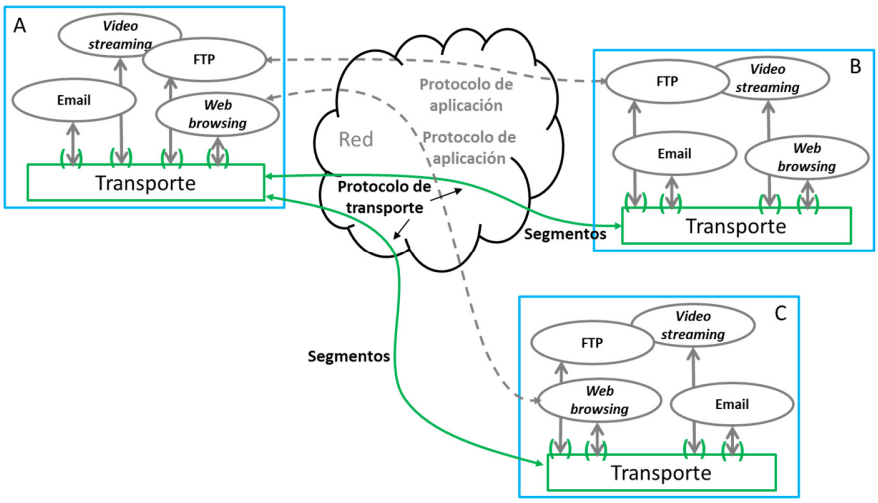
Módulo 2

Arquitectura de Redes

Capítulos



SEBASTIÀ GALMÉS OBRADOR

Universitat de les Illes Balears
Escola Politècnica Superior

CAPÍTULO 6. INTRODUCCIÓN A LAS REDES DE COMPUTADORES

Este capítulo sirve para introducir los conceptos básicos sobre **redes de computadores**, así como una parte fundamental de la terminología propia de este ámbito. Pese a ese carácter introductorio, el capítulo ya permite realizar un primer recorrido de las tecnologías de red existentes, caracterizadas según diversos aspectos: topología, medios de transmisión utilizados y escala geográfica. Adicionalmente, el capítulo presenta una clasificación de las aplicaciones que se ejecutan sobre las redes de computadores en base a sus exigencias de calidad de servicio, y una caracterización de las conexiones de red que soportan estas aplicaciones desde el punto de vista de los flujos de tráfico a que dan lugar.

ÍNDICE

6.1. RED DE COMPUTADORES: DEFINICIÓN Y TIPOLOGÍA

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.

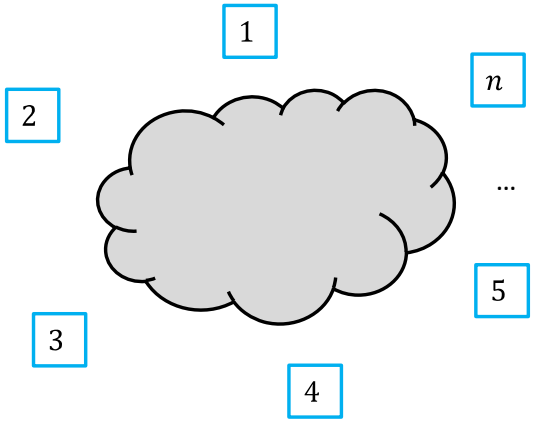


Figura 6.1. Una red de computadores va a ser una solución compleja a un problema de formulación muy sencilla.

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 sus inicios, la investigación en el área de redes ha ido proponiendo soluciones casi tan eficaces como el "todos con todos", 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 conmutadas y redes de medio compartido. Históricamente, se desarrollaron primero las redes conmutadas.

Como se ilustra en la **Figura 6.2**, una **red conmutada** 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 rotondas equivalen a los nodos de conmutación. Como veremos

más adelante, hay dos tipos de nodos de conmutación, que dan lugar a sendos tipos de redes conmutadas. Así, podremos distinguir entre nodos y redes de conmutación de circuitos, por un lado, y nodos y redes de conmutación de paquetes, por otro. Aunque ambos tipos de nodos actúan como redistribuidores de tráfico, su tecnología es totalmente distinta, siendo la de conmutación de paquetes la más utilizada con diferencia en redes de computadores. 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**. Esta diferenciación es significativa, ya que mientras los enlaces de acceso son **enlaces dedicados**, pues su ancho de banda[1] está disponible para un solo usuario, los enlaces intermedios son compartidos por múltiples conexiones, las cuales se reparten el ancho de banda total del enlace. Se dice que son **enlaces multiplexados**. Por tanto, dependiendo del número de conexiones que compartan un enlace multiplexado, el ancho de banda promedio por conexión varía. Por ello, en la práctica los enlaces intermedios tienen que ofrecer un ancho de banda total significativamente mayor que el de los enlaces de acceso.

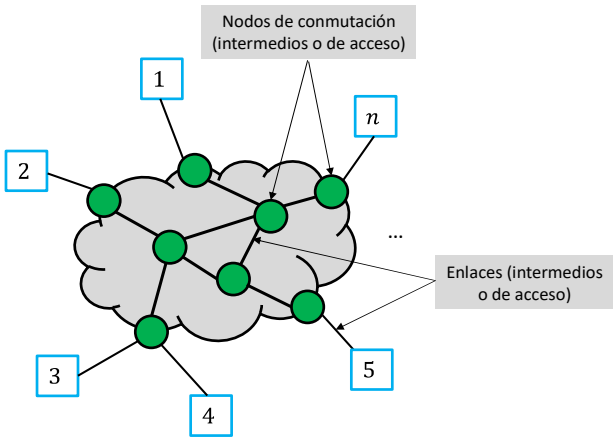


Figura 6.2. Las redes conmutadas presentan una estructura mallada formada por nodos de conmutación y enlaces de transmisión. Los computadores (dispositivos finales, en general) se conectan a la red a través de los nodos de acceso mediante los enlaces de acceso. Los nodos se conectan entre sí mediante los enlaces intermedios.

Como hemos podido observar, en comparación con la solución "todos con todos", la red conmutada requiere menos enlaces de transmisión y a cada computador le basta con tener una sola interfaz de conexión, independientemente del número de computadores restantes.

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

[1] Aunque el término "ancho de banda" se refiere estrictamente a la anchura espectral, dada su relación directa con la velocidad de transmisión (fórmula de Shannon), en la práctica se utiliza como sinónimo de esta última.

implementar una función de encaminamiento, como ocurre con las redes conmutadas: en éstas, los nodos deben ser capaces de encaminar adecuadamente los datos intercambiados por cada pareja de computadores conectados, 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 se presenta un problema que no se daba en las conmutadas: si dos o más computadores quieren transmitir al mismo tiempo, ¿a cuál de ellos se otorga el medio? En el Capítulo 9 abordaremos con más detalle esta cuestión. Por el momento, 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, y particularmente al destinatario. A continuación, se describen brevemente estas topologías.

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 segmento de cable coaxial. Esta topología se encuentra hoy en día totalmente obsoleta.

Más adelante 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 con topología en estrella/árbol. De hecho, es el resultado de la migración de la tecnología de conmutación (de paquetes) al ámbito de las redes de área local[2] centradas en *hub*.

[2] El concepto de red de área local se trata más adelante en este capítulo.

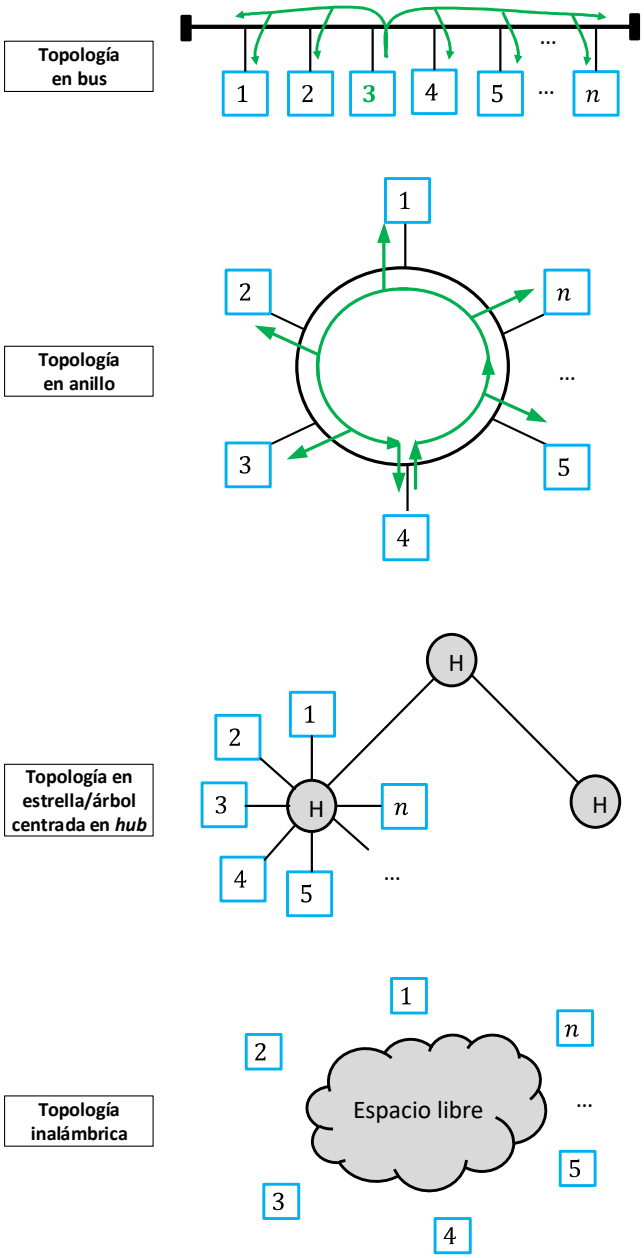


Figura 6.3. Topologías de redes sobre medio compartido.

Finalmente, una alternativa tecnológica de plena actualidad y en continuo desarrollo es la **topología inalámbrica**, pues evita el cableado que en ciertos escenarios puede ser muy costoso e incluso inviable, y además permite la movilidad de los dispositivos interconectados. La **Figura 6.4** muestra la estructura celular típica de las redes inalámbricas (una excepción a tal estructura la constituyen las redes ad-hoc, que se analizarán en el Capítulo 9 al tratar las redes wifi). Como podemos observar, 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, satélite en un servicio MSS, etc.), y otro, el ***backbone***, que suele ser cableado,

aunque no necesariamente, y permite la conexión entre dispositivos pertenecientes a celdas distintas o entre dispositivos y el exterior. Para poder comunicarse, cada dispositivo tiene que formar parte de una celda, para lo cual utiliza una antena isotrópica que le permite mantenerse enganchado al elemento central de la misma independientemente de su posición relativa. El elemento central cumple, entre otras, una función de conmutación, pues dependiendo del destino, deriva el tráfico hacia la misma celda o hacia el *backbone*. Las diferencias de unas redes inalámbricas a otras residen en múltiples aspectos: gestión del medio compartido, bandas de frecuencias de trabajo (con/sin licencia), velocidades de transmisión, alcance, etc.. Trataremos más detalladamente estos aspectos en el Capítulo 9.

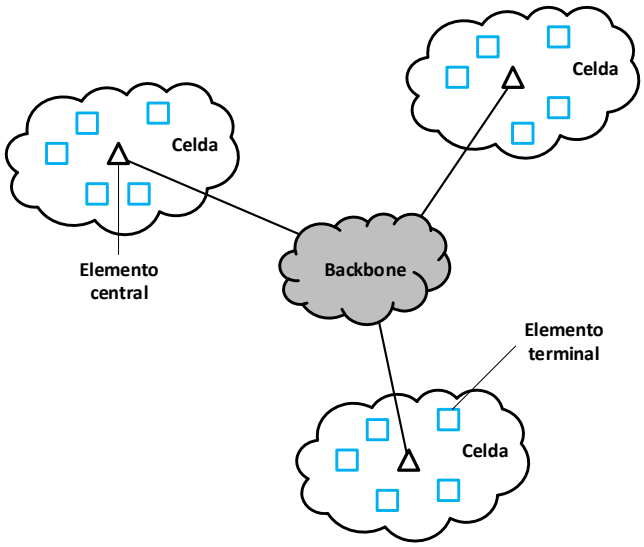


Figura 6.4. Estructura celular típica de las redes inalámbricas. El *backbone* interconecta los elementos centrales entre sí y normalmente proporciona conexión a la Internet pública.

La **Tabla 6.1** recoge las diversas soluciones de red desarrolladas hasta la actualidad, caracterizadas en base a criterios elementales (categoría, mecanismo/topología, medio de transmisión y escala o cobertura), así como los ejemplos o estándares más representativos de cada caso. En lo que a cobertura se refiere, las redes basadas en medio guiado se clasifican en tres clases:

- **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.
- **Red de área metropolitana** o **MAN** (*metropolitan area network*). Cubre un área del tamaño de una ciudad.
- **Red de área amplia** o **WAN** (*wide area network*). Cubre una extensa región geográfica, que puede abarcar desde un país hasta un continente.

En el caso de las redes inalámbricas, cuya estructura es celular, se entiende por cobertura el radio de la celda básica. La clasificación de las redes inalámbricas en cuanto a cobertura es más precisa:

Tabla 6.1. Caracterización de las diversas tecnologías de red desarrolladas hasta el momento actual. Observaciones: (a) en el caso de las redes inalámbricas, las diferencias de escala se reflejan en el tamaño de la celda básica; (b) por radioenlace se entiende un enlace inalámbrico punto a punto.

Categoría	Mecanismo/Topología	Ejemplos	Medio	Escala	Estado
Conmutada	Conmutación de circuitos	PSTN[3] ISDN[4]	Guiado Radioenlace	WAN	Vigente
	Conmutación de paquetes	*Frame Relay*	Guiado, radioenlace	WAN	En declive
		ATM[5]	Guiado, radioenlace	WAN	Vigente
		Ethernet conmutada	Par trenzado Fibra óptica	LAN MAN WAN	Vigente
		Internet	Guiado Radioenlace	Planetaria	Vigente
Medio compartido	Topología en bus	Thick Ethernet Thin Ethernet *Token-bus*	Coaxial	LAN	Obsoleto
	Topología en anillo	*Token-ring*	Par trenzado	LAN	Obsoleto
		FDDI[6]	Fibra óptica	MAN	En declive
	Topología en estrella/árbol centrada en *hub*	Ethernet sobre medio compartido	Par trenzado	LAN	En declive
	Topología inalámbrica	Bluetooth/ZigBee/ UWB[7]/IrDA[8]	Espacio libre	WPAN	Vigente
		WiFi	Espacio libre	WLAN	Vigente
		WiMAX	Espacio libre	WMAN	Vigente
		IEEE 802.20	Espacio libre	WWAN	Vigente
		Redes celulares (1G/2G/3G)			Obsoleto
		Redes celulares (4G/5G ---> 6G)			Vigente
		LPWAN[9] (LoRa/SIGFOX …)			Vigente

- **Red inalámbrica de área personal** o **WPAN** (*wireless personal area network*). Estas redes permiten interconectar dispositivos dentro del ámbito personal, como por ejemplo un computador, un teléfono móvil, una tableta, equipos periféricos, etc. Su cobertura puede variar

[3] PSTN es el acrónimo de *Public Switched Telephone Network* (Red Telefónica Conmutada o RTC, típicamente una en cada país).
[4] 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).
[5] ATM es el acrónimo de *Asynchronous Transfer Mode*.
[6] FDDI es el acrónimo de *Fiber Distributed Data Interface*.
[7] UWB es el acrónimo de *Ultra Wideband*.
[8] IrDA es el acrónimo de Infrared Data Association.
[9] LPWAN es el acrónimo de *Low-Power Wide Area*.

entre 10 cm y 10 m. Por debajo del límite inferior de las WPAN están las **redes de área corporal** o **WBAN** (*wireless body area networks*), cuyo auge es cada vez mayor por sus aplicaciones en el campo de la medicina, y las llamadas **tecnologías de corto alcance** o **de proximidad**, las cuales permiten la transferencia de datos entre dispositivos muy cercanos, del orden de unos pocos centímetros. Entre ellas, las más importantes son **NFC** (*Near Field Communications*) y **RFID** (*Radio Frequency Identification*).

- **Red inalámbrica de área local** o **WLAN** (*wireless local area network*). Su cobertura va de 10 m a 1 Km.
- **Red inalámbrica de área metropolitana** o **WMAN** (*wireless metropolitan area networks*). Su cobertura varía entre 1 Km y 50 Km.
- **Red inalámbrica de área amplia** o **WWAN** (*wireless wide area networks*). Se corresponde con coberturas mayores de 50 Km.

En el límite superior de las redes de área amplia se encuentran las de **escala planetaria**. Algunos ejemplos son las redes de difusión y las redes de datos por satélite. En la mayoría de los casos, la escala planetaria se alcanza mediante la interconexión de diversas tecnologías de área amplia. Por ejemplo, la interconexión de las redes de telefonía móvil a la RTC de cada país, y la interconexión a nivel internacional entre las RTC de todos los países, constituyen una red de telefonía a escala planetaria. No obstante, el ejemplo por excelencia de red a escala planetaria, que es una verdadera amalgama de redes, es la propia **Internet**[10]. Finalmente, en un futuro no muy lejano, quizás podamos hablar también de una **Internet interplanetaria**.

Aparte de la cobertura, hay diferencias de otra índole entre las categorías que se acaban de exponer. Por ejemplo, las redes de área local suelen ser propiedad de la misma entidad que posee 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 de su administración. Por el contrario, en el caso de las redes de área amplia, las entidades que se conectan suelen ser propietarias únicamente de los dispositivos finales, mientras que los recursos de red son aportados por un operador de telecomunicaciones. Esto permite que tales entidades queden liberadas de las tareas de gestión y mantenimiento de la red, lo que en términos económicos resulta ventajoso, pues el coste asumido por cada una 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 sobre medio guiado más populares es Ethernet, especialmente en su variante estrella/árbol centrada en conmutador. Esta variante se utiliza en todos los ámbitos (LAN, MAN, WAN), lo que no hace más que confirmar que la escala geográfica ya no es el criterio fundamental a la hora de clasificar y diferenciar las redes sobre medio guiado, sino que se trata de una característica más. En cambio, las tecnologías inalámbricas presentan una mayor diversificación en términos de cobertura. La **Figura 6.5** compara los grandes grupos de tecnologías inalámbricas del IEEE, principal organización de estandarización de las mismas, desde el punto de

[10] 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.

vista de la cobertura (radio de la celda) y la velocidad de transmisión. Esta figura pone de manifiesto que la cobertura está reñida con la velocidad de transmisión, hecho que tiene su fundamento en la fórmula de Shannon vista en el Capítulo 5 – ecuación (5.13). Sin embargo, la figura muestra tan solo una tendencia, que la propia evolución tecnológica ha ido progresivamente contrarrestando mediante la introducción de técnicas de diversificación de señal y esquemas de modulación más sofisticados, entre otras mejoras. Así, en una versión más detallada de la **Figura 6.5** que incluyera todos los estándares del IEEE y de otras organizaciones, las diversas tecnologías inalámbricas aparecerían distribuidas de manera más uniforme en todo el cuadrante.

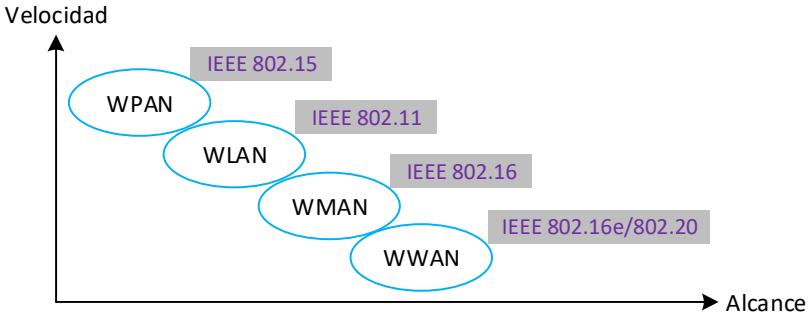


Figura 6.5. Cuadro comparativo de las tecnologías inalámbricas del IEEE en términos de cobertura y velocidad. Se señala un estándar representativo de cada caso.
Fuente: W. C. Y. Lee. Wireless & Cellular Telecommunications (p. 297, fig. 7.3).

Aunque las tecnologías señaladas en la **Tabla 6.1** tienen su razón de ser propia y pueden operar de forma autónoma, un grupo importante de ellas constituye habitualmente la primera etapa de nuestras conexiones a Internet. Por ello, las revisitaremos como tecnologías de acceso en el Capítulo 9. Veamos, a continuación, los requisitos generales de diseño de las redes.

