individuales a sendas interfaces). Así por ejemplo, la dirección IPv4 de la interfaz E2 del *router* R2 de la [Figura 10.15](#) sería 117.70.221.33. La dirección de *broadcast* en esa subred sería 117.70.221.63.

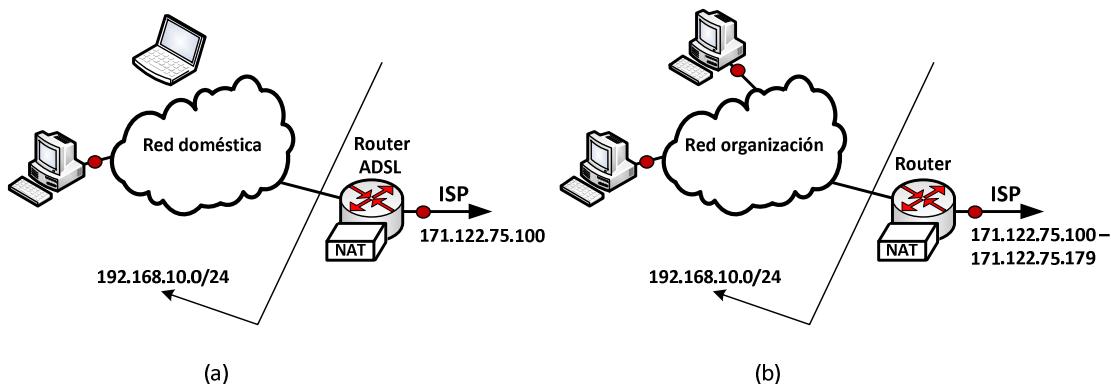
Para concluir, es importante subrayar que las direcciones IPv4 de los dispositivos, así como la dirección de *broadcast*, nunca incluyen el prefijo. El prefijo sirve para especificar bloques de direcciones IP, y es una información que tan solo maneja el administrador y los *routers* de la organización.

#### **Traducción de direcciones IPv4**

La traducción de direcciones de red o **NAT** (*Network Address Translation*) es una metodología descrita en el RFC 3022, que fue desarrollada, junto con otras técnicas, para ralentizar el consumo de direcciones IPv4. Se trataba de proporcionar una solución a corto plazo para tratar de combatir los pronósticos de agotamiento de direcciones IPv4, que lo situaban para finales de los 80 – principios de los 90. Ya a finales de los años 90 se desarrolló el primer borrador de la nueva versión IPv6, con direcciones más que suficientes de 128 bits, pero su inserción es tan lenta que, a día de hoy, IPv4 y NAT continúan plenamente vigentes.

Generalmente, el NAT es una componente software integrada en el *router* que conecta una red doméstica o la red de una organización a la Internet pública. Se ilustra en la [Figura 10.18](#). Como puede observarse, el NAT separa dos espacios de direccionamiento IPv4 claramente diferenciados: el espacio privado, donde se utiliza uno de los tres bloques definidos en el RFC 1918 (10.0.0.0/8, 172.16.0.0/12 y 192.168.0.0/16), y la Internet pública. Las direcciones privadas son de uso interno, y por tanto no son enruteables hacia la Internet pública y pueden reutilizarse en cientos de miles de redes similares. Con las direcciones privadas, los diversos dispositivos de una organización pueden comunicarse entre sí, sin necesidad siquiera de que la organización esté conectada a Internet. Si esa

conectividad es necesaria, como sucede habitualmente, la organización puede contratar un bloque de direcciones IPv4 públicas de tamaño inferior al número de dispositivos, sobre la base de una de las dos premisas siguientes: no todos los dispositivos estarán conectados a Internet al mismo tiempo, o todos los dispositivos pueden estar conectados a Internet al mismo tiempo, pero se usa la variante de NAT con traducción de puertos que se describe más adelante. Típicamente, el bloque asignado a una red doméstica es de tamaño 1, es decir, consta de una sola dirección IPv4 pública, tal como se muestra en la **Figura 10.18**.



**Figura 10.18.** Utilización de NAT en los ámbitos doméstico (a) y corporativo (b). En ambos casos, la Internet pública ve la red que está detrás del NAT como una subred IP a la que se ha asignado un bloque de direcciones públicas IPv4, aunque esa subred utiliza internamente un espacio de direcciones privadas.

El NAT mantiene en todo momento una tabla de correspondencias entre las direcciones IPv4 privadas que se utilizan internamente en la red de la organización y las direcciones IPv4 públicas que se le han asignado a la misma. Esas direcciones IPv4 privadas sólo tienen significado para los dispositivos de la red doméstica o corporativa, de forma que pueden reutilizarse en cientos de miles de redes similares, pero nunca dentro de la Internet pública.

El funcionamiento habitual del NAT es bastante simple. Supongamos que un NAT recibe un primer datagrama de algún dispositivo de la red privada, y que el destino de ese datagrama está en la Internet pública. Sea  $Private - IP_i$  la dirección IPv4 del dispositivo en la red privada, y sea  $Public - IP_j$  la primera dirección disponible dentro del bloque de direcciones IPv4 públicas asignadas (se supone que las variables  $i$  y  $j$  recorren, respectivamente, los conjuntos de direcciones privadas y públicas manejadas por la organización). Entonces, el NAT crea una asociación entre ambas direcciones y le asigna una marca temporal que corresponde al instante en que recibió el datagrama. A continuación, transforma ese datagrama de la siguiente manera antes de reenviarlo a la Internet pública:

1. En el campo correspondiente a la dirección IPv4 origen, se substituye la dirección  $Private - IP_i$  por la dirección  $Public - IP_j$ .

2. Se recalcula el campo de chequeo de errores (`Header checksum`) de la cabecera IPv4 (**Figura 10.2**).
3. Si el datagrama encapsula un segmento TCP o UDP, también hay que recalcular el campo de chequeo de errores de la cabecera de dicho segmento, ya que su evaluación engloba las direcciones IP origen y destino.

Cuando el NAT recibe un datagrama de la Internet pública, examina su dirección IPv4 de destino y busca si esta dirección aparece en alguna entrada de la tabla de correspondencias (**tabla NAT**). Si no está, descarta el datagrama, pero si coincide con la dirección *Public – IP<sub>j</sub>* referida antes, actualiza la marca temporal al instante presente y repite los tres pasos anteriores, si bien el primero de ellos lo ejecuta sobre el campo que contiene la dirección IP de destino, cambiando en este caso la dirección *Public – IP<sub>j</sub>* por la dirección *Private – IP<sub>i</sub>*. A partir de este momento, el datagrama puede ser enviado a su destino dentro de la red privada.

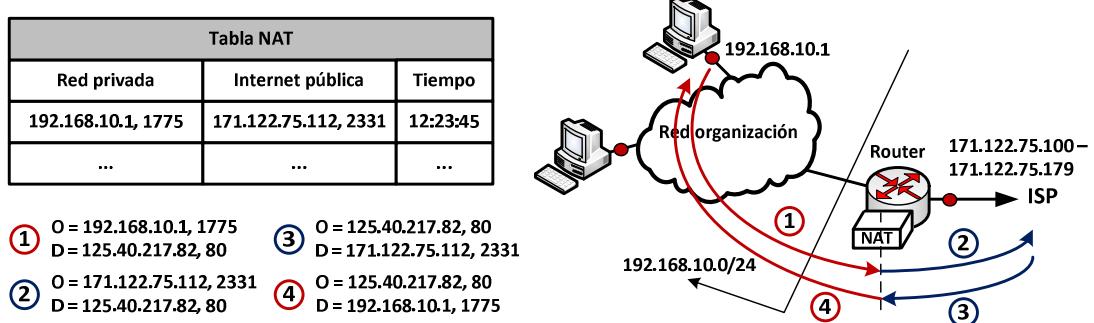
Resumiendo, el NAT crea una entrada cada vez que recibe un datagrama de la red privada cuya dirección origen no aparece en la tabla, y actualiza la marca temporal asociada a una entrada ya existente, cada vez que recibe, desde uno u otro lado, un datagrama que corresponde a esa entrada. No obstante, este funcionamiento exige que sea el dispositivo de la red privada quien haya iniciado la comunicación. Ésta es una de las desventajas del NAT, paliada por el hecho de que la mayor parte de aplicaciones que se ejecutan en Internet obedecen al modelo cliente-servidor, y en ese modelo es siempre el cliente el que inicia la comunicación (aquí podemos identificar cliente con usuario de la organización).

Cuando el NAT recibe un datagrama de la red privada cuya dirección origen no está en la tabla, y además no quedan direcciones IPv4 públicas disponibles, entonces hay que eliminar alguna de las entradas existentes para poder “traducir” el nuevo datagrama. El criterio adoptado consiste en eliminar la entrada con la marca temporal más antigua, quedando así liberada una de las direcciones IPv4 públicas. Esta gestión de la tabla de correspondencias corresponde a la versión más simple del NAT. No obstante, puesto que el bloque de direcciones públicas asignado suele ser bastante reducido en comparación con el número de computadores de la organización, la frecuencia con que las entradas de la tabla NAT son substituidas suele ser demasiado grande. Una solución para estas situaciones consiste en no tener que eliminar ninguna entrada existente cuando se va a generar una nueva. Para ello, se utiliza un procedimiento basado en traducir direcciones IPv4 y números de puerto. El procedimiento es básicamente el mismo que en el caso anterior, sólo que ahora cada entrada de la tabla es una correspondencia entre parejas del tipo (dirección IPv4, número de puerto TCP) (caso de que se utilice el protocolo de transporte TCP)<sup>10</sup>. Dado que el conjunto de puertos dinámicos utilizables como puerto origen es muy elevado, la cantidad de dispositivos que pueden compartir unas pocas direcciones IPv4 públicas es ahora muchísimo mayor. Este tipo

---

<sup>10</sup> En este nuevo NAT, sin embargo, la caducidad de una entrada se puede establecer de dos maneras: transcurrido un tiempo de inactividad, o cuando finaliza la conexión TCP (si el NAT está capacitado para realizar un seguimiento del estado de la misma).

de NAT es el más habitual y su funcionamiento para una petición HTTP generada en una red corporativa se ilustra en la **Figura 10.19**.



**Figura 10.19.** Ejemplo de NAT con traducción de puertos para una petición HTTP generada en un computador de la red corporativa (sesión de navegación web). Se reconoce la aplicación HTTP porque es la que hace uso del puerto 80. Se señalan las direcciones IP y números de puerto origen (O) y destino (D) a ambos lados del NAT.

A modo de conclusión, el NAT es una de las soluciones adoptada por el IETF para paliar la escasez de direcciones IPv4. No obstante, como ya hemos visto, presenta desventajas:

- Rompe la transparencia entre las capas de red y de transporte, ya que se maneja cada pareja (dirección IPv4, número de puerto TCP) como si fuera un identificador de host, cuando los números de puerto fueron establecidos para direccionar procesos, no hosts.
- Resulta difícil que computadores externos puedan establecer conexiones TCP con computadores detrás del NAT. Para algunos administradores de red, esto es una ventaja desde el punto de vista de la seguridad, pero no debemos olvidar que el NAT no fue concebido para este propósito, es decir, no es un cortafuegos (*firewall*). Este problema se conoce como *NAT traversal*, ya que surge cuando hay que cruzar el NAT de fuera hacia dentro en el momento que se abre una conexión. Aunque esto no perjudica las aplicaciones que operan según el modelo cliente-servidor, es un grave obstáculo para las aplicaciones en modalidad P2P (*peer-to-peer*). En estas aplicaciones, los computadores participantes juegan el doble papel de cliente y servidor, por lo que son tan frecuentes las peticiones de conexión generadas dentro de la red privada como aquellas que son generadas desde fuera hacia la red privada (aunque es cierto también que se han desarrollado soluciones para resolver este problema, más basadas en pequeñas argucias para “saltar” el NAT que no en grandes estrategias).