6.2. Requisitos de Diseño de las Redes

Cualquier red debe satisfacer unos criterios de diseño, básicamente en cuatro ejes: comportamiento (*performance*), escalabilidad (*scalability*), fiabilidad (*reliability*) y seguridad (*security*). Como vemos a continuación, la expresión más precisa de las **métricas de comportamiento** se apoya en el concepto de paquete. El **paquete** es la unidad básica de información que circula de forma autónoma por una red. Contiene generalmente un fragmento de un mensaje más grande, así como la información de control necesaria para poder ser encaminado correctamente de origen a destino, cualquiera que sea el tipo de red. Este último aspecto es importante, pues, aunque el concepto de paquete surge con las redes de conmutación de paquetes, no es exclusivo y va más allá de las mismas; de hecho, la información se transmite de forma empaquetada en todas las redes de computadores, sean redes basadas en medio compartido, redes conmutadas basadas en conmutación de paquetes (naturalmente), e incluso, como veremos más adelante al tratar los enlaces punto a punto de larga distancia, redes conmutadas basadas en conmutación de circuitos. Revisitaremos el concepto de paquete en el contexto de la arquitectura de capas de las redes en el

siguiente capítulo (**Tabla 7.2**). Por el momento, estamos ya en condiciones de definir las métricas de comportamiento de manera precisa:

- ***Throughput***. Es la velocidad media con que se transmiten los bits de una determinada conexión. Un concepto parecido es el ***goodput***, que es la velocidad media con que se transmiten los bits de datos de una determinada conexión. Por ejemplo, supongamos que en una conexión se transmiten 70 paquetes por segundo en promedio, donde cada paquete está formado por 800 bits, de los que un 20% son de control; en estas condiciones, el *throughput* de la conexión es de $70 \cdot 800 = 56$ Kbps, mientras que el *goodput* vale $56 \cdot 0.8 = 44.8$ Kbps.
- **Latencia**. Es el retardo que experimenta un paquete entre origen y destino. En su momento veremos que este retardo engloba cuatro retardos más específicos: tiempos de transmisión, retardos de propagación, retardos de procesamiento y tiempos de espera. Generalmente la latencia varía de un paquete a otro de la misma conexión, por lo que se sobreentiende que es la **latencia media**. Esto da pie a introducir la tercera métrica de comportamiento, el *jitter*.
- ***Jitter***. Es la variación de latencia que experimentan los paquetes de una misma conexión. Una forma de cuantificarla es como desviación típica de la latencia con respecto a la latencia media.

La **escalabilidad** de una red de computadores es su capacidad para crecer en cobertura y/o en número de usuarios manteniendo las métricas de comportamiento en valores aceptables. En gran medida, la escalabilidad depende de la estrategia de interconexión utilizada, siendo las redes conmutadas mucho más escalables que las redes sobre medio compartido.

La **fiabilidad** de una red de computadores es su capacidad para seguir operando correctamente a pesar de que se produzcan fallos en los componentes de la misma. Puesto que son muchos los tipos de fallos que se pueden dar, la fiabilidad en el ámbito de las redes adquiere una dimensión muy amplia, abarcando técnicas como el control de flujo, el control de errores y el control de congestión, así como las diversas estrategias para el encaminamiento robusto de los paquetes en caso de fallos de nodos y/o enlaces.

Finalmente, el último gran requisito de diseño es la **seguridad**, que a su vez abarca tres aspectos:

- **Confidencialidad**. Consiste en garantizar que la información intercambiada entre los dos extremos de una comunicación no sea accesible a terceros.
- **Integridad**. Consiste en garantizar que la información intercambiada entre los dos extremos de una comunicación no pueda ser alterada por un tercero.
- **Autenticación**. Consiste en garantizar que el usuario que accede a una red o a un recurso a través de la misma no pueda ser suplantado por otro.

Para terminar esta sección, la **Tabla 6.2** muestra el grado de exigencia de las diversas aplicaciones en cuanto a comportamiento y fiabilidad. Estas exigencias constituyen los llamados requisitos de **calidad de servicio**, que en un análisis más profundo pueden cuantificarse. Es natural que no todas las aplicaciones tengan los mismos requisitos: por ejemplo, la transferencia de ficheros exige máxima fiabilidad, pues el fichero recibido debe ser una copia exacta del fichero que fue transmitido, pero es más tolerante en términos puramente temporales (latencia y *jitter*); con las

conversaciones telefónicas pasa lo contrario, pues admiten una cierta tasa de errores de transmisión, mientras estos no sean detectados por el oído humano, pero son mucho menos tolerantes a los excesos de latencia y *jitter*, pues estos provocan una sensación desagradable en el abonado que escucha. También hay aplicaciones que presentan estrictos requisitos en todos los aspectos; es el caso, por ejemplo, de la computación distribuida.

Tabla 6.2. Requisitos de calidad de servicio de las aplicaciones. Los términos alto, medio y bajo se refieren al grado de exigencia del requisito correspondiente, no al valor del mismo. Por ejemplo, el grado de exigencia de la transferencia de ficheros en cuanto a latencia es bajo, lo que significa que dicha aplicación admite latencias relativamente altas (no bajas).
Fuente principal: A. S. Tanenbaum. *Computer Networks*. Prentice-Hall, 2011. Fifth Edition.

Aplicación	*Throughput*	Latencia	*Jitter*	Fiabilidad
Correo electrónico	Bajo	Bajo	Bajo	Alto
Transferencia de ficheros	Medio	Bajo	Bajo	Alto
Acceso web	Medio	Medio	Bajo	Alto
Acceso remoto	Bajo	Medio	Medio	Alto
Audio bajo demanda	Medio	Bajo	Alto	Bajo
Vídeo bajo demanda	Alto	Bajo	Alto	Bajo
Telefonía	Bajo	Alto	Alto	Bajo
Videoconferencia	Alto	Alto	Alto	Bajo
Computación distribuida	Alto	Alto	Alto	Alto

Este capítulo introductorio se completa con la siguiente sección, que trata de la caracterización de las conexiones de red desde el punto de vista de los flujos de datos que generan.

6.3. CARACTERIZACIÓN DE LAS CONEXIONES DE RED

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. El otro tipo de enlace es el llamado **enlace *broadcast*** (o de difusión), en el cual el medio de transmisión es compartido, y la transmisión de cualquier dispositivo alcanza a todos los demás (lo que no significa que todos los demás sean destinatarios). Teniendo en cuenta esta clasificación de los enlaces, podemos afirmar que las redes conmutadas no son más que estructuras malladas que contienen múltiples enlaces punto a punto entre nodos de conmutación (de hecho, también se las conoce como **redes punto a punto**), mientras que en una red de medio compartido el enlace es broadcast. En el caso concreto de las redes de medio compartido inalámbricas, se suele distinguir entre el enlace broadcast ascendente, o simplemente **enlace ascendente**, utilizado por los dispositivos finales para enviar paquetes al elemento central, y el enlace broadcast descendente, o simplemente **enlace descendente**, utilizado por el elemento central para enviar paquetes a los dispositivos finales.

Por otro lado, desde el punto de vista de la direccionalidad de los flujos de datos, la comunicación (o conexión) a través de un enlace puede ser de tres tipos:

- ***Simplex***: 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 (enlace punto-multipunto).
- ***Half-duplex***: La comunicación es bidireccional, pero no admite simultaneidad. Es decir, en un momento dado, solo puede haber flujo en una dirección.
- ***Full-duplex***: 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 organismo de estandarización ANSI. La UIT-T, por su parte, sólo diferencia entre enlaces ***simplex*** y ***duplex***, correspondientes, respectivamente, a los enlaces *half-dúplex* y *full-dúplex* de la terminología ANSI. Salvo que se especifique lo contrario, adoptaremos la terminología ANSI.

Una magnitud que depende de la tipología de las conexiones que acabamos de ver es la **velocidad de transferencia**, que es la cantidad de bits que se intercambian por unidad de tiempo a través de un enlace. Si el enlace es *simplex* o *half-duplex*, la velocidad de transferencia coincide con la velocidad de transmisión; en cambio, si el enlace es *full-duplex*, la velocidad de transferencia es la suma de velocidades en ambas direcciones. Por ejemplo, si la velocidad nominal de un enlace *full-duplex* es de 10 Gbps, su velocidad de transferencia nominal es de 20 Gbps.

Por último, otra cuestión relacionada con los flujos de datos es la referida al modo de direccionamiento, según el cual podemos distinguir entre los siguientes tres tipos de conexiones dependiendo de 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 de conexión son aplicables a cualquier tipo de red, sea 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.

CAPÍTULO 7. MODELOS DE CAPAS

La conexión entre computadores a través de una red o un conjunto de redes es un proceso complejo que resulta mucho más fácil de abordar subdividiéndolo en subprocesos más simples. Es la aplicación del viejo principio de "dividir y conquistar". Estos problemas menores son de naturaleza muy distinta y además tratables independientemente. Por ejemplo, nada tiene que ver el diseño de una aplicación que vaya a ejecutarse sobre Internet (transferencia de ficheros, correo electrónico, navegación web, etc.), con la codificación de línea que pueda utilizar un computador para enviar la información asociada a cualquiera de esas aplicaciones. Si nos centramos en el usuario, éste está más cerca de las aplicaciones que de los códigos de línea, por lo que entendemos que esos dos problemas, y sus correspondientes soluciones, se encuentran a diferentes niveles. De ahí que la solución final toma la forma de soluciones parciales estratificadas por niveles o capas. La **arquitectura de capas de una red** consiste precisamente en esta descomposición por capas del problema de conectar dos computadores. Los dos modelos de capas más importantes son OSI y TCP/IP, si bien el primero es un estándar oficial que constituye sólo una referencia teórica, pues no se ha llevado a la práctica, y el segundo es el estándar de facto por excelencia, pues es la base de la red Internet. De todos modos, en la parte inicial de este capítulo desarrollaremos una arquitectura genérica, que sólo al final identificaremos con la arquitectura TCP/IP.

ÍNDICE

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 entablar 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".

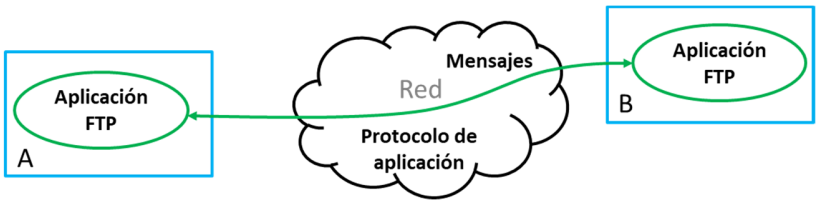


Figura 7.1. Las aplicaciones dialogan intercambiándose mensajes según las reglas de un protocolo de aplicación. En este caso, las aplicaciones son de transferencia de ficheros.

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.

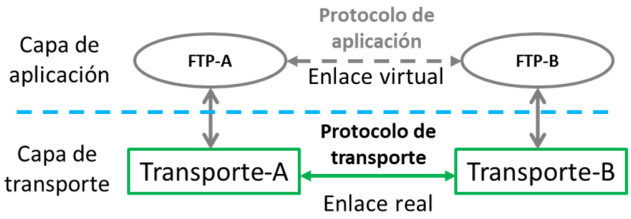


Figura 7.2. Arquitectura de dos capas: capa de aplicación y capa de transporte.

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[11] 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).

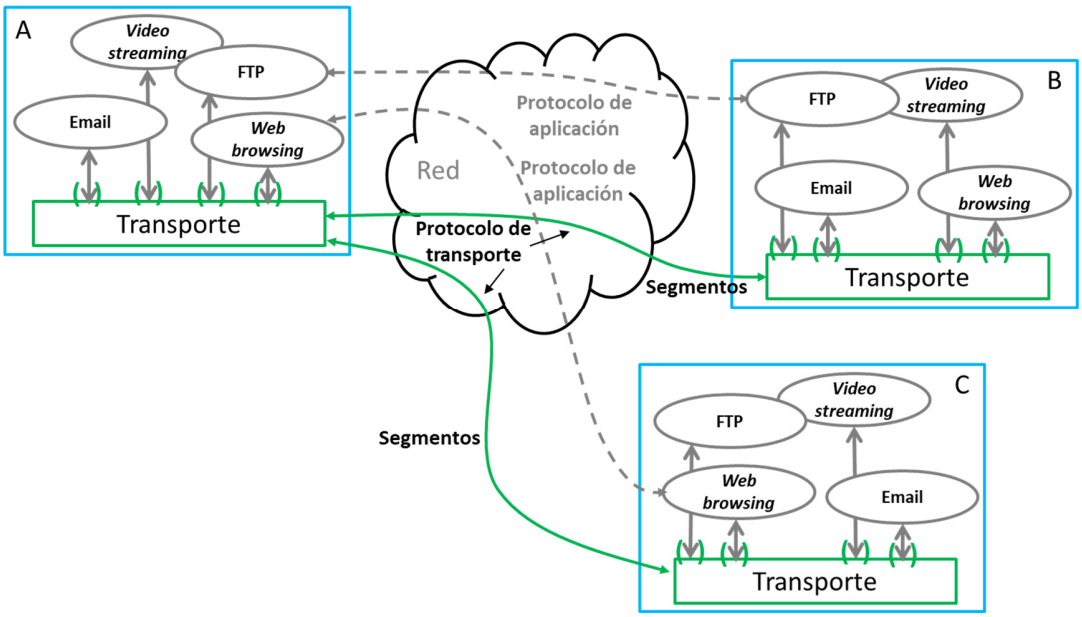


Figura 7.3. Multiplexado de conexiones a través de la capa de transporte.

[11] 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:

- Porque 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; en consecuencia, fusionar los mecanismos encaminados a satisfacer tales requisitos con cada protocolo de red existente, conllevaría un replicado innecesario de los primeros.
- Porque así liberamos a los desarrolladores de software de transporte de la necesidad de conocer e implementar los múltiples protocolos de red.

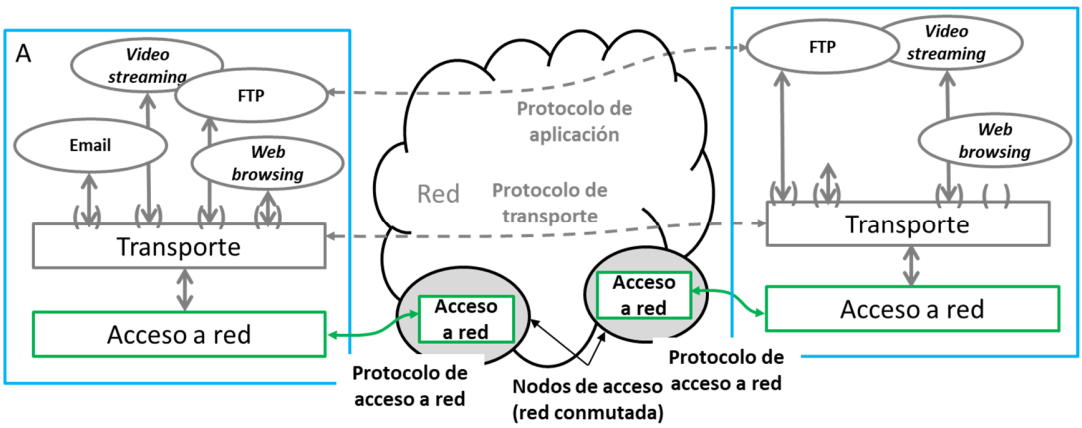


Figura 7.4. Arquitectura de tres capas: capa de aplicación, capa de transporte y capa de acceso a red. Se ha supuesto que la red es conmutada.

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.

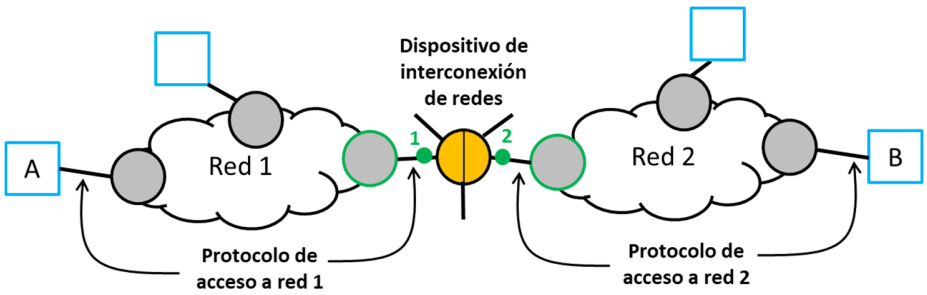


Figura 7.5. Conexión entre A y B a través de redes distintas.

¿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 unívocamente un dispositivo a escala global o mundial, mientras que una **dirección local** permite identificar unívocamente 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.

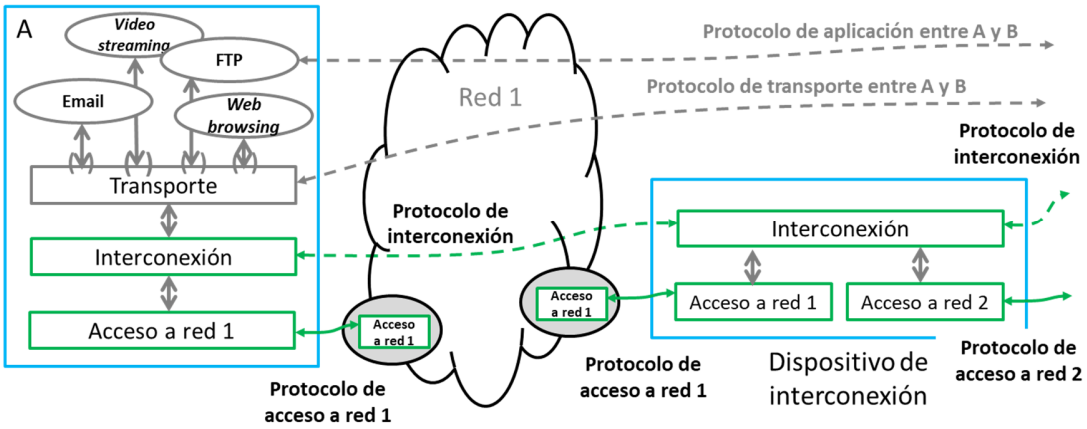


Figura 7.6. Arquitectura de cuatro capas. En el dispositivo de interconexión, sólo se implementan las dos capas inferiores.

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[12], 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 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

[12] 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.

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)[13]:

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. Esa 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.

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

[13] Douglas S. Comer: *Internetworking with TCP/IP. Volume One*. Pearson Education Limted, 2014. Sixth Edition.

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 subprocesos más sencillos de distinta jerarquía.
Capa	Cada uno de los niveles que corresponden a los subprocesos 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 (*peers*)	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 (*protocol data unit*)	Unidad básica de información de un protocolo. 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 o, cuando existe, mediante un nombre común. Tanto en las arquitecturas TCP/IP como OSI (Secciones 7.2 y 7.3, respectivamente), se utilizan las siguientes denominaciones comunes: • Mensaje: PDU de la capa de aplicación (ya introducido). • Segmento: PDU de la capa de transporte (ya introducido). • **Paquete**: PDU de la capa Internet (arquitectura TCP/IP) o de la capa de red (arquitectura OSI). • **Trama**: PDU de la capa de acceso a red o enlace (arquitectura TCP/IP) o de la capa de enlace (arquitectura OSI).
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 **interfaz física**.

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.

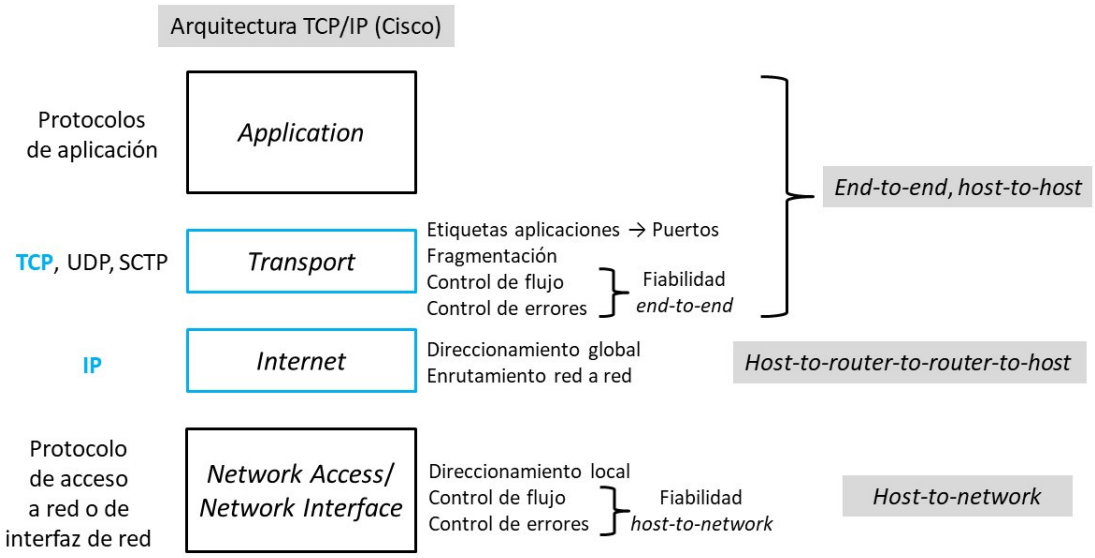


Figura 7.7. Arquitectura TCP/IP de cuatro capas (modelo de Cisco). El nombre de la arquitectura proviene precisamente de los dos protocolos más importantes del núcleo de la misma.

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 *internet*. Los principales protocolos en la capa de transporte son TCP (*Transport Control Protocol*) y UDP (*User Datagram Protocol*). 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 *internet* 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.

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.

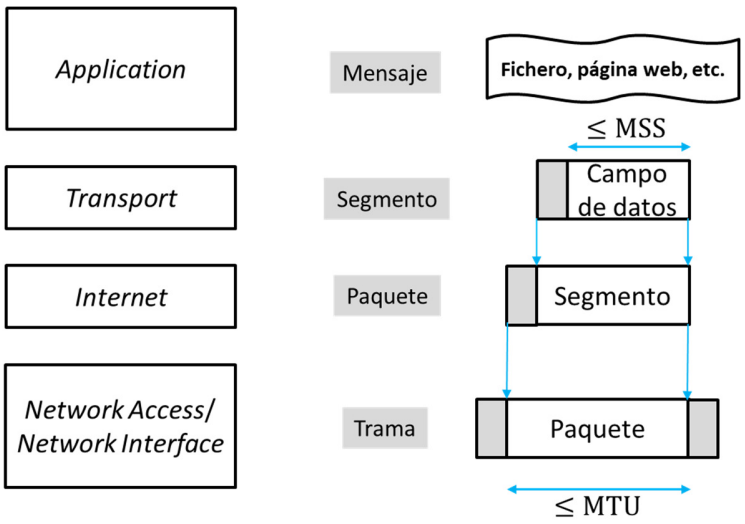


Figura 7.8. Proceso de encapsulado.