## Control de errores

El control de errores comprende dos aspectos:

- Códigos de detección y corrección de errores (**codificación de canal**, en la capa de enlace).
  - **Códigos de detección de errores**: Sólo tienen capacidad detectora, siempre limitada.
  - **Códigos de corrección de errores**: Tienen capacidad detectora y correctora, ambas limitadas, y la capacidad correctora suele ser inferior a la detectora.
- Mecanismos de retransmisión o **mecanismos ARQ** (*Automatic Retransmission reQuest*).
  - Derivan de los mecanismos de control de flujo añadiendo acuses de recibo negativos o NAK (*Negative AckNowledgment*) y temporizadores.
  - Control de flujo *stop-and-wait* → **Stop-and-wait ARQ**.
  - Control de flujo basado en ventana deslizante → **Go-Back-N, Selective Reject**.

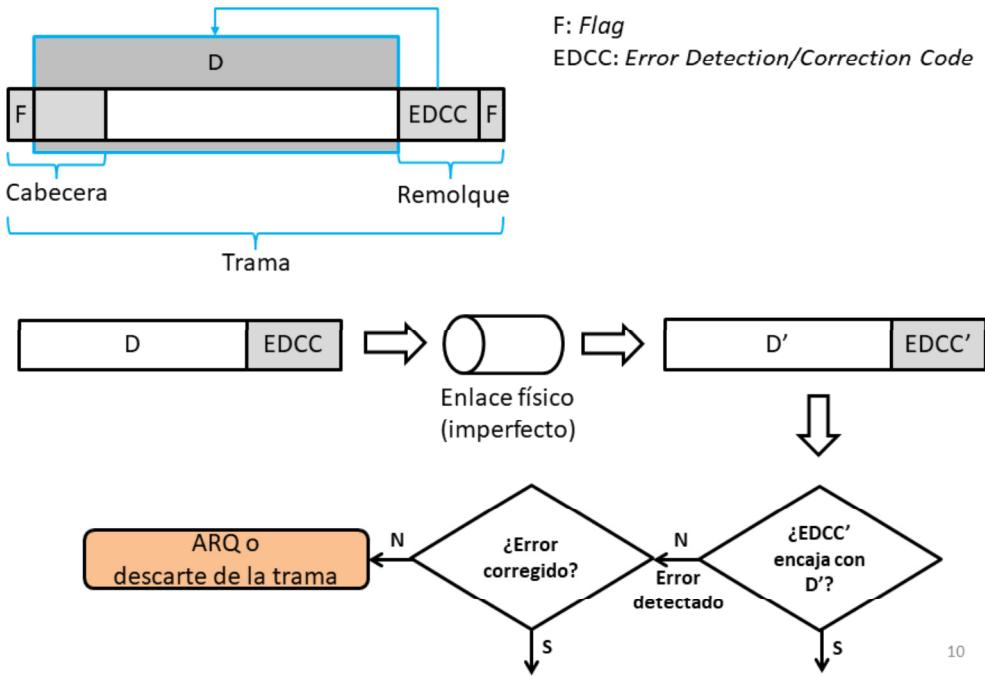
9

El control de errores, tanto en la capa de transporte como en la capa de acceso a red (capa de enlace, según el modelo OSI), abarca dos aspectos:

- El desarrollo y aplicación de códigos detectores y códigos correctores. Es lo que se denomina **codificación de canal** cuando se considera en la capa de enlace.
- Los mecanismos de retransmisión o **mecanismos ARQ**, necesarios cuando se detectan tramas erróneas que no pueden ser corregidas (porque el código utilizado carece de capacidad correctora, o porque el código es corrector pero los errores producidos exceden su capacidad correctora), y tampoco pueden ser descartadas (hay protocolos de acceso a red que descartan las tramas erróneas, pues se basan en la hipótesis de que la tasa de tramas erróneas es muy baja y se pueden relegar las retransmisiones a protocolos de capa más alta).

Los mecanismos de control de errores derivan de los mecanismos de control de flujo al introducir acuses de recibo negativos, es decir, tramas NAK, y temporizadores. Así, el mecanismo de control de errores **stop-and-wait ARQ** deriva del mecanismo de control de flujo *stop-and-wait*, y los mecanismos de control de errores **Go-Back-N** y **Selective Reject** derivan del mecanismo de control de flujo basado en ventana deslizante.

## Detección y corrección de errores en la capa de enlace



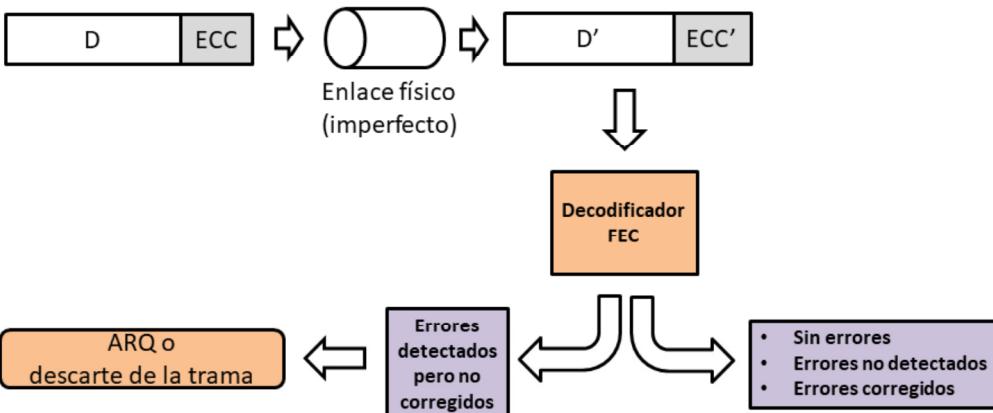
En adelante nos centraremos en la capa de acceso a red o capa de enlace. Los fundamentos son los mismos para la capa de transporte, tanto en lo referente a los códigos de protección contra errores como en lo referente a los mecanismos de retransmisión, si bien hay algunas diferencias de implementación. Se ven al abordar el protocolo de transporte TCP, brevemente en el último bloque de la asignatura, y con más profundidad en la asignatura de tercero.

La figura muestra el formato típico de una trama, que es la unidad básica de información en la capa de acceso a red. Podemos observar que consta de una cabecera, un campo de datos y un remolque o tráiler. En los extremos de la trama, es decir, al principio de la cabecera y al final del tráiler, hay un *flag* o delimitador de trama. Su función es marcar el comienzo y el final de la trama, es decir, lo que se denomina **sincronismo de trama**. Cuanto se envían tramas seguidas una tras otra, sin intervalo de tiempo entre ellas, lo que se conoce como transmisión *back-to-back*, el *flag* de final de una trama hace también el papel de *flag* de comienzo de la trama siguiente. Este *flag* suele consistir en una secuencia de octetos de valor predeterminado. Por ejemplo, en las redes Ethernet, son 7 octetos específicos. Además del *flag*, la cabecera contiene otros campos de control, y el tráiler contiene el código de protección contra errores (la razón de ponerlo detrás se explica en los apuntes) o *EDCC*. Este código se elabora siempre a partir de los bits que hay que proteger, que tal como muestra la figura, son todos los bits anteriores excepto el *flag*. Los *flags* no participan en la elaboración del código. El conjunto de todos los bits a proteger es *D*, de manera que, a efectos del control de errores, la trama puede verse como *D* U *EDCC*.

La trama se transmite por un enlace físico que es imperfecto, pues provoca atenuación, distorsión y ruido, y es el ruido principalmente la causa de los errores de bit. Así pues, en general el módulo de acceso a red en el otro extremo del enlace recibe una trama *D'* U *EDCC'*. A continuación, el receptor determina si el código *EDCC'* encaja con *D'*, aplicando el mismo algoritmo que utilizó el transmisor. Si encaja, el receptor determina que no hay error, y la transmisión sigue adelante (obsérvese que aquí pueden “colarse” errores que excedan la capacidad detectora). Si se detectan errores en la trama, el código tiene también capacidad correctora, y tales errores entran dentro de dicha capacidad, la trama es corregida y nuevamente la transmisión sigue adelante. Si se detectan errores, pero estos exceden la capacidad correctora del código (o simplemente este código carece de capacidad correctora), entonces la trama se

descarta o hay que invocar su retransmisión, según sea el caso.

## Códigos FEC



11

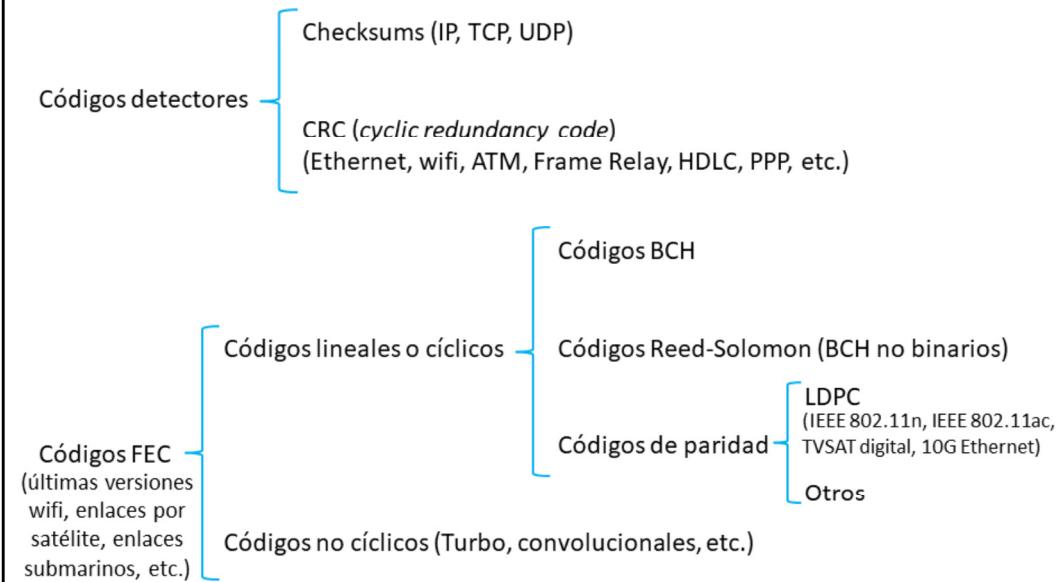
Centrándonos en los códigos correctores o códigos FEC (*forward error correction*), la transparencia muestra todas las situaciones que pueden darse. En este caso, nos referimos al código como ECC (*error correction code*). El planteamiento es el mismo que en la transparencia anterior, es decir, el decodificador FEC determina si  $ECC'$  encaja con  $D'$ . Hay tres casos en los que la transmisión sigue adelante:

- No se han producido errores, de manera que  $D' = D$  y  $ECC' = ECC$  y, por supuesto,  $ECC'$  encaja con  $D'$ . Esta será la situación más frecuente si el enlace es mínimamente bueno.
- Se han producido errores, pero estos exceden la capacidad detectora del código y el decodificador también determina que  $ECC'$  encaja con  $D'$ .
- Se han producido errores de transmisión, el decodificador los detecta comprobando que  $ECC'$  no encaja con  $D'$ , pero es capaz de corregirlos.

En cambio, la transmisión no sigue adelante cuando el decodificador detecta errores en la trama ( $ECC'$  no encaja con  $D'$ ) y estos errores exceden su capacidad correctora. En tal caso, la trama se descarta (la transmisión no sigue adelante ni hacia atrás) o hay que solicitar retransmisión (la transmisión vuelve atrás).

Si el código tiene capacidad correctora suficientemente alta, cabe esperar que el caso más frecuente dentro de los casos negativos sea el de errores detectados y corregidos, de forma que la transmisión sigue hacia delante. De ahí la denominación FEC.