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 recuerda la denominación de las diversas unidades básicas de información según la capa considerada.

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, que resulta casi evidente una vez descritas las funciones de la capa *internet* y la capa de acceso a red: las direcciones globales origen y destino son informaciones de control que debe manejar la capa *internet*, 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[14], que añade una cabecera a cada fragmento para formar un segmento.
2. El segmento se encapsula en la PDU de la capa *internet* (paquete).
3. El paquete se encapsula en la PDU de la capa de acceso a red (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 a partir de los bits a proteger (cuando el módulo de acceso a red es una tarjeta de red); para no desaprovechar esta alta velocidad, dicho cómputo puede realizarse al mismo tiempo que la trama se transmite, con sólo uno o dos bits de retraso, si el código de protección puede acoplarse por detrás 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.

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

[14] Este proceso es algo más complejo, y se analiza con detalle sobre el ejemplo de una sesión de navegación web (Sección 7.5).

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

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 *internet*	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
Application	*Application*	*Application*	*Application*	*Application*	*Application*
Transport	*Transport*	*Transport*	*Transport*	*Transport*	*Transport*
Internet	*Network*	*Internet*	*Network*	*Internet*	*Internet*
Link	*Link*	*Network Access/ Network Interface*	*Data Link*	*Network Access/ Data Link*	*Network Interface*
	Physical		*Physical*	*Physical*	*Physical hardware*

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[15], 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 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

[15] La arquitectura TCP/IP fue desarrollada inicialmente sobre máquinas Unix.

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*[16]. 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.

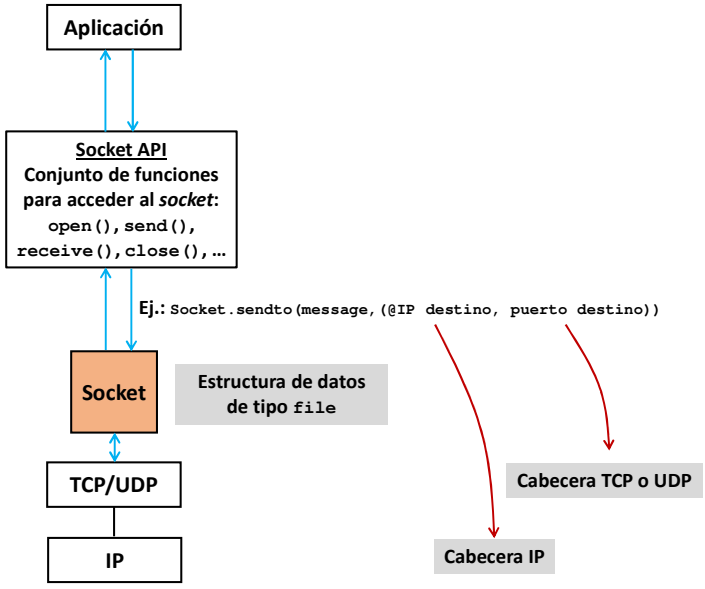


Figura 7.9. El Socket API.

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

[16] 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 muestra 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*) o función de conmutación (*switching 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 conmutación**. 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.

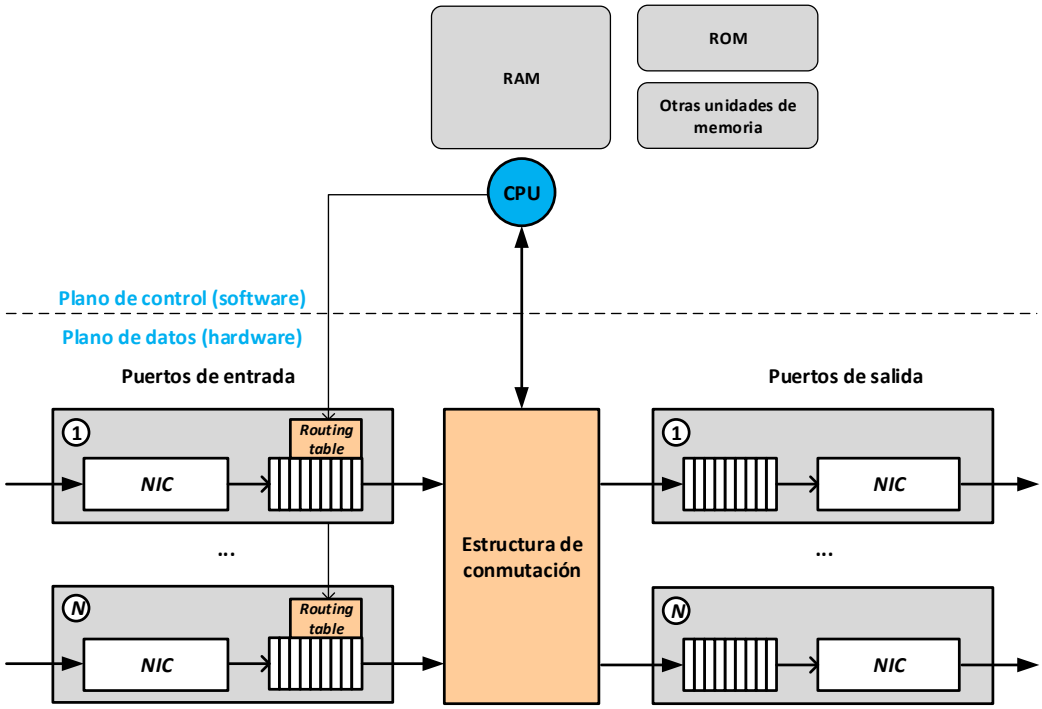


Figura 7.11. Estructura básica de un *router* moderno. En general, la NIC admite operación full-dúplex, des-encapsulando los paquetes de las tramas entrantes y volviéndolos a encapsular en las tramas salientes. Los *buffers* de entrada y salida contienen, por tanto, paquetes.

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 conmutación 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[17] 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 conmutación. 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 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

[17] 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.

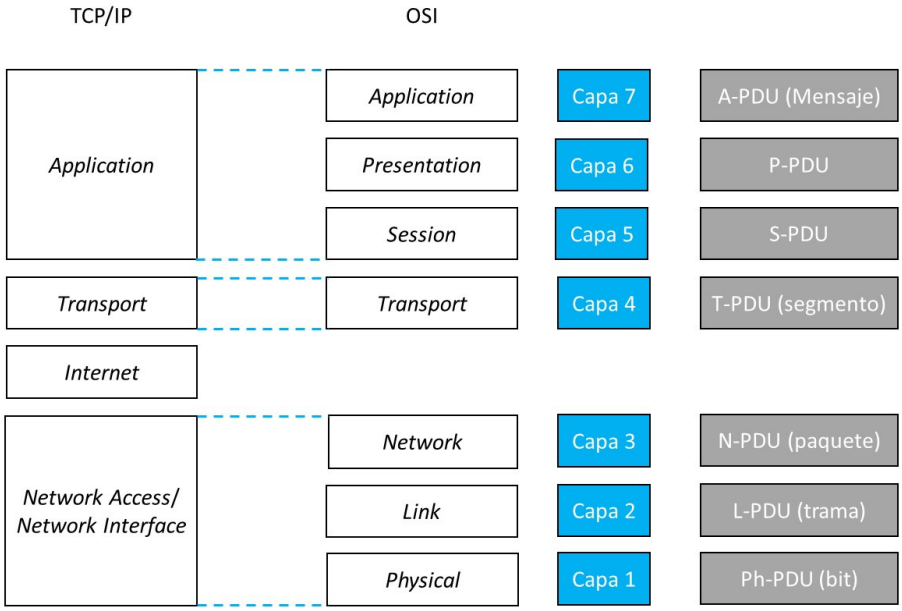


Figura 7.13. Arquitectura OSI y comparativa con la arquitectura TCP/IP (modelo Cisco).

En la **Figura 7.13** se muestra la arquitectura OSI comparada con la arquitectura TCP/IP. Como podemos observar, consta de un total de siete capas, que también se reconocen por su enumeración. Las cuatro capas superiores conllevan interacción entre extremos (*end-to-end*), mientras que en las tres capas inferiores la interacción es nodo a nodo (*node-to-node*). La figura incluye también la denominación de las unidades básicas de información según la capa OSI considerada. Aunque las arquitecturas TCP/IP y OSI se desarrollaron independientemente, la **Figura 7.13** sugiere que podemos ver la primera como una modificación de la segunda en los siguientes términos:

- La agrupación en una sola capa (capa de aplicación) de las tres capas superiores de la arquitectura OSI: 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 agrupación en una sola capa (capa de acceso a red o de interfaz de red o, en algunos modelos de la arquitectura TCP/IP, capa de enlace) de las tres capas inferiores de la arquitectura OSI: una **capa de red** encargada de las funciones de encaminamiento y control de congestión de la única red mallada a la que se suponen conectados todos los dispositivos en la arquitectura OSI, una **capa de enlace** que implementa las funciones de control de flujo y errores en cada enlace entre

nodos de esa única red, y una **capa física** que aglutina los aspectos físicos de la transmisión en cada uno de esos enlaces.

- La inserción de una nueva capa, la capa *internet*, que tiene en cuenta el hecho de que, por lo general, las conexiones cruzan múltiples redes.

Como vemos en la figura, sólo la **capa de transporte** del modelo OSI es totalmente equiparable a la capa de transporte del modelo TCP/IP.

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, las tarjetas de red utilizadas por los computadores no son más que la implementación hardware de las funciones necesarias para acceder a una determinada red, funciones directamente relacionadas con las tres capas más bajas de la arquitectura OSI. Con respecto a este último aspecto, la arquitectura TCP/IP recoge la idea de que cuando un computador se conecta a una red determinada, no importa tanto como esta red se organiza internamente, sino el hecho de implementar correctamente las reglas de acceso a la misma.

Una última observación acerca de la **Figura 7.13**. Algunos autores equiparan la capa *internet* de la arquitectura TCP/IP con la capa de red de la arquitectura OSI. Tiene sentido, pues la unidad básica de información en ambas capas es el paquete, y es en ambas capas donde reside la función de encaminar los paquetes de origen a destino. No obstante, son encaminamientos en dos niveles diferentes; de hecho, lo que es una red en OSI puede ser un simple enlace en TCP/IP, de ahí que también tiene sentido equiparar la capa de enlace (acceso a red) de la arquitectura TCP/IP con las tres capas más bajas de la arquitectura OSI.

7.4. CONMUTACIÓN DE PAQUETES

El concepto de conmutación de paquetes fue introducido en 1965 por Donald Davies, científico computacional del *National Physical Laboratory* (UK). La conmutación de paquetes es parte esencial de la función de encaminamiento de una red de computadores con estructura mallada. Básicamente, la **conmutación de paquetes** es la operación mediante la cual un nodo de la red transfiere cada paquete entrante al puerto de salida que le corresponde. La estructura de cualquier nodo en una red de conmutación de paquetes obedece, en su esencia, a la de un *router* descrita en la **Figura 7.11**. Por tanto, para realizar la función de conmutación, el nodo coteja dos informaciones por cada paquete: la información de encaminamiento contenida en la cabecera del mismo, y la información disponible en una tabla de encaminamiento.

La conmutación de paquetes no es exclusiva de Internet y las redes IP en general, sino que es la base de funcionamiento de una variada gama de redes de computadores, anteriores o coetáneas de Internet. Ejemplos de tales redes son X.25 (obsoletas), FR (*Frame Relay*) y ATM (*Asynchronous Transfer Mode*). Un elemento diferenciador que permite una primera clasificación de todas estas redes se refiere a la forma de ejecutar la función de encaminamiento, la cual repercute, a su vez, en

la información de encaminamiento que maneja la función de conmutación de cada nodo. En la **conmutación de paquetes por circuito virtual**, antes de que tenga lugar la transferencia de paquetes de datos de origen a destino, la red determina una ruta para dicha transferencia. Se trata, pues, de un **servicio orientado a la conexión**, caracterizado por tres fases:

- Fase de **establecimiento de la conexión**. La red, es decir, un conjunto de nodos de la misma cooperando entre sí, determina la ruta por la que deberá encauzarse la conexión solicitada. Esta ruta es bidireccional, pues se utilizará para el intercambio de paquetes de datos en ambos sentidos de la conexión. Además, a cada conexión establecida se le asigna un valor, mediante el cual será identificada tanto en la cabecera de los paquetes como en las tablas de encaminamiento de los nodos.
- Fase de **transferencia de datos**. Los dos extremos de la conexión intercambian paquetes de datos y otros paquetes de control a través de la ruta establecida.
- Fase de **terminación de la conexión**. Puede ser iniciada por cualquiera de los dos extremos, e implica liberar el identificador de la conexión para poder ser utilizado por otra.

Una alternativa a la conmutación de paquetes por circuito virtual es la **conmutación de paquetes en modo datagrama**. Se trata de un **servicio no orientado a la conexión**, carente de las fases de establecimiento y terminación. Por tanto, no hay búsqueda de ruta entre origen y destino y cada paquete se encamina independientemente de los demás, aunque formen parte de la misma conexión. En este caso, la cabecera de los paquetes debe contener la única información de encaminamiento posible, que son las direcciones origen y destino. Es por ello por lo que se les denomina también **datagramas**.

Mientras las redes X.25, FR y ATM son redes de conmutación de paquetes por circuito virtual, la red de redes que es Internet funciona en modo datagrama (de ahí la denominación frecuente de datagramas IP a las unidades básicas de información en este ámbito). Las dos opciones tienen sus ventajas e inconvenientes. El uso de datagramas evita el consumo de tiempo y recursos de las fases de establecimiento y terminación de la conexión, lo cual resulta especialmente ventajoso en el caso de conexiones entre extremos muy remotos, como es el caso de la red Internet. Por otro lado, el uso de circuitos virtuales asegura que todos los paquetes de una misma conexión siguen una misma ruta, de manera que no pueden llegar desordenados al destinatario; si se utilizan datagramas, no existe garantía de que todos ellos sigan la misma ruta, y puede que el orden de llegada de los mismos no coincida con el orden en que fueron enviados, con lo cual deben ser reordenados en el destinatario (si se exige transferencia 100% fiable).

Al margen de las diferencias entre el uso de circuitos virtuales y datagramas, la conmutación de paquetes presenta dos características intrínsecas, ilustradas en la **Figura 7.14**:

- Operación en modo ***store-and-forward*** (almacenamiento y reenvío). Cada nodo de una red de conmutación de paquetes tiene que recibir completamente cada paquete para poder examinar su cabecera y determinar el puerto de salida que le corresponde. Entre el instante en que el paquete es recibido completamente y el instante en que empieza a ser reenviado, transcurre

un tiempo que genéricamente se conoce como tiempo de procesado, aunque también incluye los tiempos de espera en los *buffers* de entrada y salida (**Figura 7.11**). Este modo de operación ralentiza el intercambio de paquetes entre origen y destino, pues se repite en cada nodo de paso. Para reducir el retardo extremo a extremo que ello conlleva, se introdujo posteriormente la técnica ***cut-through***, que aprovecha la circunstancia de que toda la información del paquete que el nodo debe examinar reside en la cabecera; desde este punto de vista, basta con esperar el tiempo de transmisión equivalente a la cabecera para que el paquete pueda ser procesado. No obstante, esta técnica no tuvo mucho éxito por diversas razones: en primer lugar, ignora el campo de chequeo de errores que forma parte del remolque de la trama que encapsula el paquete, con lo cual el nodo puede estar reenviando paquetes erróneos; por otra parte, en los picos de tráfico no aporta mucho, pues igualmente los paquetes se acumulan en los puertos de entrada, por lo que se vuelve a consumir un tiempo de transmisión completo al recibir cada uno de ellos.

- **Multiplexado estadístico**. Hablamos de multiplexado cuando varias conexiones comparten un mismo recurso de transmisión (enlace), como es el caso de los paquetes que se acumulan en cualquiera de los puertos de salida de un nodo (**Figura 7.11**). En principio, el único criterio de compartición de cualquier enlace en una red de conmutación de paquetes consiste en asignarlo a los paquetes según el orden de llegada de estos al *buffer* de salida, es decir, el enlace no se distribuye entre las diversas conexiones de acuerdo con un ciclo predeterminado, sino que se va otorgando paquete tras paquete según una disciplina FIFO (*First In First Out*), independientemente de la conexión a la cual pertenece cada uno.

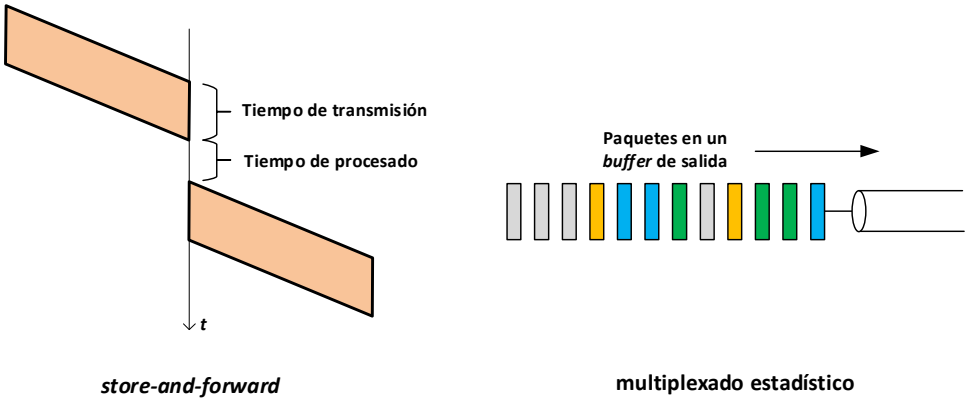


Figura 7.14. Características intrínsecas de la conmutación de paquetes: operación de los nodos en modo *store-and-forward* y compartición de los enlaces mediante la técnica de multiplexado estadístico. En la ilustración de esta última técnica, cada color representa una conexión.

Estas dos características contrastan con las que presenta otra técnica de conmutación radicalmente distinta, la conmutación de circuitos, utilizada sobre todo en redes de telefonía. Aunque dejamos los detalles de esta técnica para más adelante, cabe señalar que los nodos de conmutación de circuitos no operan en modo *store-and-forward*, y que el multiplexado de diversas conexiones en un enlace se realiza según un esquema prestablecido, de acuerdo con dos posibles modalidades:

multiplexado por división de frecuencia o FDMA (*Frequency Division Multiple Access*), típico de la telefonía analógica, y multiplexado por división de tiempo o TDMA (*Time Division Multiple Access*), típico de la telefonía digital.

7.5. 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 7.15**. 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. 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.

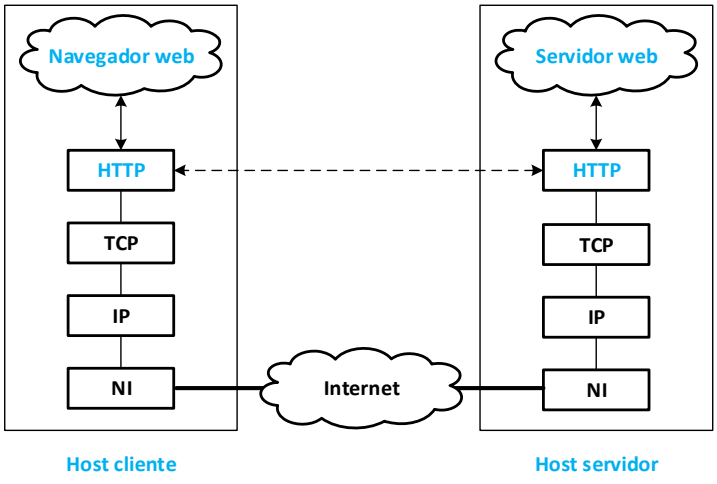


Figura 7.15. Componentes de la navegación web en el contexto de la arquitectura TCP/IP.

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[18]. Esta posición prominente se debe no

[18] Sandvine: *The Global Internet Phenomena Report*. September 2019. https://www.sandvine.com/resources

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. 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 adecuado.

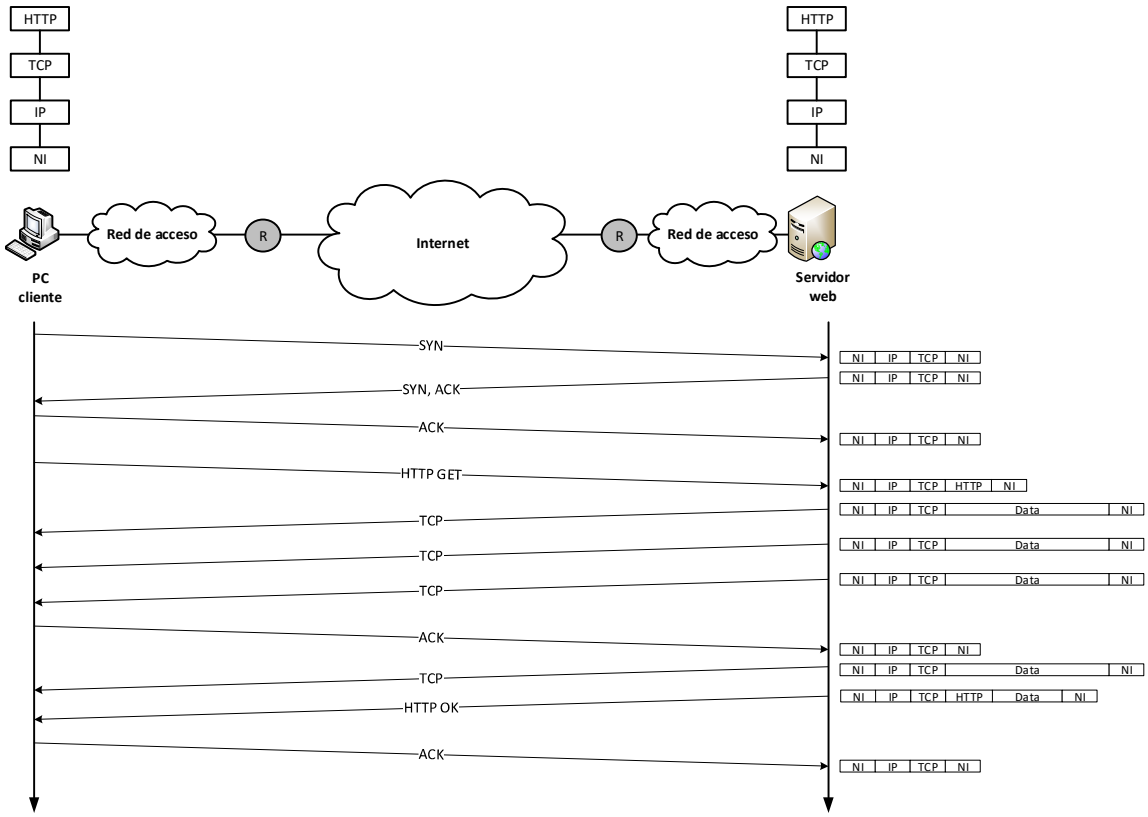


Figura 7.16. Intercambio de mensajes HTTP y segmentos TCP durante la descarga del contenido HTML de una página web, al principio de una sesión web con un determinado servidor. No se muestran las posibles descargas subsecuentes de otros objetos imbricados en la página.

El diagrama temporal de la **Figura 7.16** 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*. El diálogo mostrado requiere algunas observaciones:

- 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 7.17** 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.

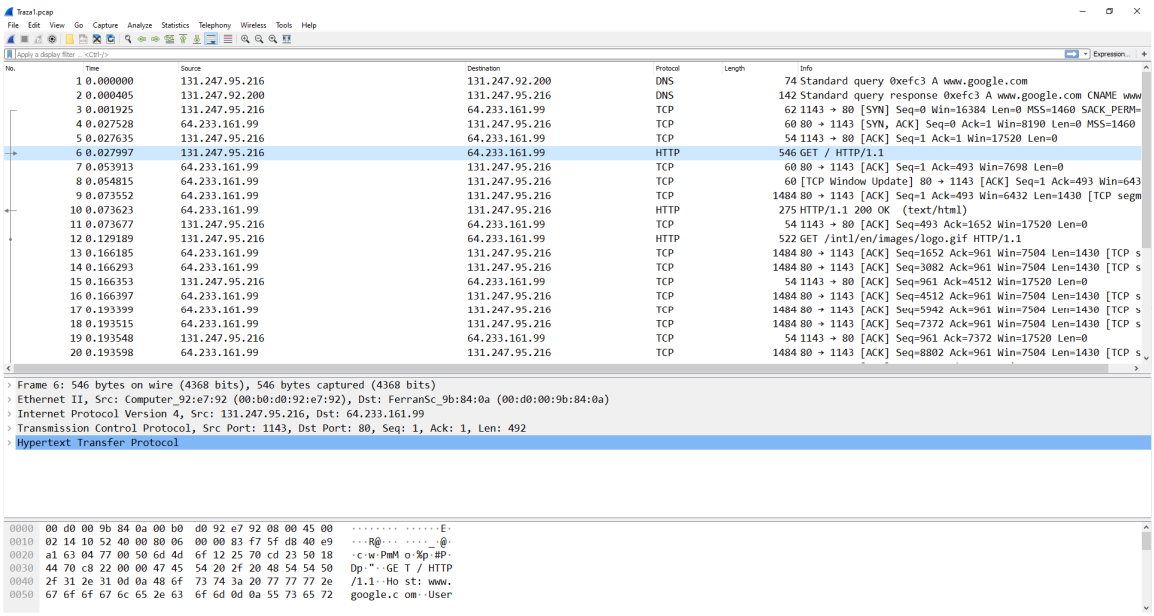


Figura 7.17. Ventana Wireshark que describe parte de la captura de una sesión web. La trama seleccionada encapsula el primer mensaje `HTTP GET` enviado.

CAPÍTULO 8. CONTROL DE FLUJO, CONTROL DE ERRORES Y CONTROL DE CONGESTIÓN

En las arquitecturas estudiadas en el capítulo anterior, algunas funciones se replican en varias capas. Es el caso del **control de flujo** y el **control de errores**, asignadas tanto a la capa de transporte como a la capa de enlace de los modelos OSI y TCP/IP. En ambos modelos, la ejecución de estas funciones en la capa de transporte es *end-to-end*, es decir, a través del enlace virtual que interconecta los dos módulos de transporte finales, mientras que la ejecución en la capa de enlace es *node-to-node* (OSI) o *host-to-network* (TCP/IP), es decir, entre dos módulos de enlace conectados directamente mediante un enlace físico. Dado, pues, el protagonismo de estas funciones, las tratamos aparte en el presente capítulo. Si bien ambas son conceptualmente distintas, los mecanismos de control de errores son una extensión de los mecanismos de control de flujo, y por eso es habitual estudiarlas juntas, empezando naturalmente por el control de flujo. Concretamente, este capítulo se ocupa de los mecanismos de control de flujo y errores de la capa de enlace, pues las versiones de tales mecanismos en la capa de transporte son esencialmente las mismas, aunque con algunas diferencias. Estas versiones se analizan con detalle al estudiar el protocolo de transporte TCP.

Por otro lado, la función de **control de congestión**, siendo conceptualmente muy similar a la de control de flujo, se implementa de forma muy distinta a ésta. Además, hay notables diferencias en cuanto a la misma función de control de congestión de una arquitectura a otra. En el caso de la arquitectura OSI, la responsabilidad de la función de control de congestión corresponde a la capa de red, mientras que en la arquitectura TCP/IP, es el protocolo TCP de la capa de transporte el encargado de implementarla. Por tanto, aunque la función de control de congestión se describe brevemente en este capítulo, el análisis detallado de la misma también se aborda al estudiar el protocolo TCP.

La parte final de este capítulo está dedicada al **protocolo HDLC**, máximo exponente de la implementación de las funciones de control de flujo y errores en la capa de enlace. Este protocolo, no sólo es importante por su uso habitual en enlaces punto a punto de larga distancia, sino también porque de él han derivado otros protocolos de enlace igualmente muy utilizados.

ÍNDICE

8.1. CONTROL DE FLUJO

Una de las ventajas de las redes de conmutación de paquetes es que permiten interconectar elementos terminales que operan a velocidades distintas. Por ejemplo, un dispositivo (ordenador, estación de trabajo, servidor, etc.) con una conexión a 1 Gbps puede enviar datos a otro con una conexión de tan solo 10 Mbps. Esto es posible porque el dispositivo que recibe dispone de un buffer que amortigua la diferencia de velocidades. No obstante, el buffer tiene una capacidad limitada, por lo que el dispositivo que recibe debe poder frenar al que transmite cuando ese buffer está lleno; de lo contrario, se produciría sobrecarga en el buffer (*buffer overflow*) y se perdería información. En esto consiste, precisamente, la función de control de flujo. Más formalmente, podemos afirmar que el **control de flujo** tiene por objeto evitar que la entidad que transmite sobrecargue el buffer de la entidad que recibe. Estas entidades pueden ser módulos de transporte (control de flujo en la capa de transporte) o módulos de enlace (control de flujo en la capa de enlace). El control de flujo descrito en el ejemplo anterior acerca de los dos dispositivos interconectados, uno a 1 Gbps y el otro a 10 Mbps, tiene lugar específicamente entre los módulos de transporte de ambos dispositivos, y por tanto se ejerce a nivel de segmentos; es decir, en este escenario la función de control de flujo tiene por objeto evitar que el buffer gestionado por el módulo de transporte destinatario se sobrecargue de segmentos. En cambio, el control de flujo en la capa de enlace tiene lugar entre los módulos de enlace de dos dispositivos conectados directamente mediante un enlace físico. Concretamente, el módulo de enlace receptor monitoriza su buffer de tramas entrantes, deteniendo el envío de las mismas cuando dicho buffer alcanza su ocupación máxima. La **Figura 8.1** ilustra las diferencias básicas entre las funciones de control de flujo de ambas capas.

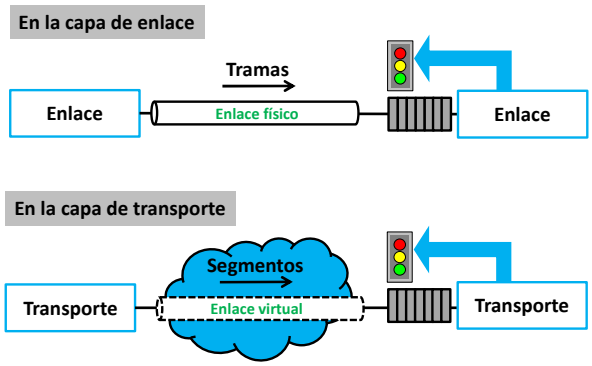


Figura 8.1. La función de control de flujo en las capas de enlace y transporte. Dado que las comunicaciones son generalmente bidireccionales, dicha función se ejecuta en ambos extremos del enlace, físico o virtual.

Cualquiera que sea la capa en la que se ejecute la función de control de flujo, es siempre la entidad que recibe la que regula el tráfico de la entidad que transmite, bajo un formato de interacción 1 a 1, es decir, de una entidad sobre otra, y a diferencia del control de congestión que, según veremos más adelante, obedece a un formato 1 a N.

Este capítulo se ocupa del control de flujo y el control de errores en la capa de enlace, relegando las especificidades de estos mismos mecanismos en la capa de transporte al estudio del protocolo TCP. Por tanto, el modelo de referencia para analizar la función de control de flujo es el mostrado en la **Figura 8.2**, donde se pone de manifiesto el papel de cada entidad, una como transmisora (T) y la otra como receptora (R), así como el intercambio de tramas de datos y tramas de control, estas últimas para regular precisamente el flujo de tramas de datos generado por la transmisora. También se introducen las variables d y Q, respectivamente, la distancia que separa ambas entidades y la capacidad del buffer de la entidad que recibe, que se entiende expresada en número de tramas.

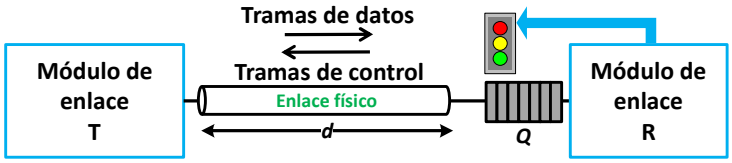


Figura 8.2. Esquema para el análisis de la función de control de flujo en la capa de enlace. Este esquema opera realmente en los dos sentidos, pues las comunicaciones son generalmente bidireccionales y cada entidad puede actuar como transmisora y como receptora, de forma alternativa o simultánea.

Control de flujo de parada y espera

El mecanismo más sencillo de control de flujo es el de parada y espera (*stop-and-wait*), cuyo funcionamiento se describe mediante el diagrama de tiempos detallado de la **Figura 8.3**. Equivale a disponer de un buffer con capacidad para una sola trama ($Q = 1$), de manera que el transmisor no puede enviar una nueva trama de datos hasta que no recibe confirmación por parte del receptor de que éste ya ha descargado del buffer la trama de datos anterior, y por tanto dicho buffer vuelve a estar disponible. La confirmación o **acuse de recibo** se envía mediante una trama de control especial, la trama **ACK** (*ACKnowledgment*). La figura también pone de manifiesto los siguientes intervalos de tiempo relevantes:

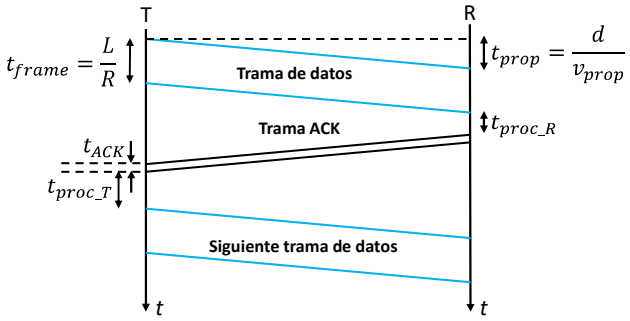


Figura 8.3. Diagrama de tiempos detallado del mecanismo de control de flujo de parada y espera. El diagrama se basa en trazar un eje vertical de tiempos para cada extremo de la comunicación, donde el tiempo evoluciona hacia abajo. Otros autores utilizan la versión horizontal de dicho diagrama.

- Tiempo de transmisión de las tramas de datos: $\boldsymbol{t_{frame}}$. Es la duración de las tramas de datos (supuesto que todas duran igual), es decir, el tiempo que tarda el transmisor en "colocar" la secuencia de bits de la trama en el enlace, o el tiempo que tarda el receptor en "extraer" esa secuencia de bits del enlace. Si R es la velocidad de transmisión a través del enlace, y L es el número total de bits de una trama de datos, su duración viene dada por la siguiente expresión:

$$t_{frame} = \frac{L}{R} \tag{8.1}$$

- Tiempo de propagación a través del enlace: $\boldsymbol{t_{prop}}$. Es el tiempo que tarda cualquier bit en viajar de un extremo al otro del enlace, sea en un sentido o en el otro. Siendo d la distancia que separa el receptor del transmisor, y v_{prop} la velocidad de propagación de la señal a través del medio físico que constituye el enlace (medio metálico, fibra óptica, inalámbrico, etc.), el tiempo de propagación obedece a la fórmula siguiente:

$$t_{prop} = \frac{d}{v_{prop}} \tag{8.2}$$

- Tiempos de procesado en transmisión y recepción: $\boldsymbol{t_{proc\_T}}$ y $\boldsymbol{t_{proc\_R}}$, respectivamente. El tiempo de procesado en recepción es el que transcurre entre que el receptor termina de recibir el último bit de una trama de datos y empieza a transmitir el primer bit de una trama de confirmación. Por su parte, el tiempo de procesado en transmisión es el tiempo que transcurre entre que el transmisor termina de recibir el último bit de una trama de confirmación y empieza a transmitir el primer bit de una nueva trama de datos.
- Duración de las tramas ACK: $\boldsymbol{t_{ACK}}$. Es equivalente al tiempo t_{frame}, pero aplicado a las tramas ACK.

Observemos que, puesto que todos los bits experimentan el mismo tiempo de propagación, el envío de cualquier trama queda reflejado en el diagrama de tiempos como una franja inclinada de grosor fijo, delimitada por las líneas que muestran la propagación del primer y último bit de esa trama. El grosor de la misma viene determinado por el tiempo de transmisión, mientras que su desnivel depende del tiempo de propagación. Como prueban las expresiones (8.1) y (8.2), los tiempos de transmisión y propagación dependen de factores distintos (tamaño de las tramas y velocidad de transmisión, por un lado, y distancia y velocidad de propagación por otro), y además esos factores son totalmente independientes unos de otros, por lo que todas las combinaciones de grosor y desnivel son posibles (grosor grande y desnivel alto o bajo, y grosor pequeño y desnivel alto o bajo). Esto explica también que las tramas de datos y las tramas ACK presenten grosores muy distintos en el diagrama, pues el tamaño (número de bits) de las tramas de datos suele ser muy superior al de las tramas ACK y las tramas de control en general, pero sin embargo muestren el mismo desnivel, y es que ambas experimentan el mismo tiempo de propagación. Téngase en cuenta que, si bien el tiempo de transmisión es acumulativo, es decir, el tiempo de transmisión de una trama es la suma de los tiempos de transmisión de cada uno de los bits que contiene, el tiempo de propagación no lo es, lo que significa que el hecho de propagar más o menos bits a través del enlace no influye en dicho tiempo.

Eficiencia del mecanismo de control de flujo de parada y espera

Un parámetro de comportamiento fundamental en la caracterización de los mecanismos de control de flujo y de control de errores de la capa de enlace es la **eficiencia**, definida como la relación entre el tiempo que idealmente se invertiría en la transmisión de una trama y el tiempo realmente invertido (cuando se trata de la capa de transporte, el concepto de eficiencia se transforma en la velocidad efectiva o *throughput* del protocolo TCP, pues éste es el único protocolo de transporte que implementa tales mecanismos). Por tanto, la eficiencia mide el grado de aprovechamiento de un enlace, y como casi todas las eficiencias, tiene un valor entre 0 y 1. Los cálculos de eficiencia se realizan en condiciones de **tráfico de saturación**, es decir, suponiendo que la entidad que transmite siempre tiene tramas por enviar. De esta manera, se puede evaluar lo máximo que puede dar de sí el mecanismo que se está analizando, sin que esta evaluación se vea penalizada por el hecho de que la entidad transmisora genere poco tráfico. En esa misma línea, cuando el análisis se centra en algún mecanismo de control de flujo, se supone que no se producen errores de transmisión en las tramas, pues esto forma parte del ámbito del control de errores, cuyos mecanismos siempre degradan la eficiencia de los mecanismos de control de flujo de los cuales derivan (recuérdese que los mecanismos de control de errores son extensiones de los mecanismos de control de flujo). Finalmente, una hipótesis habitual y realista, cuyo objetivo es centrar la atención en las variables más relevantes en los cálculos de eficiencia, consiste en admitir que, dado su reducido valor, los tiempos de procesado y la duración de los acuses de recibo son despreciables.

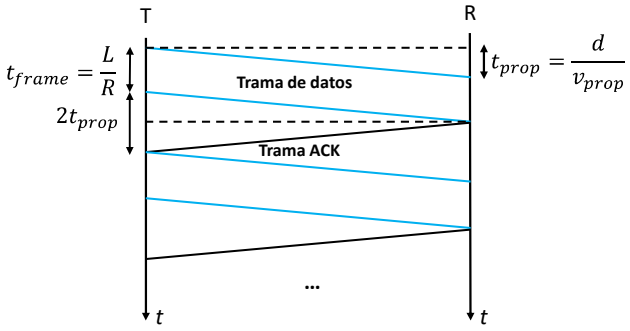


Figura 8.4. Diagrama de tiempos para el análisis de eficiencia del mecanismo de control de flujo de parada y espera.

Bajo las hipótesis anteriores, el cálculo de eficiencia del mecanismo de control de flujo de parada y espera toma como referencia el diagrama mostrado en la **Figura 8.4**. Obsérvese que, efectivamente se ha despreciado la duración de las tramas ACK (se dibujan como una sola línea o franja de grosor nulo), ¡pero no su propagación! Según dicha figura y la definición de eficiencia, podemos formular la eficiencia del mecanismo de control de flujo de parada y espera ($\eta_{S\&W}$) de la siguiente manera:

$$\eta_{S\&W} = \frac{t_{frame}}{t_{frame}+2t_{prop}} \tag{8.3}$$

Dividiendo numerador y denominador de la expresión (8.3) por t_{frame}, la eficiencia queda reformulada como sigue:

$$\eta_{S\&W} = \frac{1}{1+2\frac{t_{prop}}{t_{frame}}} \tag{8.4}$$

Esta expresión pone de manifiesto que lo significativo no son los valores del tiempo de propagación y el tiempo de transmisión por separado, sino la relación entre ellos. Dicha relación aparece en el denominador de casi todas las fórmulas de eficiencia de los mecanismos de control de flujo y errores, por lo que su relevancia es máxima. Es el llamado **parámetro a** del enlace, es decir:

$$a = \frac{t_{prop}}{t_{frame}} = \frac{d \cdot R}{L \cdot v_{prop}} \tag{8.5}$$

Naturalmente este parámetro es adimensional, y su valor es no negativo. Recoge los parámetros tecnológicos básicos del enlace (distancia, velocidad de transmisión y velocidad de propagación), así como el tamaño de las tramas, fijado por el protocolo de enlace que se utilice. En función de dicho parámetro, la eficiencia del mecanismo de control de flujo de parada y espera queda de la siguiente manera:

$$\eta_{S\&W} = \frac{1}{1+2a} \tag{8.6}$$

Esta eficiencia sólo arroja resultados aceptables para valores muy bajos del parámetro a, por lo que la aplicabilidad del mecanismo de control de flujo de parada y espera es en la práctica limitada.

La expresión (8.6) y la mayoría de las expresiones que vamos a obtener para la eficiencia de los diversos mecanismos de control de flujo y errores, merecen dos observaciones importantes:

- Su relación inversa con la distancia, es decir, la eficiencia decrece con la longitud del enlace.
- Su relación también inversa con la velocidad de transmisión, lo cual puede parecer un contrasentido, pues una mayor velocidad de transmisión resulta siempre deseable. No obstante, la interpretación correcta es que la calidad del enlace no se aprovecha lo suficiente, por lo que, a velocidades de transmisión altas, conviene utilizar mecanismos que conduzcan a expresiones matemáticas de eficiencia que, a su vez, den mejores resultados. Es decir, no se trata de reducir la velocidad de transmisión para lograr una mayor eficiencia, sino de mejorar el mecanismo.

El producto de la velocidad de transmisión nominal de un enlace por su eficiencia es la llamada velocidad efectiva de transmisión o ***throughput***. Así, en el caso del mecanismo de control de flujo de parada y espera, el *throughput* ($th_{S\&W}$) viene dado por la siguiente expresión:

$$th_{S\&W} = R \cdot \eta_{S\&W} \tag{8.7}$$

Por ejemplo, si la velocidad de transmisión nominal es de 100 Mbps y la eficiencia es del 20%, la velocidad efectiva es tan solo de 20 Mbps.