## Taxonomía de códigos de control de errores en redes



12

La transparencia muestra la taxonomía de códigos de control de errores utilizados en las redes de computadores. Entre los códigos detectores, los más utilizados son los códigos basados en sumas binarias o códigos *checksum* y los códigos CRC. Los primeros son computacionalmente menos exigentes, pero también menos potentes, y se utilizan en las capas más altas de la arquitectura TCP/IP (transporte e internet). Lo contrario ocurre con los códigos CRC, que son computacionalmente más complejos, tienen más capacidad detectora, y se utilizan en la mayor parte de estándares de capa de acceso a red (estándares de red). Uno de los motivos por el que estas dos clases de códigos se distribuyen por capas de la manera indicada tiene que ver con la implementación habitual de dichas capas:

- Las capas superiores del modelo TCP/IP se suelen implementar en software, y consumen CPU del dispositivo.
- La capa de acceso a red suele implementarse en hardware, con lo que el procesamiento de las tramas es muy rápido al realizarse con una CPU dedicada.

En consecuencia, es mejor aprovechar la velocidad de proceso de la capa de acceso a red para implementar una codificación de canal computacionalmente costosa, y relegar la codificación menos compleja a las capas superiores, donde el procesamiento es más lento. El otro motivo es que resulta más eficiente atajar la mayor parte de errores en la capa de acceso a red, donde las posibles retransmisiones conllevan menores retardos, y dejar solamente los errores residuales para la capa de transporte, donde las retransmisiones son *end-to-end*.

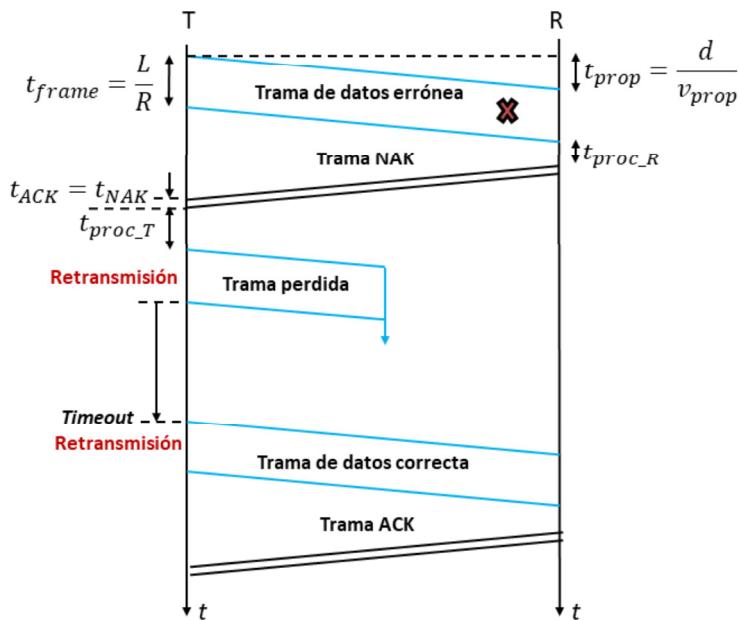
La transparencia también pone de manifiesto que la mayor parte de códigos utilizados en redes de computadores tienen solamente capacidad detectora. Sin embargo, hay casos en que el uso de códigos FEC resulta especialmente recomendable:

- En estándares de red de muy alta velocidad, para evitar penalizar esta velocidad con continuas retransmisiones de tramas. Algunos ejemplos son las versiones wifi más modernas (IEEE 802.11n, IEEE 802.11ac) y las redes Ethernet a 10 Gbps.
- Cuando el retardo de propagación es muy elevado. A su vez, esto puede ser debido a dos causas: la distancia que debe recorrer la señal es muy elevada, como ocurre en el caso de los enlaces por satélite geoestacionario (retardo global de subida y bajada igual a 240 ms), o la velocidad de propagación es muy baja, como ocurre en los enlaces submarinos que hacen uso de señales acústicas.

En los apuntes se describen los códigos más utilizados, es decir, los códigos *checksum* y los códigos CRC. También se describen brevemente los códigos de paridad más sencillos, si bien los códigos de paridad que, desde finales de los 90, han ido atrayendo cada vez más interés, son los códigos LDPC (*low density parity check*). Estos se utilizan en los estándares de muy alta velocidad señalados anteriormente, y también en la distribución de TV digital por satélite. En este último caso, la razón es doble, pues se trata de un servicio de muy alta velocidad (ancho de banda muy grande) a

través de un enlace por satélite.

## Mecanismo stop & wait ARQ



13

A continuación, pasamos a describir los mecanismos de retransmisión de tramas. El diagrama muestra el funcionamiento del mecanismo ARQ más sencillo, llamado *stop-and-wait ARQ*. En este caso, hay que contemplar la posibilidad de que se produzcan errores en la transmisión de las tramas. Si el destinatario detecta una trama errónea (marcada con una cruz) y no puede corregirla, ni descartarla, entonces solicita su retransmisión enviando una trama de control de tipo NAK al transmisor. También puede darse un caso extremo de transmisión fallida, que es el de trama perdida. En un enlace directo entre dos dispositivos, una trama puede perderse por varias razones:

- El nivel de ruido durante la transmisión de esa trama es tan alto que ésta queda enmascarada completamente y no puede ser detectada por el receptor. De hecho, basta que esto ocurra durante la transmisión del *flag* que marca el comienzo de la trama (si no hay comienzo de trama, no hay trama).
- La potencia media de señal con que se recibe dicha trama está por debajo del nivel de sensibilidad del receptor. Esto incluye el caso en que se produce una caída de tensión en el circuito que conecta transmisor con receptor. También puede ocurrir que la potencia recibida sea demasiado elevada, hasta el punto de saturar el amplificador a la entrada del receptor; en tal caso, la señal resultante está completamente distorsionada y no posibilita la detección de los bits.
- La trama colisiona con otras (caso de que el medio que conecta transmisor y receptor sea compartido).

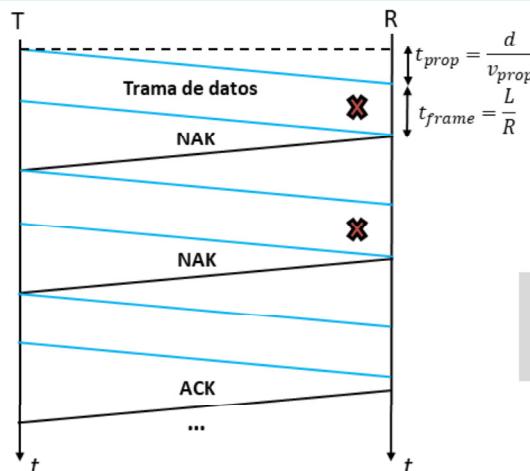
La posibilidad de tramas perdidas, o de que los acuses de recibo también se pierdan, o simplemente sean detectados erróneos, genera la necesidad de incluir un temporizador por cada trama enviada. De lo contrario, en el ejemplo de la figura, el transmisor estará esperando indefinidamente un acuse de recibo que nunca va a llegar (situación de **bloqueo del transmisor**). El valor del temporizador empieza a decrecer desde un valor inicial cada vez que termina de transmitirse una trama, de manera que si alcanza el valor 0 sin que se haya recibido confirmación alguna (ACK o NAK), entonces la trama se retransmite automáticamente (el transmisor siempre guarda una copia de cada trama enviada hasta que recibe una confirmación positiva para la misma). Se trata, pues, de una retransmisión por expiración del temporizador (vía **timeout**), en lugar de una retransmisión vía NAK.

## Eficiencia del mecanismo *stop-and-wait* ARQ

Hipótesis:

- Tráfico de saturación.
- $t_{proc} \cong 0$ ,  $t_{ACK} = t_{NAK} \cong 0$ .
- Los acuses de recibo, ACK o NAK, se transmiten correctamente.
- Ausencia de tramas perdidas.

} Evitamos el *timeout*  
en el cálculo de  
eficiencia



$$\eta = \frac{1 - p_{retr}}{1 + 2a}$$

Falta caracterizar la  
probabilidad de  
retransmisión  $p_{retr}$

14

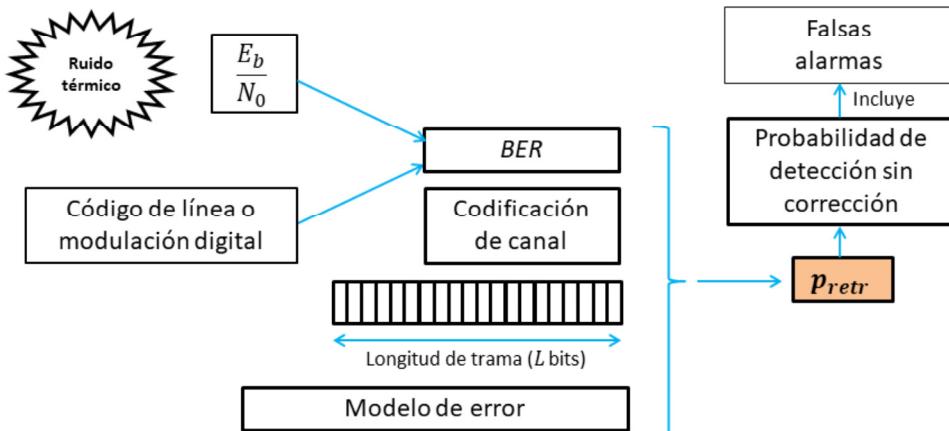
Para obtener la eficiencia del mecanismo de control de errores *stop-and-wait* ARQ, introducimos las siguientes hipótesis, algunas de las cuales ya las planteamos al calcular la eficiencia del mecanismo de control de flujo homólogo:

- Tráfico de saturación: la fuente siempre tiene tramas por enviar (queremos obtener la máxima eficiencia que el mecanismo puede dar de sí, con lo que hay que suponer que las pérdidas de tiempo se deban al propio mecanismo y no al hecho de que la fuente se quede sin tramas por enviar).
- Tiempos de procesos y duraciones de los acuses de recibo despreciables.
- Las tramas de confirmación (ACK/NAK) siempre se transmiten correctamente, y las tramas de datos nunca se pierden. Con ambas hipótesis conseguimos simplificar el cálculo de la eficiencia al no tener que incluir el *timeout*. Esto también es coherente con el objetivo de evaluar la máxima eficiencia que el mecanismo puede ofrecer, ya que las retransmisiones vía *timeout* suelen generar un mayor retardo (el valor inicial del temporizador se elige lo suficientemente grande como para asegurar que la trama de confirmación hubiera podido llegar).

Bajo las hipótesis anteriores, el diagrama de tiempos a considerar en el análisis es el mostrado en la transparencia. Lógicamente entra en juego un nuevo parámetro, que es la **probabilidad de retransmisión de trama** ( $p_{retr}$ ). El diagrama refleja la situación general de una trama de datos que tiene que ser transmitida varias veces (en el dibujo dos veces, pero entiéndase en sentido general) hasta que se recibe una confirmación positiva por parte del receptor. Obsérvese que el tiempo realmente consumido en la transmisión de una trama siempre será un múltiplo entero de veces ( $n$ ) el tiempo  $t_{frame} + 2t_{prop}$ , es decir,  $n(t_{frame} + 2t_{prop})$ , donde  $n$  es una variable aleatoria. Si el tiempo realmente gastado en la transmisión de una trama es aleatorio, calcularemos la eficiencia como el tiempo ideal ( $t_{frame}$ ), dividido por el tiempo medio gastado (el parámetro más representativo de una variable aleatoria es su media).

La caracterización del número de transmisiones hasta que la transmisión es correcta es exactamente la misma que en el experimento de lanzar una moneda tantas veces como sea necesario hasta que sale cara (la probabilidad de que salga cruz sería la equivalente a  $p_{retr}$ ). Esta variable aleatoria es discreta y presenta una distribución geométrica. Si  $\bar{n}$  es el número medio de veces que se requiere transmitir la trama (que se tiene que lanzar la moneda en el experimento equivalente), entonces el tiempo medio realmente gastado para enviarla es  $\bar{n}(t_{frame} + 2t_{prop})$ . Se puede demostrar que  $\bar{n} = \frac{1}{1-p_{retr}}$  (resultado conocido para la distribución geométrica), con lo que la eficiencia media resultante es  $\eta = \frac{t_{frame}}{\bar{n}(t_{frame} + 2t_{prop})} = \frac{1-p_{retr}}{1+2a}$ . Nótese que si  $p_{retr} = 0$ , se obtiene la eficiencia del mecanismo de control de flujo *stop-and-wait*. Esto es lógico, pues al analizar este mecanismo supusimos que no había errores.