El producto ancho de banda-retardo

Existe una cierta similitud entre un enlace de comunicaciones que transporta bits y una tubería que conduce agua (o cualquier otro líquido), como la dibujada en la **Figura 8.5**. Cuanta mayor sección tiene la tubería, mayor cantidad de agua puede conducir por unidad de tiempo (mayor caudal), lo que equivale a una velocidad de transmisión más alta. Por tanto, sección transversal de la tubería y velocidad de transmisión o ancho de banda nominal del enlace son equiparables. Por otro lado, la longitud de la tubería es claramente equiparable a la longitud del enlace enlace. Puesto que el tiempo de transmisión depende de la velocidad de transmisión (sección de la tubería) y el tiempo de propagación de la distancia recorrida (longitud de la tubería), la **Figura 8.5** no hace más que confirmar la diferencia e independencia que hay entre tiempo de transmisión y tiempo de propagación, pues una tubería puede ser ancha y corta, ancha y larga, estrecha y corta o estrecha y larga.

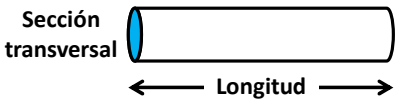


Figura 8.5. Un enlace de comunicaciones es similar a una tubería.

Por otro lado, el parámetro a se puede re-expresar de la siguiente forma, sin substituir el tiempo de propagación:

$$a = \frac{R \cdot t_{prop}}{L} \tag{8.8}$$

El numerador de esta expresión es el llamado **producto ancho de banda-retardo**; no es más que el número de bits transmitidos durante el tiempo de propagación, es decir, durante el tiempo que tarda el primero de esos bits en alcanzar el otro extremo del enlace, lo que significa que ese producto es, en definitiva, el número de bits que caben en el enlace. Así pues, el producto ancho de banda-retardo es equiparable al volumen de la tubería (a mayor volumen, más cantidad de agua cabe en ella). Finalmente, también observamos que el parámetro a resulta ser el cociente entre el número de bits que caben en el enlace y el número de bits de las tramas, pudiendo ser este resultado mayor o menor que la unidad.

Control de flujo basado en ventana deslizante

El objetivo del mecanismo de control de flujo basado en ventana deslizante (*sliding window*) es aumentar la eficiencia con respecto al mecanismo de parada y espera, permitiendo el envío de más tramas sin recibir confirmación. El máximo número de tramas que el transmisor puede enviar sin recibir confirmación es el llamado **tamaño máximo de ventana** (N). Lógicamente, el *buffer* del receptor tiene que poder albergar ese número de tramas, es decir, $Q \geq N$.

El máximo provecho del mecanismo de control de flujo basado en ventana se obtiene sobre enlaces *full-duplex* y utilizando tramas ACK que puedan confirmar varias tramas de datos a la vez, es decir,

tramas ACK con efecto acumulativo. El funcionamiento se ilustra en la **Figura 8.6**, para un tamaño máximo de ventana de valor 7 (en el dibujo se ha omitido el grosor de las tramas para mayor simplicidad). En dicha figura, el transmisor va enviando tramas de datos al tiempo que va recibiendo tramas de confirmación, de manera que, cada vez que recibe una de éstas últimas, su ventana de transmisión vuelve al valor máximo. Solamente cuando envía 7 tramas posteriores a la última confirmada, sin recibir nueva confirmación, el transmisor se queda bloqueado.

Comparado con el mecanismo de parada y espera, el mecanismo basado en ventana deslizante es más dinámico, flexible y eficiente, pero su correcto funcionamiento exige enumerar las tramas de datos. Esto significa que el formato de tales tramas debe incluir un campo de control que permita su enumeración, al tiempo que las tramas ACK deberán incluir ese mismo campo para indicar la siguiente trama de datos que el receptor espera procesar (convenio habitualmente adoptado para las tramas de confirmación). Concretamente, si k es el tamaño en bits del campo de enumeración, el número de trama puede variar desde 0 hasta $2^k - 1$ (2^k números de secuencia distintos). Se trata, pues, de una enumeración módulo 2^k, en que la trama 0 sigue a la trama $2^k - 1$. Aparentemente, con el objeto de evitar ambigüedades, el tamaño máximo de ventana debería satisfacer la condición $N \leq 2^k$, no obstante, la condición correcta para evitar toda posible ambigüedad es $N \leq 2^k - 1$. Puesto que el tamaño máximo de ventana también debía satisfacer la condición $N \leq Q$, la condición final para dicho tamaño es la siguiente:

$$N \leq \min(2^k - 1, Q) \qquad (8.9)$$

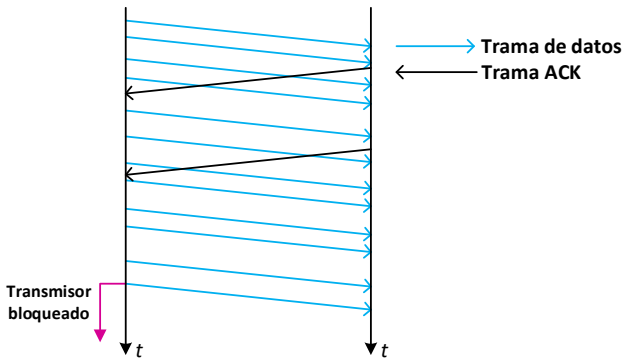


Figura 8.6. Mecanismo de control de flujo basado en ventana deslizante, sobre un enlace *full-duplex* y para un tamaño máximo de ventana de valor 7.

Normalmente el tamaño del *buffer* del receptor es muy grande, de manera que la condición anterior se transforma en la siguiente:

$$N \leq 2^k - 1 \qquad (8.10)$$

Como regla general, el tamaño máximo de ventana se elige de forma que se satisfaga la inecuación anterior en condiciones de igualdad. La **Figura 8.7** ilustra el funcionamiento de los números de

secuencia en ambos tipos de tramas para el caso $k = 3$ (al que corresponde, precisamente, $N = 7$). En esta figura, con el objeto de facilitar la comprensión del intercambio de números de secuencia, se ha supuesto que el enlace es *half-duplex*.

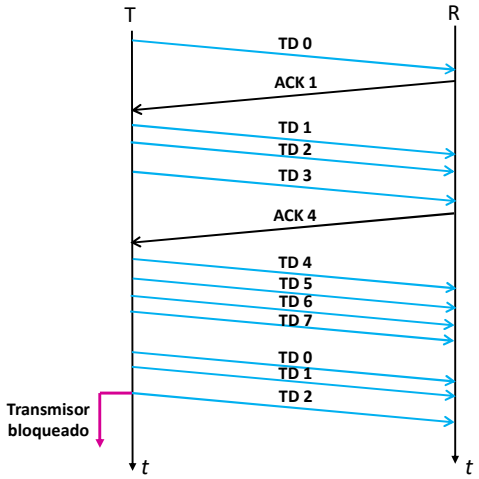


Figura 8.7. Enumeración de las tramas de datos (TD) y de confirmación (ACK) en el mecanismo de control de flujo basado en ventana deslizante, para un tamaño máximo de ventana de valor 7.

La denominación del mecanismo obedece a que tanto transmisor como receptor manejan sendas ventanas o grupos de tramas de datos. En el caso del transmisor, su ventana contiene las tramas que puede enviar sin esperar confirmación adicional; en el caso del receptor, su ventana contiene las tramas que espera recibir. Estas ventanas evolucionan en el tiempo de forma más o menos sincronizada, pero no tienen por qué coincidir instantáneamente. La **Figura 8.8** muestra la evolución de las ventanas de transmisión y recepción para el flujo de tramas de la **Figura 8.7**. Como podemos observar, ambas ventanas se deslizan más o menos en paralelo siguiendo un movimiento de gusano, acortándose por detrás y alargándose por delante. Concretamente, la ventana de transmisión se acorta cada vez que el transmisor envía una trama de datos, y se alarga cada vez que recibe una trama de confirmación; por su parte, la ventana de recepción se acorta cada vez que el receptor recibe una trama de datos y se alarga cada vez que envía una trama de confirmación. Así, ambas ventanas tienen un tamaño que va cambiando, sin superar el valor máximo de 7 (N, en general).

Eficiencia del mecanismo de control de flujo basado en ventana deslizante

Para analizar la eficiencia del mecanismo de control de flujo basado en ventana deslizante, nos apoyaremos en las mismas hipótesis que en el caso del mecanismo de parada y espera, es decir:

- Tráfico de saturación. Tratándose de un mecanismo de ventana, esta condición también implica que cuando el transmisor puede enviar varias tramas, lo hace una tras otra sin intervalos de tiempo entre ellas (transmisión *back-to-back*).
- Ausencia de errores de transmisión.
- Tiempos de procesado y duración de los acuses de recibo despreciables.

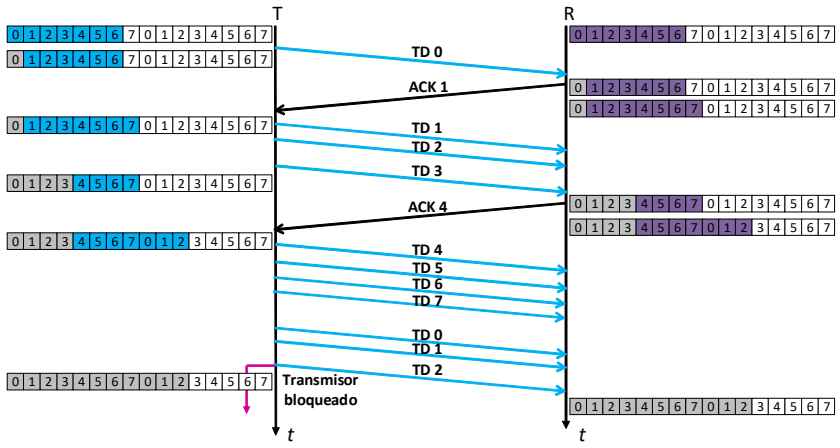


Figura 8.8. Evolución de las ventanas de transmisión y recepción en el mecanismo de control de flujo por ventana deslizante (caso $k = 3, N = 7$). Cada ventana se muestra lo más a continuación posible del acontecimiento que la provoca.

Además, con el fin de asegurar la obtención de la máxima eficiencia que puede dar de sí el mecanismo, añadiremos una nueva hipótesis, consistente en que el receptor devuelve una trama ACK por cada trama de datos recibida. Esto implica, inevitablemente, asumir que el enlace es *full-duplex*. Todas estas hipótesis conducen al diagrama de la **Figura 8.9** como referencia para el análisis de eficiencia.

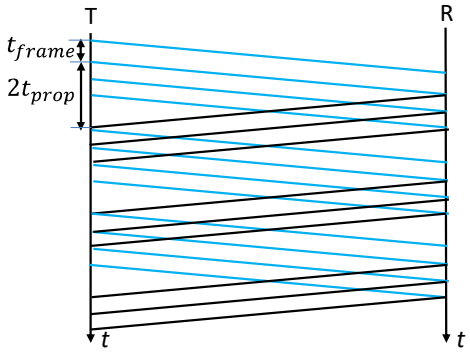


Figura 8.9. Diagrama temporal para el análisis de eficiencia del mecanismo de control de flujo basado en ventana deslizante.

La **Figura 8.9** muestra el caso en que el transmisor recibe el primer acuse de recibo cuando ya ha agotado su máxima ventana de transmisión, y por tanto se queda bloqueado un cierto intervalo de tiempo. Matemáticamente, esto ocurre cuando $N \cdot t_{frame} < t_{frame} + 2 \cdot t_{prop}$. En tal caso, la eficiencia resultante (η_{SW}) es precisamente el cociente entre ambos términos:

$$\eta_{SW} = \frac{N \cdot t_{frame}}{t_{frame} + 2 \cdot t_{prop}} \qquad (8.11)$$

Por el contrario, si el transmisor no hubiera agotado su máxima ventana a la llegada del primer acuse de recibo, no se hubiera quedado bloqueado, podría enviar tramas de forma continuada, y la eficiencia alcanzada sería del 100%. Esto ocurre cuando $N \cdot t_{frame} \geq 2 \cdot t_{frame} + t_{prop}$. Los dos casos expuestos se pueden resumir en la siguiente expresión:

$$\eta_{SW} = \begin{cases} \frac{N \cdot t_{frame}}{t_{frame} + 2 \cdot t_{prop}}, N \cdot t_{frame} < t_{frame} + 2 \cdot t_{prop} \\ 1, N \cdot t_{frame} \geq t_{frame} + 2 \cdot t_{prop} \end{cases} \quad (8.12)$$

En términos del parámetro a, la expresión final queda así:

$$\eta_{SW} = \begin{cases} \frac{N}{2a+1}, N < 2a + 1 \\ 1, N \geq 2a + 1 \end{cases} \quad (8.13)$$

Obsérvese que el mecanismo de control de flujo de parada y espera no es más que el caso particular para $N = 1$ del mecanismo basado en ventana deslizante.

8.2. CONTROL DE ERRORES

Ya hemos visto que, debido principalmente al ruido y las interferencias, la capa física de las comunicaciones no es totalmente fiable, por lo que cualquier bit puede resultar alterado en su camino de origen a destinatario. El **control de errores** no es más que el conjunto de técnicas desarrolladas para gestionar esos errores de transmisión. Tanto en la capa de enlace como en la capa de transporte (capas más habituales en las que se implementa este control), esta gestión puede consistir en que la entidad destinataria llegue a disponer de una versión correcta de la unidad de información enviada, o que dicha entidad simplemente descarte la unidad de información recibida cuando detecta que es errónea. Las técnicas utilizadas en control de errores se agrupan en dos grandes conjuntos:

- Esquemas de codificación. Cuando una entidad envía una secuencia de bits a otra, la alteración de uno o varios de esos bits no hace más que transformar dicha secuencia en otra perfectamente válida, pues en principio todas las combinaciones binarias son potencialmente legítimas. En estas condiciones no es posible, por tanto, detectar errores. Dicho de otra manera, los bits enviados no son verificables como correctos o incorrectos por sí solos. La detección (y posible corrección) de errores sólo puede tener lugar transformando las secuencias de bits a proteger en secuencias más largas, de modo que las alteraciones experimentadas por estas últimas conduzcan a combinaciones no válidas. Estas secuencias contienen, pues, más bits de los necesarios para transportar la misma información que las secuencias originales, es decir, contienen un cierto grado de **redundancia**. Como la misma palabra indica, la redundancia no aporta información, pero es necesaria para posibilitar la detección de errores. Las reglas que

establecen cómo se añade la redundancia quedan definidas por un algoritmo, esquema de codificación o, simplemente, un **código**. Hay códigos que solamente tienen capacidad detectora, es decir, permiten determinar la presencia de errores en la secuencia de bits recibidos, pero no el número y, mucho menos, la ubicación de esos errores. Son los llamados **códigos detectores**. Alternativamente, los **códigos correctores** tienen capacidad correctora, es decir, además de detectar la presencia de errores, son capaces de determinar su número y posición dentro de la secuencia recibida. No obstante, el nivel de redundancia exigido por los códigos correctores es superior al de los códigos detectores, por lo que la prevalencia de una u otra opción dependerá de cada caso. Códigos detectores y correctores, cuando se consideran en la capa de enlace, constituyen la llamada **codificación de canal**.

- Mecanismos de retransmisión. Las secuencias de bits enviadas de una entidad a otra no son más que las unidades de información características de la capa en que se encuentran dichas entidades. Dicho de otro modo, estas secuencias son las tramas en el caso de la capa de enlace y los segmentos en el caso de la capa de transporte. La redundancia introducida por un código de control de errores sirve para proteger toda la unidad de información o una parte de la misma. Tanto los códigos detectores como los correctores tienen capacidades limitadas, por lo que pueden darse situaciones en que los errores de transmisión producidos excedan estas capacidades. En particular, cuando el receptor detecta que la unidad de información recibida es errónea, pero no puede corregirla, bien porque el código utilizado carece de capacidad correctora, bien porque los errores exceden su capacidad correctora, entonces caben dos opciones: descartar la unidad de información o solicitar su retransmisión. Esta última opción es la única posible cuando la aplicación exige que la entidad destinataria disponga de la versión correcta de cada unidad de información enviada. Los mecanismos de retransmisión o mecanismos ARQ están diseñados como extensiones de los mecanismos de control de flujo. Tales extensiones consisten esencialmente en introducir acuses de recibo o confirmaciones de carácter negativo, es decir, del tipo **NAK** (*Negative AcKnowledgment*), junto a los acuses o confirmaciones del tipo ACK. Así, el mecanismo de control de flujo de parada y espera (*stop-and-wait*) se transforma en el mecanismo de control de errores *stop-and-wait ARQ*, mientras que el mecanismo de control de flujo basado en ventana deslizante da lugar a dos mecanismos de control de errores, volver-atrás-N (*go-back-N*) y rechazo selectivo (*selective reject*).

Codificación y retransmisión son aspectos distintos del control de errores, pero complementarios; como veremos más adelante, el tipo de aplicación y las características del canal son determinantes a la hora de seleccionar la mejor combinación en cuanto a tipo de código y opción de retransmisión (incluyendo aquí la posibilidad de no retransmisión, es decir, el descarte). La complementariedad también se da entre las dos capas implicadas, es decir, enlace y transporte, de forma que el control de errores final es fruto de la acción combinada de los controles de errores implementados en ambas capas. Analizamos primero los esquemas de codificación y, a continuación, los mecanismos de retransmisión. En ambos casos centraremos el análisis en la capa de enlace, si bien los conceptos básicos son totalmente extrapolables a la capa de transporte.

Códigos de control de errores

La calidad de un código viene determinada por un conjunto de factores, tanto cualitativos como cuantitativos. Son los siguientes:

- Tipo de código, detector o corrector. En principio, un código con capacidad correctora es preferible a un código con capacidad sólo detectora.
- Robustez. Se refiere a la capacidad detectora en el caso de los códigos detectores, y a las capacidades detectora y correctora en el caso de los códigos correctores. Tanto la capacidad detectora como la capacidad correctora se suelen cuantificar para dos tipos de errores: errores aislados y ráfagas de errores. De hecho, un código puede ser robusto frente a errores aislados, pero débil frente a errores a ráfagas. Errores aislados y errores a ráfagas son realmente dos estadísticas de errores. Así, hablamos de **errores aislados** (*single-bit errors*) cuando no hay correlación de errores entre bits vecinos, es decir, el hecho de que el valor de un determinado bit haya resultado alterado no aporta ninguna información sobre lo que haya ocurrido con sus bits vecinos. Con ruido térmico y velocidades de transmisión no demasiado elevadas, la hipótesis de errores aislados es aceptable. No obstante, el caso más típico en comunicaciones es el de **errores a ráfagas** (*burst errors*), lo que significa que no hay independencia de errores a nivel de bit. Esto ocurre, por ejemplo, con el ruido impulsivo o con el desvanecimiento (*fading*) de la señal en transmisiones inalámbricas, o simplemente cuando la velocidad de transmisión es muy alta, incluso aunque la única fuente perturbadora sea el ruido térmico. En todos estos casos, cuando un bit es erróneo, es probable que también lo sean sus vecinos. La cuantificación de la robustez de un código frente a las ráfagas de errores exige una definición puntualizada de las mismas. El IEEE define una **ráfaga de errores** como un grupo de bits en el que dos bits erróneos cualesquiera están separados por menos de un número c de bits correctos, de manera que entre el último bit erróneo de una ráfaga y el primer bit erróneo de la siguiente ráfaga hay como mínimo c bits correctos (en la definición no se concreta ningún valor para c). La **longitud de una ráfaga** es el número total de bits entre el primer y el último bit erróneo de la misma, ambos inclusive.
- Grado de redundancia. Una forma de cuantificar la redundancia introducida por un código es mediante la llamada **tasa de codificación**, que no es más que la proporción de bits de información con respecto al total de bits enviados (por bits de información se entiende aquí los bits a proteger, tanto si contienen información de datos como si contienen información de control). Por ello, la tasa de codificación determina la fracción del ancho de banda disponible para los bits de información. Por ejemplo, si la tasa de codificación utilizada en un enlace de 150 Kbps (ancho de banda nominal) es del 80%, entonces el ancho de banda disponible para los bits de información en ese enlace es de $150 \cdot 0.8 = 120$ Kbps, es decir, se ha reducido un 20%. Por tanto, desde el punto de vista del consumo de ancho de banda por parte de los bits de información, interesa que la tasa de codificación sea lo mayor posible, es decir, cuanta menor redundancia mejor, en contraposición a lo que exige la robustez (de ahí que en la práctica haya que encontrar una solución de compromiso entre ambos objetivos). Otra forma de evaluar el

grado de redundancia es mediante la proporción de bits redundantes con respecto a bits de información, medida conocida como **redundancia** (propiamente dicha).

- Ganancia de codificación. Este concepto es específico de la corrección de errores en la capa de enlace, pues nos lleva a las curvas de probabilidad de error de bit (BER) en función de la relación E_b/N_0 para los diversos esquemas de codificación de línea o modulación digital. Concretamente, la **ganancia de codificación** mide la reducción en el valor de E_b/N_0 necesario para alcanzar una BER dada con un código de corrección de errores determinado, en comparación con el valor de E_b/N_0 necesario para obtener la misma BER en un sistema no codificado que utilice la misma codificación de línea o el mismo esquema de modulación. Dicha reducción se expresa en dB.
- Complejidad. La complejidad del algoritmo de codificación es un factor importante, no sólo porque afecta directamente a la complejidad de implementación (hardware o software) de los procesos de codificación y decodificación, sino porque también afecta al retardo introducido por tales procesos. Este retardo es crítico en el caso de las aplicaciones interactivas, ya que éstas se basan en frecuentes intercambios de mensajes en tiempo real entre origen y destinatario.