## Probabilidad de retransmisión



Hipótesis simplificadoras:

- (1) Código con máxima capacidad detectora y nula capacidad correctora.
- (2) Independencia de errores de bit (modelo compatible con sólo ruido térmico en el canal).

$$p_{retr} \stackrel{(1)}{=} p_{frame} \stackrel{(2)}{=} p = 1 - (1 - BER)^L$$

15

Queda por determinar una expresión de la probabilidad de retransmisión de trama en función de parámetros del enlace. La probabilidad de retransmisión de trama depende de cuatro factores:

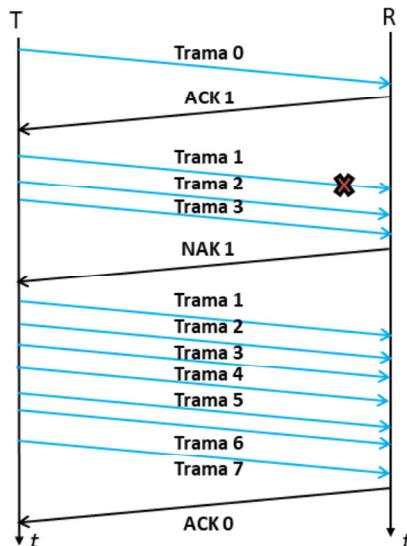
- La probabilidad de error de bit, o tasa de errores de bit (*BER*), que obtuvimos al final del Bloque 1. Recordemos que esa tasa de errores de bit dependía de la relación señal-ruido (considerando sólo ruido térmico), a través del cociente  $\frac{E_b}{N_0}$ , y de la codificación de línea o modulación digital utilizadas.
- La codificación de canal, es decir, el tipo de código detector o corrector. Por ejemplo, un código sin capacidad correctora conllevará una probabilidad de retransmisión mayor que la de un código FEC. O, considerando los códigos detectores, la probabilidad de retransmisión será tanto mayor cuanto mayor sea la capacidad detectora.
- La longitud de la trama. Naturalmente, cuanto mayor es el número de bits contenidos en la trama, mayor es el número de errores de bit que pueden producirse.
- La estadística de los errores de bit, recogida en el llamado **modelo de error**. Este modelo captura la estadística de los patrones de bits erróneos. Por ejemplo, cuando hay mucho ruido impulsivo en el canal, y especialmente si la velocidad de transmisión es muy alta, los errores de bit se producen a ráfagas. Esto equivale a decir que hay correlación entre los errores de bit, es decir, si un bit es erróneo, es probable que también lo sean los bits vecinos. Este caso es matemáticamente más difícil de tratar. En el otro extremo está el modelo de error en que los errores de bit se producen independientemente unos de otros. Es el modelo más sencillo, y es aceptable cuando la única fuente de ruido es el ruido térmico.

Los cuatro factores mencionados determinan la probabilidad de que una trama tenga que ser retransmitida. Esto incluye, desafortunadamente, las falsas alarmas, es decir, situaciones en las que la trama se retransmite sin que fuera necesario, como por ejemplo cuando los errores de bit se producen solamente en el propio código de protección contra errores. Supondremos que la tasa de falsas alarmas es muy baja, y adoptaremos dos hipótesis adicionales para simplificar el cálculo de la probabilidad de retransmisión:

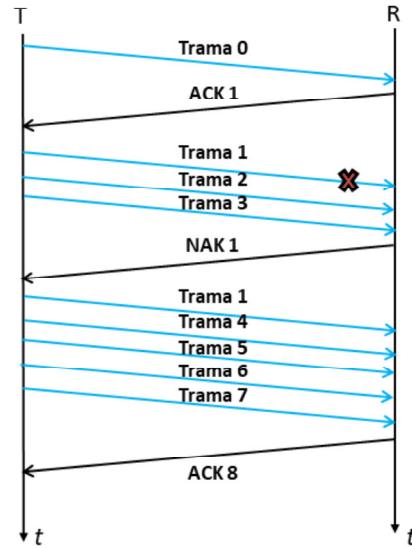
1. El código utilizado tiene máxima capacidad detectora y nula capacidad correctora (los códigos CRC, frecuentemente utilizados, se acercan a este comportamiento). Esto significa que los acontecimientos "trama retransmitida" y "trama errónea" son equivalentes:  $p_{retr} = p_{frame} = p$ . En adelante nos referiremos a ella como **probabilidad de error de trama**.
2. Modelo de independencia de errores de bit. En tal caso, la probabilidad de error de trama se calcula fácilmente como  $p = 1 - (1 - BER)^L$ , donde  $L$  es el número total de bits que contiene la trama.

## Mecanismos *go-back-N* y *selective reject*

*Go-back-N* con  $N = 7$  (suponiendo  $k = 3$ )



*Selective reject* con  $N = 7$  (requiere  $k \geq 4$ )



16

Como ya ocurría en el caso del control de flujo, podemos mejorar la eficiencia del mecanismo de control de errores *stop-and-wait* ARQ aumentando el tamaño de la ventana. En este caso se dan, no obstante, dos posibilidades: que la recepción de un acuse de recibo negativo conlleve la retransmisión de la trama indicada por dicho acuse, y de todas las que le siguieron (caso *go-back-N*), o que la retransmisión sea únicamente de la trama indicada por el acuse de recibo, es decir, retransmisión de forma selectiva (*selective reject*). Este último caso requiere que el destinatario sea capaz de reordenar las tramas recibidas a partir de sus números de secuencia. La figura muestra los dos mecanismos para el mismo tamaño de ventana y la misma situación de trama errónea.

Aparentemente, es más eficiente el mecanismo de rechazo selectivo; sin embargo, para evitar situaciones de ambigüedad, el mecanismo de rechazo selectivo impone una ventana más restrictiva (menor) que la del mecanismo *go-back-N* (no vemos la demostración). Específicamente, fijado el número de bits ( $k$ ) para secuenciar las tramas, el tamaño máximo de ventana en *go-back-N* es  $2^k - 1$  (igual que la del mecanismo de control de flujo basado en ventana deslizante del cual deriva), mayor que el tamaño máximo de ventana en *selective reject*, que viene dado por  $2^{k-1}$ . Refiriéndonos al ejemplo de la figura, el mismo tamaño de ventana de 7 exige 3 bits de secuenciación en *go-back-N* y en cambio 4 bits en *selective reject*. Por lo tanto, si lo que viene fijado es el valor de  $k$  (por el protocolo de enlace), no podemos afirmar que un mecanismo proporcione siempre mayor eficiencia que el otro, sino que se tiene que evaluar en cada caso. La tabla de la página siguiente resume los resultados, incluyendo los correspondientes al mecanismo *stop-and-wait* ARQ.

## Eficiencias ARQ

Mecanismo	Tamaño de ventana ( $N$ )	Eficiencia ( $\eta$ )
<i>Stop-and-wait ARQ</i>	1	$\frac{1-p}{1+2a}$
<i>Go-back-N</i>	$\leq 2^k - 1$	$\begin{cases} \frac{1-p}{1+2a \cdot p}, & N \geq 2a + 1 \\ \frac{N(1-p)}{(2a+1)(1-p+N \cdot p)}, & N < 2a + 1 \end{cases}$
<i>Selective reject</i>	$\leq 2^{k-1}$	$\begin{cases} 1-p, & N \geq 2a + 1 \\ \frac{N(1-p)}{2a+1}, & N < 2a + 1 \end{cases}$

$k$ : número de bits de enumeración de las tramas.

Estas fórmulas asumen que  $p_{retr} = p_{frame} = p$ .

Si además hay independencia de errores a nivel de bit:  $p = 1 - (1 - BER)^L$ .

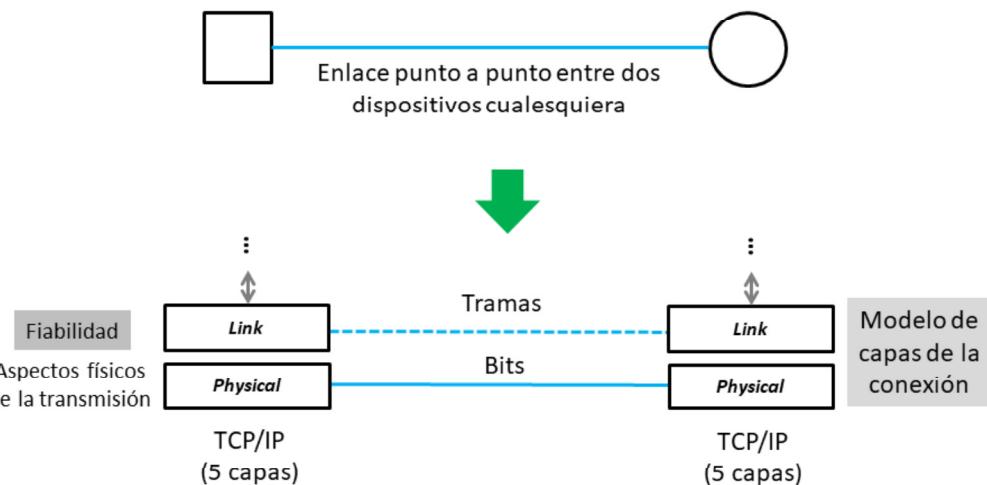
17

Esta tabla resume los resultados para todos los mecanismos ARQ. Tanto en el caso *go-back-N* como en el de *selective reject*, la eficiencia obedece a dos fórmulas dependiendo de cómo es el tamaño de ventana ( $N$ ) comparado con el término  $2a + 1$  (omitimos la demostración). En ambos casos, la fórmula superior, válida cuando se cumple  $N \geq 2a + 1$ , proporciona una mayor eficiencia que la fórmula inferior, en la que el tamaño de ventana no alcanza un valor suficientemente grande ( $N < 2a + 1$ ).

En general, salvo que se especifique lo contrario, supondremos que los mecanismos ARQ trabajan con el máximo tamaño de ventana permitido, es decir,  $N$  alcanza la cota superior que aparece en la columna intermedia de la tabla.

## Protocolos de enlace

- Función: fiabilidad de las comunicaciones sobre conexiones físicas.
- Técnicas básicas: control de flujo y control de errores a nivel de tramas.



18

La función de todo protocolo de enlace es asegurar la fiabilidad de la conexión entre cualquier pareja de dispositivos enlazados directamente, fiabilidad que no puede garantizar el medio físico debido a las perturbaciones a las que está sometido. Esas perturbaciones provocan errores de transmisión, y por ello se necesita disponer de un mecanismo de control de errores que los corrija. El concepto de fiabilidad abarca control de errores y también control de flujo, porque además se requiere que cada dispositivo, cuando transmite, no sobrecargue el *buffer* del dispositivo que recibe (cuando ese *buffer* se llena y siguen llegando tramas, éstas se pierden, y el efecto es similar a que se hubieran transmitido de forma errónea).

Todo esto es desde la teoría, pues luego cada protocolo de enlace concreto realiza estas funciones de manera completa o sólo parcialmente.