Aunque desde el punto de vista de las comunicaciones y las redes de computadores, la característica más llamativa de un código es su capacidad detectora o correctora, la taxonomía de códigos de control de errores obedece a otros criterios, de cuya base matemática sólo vislumbraremos la superficie. Dada la extensión de dicha taxonomía, en lugar de representarla a través de una tabla, se ha optado por el mapa conceptual de la **Figura 8.10**. Como ya ocurría con la codificación de línea, los códigos de control de errores pueden ser binarios o no binarios, según estén construidos sobre un alfabeto de 2 o más símbolos, respectivamente. La mayor parte de códigos señalados en la **Figura 8.10** admiten una formulación binaria y no binaria, siendo la primera un caso particular de la segunda (códigos Reed-Solomon, por ejemplo) o la segunda una generalización de la primera (códigos Hamming, por ejemplo) (todo depende de la formulación que se introdujera en primer lugar).

Las dos tipologías básicas de códigos de control de errores son los códigos bloque y los códigos convolucionales, pues los códigos concatenados, multidimensionales y espacio-temporales son realmente combinaciones y/o extensiones de los dos primeros. Estos códigos compuestos pueden retener o no algunas de las propiedades de los códigos básicos constituyentes.

Dentro de la categoría de códigos bloque están los códigos más comúnmente utilizados en redes de computadores, es decir, los códigos CRC en el caso de los protocolos de enlace y los *checksums*, particularmente los basados en la suma complemento a 1, en el caso de los protocolos de capas más altas (IP, TCP y UDP). Pese a su popularidad, estos códigos sólo tienen capacidad detectora; cuando se requiere capacidad correctora, hay muchas otras opciones, pero actualmente las alternativas que ofrecen más alto rendimiento son los códigos Turbo y, sobretodo, los códigos LDPC (*Low Density Parity Check*). Unos y otros permiten alcanzar velocidades de transmisión muy cercanas al límite teórico de Shannon – fórmula de capacidad (5.13).

Para terminar esta introducción a los códigos de control de errores, la **Tabla 8.1** indica la capacidad, detectora o correctora, así como los usos principales, de la mayor parte de códigos indicados en la **Figura 8.10**. El primero de los códigos que aparecen en dicha tabla es el más sencillo de todos, pues se basa en utilizar un solo bit de paridad. Es el código SPC (*Single-Parity Check*).

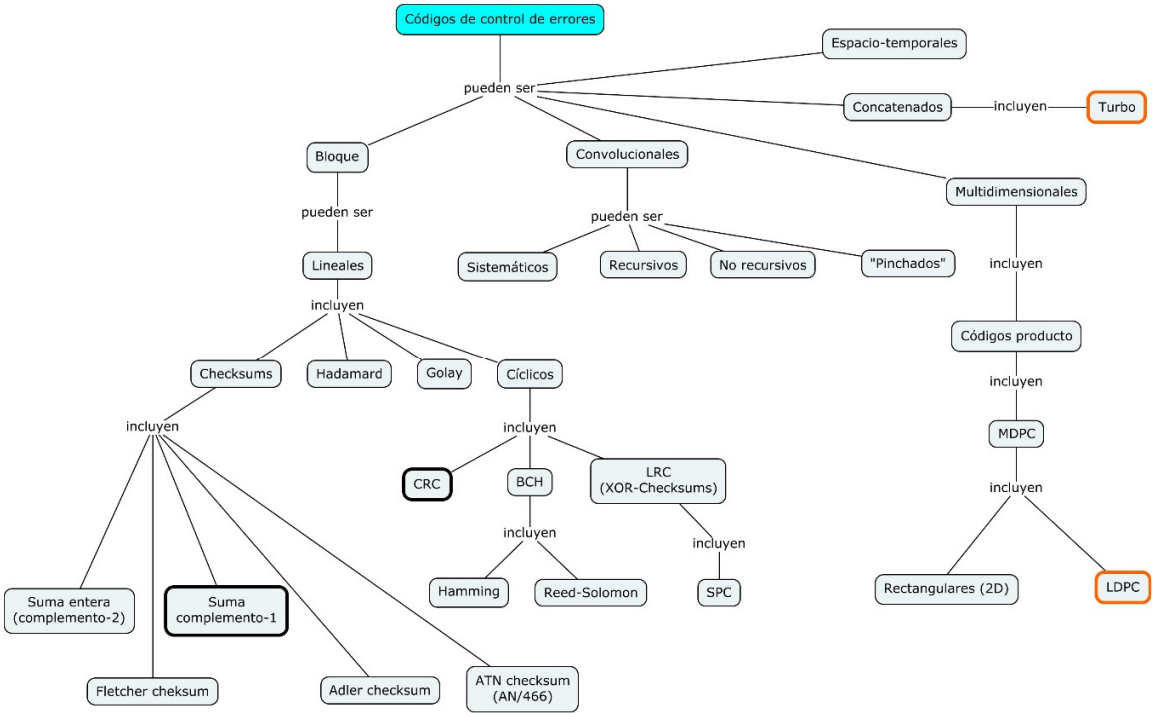


Figura 8.10. Mapa conceptual de los códigos de control de errores. Se señalan los códigos más comúnmente utilizados en redes de computadores (CRC y *checksums* basados en suma complemento a 1), y los más avanzados (Turbo y LDPC).

Tabla 8.1. Capacidad detectora o correctora y usos principales de los códigos de control de errores.

Códigos	Capacidad	Usos principales
SPC	Detectora	Comunicación serie asíncrona de caracteres ASCII (y otros formatos), hardware de computación (buses, memorias, interfaces, etc.)
MDPC	Correctora	Dispositivos de almacenamiento (2DPC), buses (2DPC), CubeSats
Checksums	Detectora	IP, TCP, UDP
Hadamard	Correctora	Como *spread sequences* en comunicaciones basadas en CDMA
Golay	Correctora	*NASA Deep Space Network*
CRC	Detectora	Ethernet (IEEE 802.3), WiFi (IEEE 802.11), ATM, Frame Relay, HDLC, PPP
BCH	Correctora	Comunicaciones por satélite, dispositivos de almacenamiento
LDPC	Correctora	IEEE 802.11n, IEEE 802.11ac, TVSAT digital, 10G Ethernet
Turbo	Correctora	Telefonía móvil 3G/4G, *NASA Deep Space Network*

A continuación, se constatan los aspectos principales de los códigos bloque y se describen los códigos CRC y los *checksums* basados en aritmética complemento a 1.

Códigos bloque

Un **código bloque** asigna a cada combinación única de k bits de información o **mensaje**, una combinación de n bits, con $n > k$, de acuerdo con un cierto algoritmo. Hablamos entonces de un **código bloque (n, k)**. A cada bloque de n bits asociado a un bloque de k bits se le denomina **palabra código**. Por tanto, un código bloque (n, k) contiene un total de 2^k palabras código, que constituyen un subconjunto de las 2^n posibles combinaciones con n bits. El resto, $2^n - 2^k$, son palabras no utilizadas. La estrategia fundamental de un código bloque consiste precisamente en seleccionar las palabras código de manera que cada una de ellas sea lo más distinta posible de las restantes, a fin de que, si se alteran uno o varios bits durante la transmisión, la probabilidad de recibir una palabra código inválida sea mayor. Esto nos lleva al concepto de **distancia de Hamming $d(x, y)$** entre dos palabras x e y del mismo tamaño, como el número de bits en los cuales ambas palabras difieren. Por ejemplo, si se transmite la palabra código 01001101 y se recibe la palabra 01011001, el número de bits erróneos es 2 y la distancia de Hamming entre ambas palabras es precisamente $d(01001101,01011001) = 2$. Es decir, la distancia de Hamming entre la palabra recibida y la palabra código enviada es el número de bits corrompidos durante la transmisión, de ahí la importancia de este concepto en detección de errores. La distancia de Hamming entre dos palabras se puede calcular aplicando la operación XOR entre ambas y contando el número de 1s en el resultado, es decir, su peso[19].

Se define también la **mínima distancia de Hamming** de un código (d_{min}), como la mínima de las distancias de Hamming entre todas las posibles parejas de palabras código que lo constituyen. La relación entre la mínima distancia de Hamming de un código y la distancia de Hamming entre la palabra recibida y la palabra código enviada, determina la capacidad de dicho código para reconocer la palabra recibida como errónea. Por ejemplo, si la distancia de Hamming entre la palabra recibida y la palabra código enviada es 3, es decir, se han producido 3 errores de bit, y la mínima distancia de Hamming del código utilizado es 4, entonces la palabra recibida será reconocida como inválida y, por tanto, errónea. Esto no significa, sin embargo, que con un código cuya distancia mínima de Hamming sea 4, no puedan reconocerse más de 3 errores de bit en casos especiales, pero no en todos los casos. En general, podemos enunciar que, para garantizar la detección de cualquier combinación de hasta e errores, la mínima distancia de Hamming de un código debe satisfacer la condición $d_{min} \geq e + 1$. Esta afirmación admite una interpretación geométrica sencilla, como se muestra en la **Figura 8.11**. En ella, X e Y son dos palabras código cualesquiera, y cada circunferencia representa el lugar geométrico de palabras que están a una cierta distancia de Hamming (1, 2, 3, … hasta e) de la palabra código X. Los puntos corresponden a palabras no válidas, que serán reconocidas como tales, cualesquiera que sean X e Y, si la distancia mínima de Hamming del código cumple la condición señalada, es decir, para cualquier palabra X, cualquier palabra Y se encuentra más allá de la circunferencia más externa centrada en X.

[19] El número de 1s de una palabra x se denomina **peso** de la misma, y se representa mediante $w(x)$. Por tanto, el peso de una palabra de n bits puede variar entre 0 y n.

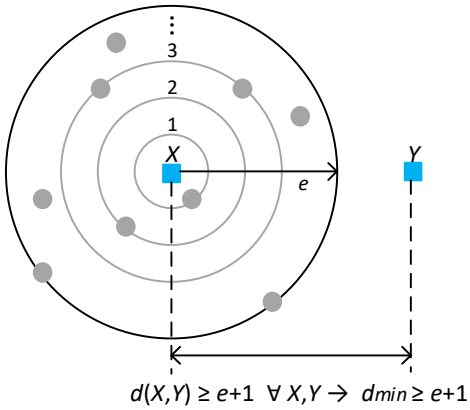


Figura 8.11. Interpretación geométrica de la distancia de Hamming en detección de errores.

En la **Figura 8.11**, el hipotético círculo de radio $e + 1$ centrado en la palabra código Y se solapa con el círculo de la palabra código X. Si bien esto no representa ningún obstáculo para la detección de errores, cuando se trata de corrección de errores, ni siquiera puede haber contacto entre los círculos. La **Figura 8.12** ilustra esta afirmación. Como puede observarse, para cualquier palabra no válida detectada dentro del círculo de X, el receptor decide que la palabra código transmitida fue precisamente X, pues es la más cercana. Si efectivamente esa fue la palabra transmitida, se habrán corregido todos los errores. Esto no podíamos afirmarlo en el caso de solapamiento entre círculos, al menos dentro de la zona de solapamiento. Lógicamente, el criterio aplicado por el receptor en corrección de errores no es infalible, pues la detección de una palabra no válida dentro del círculo de X podría corresponder a una transmisión de la palabra código Y que hubiera experimentado muchos errores. No obstante, el potencial del código reside en que, si se transmite la palabra código X y se producen hasta e errores de bit, la corrección está asegurada. En definitiva, como indica la **Figura 8.12**, podemos afirmar que, para garantizar la corrección de cualquier combinación de hasta e errores, la mínima distancia de Hamming del código debe satisfacer la condición $d_{min} \geq 2e + 1$.

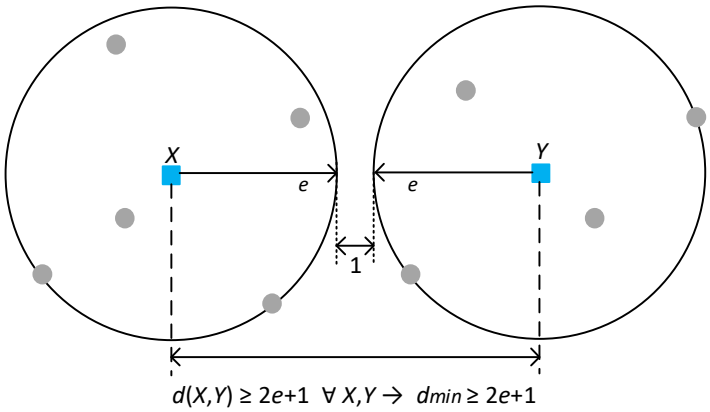


Figura 8.12. Interpretación geométrica de la distancia de Hamming en corrección de errores.

Es obvio que cuanto mayor es la mínima distancia de Hamming de un código, mayores son sus capacidades detectora y correctora, es decir, mayor es su robustez. No obstante, si bien la mínima distancia de Hamming de un código es un parámetro fundamental, no es el único, y de hecho queda confrontado con otros, como la tasa de codificación, que en un código bloque (n, k) vale k/n, o la redundancia, cuyo valor es $(n - k)/k$. La razón es que el grado de redundancia de un código es tanto mayor cuanto mayor es su mínima distancia de Hamming, por lo que siempre hay que alcanzar una solución de compromiso, que además tenga en cuenta el resto de factores que determinan la calidad de un código.

Uno de esos restantes factores es la complejidad. Además de un equilibrio entre robustez y grado de redundancia, es importante que el código elegido posea una cierta estructura que facilite su tratamiento analítico, así como la implementación de los procesos de codificación y decodificación. Por ello, prácticamente la totalidad de códigos bloque pertenecen a la clase de códigos bloque lineales sistemáticos. El tratamiento riguroso de los códigos bloque lineales y sus variantes se sustenta en una rama del álgebra abstracta conocida como *campos de Galois*, si bien nuestro enfoque será mucho menos formal. Para nuestros propósitos, un código bloque es **lineal** si incluye la palabra código todo 0s y si la suma módulo 2 (operación XOR) de dos palabras código cualesquiera es otra palabra código. Además, se puede demostrar que la mínima distancia de Hamming de un código bloque lineal es el número de 1s de la palabra código no nula con menor número de 1s, es decir, con menor peso. Por otro lado, un código bloque es **sistemático** si cada palabra código incluye explícitamente los k bits de información a partir de los cuales se ha obtenido dicha palabra. Estos k bits suelen ubicarse al principio de la palabra código, quedando reservadas las $n - k$ posiciones restantes a los bits redundantes.

Tanto los códigos CRC como los códigos *checksum* que veremos a continuación son códigos bloque lineales sistemáticos. Obsérvese en la Tabla 8.1 que ambos códigos sólo tienen capacidad detectora.

Códigos CRC

Los códigos CRC (*Cyclic Redundancy Check*) pertenecen a la categoría de códigos cíclicos, que a su vez son una clase especial de códigos bloque lineales. En un **código cíclico**, al desplazar cíclicamente (rotar) una palabra código, se obtiene otra palabra código. Por ejemplo, si 01001101 es una palabra código, también lo es 10011010, pues resulta de una rotación de 1 bit hacia la izquierda, de forma que el bit de mayor peso en la palabra original pasa a ser el de menor peso en la nueva palabra. Esta estructura cíclica conlleva también una mayor facilidad de implementación, pues estará basada en el uso de registros de desplazamiento. Un código cíclico puede ser sistemático o no. La imposición del carácter sistemático conduce, precisamente, a los códigos CRC.

La idea fundamental de un código CRC es que el transmisor, a partir de la secuencia de k bits de información (mensaje), genera $n - k$ bits redundantes (bits de comprobación), de tal manera que la palabra código resultante de n bits es divisible por algún valor binario prestablecido y conocido también por el receptor. Este último, al recibir una palabra de n bits, la divide por el valor prestablecido y si el resto es 0 decide que la transmisión ha sido correcta; en caso contrario, decide que la palabra recibida es errónea. Este funcionamiento se puede explicar de dos formas: mediante

aritmética módulo 2 (lógica XOR) sobre los valores binarios, o utilizando polinomios equivalentes. Consideraremos esta última opción, pues facilita el análisis del código y su capacidad detectora. Así, dado un valor binario cualquiera de longitud l bits $b_{l-1}b_{l-2}b_{l-3}\dots b_1b_0$, es decir, $b_i = 0{,}1\ \forall i = 0 \dots l-1$, podemos asociarle el siguiente polinomio $p(x)$, donde x es una variable muda:

$$p(x) = b_{l-1}\cdot x^{l-1} + b_{l-2}\cdot x^{l-2} + b_{l-3}\cdot x^{l-3} \dots b_1\cdot x^1 + b_0 \qquad (8.14)$$

Como podemos observar, el grado del polinomio coincide con el número de bits que contiene el valor binario (l) menos 1. También podemos observar que las potencias indican las posiciones de los bits, mientras que los coeficientes corresponden a sus valores. Por ejemplo, a la secuencia 10011101 ($l = 8$) le asociaríamos el polinomio $x^7 + x^4 + x^3 + x^2 + 1$.

Específicamente, para la codificación CRC definimos los siguientes polinomios:

- $M(x)$: Polinomio asociado al mensaje de k bits que se pretende enviar. Es un polinomio de grado $k-1$.
- $G(x)$: **Polinomio generador**. Es el polinomio asociado a un patrón de bits prestablecido y conocido por los dos lados de la comunicación, cuya longitud es de $n-k+1$ bits. Este polinomio actúa como divisor en el proceso de codificación y su grado es $n-k$.
- $Q(x)$: Polinomio cociente.
- $R(x)$: Polinomio asociado a los $n-k$ bits redundantes. El proceso de codificación tiene como objetivo generar esos bits para concatenarlos con los k bits del mensaje y dar lugar a la palabra código de n bits. El polinomio $R(x)$ resulta del resto de la división que se lleva a cabo en el proceso de codificación, y su grado es $n-k-1$.
- $P(x)$: Polinomio asociado a la palabra código. Su grado es $n-1$.

A partir de estas definiciones, el proceso de codificación se puede describir algebraicamente en dos pasos, teniendo en cuenta, sin embargo, que todas las operaciones se realizan en aritmética módulo 2 (el uso de aritmética módulo 2 equivale a utilizar el álgebra ordinaria, pero teniendo en cuenta que $1 \oplus 1 = 0$, por lo que en dicha aritmética la suma y la substracción son coincidentes). Los dos pasos a realizar son los siguientes:

$$\frac{x^{n-k}\cdot M(x)}{G(x)} = Q(x) + \frac{R(x)}{G(x)} \qquad (8.15a)$$

$$P(x) = x^{n-k}\cdot M(x) + R(x) \qquad (8.15b)$$

Por ejemplo, supongamos que el mensaje a enviar es el byte 10011101 considerado más arriba, representado por el polinomio $x^7 + x^4 + x^3 + x^2 + 1$, y que el patrón de bits es 1011, cuyo polinomio asociado es $x^3 + x + 1$ (polinomio generador). Como el polinomio generador es de grado 3, se tiene que $n-k = 3$, lo que significa que vamos a añadir 3 bits de redundancia para obtener palabras código de 11 bits ($n = k + 3 = 8 + 3 = 11$). Al realizar las operaciones (8.15), el resultado que se obtiene es el siguiente (compruébese a modo de ejercicio):

- $Q(x) = x^7 + x^5 + 1$

- $R(x) = x + 1$
- $P(x) = x^{10} + x^7 + x^6 + x^5 + x^3 + x + 1$

La expresión alcanzada para $P(x)$ indica que la palabra código generada es 10011101011. También se puede razonar a partir de la expresión alcanzada para el polinomio $R(x)$. Su grado es inferior en una unidad al grado del polinomio generador, es decir, $n - k - 1 = 2$ y, por tanto, la combinación binaria asociada al mismo es 011.

Cuando una palabra código generada se transmite, algunos bits pueden verse alterados. Esto también puede expresarse de forma polinómica según se expone a continuación, donde $P'(x)$ es la palabra código recibida y $E(x)$ es el polinomio error, es decir, el polinomio asociado a una secuencia de n bits en la que los 1s marcan las posiciones de los errores de bit en la palabra código enviada:

$$P'(x) = P(x) + E(x) \tag{8.16}$$

Por ejemplo, para la palabra código del ejemplo anterior, si la que se recibe es 10001111001, se tiene $E(x) = x^7 + x^4 + x$, pues en este caso el patrón de error es 00010010010. Para determinar si la palabra código recibida es errónea, el receptor calcula el resto de la división de $P'(x)$ entre $G(x)$, es decir:

$$R'(x) = rem\left(\frac{P'(x)}{G(x)}\right) \tag{8.17}$$

Si $R'(x) \neq 0$, el receptor determina que la transmisión ha sido errónea; si $R'(x) = 0$, acepta la palabra código recibida como válida, aunque existe una probabilidad no nula de que el patrón de bits erróneos sea tal que conduzca a un resto 0. Esto último ocurre, precisamente, cuando el polinomio $E(x)$ resulta ser divisible por $G(x)$ (de ahí la ventaja de la formulación basada en polinomios, pues permite analizar la capacidad detectora en términos de divisibilidad). La secuencia binaria asociada a $R'(x)$, de cuyo valor depende la aceptación o rechazo de la palabra código recibida, se denomina síndrome. De forma general, en detección de errores, el **síndrome** es la secuencia binaria que sirve de base para decidir si la palabra código recibida es aceptada o rechazada. Habitualmente, la condición de aceptación es que el síndrome sea nulo, como ocurre precisamente en el caso de los códigos CRC.

La capacidad de detección de errores de los códigos CRC reside en la selección del polinomio $G(x)$. Concretamente, se pueden demostrar las siguientes propiedades, algunas de ellas formuladas en términos de la descomposición factorial de dicho polinomio (en aritmética módulo 2):

- Si $G(x)$ tiene más de un término distinto de 0, entonces se detectan todos los errores de un único bit.
- Si $G(x)$ no divide a $x^t + 1$, para $t = 0 \ldots n - 1$, entonces se detectan todos los errores dobles.
- Si $G(x)$ contiene el factor $(x + 1)$, se detecta cualquier número impar de errores.

Las siguientes condiciones se refieren a la capacidad de detectar ráfagas de errores. Sea $g = n - k$ el grado del polinomio generador, y r la longitud de una ráfaga:

- Todas las ráfagas de errores tales que $r \leq g$ son detectadas.
- Todas las ráfagas de errores de longitud $r = g + 1$ son detectadas con probabilidad $1 - 0.5^{g-1}$.
- Todas las ráfagas de errores de longitud $r > g + 1$ son detectadas con probabilidad $1 - 0.5^{g}$.

Las condiciones anteriores ponen de manifiesto que la selección de un polinomio generador que favorezca la robustez del código no es un problema trivial. En la **Tabla 8.2** se listan algunos de los polinomios generadores estandarizados más importantes, juntamente con los casos de uso. Obsérvese que el código de paridad simple equivale a un CRC-1.

Tabla 8.2. Algunos polinomios estándares y casos de uso.

Código	Polinomio $G(x)$	Uso
CRC-1 ≡ SPC	$x + 1$	Ver **Tabla 8.1**
CRC-5-EPC	$x^5 + x^3 + 1$	RFID
CRC-5-USB	$x^5 + x^2 + 1$	USB (paquetes *token*)
CRC-8-Bluetooth	$x^8 + x^7 + x^5 + x^2 + x + 1$	Bluetooth
CRC-8-LTE	$x^8 + x^7 + x^4 + x^3 + x + 1$	4G LTE/LTE-Advanced
CRC-16-IBM	$x^{16} + x^{15} + x^2 + 1$	Bysinc, USB (paquetes de datos)
CRC-CCITT	$x^{16} + x^{12} + x^5 + 1$	HDLC, RFID, 4G LTE/LTE-Advanced, 5G NR
CRC-24a	$x^{24} + x^{23} + x^{18} + x^{17} + x^{14} + x^{11} + x^{10} + x^7 + x^6 + x^5 + x^4 + x^3 + x + 1$	4G LTE/LTE-Advanced, 5G NR
CRC-24b	$x^{24} + x^{23} + x^6 + x^5 + x + 1$	
CRC-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$	HDLC, estándares LAN

Códigos *checksum*

En los códigos *checksum*, el mensaje se divide en unidades (palabras) de m bits, a partir de las cuales se genera una unidad extra de m bits (*checksum*) mediante un procedimiento basado en sumas. La unidad extra se envía a continuación del mensaje. Básicamente, el receptor comprueba que el *checksum* encaja con el mensaje enviado. A igualdad de redundancia, los códigos *checksum* son menos potentes que los códigos CRC, pero computacionalmente más sencillos. Por ello, su uso se reserva a las capas altas de la arquitectura TCP/IP (protocolos IP y TCP, concretamente), pues habitualmente están implementadas en software y consumen tiempo de CPU del dispositivo (por el contrario, la capa de enlace se implementa típicamente en hardware y cuenta con una CPU dedicada, pudiendo alcanzar una velocidad de proceso muy alta, compatible con la complejidad computacional de los códigos CRC). Otra razón por la que resulta apropiado utilizar los códigos CRC en la capa de enlace es su alta capacidad detectora, la cual permite que la mayor parte de los errores de transmisión sean detectados en dicha capa, donde las retransmisiones conllevan menor retardo, quedando relegados los errores residuales al protocolo TCP, que conlleva retransmisiones extremo a extremo.

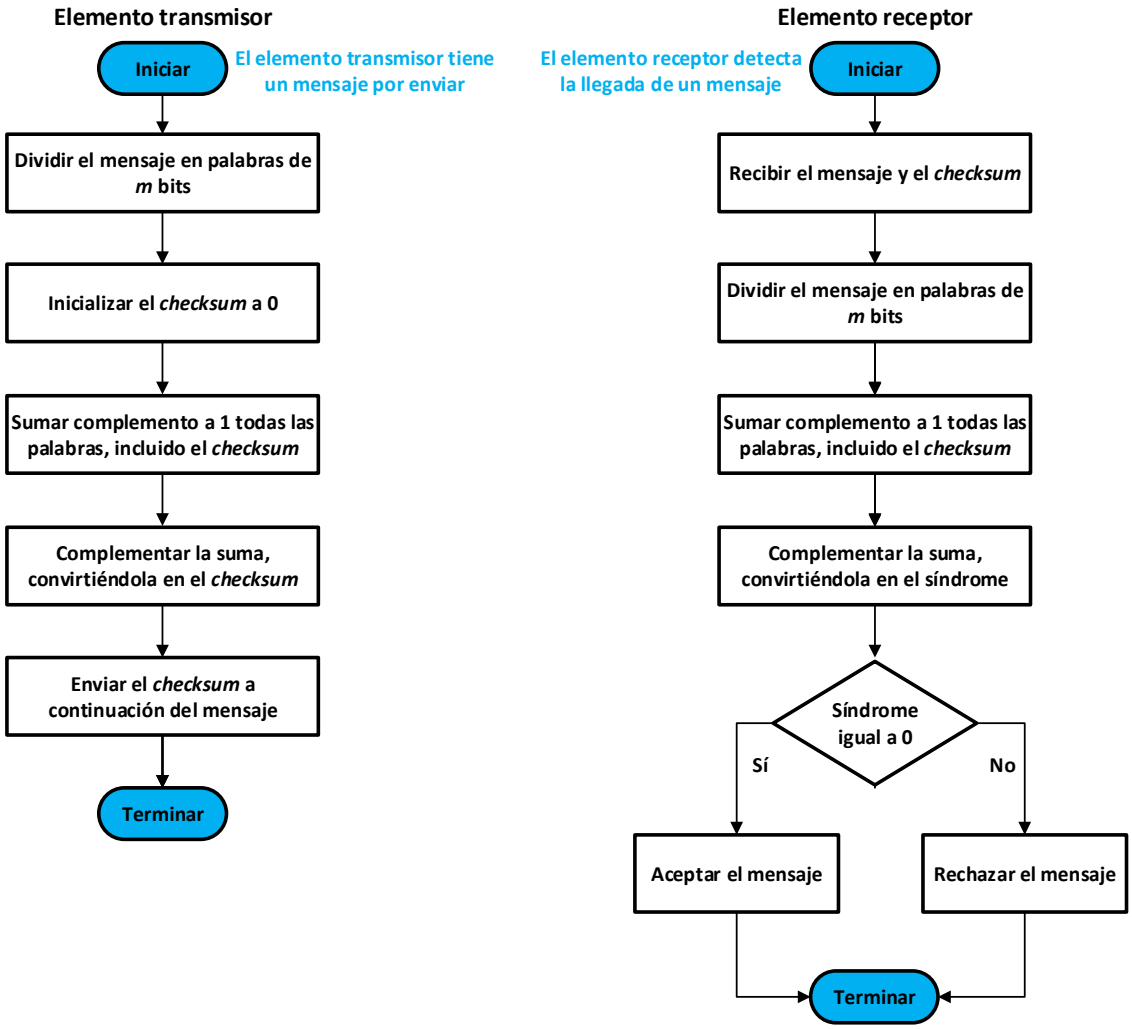


Figura 8.13. Algoritmos ejecutados por los elementos transmisor y receptor por cada mensaje intercambiado, suponiendo codificación *checksum* basada en complemento a 1.

Existen varias modalidades de suma en la generación del *checksum*, siendo la suma complemento a 1 la más habitual. En aritmética complemento a 1, con sólo m bits podemos representar números sin signo entre 0 y 2^{m-1}, por lo que, si el número a representar es mayor, los bits de mayor peso sobrantes se suman a los m bits de menor peso (en inglés, *wrapping*). Por ejemplo, supongamos que el mensaje a enviar es $1000110101000111010 1$ y se trocea en *nibbles*, es decir, $m = 4$. Si realizamos la suma binaria (complemento a 2) de los 5 *nibbles* que constituyen el mensaje, el resultado es 100101 (naturalmente, podemos verificar esta suma en términos de los valores decimales, es decir, $8 + 13 + 4 + 7 + 5 = 37$). Para simular la suma complemento a 1, realizamos el *wrapping*, es decir, $0101 + 10 = 0111$. El transmisor podría acoplar el valor 0111 como *checksum*, justo a continuación de los 20 dígitos binarios que constituyen el mensaje, de manera que el receptor tendría que realizar la suma complemento a 1 de los 5 nibbles recibidos y comprobar si el resultado coincide con el *checksum* recibido. Si coincide, el mensaje recibido se acepta como válido; en caso contrario, se descarta. No obstante, la tarea del receptor puede simplificarse

ligeramente aprovechando la propiedad de que la suma de un número binario y su complemento (es decir, el que resulta de cambiar los 0s por 1s y los 1s por 0s) es siempre un número con todos sus bits iguales a 1. Por tanto, volviendo al ejemplo anterior, el transmisor podría enviar el valor 1000 como *checksum*, pues es el número que resulta de complementar 0111. Bastaría entonces que el receptor realizara la suma complemento a 1 de los 6 nibbles recibidos, es decir, incluyendo el *checksum*, para obtener el síndrome, que debería ser 1111 para que el mensaje fuera aceptado como válido. Finalmente, para equiparar totalmente el algoritmo del receptor con el del transmisor, consistente el de este último en sumar complemento a 1 y complementar, el receptor debería complementar el resultado de la suma realizada, por lo que el síndrome esperado sería 0000 en lugar de 1111. Los algoritmos mostrados en la **Figura 8.13**, uno ejecutado por el transmisor y el otro por el receptor, son esencialmente equivalentes.

La sencillez de la codificación *checksum* también revela fácilmente sus debilidades para la detección de errores. En efecto, si una o varias palabras se incrementan y otra u otras se reducen en la misma cantidad, el *checksum* no cambia y, por tanto, los errores no son detectados. Para paliar esta deficiencia se introdujeron otras modalidades de suma, básicamente consistentes en asignar un peso a cada palabra según su posición dentro del mensaje. Los ejemplos más sobresalientes son el *checksum* de Fletcher y el *checksum* de Adler; no obstante, la mejora introducida por estos códigos en cuanto a capacidad detectora no compensa el aumento de complejidad que conllevan, por lo que su uso es más bien marginal. De hecho, el *checksum* estándar de Internet se basa en la suma complemento a 1 en base a palabras de 16 bits ($m = 16$), independientemente del tamaño del mensaje (RFC 1071).

Mecanismos ARQ

Cuando se detecta que el mensaje recibido es erróneo y no puede ser corregido (porque el código utilizado sólo tiene capacidad detectora o porque el patrón de bits erróneos excede su capacidad correctora), y se requiere disponer de la versión correcta de dicho mensaje, la única solución es solicitar su retransmisión. De esto se encargan los **mecanismos ARQ** (*Automatic Retransmission reQuest*), también conocidos como **mecanismos BEC** (*Backward Error Correction*), pues en definitiva consisten en corregir los errores volviendo hacia atrás en el proceso de transmisión del mensaje. Alternativamente, los **mecanismos FEC** (*Forward Error Correction*) no son más que el conjunto de mecanismos orientados a posibilitar que la corrección de errores pueda realizarse sin volver hacia atrás, por ejemplo, enviando cada vez dos réplicas del mismo mensaje (solución poco eficiente) o, sobretodo, utilizando códigos correctores. Mecanismos ARQ y mecanismos FEC no son necesariamente excluyentes; de hecho, como se apunta más arriba, cuando un mensaje erróneo no puede ser corregido porque el error excede la capacidad correctora del código, es decir, el mecanismo FEC falla, se debe recurrir a un mecanismo ARQ si se quiere garantizar la fiabilidad de la transmisión al 100%.

Los mecanismos ARQ son aplicables tanto en la capa de enlace como en la capa de transporte de las arquitecturas OSI y TCP/IP; no obstante, como ya hicimos con los mecanismos de control de flujo, nos centraremos en los mecanismos ARQ de la capa de enlace, teniendo presente que

esencialmente son los mismos que los utilizados en la capa de transporte, salvo diferencias poco sustanciales. Así pues, el diagrama de la **Figura 8.14** muestra la versión completa del proceso de control de errores en la capa de enlace, suponiendo la utilización de un código bloque sistemático. Como ya sabemos, la unidad básica de información en dicha capa es la trama, cuya estructura se recuerda en la misma figura. Dicha estructura pone de manifiesto que los delimitadores de comienzo y final de trama (*flags*) no intervienen en la codificación de canal, por lo que la estructura de la trama a efectos de control de errores queda circunscrita a los dos siguientes conjuntos de bits:

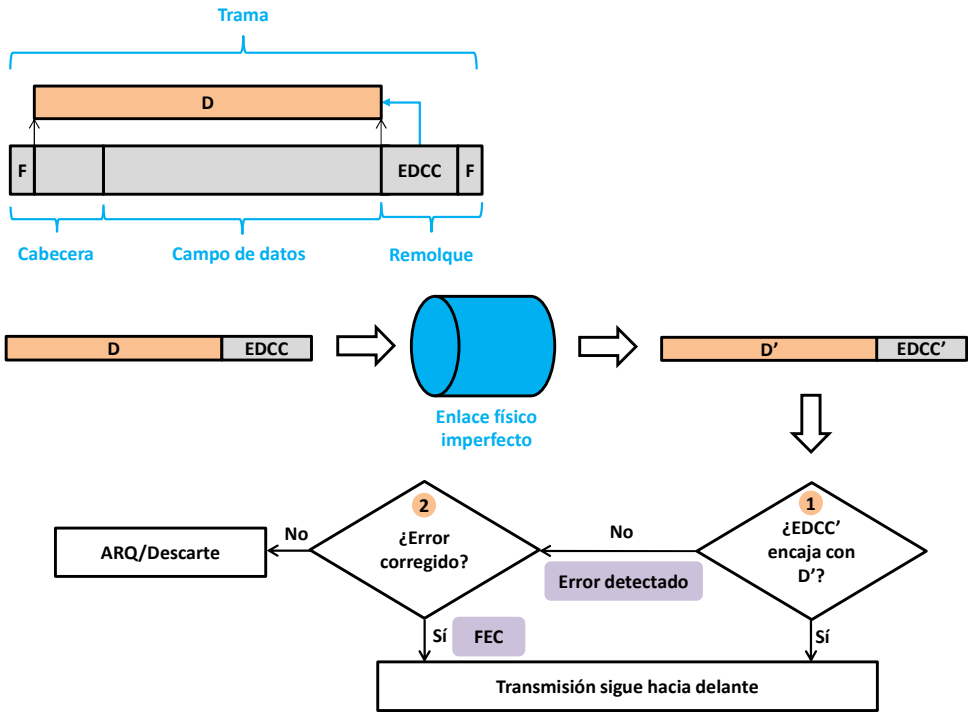


Figura 8.14. Proceso de control de errores en la capa de enlace, suponiendo que se utiliza un código bloque sistemático.

- D (*Data*): Es el conjunto de bits a proteger, genéricamente, dato o mensaje: cabecera, salvo el delimitador de comienzo de trama, más el campo de datos.
- EDCC (*Error Detection/Correction Code*): Es la redundancia, es decir, el conjunto de bits calculado a partir del conjunto D de acuerdo con el algoritmo de detección o corrección de errores.

Como se desprende de la **Figura 8.14**, el efecto de transmitir una trama a través del enlace físico imperfecto es la recepción de sendos conjuntos D' y EDCC', que en general no tienen por qué coincidir con D y EDCC respectivamente. Dependiendo del patrón de errores y del tipo de código de protección contra los mismos, pueden darse diversas situaciones reflejadas en las salidas de las decisiones 1 y 2:

- Decisión 1. Cualquiera que sea el código bloque sistemático utilizado, la detección de errores se basa en verificar si EDCC' encaja con D'. En caso afirmativo, la trama se da por correcta y la transmisión sigue hacia delante, incluso si casualmente EDCC' encaja con D' aun siendo el mensaje erróneo. Esto ocurre cuando el patrón de errores excede la capacidad detectora del código. En caso negativo, la trama se da por incorrecta, incluso en el caso de que los errores de bit se hayan producido exclusivamente en el campo EDCC' (falsa alarma).
- Decisión 2. En esta decisión se pone de manifiesto la capacidad correctora del código utilizado. Si solamente tiene capacidad detectora, el error no puede ser corregido y hay que invocar algún mecanismo de retransmisión si se quiere garantizar una transmisión 100% fiable. Si el código tiene capacidad correctora, pero el patrón de errores excede dicha capacidad, se repite la situación anterior. Finalmente, si el patrón de errores encaja en la capacidad correctora del código, los errores son corregidos y la transmisión sigue hacia delante (el mecanismo FEC resulta exitoso).

La **Figura 8.14** muestra también que una alternativa a la retransmisión es el descarte de la trama. En general, la opción del descarte es preferible en dos tipos de situaciones:

- Cuando la tasa de errores de transmisión es muy baja, y con el objeto de preservar la simplicidad del protocolo de enlace y no ralentizar demasiado la transmisión, se relega la gestión de tales errores al protocolo TCP de la capa de transporte, el cual detecta la ausencia de los segmentos encapsulados en las tramas descartadas. Es lo que ocurre con algunos protocolos de enlace diseñados para medios guiados, como es el caso de Ethernet.
- Cuando se trata de transmisiones multimedia con requisitos de tiempo real, ya que las retransmisiones generan retardos inaceptables en el proceso de reproducción que tiene lugar en el destinatario. Por la misma razón, tampoco se relegan las retransmisiones a la capa de transporte; de hecho, en estos escenarios se utilizan otros protocolos de transporte como UDP o RTP, que no invocan retransmisiones. Con el objeto de acotar la tasa de información perdida, para este tipo de aplicaciones puede ser más interesante utilizar en la capa de enlace un código con capacidad correctora que uno con capacidad sólo detectora. Esto es lo previsto en algunos estándares WiFi de alta velocidad, como IEEE 802.11n e IEEE 802.11ac (la tasa de errores de bit en transmisiones inalámbricas es alta, y si hubiera que descartar todas las tramas erróneas al no poder corregirlas, la calidad de servicio se vería seriamente mermada).

Como ya hemos visto, la mayor parte de códigos de control de errores utilizados en redes de computadores tienen solamente capacidad detectora; no obstante, el uso de códigos correctores se ha expandido en los últimos años. Actualmente son tres los escenarios en que los códigos correctores resultan preferibles a los sólo detectores:

- Cuando el retardo de propagación es muy alto, bien porque la distancia que tiene que recorrer la señal es muy elevada (transmisiones por satélite), bien porque la velocidad de propagación es muy baja (comunicaciones submarinas basadas en señales acústicas).

- En estándares de red de muy alta velocidad, con el fin de evitar penalizar esta velocidad con frecuentes retransmisiones de tramas. Algunos ejemplos son las versiones wifi más modernas (IEEE 802.11n, IEEE 802.11ac, y otras) y las redes Ethernet a 10 Gbps.
- Cuando las tramas erróneas van a descartarse, con el objeto de reducir la tasa de tramas descartadas.
- En transmisiones unidireccionales, donde no existe la posibilidad de solicitar retransmisión. Es el caso, por ejemplo, de las transmisiones *broadcast* de TV.

Como se apuntó al principio de esta sección, los mecanismos ARQ se implementan como extensiones de los mecanismos de control de flujo, básicamente mediante la incorporación de acuses de recibo negativos (NAK). Empezamos con el estudio del mecanismo de control de errores de parada y espera (*stop-and-wait ARQ*), para continuar con los que resultan de la extensión del mecanismo de control de flujo basado en ventana deslizante.

Mecanismo *stop-and-wait ARQ*

El mecanismo ***stop-and-wait ARQ*** es el mecanismo de retransmisión más sencillo y su descripción se muestra a través del diagrama de tiempos detallado de la **Figura 8.15**. Naturalmente, dicho diagrama debe contemplar la situación en que la trama enviada (trama de datos, la que contiene el mensaje) es detectada errónea, lo que queda reflejado con una cruz sobre la misma. Si el receptor de esa trama no puede corregirla y tampoco descartarla, debe solicitar su retransmisión, para lo cual envía una trama de control específica, la trama NAK, indicando que la transmisión ha sido fallida. El transmisor, que debe guardar una copia de la trama de datos enviada, la retransmitirá tantas veces como sea necesario (una sola vez en la figura) hasta recibir una confirmación positiva en forma de trama ACK. En dicho momento puede eliminar la copia de la trama de datos y proceder a la transmisión de la siguiente.

La **Figura 8.15** también pone de manifiesto una situación extrema de transmisión fallida, en la cual la trama se pierde. Pero, ¿qué significa que una trama se pierda en un enlace directo entre dos dispositivos? Son varias las posibles razones:

- El nivel de ruido es momentáneamente tan alto que el receptor no es capaz de identificar el delimitador de comienzo de trama, con lo que ésta queda inadvertida.
- La potencia media de señal con que se recibe la trama está por debajo de la sensibilidad del receptor, por lo que éste no la detecta. Esto puede producirse, por ejemplo, por una caída de tensión en el circuito que conecta transmisor con receptor.
- La trama colisiona con otras y su delimitador de comienzo de trama queda destruido. Esto puede ocurrir cuando el medio que conecta transmisor con receptor es compartido.