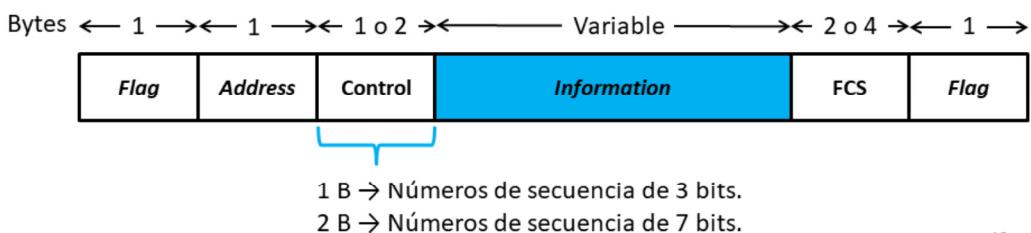
La figura muestra el desglose por capas de la conexión directa entre los dos dispositivos. Se utiliza un modelo TCP/IP de 5 capas para poner de manifiesto los dos aspectos implicados en toda conexión directa entre dispositivos: por un lado aquéllos puramente físicos de la transmisión y, por otro, los relativos al protocolo de enlace. Cuando ambos aspectos vienen definidos en un mismo módulo, normalmente implementado en hardware en forma de tarjeta de red (caso de las redes Ethernet y Wi-Fi), entonces encaja mejor cualquier modelo TCP/IP de 4 capas.

## Ejemplos de protocolos de enlace (1)

### HDLC (High-level Data Link Control)

- Recogido inicialmente en los siguientes estándares:
  - ISO 3309 (*Frame Structure*), ISO 4335 (*Elements of Procedure*), ISO 6159 (*Unbalanced Classes of Procedure*), ISO 6256 (*Balanced Classes of Procedure*).
  - Revisados y recopilados en estándar ISO/IEC 13239:2002.
- Base de otros protocolos de enlace: [PPP](#), [LLC](#), [Cisco HDLC](#), LAPB, LAPD, LAPM, SDLC.
- Utilizado en líneas serie.

Trama HDLC



19

El protocolo de enlace más completo es HDLC, y de ese protocolo derivan prácticamente la totalidad del resto de protocolos de enlace estandarizados hasta el momento, entre los cuales, los que están en uso son PPP, LLC y Cisco HDLC casi exclusivamente. Cisco HDLC es una versión simplificada desarrollada por Cisco y se trata, por tanto, de una solución propietaria. HDLC estaba documentado en una multiplicidad de estándares ISO, hasta que en un momento dado se recopilaron en un solo (ISO/IEC 13239:2002).

HDLC se utiliza principalmente en líneas serie. El formato de trama es bastante sencillo:

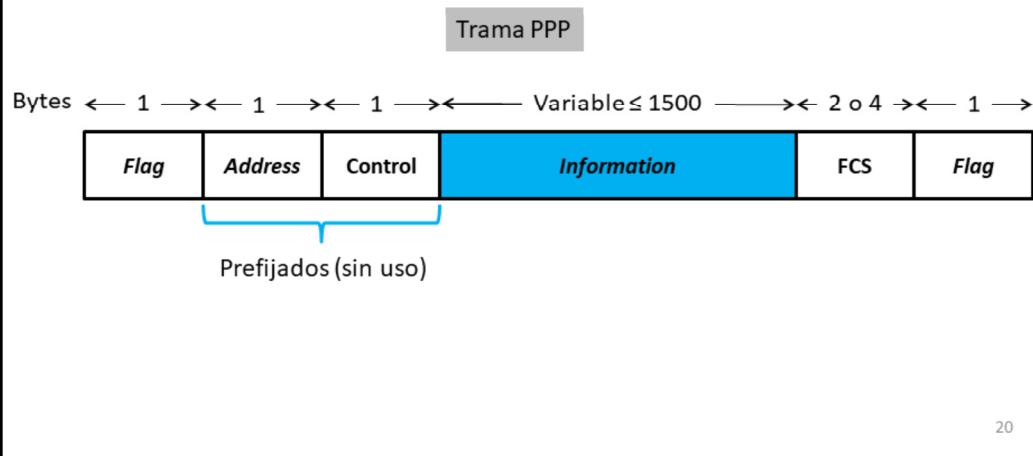
- **Flag:** Es 1 byte concreto que marca tanto el comienzo como el final de trama (sincronismo de trama). El *flag* de final hace el papel de *flag* de comienzo en la transmisión de tramas seguidas (*back-to-back*).
- **Address:** La presencia de este campo puede sorprender, pues qué papel puede jugar un campo de dirección en un enlace punto a punto. La razón de ser es que HDLC fue también desarrollado para configuraciones punto a multipunto (o simplemente multipunto), en las que múltiples estaciones, llamadas secundarias, se conectan a una única estación, llamada primaria. La primaria gobierna el enlace y va otorgando turnos a las secundarias tanto para transmitir como para recibir. Cualquier comunicación sólo puede tener lugar entre la primaria y una secundaria, o entre una secundaria y la primaria, pero nunca entre dos secundarias. Un ejemplo de este tipo de configuraciones es el de los viejos sistemas centralizados, cuyos usuarios se conectan a un computador central desde sus terminales consistentes simplemente en teclado y pantalla. En tales casos, la dirección contenida en la trama HDLC identifica siempre la secundaria, bien como dirección origen cuando la secundaria transmite, bien como dirección de destino cuando la primaria transmite.
- **Control:** Este campo tiene dos funciones, por un lado sirve para especificar el tipo de trama HDLC y, por otro, la secuenciación de las tramas, necesaria en la función de control de flujo y en los mecanismos de retransmisión. Puede tener dos tamaños, 1 o 2 bytes. Cuando tiene el tamaño de 1 byte, los números de secuencia contenidos en el campo de control son de 3 bits; cuando tiene un tamaño de 2 bytes, los números de secuencia son de 7 bits. No hay opciones intermedias.
- **Information:** Es el campo de datos. El estándar no especifica ninguna acotación para el tamaño de ese campo de datos.
- **FCS (frame check sequence):** Puede ser de 2 ó 4 bytes, según la cantidad de bits que haya que

proteger. Se utilizan códigos CRC estandarizados (CRC-16 o CRC-32 respectivamente).

## Ejemplos de protocolos de enlace (2)

### PPP (Point-to-Point Protocol)

- Versión simplificada de HDLC + procedimientos de autenticación.
- Utilizado en líneas serie y ADSL.



PPP es un protocolo de enlace derivado de HDLC. Es más simple que éste, pero incorpora procedimientos de autenticación, pues fue desarrollado especialmente para enlaces punto a punto cuyos extremos corresponden a entidades distintas, una proveedora de un servicio y la otra usuaria de ese servicio. El servicio se ofrece vía contrato y, por ello, se requiere autenticación (nombre de usuario y contraseña, básicamente). Es el caso, por ejemplo, de las líneas ADSL y las líneas serie.

El formato de una trama PPP es parecido al de HDLC. Tan solo significar las diferencias principales:

- Tanto el campo Address como el campo Control quedan inhabilitados (se les asigna una combinación fija irrelevante).
- Se impone un tamaño máximo de 1500 bytes al campo de datos. Por tanto, la *MTU* de PPP son 1500 bytes.

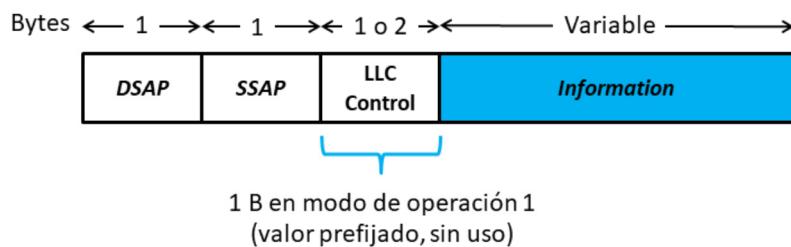
No obstante, llama la atención el hecho de que exista un campo FCS para detectar errores, pero no se utilicen números de secuencia (fueron inhabilitados al inhabilitar el campo Control). Si no hay números de secuencia, no pueden funcionar los mecanismos de control de flujo y errores, pero entonces ¿qué sucede con las tramas erróneas? Simplemente se descartan. Esta es una opción de diseño habitual cuando se asume que el medio de transmisión sobre el que va a ejecutarse el protocolo de enlace presenta una tasa de errores de bit muy baja, como ocurre con los medios guiados. En tal caso, es preferible aligerar el diseño y la operación del protocolo de enlace y delegar, en todo caso, la fiabilidad a la capa superior de transporte, concretamente, al protocolo TCP. Téngase en cuenta que si se descarta una trama, también se descarta el segmento encapsulado en esa trama, pero el módulo TCP destinatario detectará el segmento perdido y solicitará su retransmisión al módulo TCP origen (obsérvese que será ya una retransmisión entre extremos, es decir, *end-to-end*). El segmento a retransmitir volverá a descender sobre la pila de protocolos del host origen y será finalmente encapsulado en una nueva trama a transmitir.

## Ejemplos de protocolos de enlace (3)

### LLC (*Logical Link Control*)

- Estándar IEEE 802.2.
- Derivado de HDLC.
- 3 modos de operación:
  - **Tipo 1:** redes de área local en entornos TCP/IP (funcionalidad mínima, sin control de flujo ni errores).
  - Tipos 2 y 3: redes de área local en entornos industriales (sin TCP/IP).

Trama LLC (común a los 3 modos de operación)



21

El protocolo LLC es otro derivado de HDLC, desarrollado en este caso por el comité 802 del IEEE, para redes sobre medio compartido (redes de área local estandarizadas por el mismo comité). Téngase en cuenta que la capa MAC de estas redes es la encargada de regular el acceso al medio de transmisión, por el que generalmente compiten múltiples estaciones. Cuando una estación consigue hacerse con el medio de transmisión, entonces es como si las demás no estuvieran, salvo la destinataria de la comunicación. Es como si temporalmente se hubiera establecido un enlace punto a punto entre ambas estaciones. Nuevamente, como este enlace no está exento de errores de transmisión, requiere un protocolo de enlace si se quiere garantizar la fiabilidad de la conexión a través de la red. A diferencia de HDLC, el estándar desarrollado, oficialmente IEEE 802.2, admite 3 modos de operación, dependiendo del escenario donde vaya a ser utilizado:

- El Tipo 1 se trata nuevamente de una versión minimizada, carente de las funciones de control de flujo y errores. Es el utilizado en redes de área local en entornos TCP/IP, es decir, conectadas a Internet, y por la misma razón que en el caso de PPP, se delega la fiabilidad al protocolo TCP.
- Los Tipos 2 y 3 se utilizan en redes de área local que trabajan en entornos industriales, donde no se utiliza la arquitectura TCP/IP, y por ello no puede preverse la presencia de un protocolo de capa superior que garantice la fiabilidad. El Tipo 2 incluye la funcionalidad completa para lograr esa fiabilidad, mientras que el Tipo 3 se sitúa en un punto intermedio entre los Tipos 1 y 2.

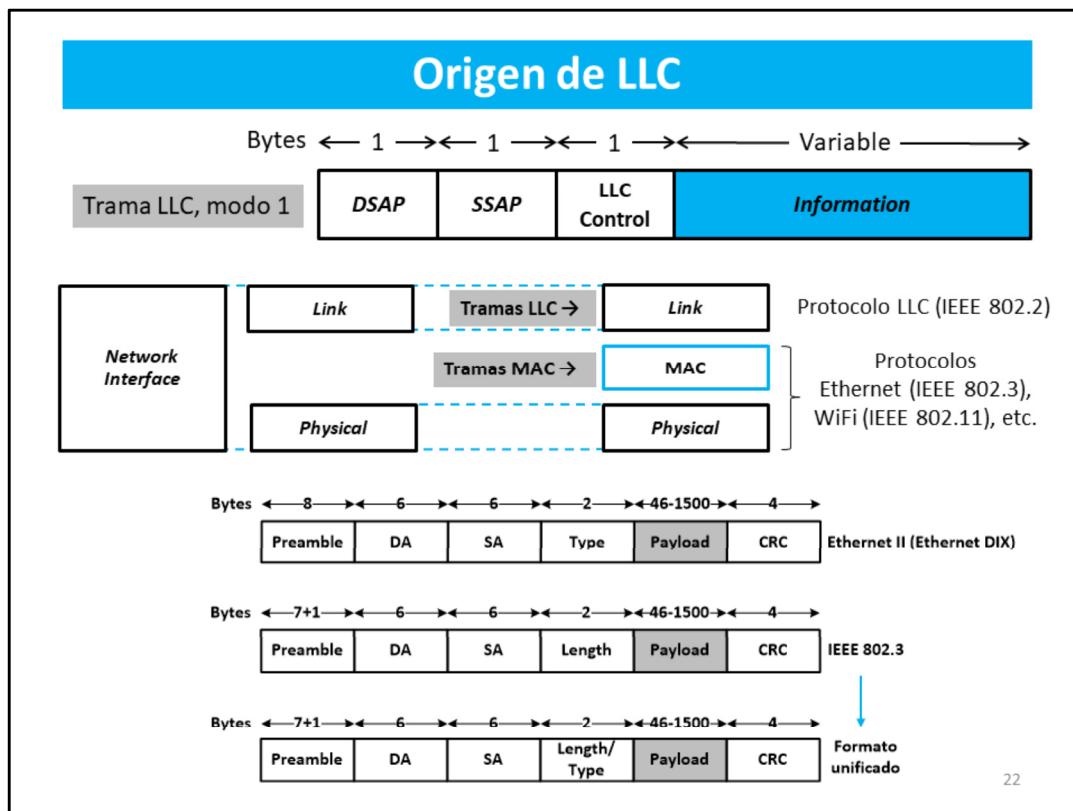
El formato de trama LLC es el mismo para los tres modos de operación. Los campos **DSAP** (*destination service access point*) y **SSAP** (*source service access point*) sirven para identificar los procesos de capa inmediatamente superior en los dos extremos de la comunicación, de forma similar a los puertos origen y destino de los protocolos de transporte TCP o UDP.

En el ámbito de las redes de área local, podemos hablar de tramas a dos niveles, es decir, tramas LLC y tramas MAC. En general, las tramas Ethernet y las tramas Wi-Fi son fusiones de ambas tramas, con matices propios en cada caso. De todos modos, en esa fusión hay más funcionalidades del protocolo de capa MAC que del protocolo LLC, por lo que tramas MAC y tramas Ethernet o Wi-Fi suelen utilizarse indistintamente.

Finalmente, puede llamar la atención que se utilice el modo de operación Tipo 1 (protocolo de enlace sin funcionalidades de control de flujo y errores) sobre redes Wi-Fi, cuando en éstas la tasa de errores de bit, y por tanto la tasa de errores de trama, son relativamente altas. Puede tener sentido en Ethernet, ya que trabaja sobre medio guiado, pero no en Wi-Fi, donde el medio es el espacio libre. Sin embargo, esto es solo aparentemente una incoherencia, ya que la capa MAC de las redes Wi-Fi ya contempla un mecanismo propio de control de errores,

consistente en que cada trama de datos enviada debe ser confirmada mediante una trama ACK, es decir, se trata de un *stop-and-wait*; transcurrido un tiempo sin la recepción del ACK, la estación transmisora decide retransmitir la trama de datos.

## Origen de LLC



El origen del protocolo LLC y su inserción en las tramas Ethernet está aquí a modo de curiosidad. Se describirá en un documento aparte.

## Control de congestión

- Control de flujo ejercido desde un nodo de conmutación (de paquetes).
- Es un control de 1 (nodo) sobre N (hosts que generan tráfico sobre ese nodo).
- Mecanismos de *backpressure*.
- Peculiaridades muy específicas en Internet.
  - Control de congestión delegado en TCP.
  - Mecanismos de control de congestión de TCP ---> Múltiples versiones de TCP.

23

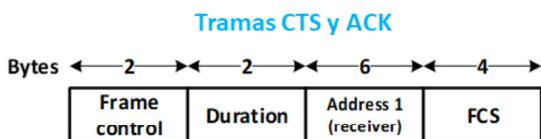
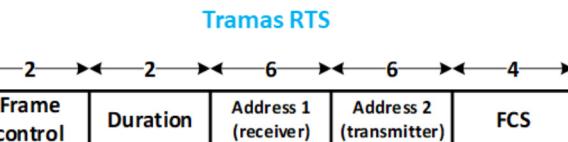
El control de congestión es el control de flujo ejercido desde un nodo de red, por ejemplo, en el caso de Internet, ese nodo sería un *router*. El *router* soporta múltiples conexiones que pueden sobrecargar cualquier *buffer* de entrada o salida. Cuando un paquete llega a un *buffer* lleno, se pierde. Para evitar esta situación, el nodo debe regular la velocidad con que las fuentes envían su tráfico. Es similar al control de flujo, pero mientras este último conlleva una interacción 1 a 1 (la entidad que recibe controla la entidad que transmite), en el caso del control de congestión esa interacción es de 1 sobre N, si N es el número de fuentes que provocan el estado de congestión en el nodo.

En general, las técnicas de control de congestión consisten en que el nodo afectado ejerce una presión hacia las fuentes, es decir, hacia atrás (*backpressure*), con el objetivo de reducir la velocidad con que dichas fuentes inyectan tráfico en la red.

En el ámbito de Internet, no obstante, el control de congestión tiene peculiaridades muy específicas que lo diferencian significativamente de la descripción formal que se acaba de dar. Si tuviera que obedecer a dicha descripción, cualquier *router* en estado de congestión (o pre-congestión) debería iniciar el *backpressure*, por ejemplo enviando paquetes IP de frenado a las fuentes. Estas fuentes estarían identificadas por las direcciones IP origen contenidas en las cabeceras de los paquetes, que son analizadas y reconocidas por los *routers*. No obstante, si bien en los paquetes IP existe algún campo relacionado con la función de control de congestión, en la práctica no se utiliza y se relega completamente el control de congestión al protocolo TCP. Por lo tanto, en la actualidad el estudio del control de congestión se centra fundamentalmente en los mecanismos de control de congestión de TCP, los cuales han dado lugar a múltiples

versiones de dicho protocolo. Se estudian, pues, al tratar TCP en profundidad.

## Tramas de control y tramas de gestión



### Tramas beacon

- Enviadas regularmente por el punto de acceso (cada 100 ms típicamente).
- Proporcionan las siguientes informaciones:
  - SSID (*Service Set Identity*).
  - Velocidades de transmisión soportadas.
  - TIM (*Traffic Indication Map*).
  - *Timestamp*.
  - Intervalo de tiempo hasta el siguiente *beacon*.

24

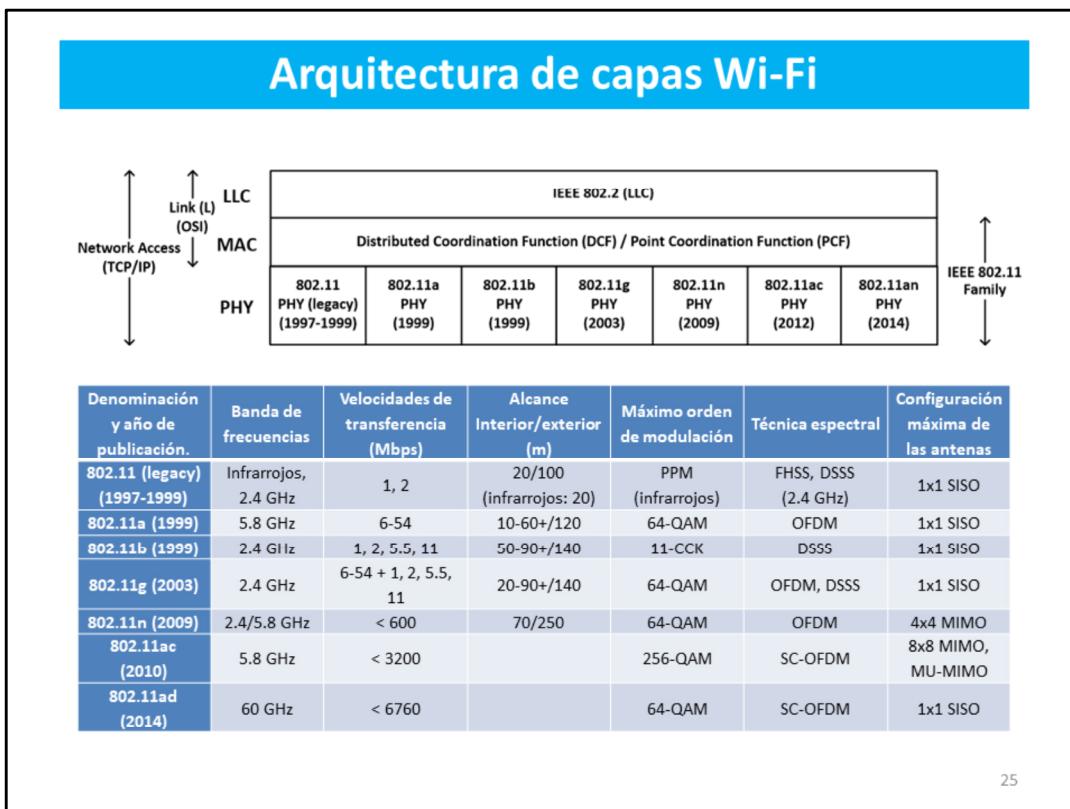
El protocolo IEEE 802.11 especifica tres tipos de tramas: tramas de datos, tramas de control y tramas de gestión. Las tramas de control más importantes son las que aparecen en los diagramas de tiempo descritos en la transparencia anterior, es decir, las tramas ACK, utilizadas tanto en el modo de acceso básico como en el modo RTS-CTS, y las tramas CTS y ACK específicas de este último modo.

A diferencia de las tramas de datos, las tramas ACK, RTS y CTS solamente incluyen los extremos inmediatos de la transmisión. En el caso de la trama RTS, el origen y el destino inmediatos de la misma, y en el caso de las tramas CTS y ACK, únicamente su receptor inmediato.

La principal trama de gestión en una celda Wi-Fi es la trama *beacon*, enviada periódicamente por el punto de acceso (por defecto cada 100 ms), en el modo infraestructura (en el modo ad-hoc, las estaciones participantes se reparten la tarea de enviar la trama *beacon*). Esa es la trama cuya potencia miden las estaciones para engancharse a uno u otro punto de acceso, o para des-asociarse de uno y asociarse a uno nuevo de la misma red Wi-Fi (este último procedimiento es el *handoff*). Además, la trama *beacon* aporta diversas informaciones relativas a la celda:

- SSID: Es el identificador de la red Wi-Fi, es decir, del *extended basic service set*. Además, cada celda de una misma red Wi-Fi tiene su propio identificador BSSID (*basic service set identity*), que no es más que la dirección MAC asignada a su punto de acceso.
- Una lista de las velocidades de transmisión soportadas. En general, cada especificación de capa física soporta varias velocidades.
- TIM: es un listado de las estaciones para las cuales el punto de acceso ha recibido tráfico. Con esta información, una estación sabe si en el punto de acceso hay información dirigida a ella, y en caso negativo dicha estación puede comutar a un modo de bajo consumo hasta el siguiente *beacon*.
- El *timestamp* es una referencia temporal enviada con el objeto de que todas las estaciones estén sincronizadas.
- El intervalo hasta el siguiente *beacon* permite que una estación pueda comutar al modo de bajo consumo con la información de cuándo puede volver a despertarse para detectar si hay tráfico pendiente para ella.

## Arquitectura de capas Wi-Fi



25

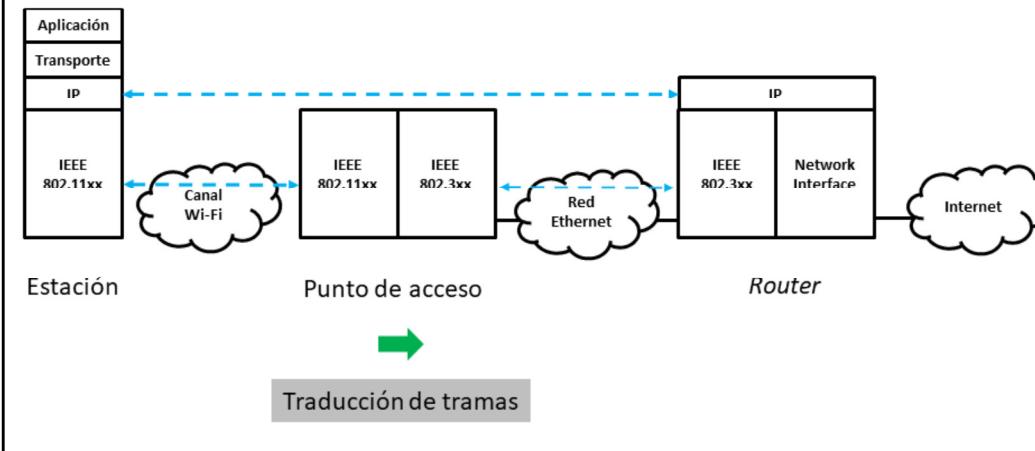
La figura muestra la arquitectura de capas en el caso de las redes Wi-Fi. Ocurre lo mismo que en Ethernet, es decir, consiste en una única capa MAC sobre diversas especificaciones de capa física, coherentes con la evolución tecnológica. No obstante, la capa MAC de las redes Wi-Fi admite dos modos de operación que pueden darse simultáneamente. En el modo DCF, el punto de acceso es una estación más tanto a la hora de transmitir como a la hora de recibir, con la única salvedad de que, por tratarse del punto de acceso, su cobertura es mayo y recubre el resto de estaciones. En el modo PCF, el punto de acceso adquiere un nivel jerárquico superior, de modo que pasa a controlar la celda como si de una estación primaria se tratara. Este modo es útil cuando se quiere garantizar el acceso a un conjunto de estaciones, por ejemplo porque envían y/o reciben tráfico con requisitos de tiempo real (*tráfico de streaming*). El tiempo en una red Wi-Fi está organizado en un sucesivos intervalos temporales de super-trama. Cuando se utiliza el modo PCF, el intervalo de super-trama se divide en una primera parte en la que la celda trabaja en modo DCF, y una segunda parte en la que trabaja en el modo PCF. El intervalo de super-trama tiene una longitud fija, pudiendo variar su distribución en los dos modos. La otra opción es que la red trabaje íntegramente en el modo DCF durante todo el intervalo de super-trama; ésta es la opción por defecto.

La tabla muestra las características técnicas principales de las diversas especificaciones de capa física. Cada especificación se corresponde con una o dos letras a continuación de IEEE 802.11. Son especialmente significativas las tres primeras columnas:

- La banda de frecuencias de trabajo. Todas las bandas de frecuencias en las que operan las redes Wi-Fi son bandas ISM (*industrial, scientific and medical purposes*), caracterizadas porque no requieren el pago de una licencia. Por ello, son franjas del espectro en general muy densamente ocupadas, de ahí que la normativa de instalación en cualquiera de ella impone fuertes restricciones de potencia transmitida, tanto al punto de acceso como a los equipos de usuario. De esta manera, se reducen los riesgos de interferencia entre las múltiples redes Wi-Fi que pueden coexistir en una zona, así como las interferencias que puedan provocar sobre otros equipos, o las que estos puedan provocar sobre las redes Wi-Fi.
- Las diversas velocidades ofrecidas en cada opción de capa física.
- El alcance, es decir, la distancia máxima entre el punto de acceso y una estación. Esta distancia suele ser inferior en los escenarios de interior, pues presentan más obstáculos al camino de la señal (paredes

y otros objetos).

## Arquitectura de una conexión Wi-Fi – Ethernet - Internet



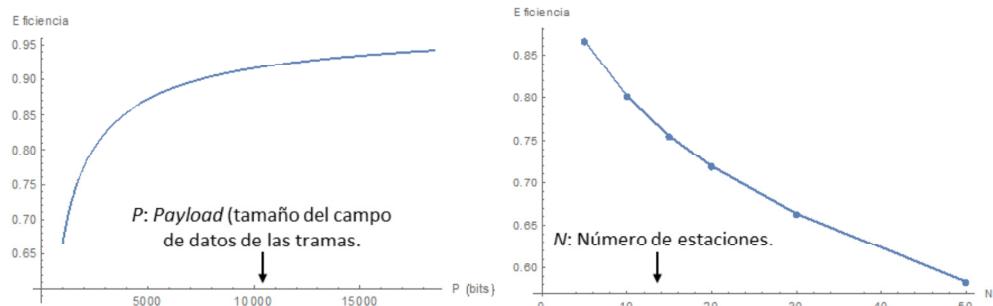
26

La figura muestra la arquitectura de capas típica de una conexión que cruza un ESS (*extended service set*), es decir, una conexión a Internet desde una estación directamente conectada a una red WiFi. Se ha supuesto que el sistema de distribución es una red Ethernet, a la cual está conectado el *router* que enlaza con la Internet pública. Como podemos observar, el punto de acceso realiza una conmutación de tramas al igual que cualquier switch de Ethernet, con la salvedad de que dicha conmutación tiene lugar entre dos estructuras de trama diferentes. Es decir, en el punto de acceso de la figura tiene lugar una traducción de tramas, del formato WiFi al formato Ethernet y viceversa. La necesidad de esta traducción desaparece cuando el sistema de distribución es también una celda WiFi (caso de un WDS).

## Eficiencia Wi-Fi

- Análisis complejo: Solución aportada por Bianchi.
- Depende de la configuración MAC (DCF/PCF, acceso básico/RTS-CTS) y de muchos parámetros de la capa física.

Modo DCF y acceso básico  
IEEE 802.11b 11 Mbps



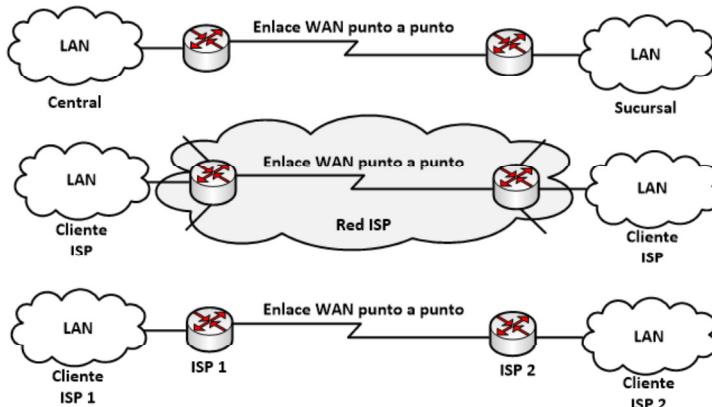
27

La eficiencia de las redes Wi-Fi depende de diversos aspectos de configuración (sólo DCF o DCF/PCF, acceso básico o acceso RTS-CTS) y también de múltiples parámetros de la capa física. El análisis es más complejo que en el caso de las redes Ethernet. El análisis más completo fue el realizado por Giuseppe Bianchi ("Performance Analysis of the IEEE 802.11 Distributed Coordination Function", *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 3, March 2000). El análisis propone la aplicación de un algoritmo para obtener la eficiencia. El resultado, para una red IEEE 802.11b a 11 Mbps, en modo DCF y acceso básico, se muestra en las dos curvas mostradas, una obtenida en función del tamaño del campo de datos de las tramas ( $P$ ), la otra en función del número de estaciones ( $N$ ). Los resultados son bastante lógicos: cuanto mayor es el campo de datos, mejor se aprovecha el canal cuando una estación acapara el medio de transmisión inalámbrico, y por tanto más alta es la eficiencia con que se utiliza dicho canal; en cambio, cuanto mayor es el número de estaciones, menor es la eficiencia, pues más estaciones compiten por el medio, y eso provoca que una fracción cada vez mayor del tiempo del canal se dedica a la solución de conflictos de acceso.

## Líneas serie

- Solución utilizada para enlaces punto a punto de larga distancia entre routers.
- Contratada como servicio de conexión a un TelCo.
- Tecnología basada en conmutación de circuitos.
- Múltiples denominaciones: *leased line*, *leased circuit*, *serial link*, *serial line*, *point-to-point link*, *point-to-point line*, *WAN link*, *private line*.

Ejemplos de uso



28

Las líneas serie constituyen una solución habitualmente utilizada para enlaces punto a punto de larga distancia entre routers. Se contratan como un servicio de conexión a un TelCo, es decir, un operador de telecomunicaciones. En cada país operan uno o varios grandes operadores de telecomunicaciones, cuya infraestructura se desarrolló inicialmente para telefonía. Es decir, consistía en la red telefónica de cada país, a la que posteriormente se fueron añadiendo servicios de transmisión de datos. Esas redes se basan en la tecnología de conmutación de circuitos, una tecnología que precedió a la conmutación de paquetes. Por ello, aunque su uso en redes de computadores es más bien marginal, la conmutación de circuitos es la base del servicio de línea serie proporcionado por un TelCo en el ámbito de las redes corporativas.

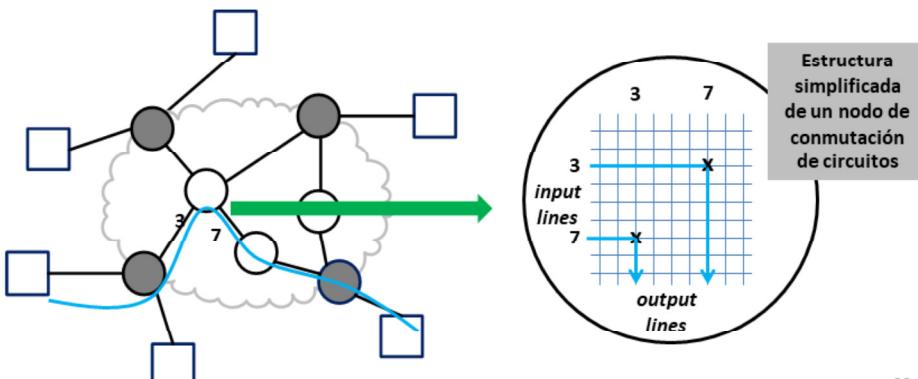
La figura muestra diversos usos del servicio de línea serie:

- Como enlace privado entre dos sedes de una misma organización, geográficamente dispersas. Esta opción es una alternativa al uso de un servicio de túnel VPN a través de Internet (el túnel VPN se crea al encriptar la información intercambiada entre los dos extremos del mismo).
- Como enlace privado dentro de la infraestructura de un ISP. En realidad, este escenario es un caso particular del anterior, en el que la organización es un ISP. Obsérvese que, en este caso, el NSP que da servicio de conexión de red al ISP es un TelCo.
- Como enlace privado entre dos ISP distintos del mismo nivel, de manera que el tráfico de uno a otro no tenga que cursarse a través de un ISP de jerarquía superior, o a través de otros ISP del mismo nivel.
- Un cuarto uso, no mostrado en la figura, es como enlace punto a punto entre el router de una red de empresa y el router de su correspondiente ISP (POP). Aunque factible, el servicio punto a punto no se utiliza para enlazar una red doméstica (SOHO) con su ISP, pues se trata de una solución relativamente cara.

El término “línea serie” puede aparecer en inglés bajo las múltiples denominaciones mostradas, y es posible que alguna más.

## Comutación de circuitos

- Tecnología para redes malladas, anterior a la comutación de paquetes.
- Idea preliminar:
  - “Puenteo” entrada-salida en cada nodo de comutación.
  - Camino físico entre elementos finales.
  - Retardos de transmisión y propagación únicamente.
- Fase de establecimiento de la conexión para la definición del camino entre los dos extremos (en el caso de conexiones no permanentes).



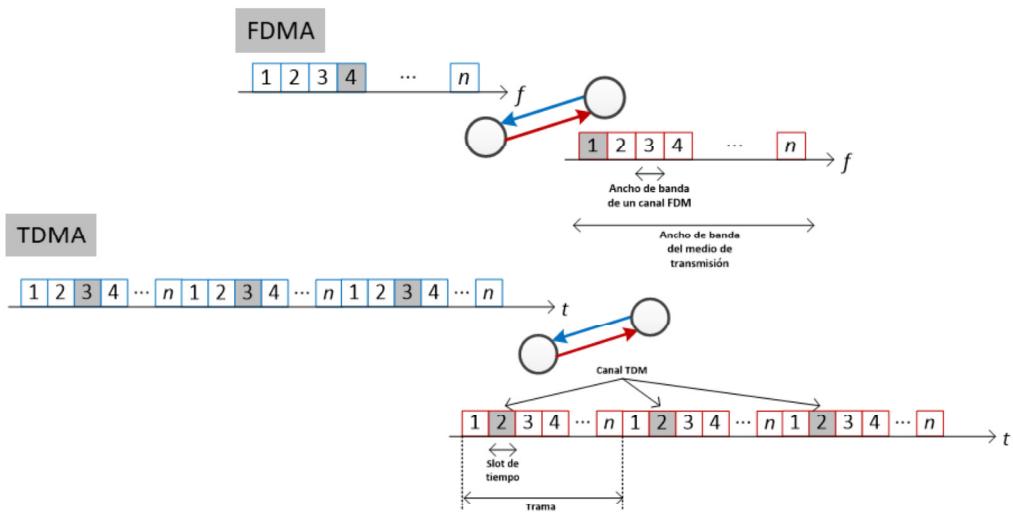
29

La comutación de circuitos es una tecnología desarrollada para redes malladas, por lo que, desde el punto de vista topológico, no hay diferencias entre una red de comutación de circuitos y una de paquetes. Las diferencias son, sin embargo, muy significativas en cuanto al funcionamiento interno de los nodos. Básicamente, en un nodo de comutación de circuitos se “puentea” un puerto de entrada con un puerto de salida, de forma que el nodo asegura la continuidad física entre los dos puertos (a diferencia de la continuidad lógica, pero no física, entre los puertos de un nodo de comutación de paquetes). El “puenteo” entre dos puertos *x* e *y* consiste en conectar la línea de entrada del puerto *x* con la línea de salida del puerto *y*, así como la línea de entrada del puerto *y* con la línea de salida del puerto *x* (recuérdese que, en general, las comunicaciones son bidireccionales). En el ejemplo de la figura, estos puertos son 3 y 7.

La acción conjunta de todos los nodos permite emular un enlace punto a punto entre los dos dispositivos interconectados. Previamente, la red ha tenido que buscar una ruta entre ambos dispositivos, durante la denominada fase de establecimiento de la conexión. Sólo después tiene lugar la transferencia de datos, garantizada si cada nodo implicado en la conexión realiza correctamente el “puenteo” entre puertos. El ejemplo más evidente es el proporcionado por la red telefónica: cuando un abonado descuelga el teléfono para realizar una llamada, después de marcar el número, la red busca un camino hasta el abonado llamado. Al alcanzarlo, se generan tonos de llamada en ambos extremos, y si el abonado llamado descuelga, puede iniciarse la conversación (fase de transferencia de datos, en general). En esto consiste la fase de establecimiento de la conexión, es decir, el establecimiento de llamada que se suele cobrar en telefonía.

No obstante, el funcionamiento descrito resultaría muy poco rentable para el TelCo, pues los enlaces intermedios de una conexión estarían dedicados a la misma durante todo el tiempo que durara dicha conexión. Para hacer un uso rentable de los recursos de transmisión, el TelCo necesita que estos estén compartidos por un número elevado de conexiones. Se trata, pues, de una nueva situación de acceso múltiple. En comutación de circuitos, las técnicas utilizadas son de partición de canal.

## Partición de canal en conmutación de circuitos



- Efecto final en ambos casos: enlace físico entre extremos.
- Requieren un protocolo de enlace si se quiere garantizar la fiabilidad de las transmisiones (funciones de control de flujo y errores).
- Protocolos de enlace típicos: HDLC, PPP, Cisco HDLC, etc.

30

La figura muestra las dos técnicas de partición de canal utilizadas en conmutación de circuitos: FDMA y TDMA. FDMA es anterior a TDMA, y consiste en una partición del eje de frecuencias, es decir, se basa en subdividir el ancho de banda del medio de transmisión en múltiples sub-bandas menores, cada una de las cuales es un canal que puede estar ocupado o no por una conexión. En un enlace de transmisión dado, la misma conexión puede tener un número de canal en un sentido y otro número de canal en el sentido opuesto, y ambos números pueden ser distintos para la misma conexión en un enlace vecino. Lo importante es que cada nodo de conmutación registre con qué canal de salida hay que conectar cada canal de entrada.

FDMA fue la técnica utilizada en la primera versión de la RTC (red telefónica conmutada), la llamada RTC analógica. Con el proceso de digitalización de las redes y servicios, FDMA fue substituido por TDMA para dar lugar a la RTC digital. TDMA es también una técnica de partición de canales, pero en este caso basada en la subdivisión del eje de tiempos en una estructura de trama consistente en  $n$  slots de tiempo (este concepto de trama nada tiene que ver con el de trama como unidad básica de información de un protocolo de enlace). Cada slot corresponde a un canal, de manera que, cuando a una conexión, en un sentido, se le asigna un canal, entonces hace uso del mismo slot dentro de la estructura de trama. Al igual que en el caso de FDMA, el número de canal asignado a una conexión en uno de los dos sentidos de un enlace no tiene por qué coincidir con el número de canal asignado a la misma conexión y en el mismo enlace, pero en el sentido opuesto. Y además, ambos números pueden cambiar de un enlace a otro, lo cual no es ningún problema si, al igual que antes, cada nodo de conmutación de circuitos enlaza correctamente cada canal de entrada con su canal de salida.

Cualquiera que sea la técnica de partición de canal, el efecto final es como si se hubiera extendido un cable entre los dos dispositivos interconectados. La distancia cubierta podría ser arbitrariamente grande, pero a un coste infinitamente inferior al que se derivaría de utilizar realmente un cable a larga distancia. En definitiva, la red de conmutación de circuitos permite emular un enlace físico directo entre dos dispositivos a un coste asequible, al menos en el ámbito corporativo.

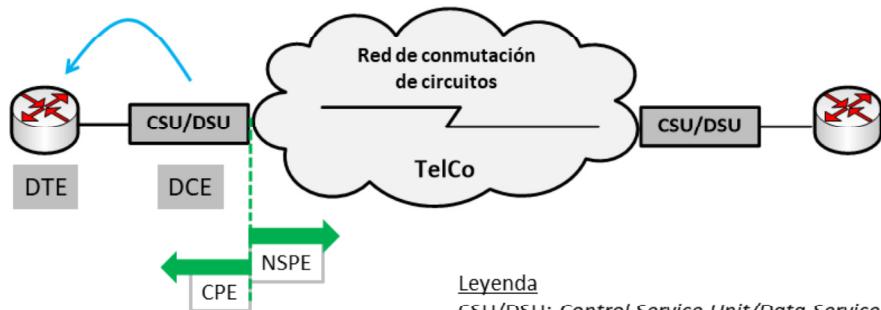
El enlace físico establecido por la red de conmutación de circuitos, está también sujeto a perturbaciones, de modo que se requiere un protocolo de enlace si se pretende garantizar la fiabilidad de la conexión. Tres son los principales protocolos de enlace utilizados sobre líneas serie: HDLC, PPP y Cisco HDLC. Este último consiste en una modificación del HDLC original desarrollada por Cisco. Se trata, pues, de una solución propietaria.

Para terminar esta descripción de la conmutación de circuitos, conviene destacar una desventaja de esta tecnología

frente a la conmutación de paquetes. En esta última, no es necesario que los dispositivos conectados trabajen a la misma velocidad, pues los nodos de conmutación de paquetes rompen la continuidad física entre extremos: un usuario se puede conectar a través de Internet utilizando una tarjeta Ethernet a 100 Mbps, mientras que en el otro extremo de la conexión, el usuario puede estar utilizando una tarjeta Ethernet a 1 Gbps. Por el contrario, en el caso de la conmutación de circuitos, hay continuidad física entre extremos, lo cual exige que trabajen a la misma velocidad.

## Equipamiento en líneas serie

La CSU/DSU puede estar integrada en el puerto del *router* utilizado para esa conexión a la línea serie (*puerto serie, serial WAN port*). Esta integración puede ser en forma de *tarjeta de red*, en inglés *WIC – WAN interface card*, que se inserta en un slot de expansión situado habitualmente en la parte posterior del *router*.



### Leyenda

- CSU/DSU: Control Service Unit/Data Service Unit.
- DTE: Data Terminal Equipment.
- DCE: Data Circuit-Terminating Equipment.
- NSPE: Network Service Provider Equipment.
- CPE: Customer Premises Equipment.

31

El equipamiento correspondiente a un servicio de conexión punto a punto consiste principalmente en una caja de doble funcionalidad, CSU por un lado y DSU por otro. CSU/DSU es un dispositivo cuya función es convertir una señal digital de un formato a otro, tanto en un sentido como en el sentido opuesto. El *router* y el módulo CSU/DSU son ejemplos de DTE y DCE respectivamente. Básicamente, el DTE es la fuente/destinatario y el DCE el transmisor/receptor del modelo básico de comunicaciones introducido al principio de curso. Por ello, DTE y DCE no son denominaciones exclusivas del ámbito de las líneas serie, sino que resultan aplicables en cualquier escenario de transmisión de datos entre dos dispositivos.

Una línea serie es un servicio de conexión punto a punto contratado a un TelCo. En toda conexión abonado-proveedor consistente en un servicio de comunicaciones, existe siempre un punto de demarcación que marca la frontera entre la responsabilidad del usuario del servicio y la del proveedor del servicio. El ámbito de responsabilidad del usuario es el equipamiento de su instalación local, el llamado CPE, y más allá del CPE, todo el equipamiento es responsabilidad del NSPE. Es lo que ocurre, también, al contratar una línea ADSL o una fibra simétrica.

De la figura, se deduce que el dispositivo CSU/DSU forma parte de la instalación del usuario; de hecho, es frecuente que el módulo CSU/DSU esté ya integrado en uno o varios de los puertos del *router* (puertos serie o *serial WAN ports*), o que pueda conectarse en forma de tarjeta (*tarjeta WIC – WAN interface card*) a través de una de las ranuras de expansión previstas para tal fin (situadas habitualmente en la parte posterior del *router*).

## Jerarquías digitales TDM

E-carrier		T-carrier		SONET/SDH		
Nivel	Velocidad (Kbps)	Nivel	Velocidad (Kbps)	Nivel SONET	Nivel SDH	Velocidad (Kbps)
		DS0	64	OC-1	STM-0	51840
		Fractional T1	Múltiplos de 64, hasta 24X	OC-3	STM-1	155520
E1	2048	DS1 (T1)	1544	OC-12	STM-4	622080
		Fractional T3	Múltiplos de 1536, hasta 28X	OC-24	-	1244160
E2	8448	T2	6312	OC-48	STM-16	2488320
E3	34368	DS3 (T3)	44736	OC-192	STM-64	9953280
E4	139264	T4	274760	OC-768	STM-256	39813120

32

Al contratar el servicio de línea serie a un TelCo, éste ofrece una gama de opciones en términos de velocidad (y de precios). Estas opciones están estandarizadas en diversas jerarquías, como por ejemplo las jerarquías *E-carrier* y *T-carrier*, cuyos niveles de velocidad se muestran en la tabla. La jerarquía *E-carrier* es vigente en Europa, Australia, Japón y otros países, mientras que *T-carrier* es la utilizada en Estados Unidos.

SDH y SONET son jerarquías desarrolladas específicamente para fibra óptica (nótese que las velocidades son mayores). Análogamente, SDH es el estándar europeo y SONET el americano. A diferencia de lo que ocurre entre *E-carrier* y *T-carrier*, entre SDH y SONET cambian las denominaciones de los niveles, pero sus velocidades son las mismas.