La posibilidad de que se pierdan tramas de datos, o de que se pierdan los acuses de recibo, o simplemente que estos sean detectados erróneos, genera la necesidad de asignar un temporizador a cada trama de datos enviada. El uso del mismo se pone de manifiesto para la trama perdida del diagrama de la **Figura 8.15**. De no existir tal temporizador, el transmisor estaría esperando indefinidamente un acuse de recibo que nunca iba a llegar (situación de bloqueo del transmisor).

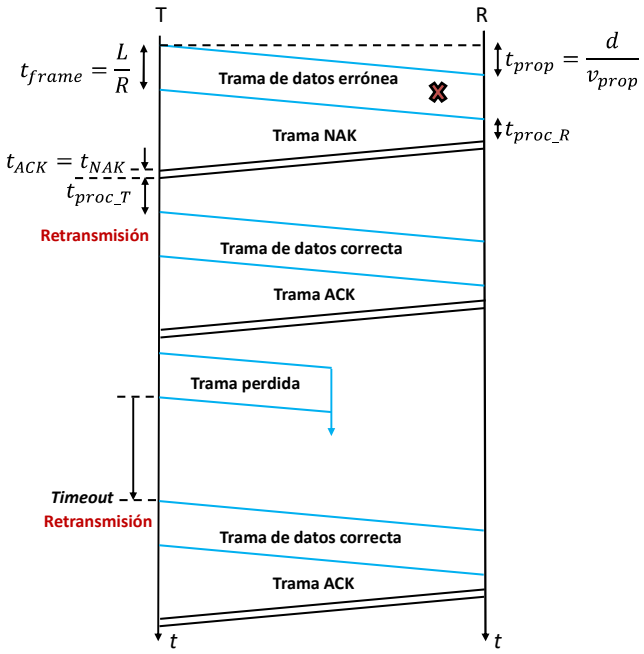


Figura 8.15. Diagrama de tiempos detallado del mecanismo *stop-and-wait* ARQ.

El funcionamiento del temporizador es sumamente sencillo. Cada vez que se completa la transmisión de una trama de datos, su valor empieza a decrecer desde un valor inicial prestablecido, de forma que si alcanza el valor nulo sin que el transmisor reciba un acuse de recibo (positivo o negativo), la trama de datos se retransmite automáticamente. Se trata, pues, de una retransmisión vía expiración del temporizador (*timeout*), en lugar de una retransmisión vía NAK.

Eficiencia del mecanismo *stop-and-wait ARQ*

Para calcular la eficiencia del mecanismo *stop-and-wait ARQ*, introducimos las siguientes hipótesis, algunas de las cuales ya fueron planteadas en los cálculos de eficiencia de los mecanismos de control de flujo:

- Tráfico de saturación.
- Tiempos de procesado y duración de los acuses de recibo despreciables.
- Las tramas de confirmación siempre se transmiten correctamente, y las tramas de datos nunca se pierden. Con ambas hipótesis conseguimos simplificar el cálculo de eficiencia, al no tener que incluir el *timeout*. Ello es también coherente con el objetivo de evaluar la máxima eficiencia que el mecanismo puede dar de sí, pues las retransmisiones vía *timeout* generan mayor retardo que las retransmisiones vía NAK.

Bajo estas hipótesis, el diagrama de tiempos a considerar en el cálculo de eficiencia es el mostrado en la **Figura 8.16**. Lógicamente aparece una nueva variable en el análisis, que es la probabilidad de retransmisión (p_{retr}), es decir, la probabilidad de que el receptor decida solicitar retransmisión, cualquiera que sea la situación que conduzca a ello. A diferencia de lo que ocurría con los

mecanismos de control de flujo, el análisis adquiere carácter estocástico, pues la ocurrencia de errores es un fenómeno totalmente aleatorio, como lo es el número de veces que una trama puede tener que retransmitirse.

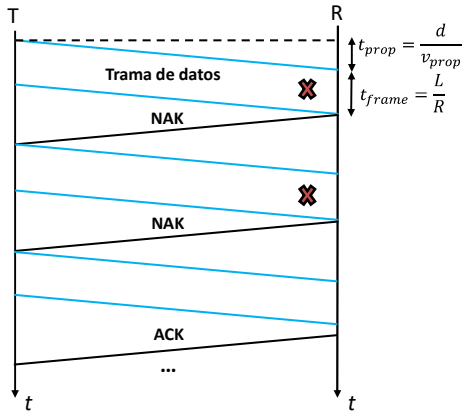


Figura 8.16. Diagrama de tiempos detallado para el cálculo de eficiencia del mecanismo *stop-and-wait* ARQ.

Para obtener la expresión de la eficiencia, realizamos una hipótesis adicional, consistente en admitir que las sucesivas transmisiones (o retransmisiones) de tramas son mutuamente independientes. La situación general queda descrita en la **Figura 8.16**, donde una misma trama se transmite un número variable de veces hasta que el transmisor recibe un ACK, momento en que puede pasar a transmitir la siguiente trama. La transmisión de una trama de datos es tanto más eficiente cuantas menos veces haya que enviarla al receptor. Como queremos realizar una evaluación global, calcularemos la eficiencia basada en el número medio de veces que hay que transmitir una trama de datos hasta que se recibe su ACK. Obsérvese que el escenario es equivalente al experimento de lanzar una moneda hasta que sale cara, suponiendo que la probabilidad de cruz es p_{retr}. El número de veces que hay que lanzarla obedece a una distribución conocida, la distribución geométrica; según esta distribución, el número medio de veces ($\bar{n}$) que hay que lanzar la moneda, es decir, transmitir una misma trama de datos en nuestro caso, viene dado por $\bar{n} = \frac{1}{1-p_{retr}}$; como por cada transmisión se consume un tiempo $t_{frame} + 2t_{prop}$, el tiempo medio invertido en la transmisión de una misma trama de datos es $\bar{n}(t_{frame} + 2t_{prop})$. Por tanto, la eficiencia del mecanismo se puede expresar del siguiente modo:

$$\eta_{S\&WARQ} = \frac{t_{frame}}{\bar{n}(t_{frame}+2t_{prop})} = \frac{1-p_{retr}}{1+2a} \qquad (8.18)$$

Obsérvese que, si $p_{retr} = 0$, se obtiene la eficiencia del mecanismo de control de flujo *stop-and-wait* – ecuación (8.6).

En la expresión (8.18), sabemos cómo calcular la magnitud a a partir de parámetros técnicos del enlace, pero no podemos afirmar lo mismo sobre p_{retr}. Por lo general, la probabilidad de que una trama tenga que ser retransmitida depende de varios factores:

- La tasa de errores de bit (BER).
- El tipo de codificación de canal (capacidad detectora y/o correctora del código utilizado).
- La longitud de la trama en bits (L).
- El patrón de bits erróneos, que obedece a un determinado **modelo de error** (modelo matemático que caracteriza la ocurrencia de errores de bit).

En el análisis que sigue a continuación, adoptaremos dos hipótesis simplificadoras con el objeto de obtener una expresión práctica de la probabilidad de retransmisión:

1. El código de control de errores tiene máxima capacidad detectora y nula capacidad correctora. Los códigos CRC se acercan a esta condición.
2. Independencia de errores a nivel de bit. Éste es el modelo de error más sencillo, y es compatible con la presencia de ruido térmico en el canal como única fuente perturbadora.

La primera hipótesis permite identificar la probabilidad de retransmisión con la probabilidad de trama errónea (p). La segunda hipótesis permite expresar fácilmente la probabilidad de trama errónea en términos de la probabilidad de error de bit y la longitud de las tramas. El resultado es:

$$p_{retr} = p = 1 - (1 - BER)^L \qquad (8.19)$$

Con esta expresión de la probabilidad de retransmisión, la eficiencia del mecanismo *stop-and-wait ARQ* queda expresada definitivamente como sigue:

$$\eta_{S\&WARQ} = \frac{1-p}{1+2a} \qquad (8.20)$$

Mecanismos de control de errores basados en ventana deslizante

La eficiencia que se obtiene al aplicar la expresión (8.20) suele arrojar valores bastante bajos, salvo en situaciones especiales. Como ya vimos en control de flujo, el aumento de la ventana de transmisión contribuye a mejorar la eficiencia. Concretamente, hay dos mecanismos de control de errores basados en ventana deslizante, que resultan de la extensión del mecanismo homólogo en control de flujo: volver-atrás-N (*go-back-N*) y rechazo selectivo (*selective reject*) o repetición selectiva (*selective repeat*). En el mecanismo **volver-atrás-N**, al recibir un acuse de recibo negativo, el transmisor retransmite la trama indicada por dicho acuse y todas las que le siguieron, aunque éstas se hubieran transmitido correctamente. En cambio, en el mecanismo **rechazo selectivo**, el transmisor sólo retransmite la trama indicada por el acuse de recibo. Este último caso requiere que el receptor sea capaz de reordenar las tramas a partir de sus números de secuencia. La **Figura 8.17** ilustra los dos mecanismos para el mismo tamaño de ventana y la misma situación de trama errónea. Obsérvese que, al igual que el ACK, el NAK especifica el siguiente número de trama que el receptor espera recibir.

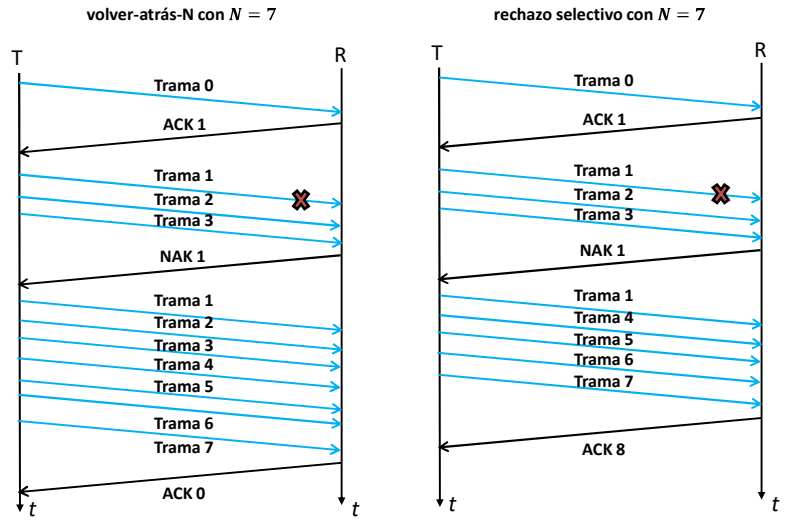


Figura 8.17. Diagrama de tiempos de los mecanismos volver-atrás-*N* y rechazo selectivo para un tamaño de ventana igual a 7. Estos diagramas de tiempo no son detallados, sino puramente funcionales, pues su único objeto es poner de manifiesto el intercambio de tramas de datos y acuses de recibo.

Aparentemente, resulta más eficiente el mecanismo de rechazo selectivo; sin embargo, a fin de evitar situaciones de ambigüedad en la identificación de las tramas, el mecanismo de rechazo selectivo impone una ventana más restrictiva que el mecanismo volver-atrás-N para un mismo número de bits (k) de secuenciación de las tramas. En efecto, fijados k bits para secuenciar las tramas (números de secuencia entre 0 y 2^{k-1}), el tamaño de ventana permitido por ambos mecanismos es $N \leq 2^k - 1$ en el caso de volver-atrás-N y $N \leq 2^{k-1}$ en el caso de rechazo selectivo. A la inversa, para un determinado valor del tamaño de ventana, el mecanismo de rechazo selectivo exige más bits de secuenciación ($k \geq \log_2 N + 1$) que el mecanismo volver-atrás-N ($k \geq \log_2(N+1)$). Por ejemplo, en el caso de $N = 7$ (**Figura 8.17**), los mecanismos volver-atrás-N y rechazo selectivo exigen $k \geq 3$ y $k \geq 4$ respectivamente. En la figura se ha supuesto exactamente $k = 3$ en el caso de volver-atrás-N, de ahí que, tras recibir la trama 7, el receptor envía un ACK 0; en el caso de rechazo selectivo, como $k \geq 4$, el receptor envía un ACK 8 tras recibir la trama 7. En conclusión, si el número de bits de secuenciación de las tramas viene fijado por el protocolo de enlace, a priori no se puede saber qué mecanismo proporcionará una mayor eficiencia, si volver-atrás-N con una ventana mayor o rechazo selectivo con una ventana menor. Afortunadamente, disponemos de fórmulas de eficiencia que nos permiten determinar en cada caso la opción más apropiada. Estas fórmulas se resumen en la **Tabla 8.3** (omitimos la demostración de las mismas). Observamos que en cada caso hay dos fórmulas de eficiencia, una correspondiente a $N \geq 2a + 1$, la otra correspondiente a $N < 2a + 1$, de las cuales siempre la primera proporciona resultados más altos. En general, salvo que se especifique lo contrario, supondremos que los mecanismos de control de errores basados en ventana deslizante trabajan con la ventana máxima permitida, es decir, N alcanza la cota superior indicada en la columna intermedia de la tabla.

Tabla 8.3. Eficiencia de los mecanismos de control de errores basados en ventana deslizante.

Mecanismo	Tamaño de ventana	Eficiencia
volver-atrás-N	$N \leq 2^k - 1$	$\eta_{GBN} = \begin{cases} \frac{1-p}{1+2a \cdot p}, & N \geq 2a+1 \\ \frac{N(1-p)}{(2a+1)(1-p+N \cdot p)}, & N < 2a+1 \end{cases}$
rechazo selectivo	$N \leq 2^{k-1}$	$\eta_{SREJ} = \begin{cases} 1-p, & N \geq 2a+1 \\ \frac{N(1-p)}{2a+1}, & N < 2a+1 \end{cases}$

8.3. CONTROL DE CONGESTIÓN

Conceptualmente, el **control de congestión** es similar al control de flujo, con la diferencia de que el elemento en situación de congestión es la propia red o una parte de la misma (un nodo, por ejemplo). En control de flujo, tanto en la capa de enlace como en la capa de transporte, la acción reguladora es 1 a 1, del elemento que recibe sobre el que transmite; en cambio, en control de congestión, la acción reguladora es 1 a N, del elemento de red que acumula tráfico sobre las fuentes que lo generan.

La propia función de encaminamiento puede aliviar una situación de congestión, simplemente redistribuyendo el tráfico que se acumula en un nodo hacia otros nodos. No obstante, en no pocas ocasiones esta técnica tan solo consigue trasladar el punto de congestión a otra parte de la red. El hecho es que cualquier red de datos tiene una capacidad limitada, y cuando el tráfico entrante crece y se acerca a dicha capacidad, el nivel de ocupación de los *buffers* de todos los nodos también crece, los retardos experimentados por las conexiones son cada vez mayores y la tasa de paquetes perdidos aumenta sin control. En tales circunstancias, la verdadera solución consiste en frenar el tráfico generado por las fuentes.

Existen diversas técnicas de control de congestión, aplicables cualquiera que sea la magnitud de la zona afectada, desde aquella que comprende un solo nodo hasta la que abarca un número de nodos arbitrariamente grande. Son las siguientes:

- **Contrapresión** (*backpressure*). Este mecanismo es aplicable cuando la red de conmutación de paquetes implementa control de flujo a nivel de enlace. En este caso, como se muestra en la **Figura 8.18**, el nodo en estado de congestión (por ejemplo, cuando el nivel de ocupación de al menos uno de sus *buffers* de entrada alcanza el 80%) frena el tráfico entrante procedente de sus nodos vecinos; estos acaban entrando en ese mismo estado de congestión y, a su vez, ejercen el control de flujo sobre los nodos vecinos de los que reciben tráfico, y así sucesivamente hasta que el control de flujo acaba ejerciéndose sobre las fuentes. Por tanto, la contrapresión siempre se ejerce en sentido contrario al flujo de tráfico que provoca la congestión.

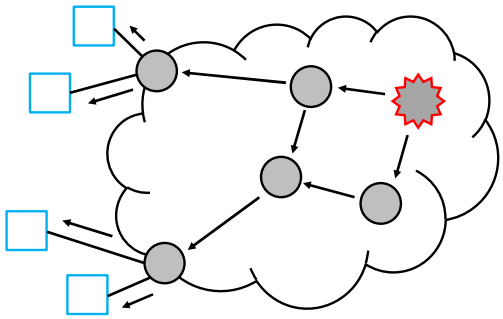


Figura 8.18. Mecanismo de control de congestión basado en contrapresión.

La contrapresión se puede implementar de forma selectiva si la red de conmutación de paquetes trabaja con circuitos virtuales. En este caso, cada nodo en estado de congestión ejerce la acción de control de flujo sobre determinadas conexiones lógicas, habitualmente las que generan mayor tráfico, de manera que finalmente sólo las fuentes de esas conexiones experimentan las restricciones de tráfico.

- **Paquetes de frenado** (*choke packets*). Según este mecanismo, el nodo en estado de congestión envía paquetes específicos de frenado a la fuente o fuentes que generan tráfico en exceso. El envío de este paquete puede producirse cuando el nodo afectado empieza a perder paquetes debido al llenado de uno de sus *buffers* de entrada; en este caso, el nodo envía un paquete de frenado por cada paquete que haya tenido que descartar. Alternativamente, el nodo afectado puede ejercer el control de congestión de forma preventiva, generando paquetes de frenado mientras la ocupación de al menos uno de sus *buffers* de entrada exceda un umbral determinado (por ejemplo, el 80%). Cualquiera que sea la implementación, las fuentes deberán ralentizar el envío de tráfico mientras reciban paquetes de frenado.
- **Señalización implícita de congestión**. Este mecanismo se basa en el reconocimiento de la situación de congestión por parte de los mismos sistemas finales. Cuando una parte de la red está en congestión, los retardos extremo-a-extremo aumentan y/o se pierden paquetes. Si una fuente es capaz de detectar esta situación, gracias a la conexión de capa de transporte con su destinatario, puede ralentizar el flujo de tráfico que genera sobre la red, hasta que detecta que la situación se ha normalizado. Este mecanismo no requiere, por tanto, ninguna acción por parte de los nodos afectados por la congestión, y representa la mejor opción en redes de conmutación de paquetes que trabajan en modo datagrama.
- **Señalización explícita de la congestión**. Es el mecanismo más preventivo de todos, pues se basa en señalizaciones explícitas del estado de la red, capaces de revelar una congestión incipiente. Generalmente, la señalización explícita se aplica en redes de conmutación de paquetes basadas en circuitos virtuales, de manera que el control de congestión puede ejercerse sobre conexiones individuales. La señalización explícita se basa en activar un campo previsto en la cabecera de los paquetes de datos o paquetes de control. Puede ser hacia delante o hacia atrás. La **señalización explícita hacia atrás** (BECN: *Backward Explicit Congestion Notification*) consiste en que el nodo en estado de riesgo de congestión marca los paquetes que, pertenecientes a una o varias conexiones lógicas, circulan hacia las fuentes que generan tráfico en exceso (aprovechando la

doble direccionalidad de las comunicaciones). Por tanto, en la señalización explícita hacia atrás, las notificaciones se envían en sentido contrario al tráfico que provoca la congestión. Por el contrario, en la **señalización explícita hacia delante** (FECN: *Forward Explicit Congestion Notification*), las notificaciones se envían en el mismo sentido que el tráfico que provoca la congestión, de manera que son los destinatarios del mismo los que las reciben. A continuación, estos destinatarios pueden ejercer acciones de control de flujo sobre las fuentes a través de las conexiones de capa de transporte.

En la **Tabla 8.4** se especifican los mecanismos de control de congestión utilizados por diversas redes de conmutación de paquetes. La tabla también incluye la modalidad de conmutación utilizada por cada una de ellas, así como la capa TCP/IP en la que operan los mecanismos correspondientes.

Tabla 8.4. Mecanismos de control de congestión. En algunos casos se señala entre paréntesis el protocolo que ejecuta el mecanismo de control de congestión para la red considerada.

Red	Modalidad	Mecanismo de control de congestión	Capa(s) TCP/IP
X.25	Circuitos virtuales	Contrapresión selectiva	Enlace
Frame Relay	Circuitos virtuales	Señalización implícita[20] (LAPF)	
		Señalización explícita hacia atrás	
		Señalización explícita hacia delante	
ATM	Circuitos virtuales	Señalización explícita hacia atrás	
		Señalización explícita hacia delante	
Internet	Datagramas	Paquetes de frenado (ICMP)	Red
		Señalización implícita (TCP)	Transporte
		Señalización explícita hacia delante (IP y TCP)	Red y Transporte

Algunos de los mecanismos de control de congestión que se acaban de describir posibilitan un tratamiento selectivo de las conexiones lógicas, penalizando aquellas que generan más tráfico. Se pretende así aplicar un **criterio de imparcialidad** a la hora de tratar las diversas conexiones establecidas, de modo que todas ellas ocupen proporcionalmente la misma cantidad de recursos, evitando que estos sean monopolizados por las conexiones más activas. Una forma de facilitar la implementación de este criterio es que cada nodo mantenga una cola separada por cada conexión lógica establecida.

No obstante, el puro criterio de imparcialidad puede no ser el más adecuado cuando las diversas conexiones establecidas presentan requisitos de calidad de servicio distintos. Recordando la **Tabla 6.2**, hay aplicaciones que son exigentes en términos de fiabilidad y más tolerantes en las diversas métricas de comportamiento; con otras aplicaciones ocurre al revés, e incluso hay aplicaciones que son exigentes en todos los aspectos. Tener en cuenta estas diferencias conlleva un tratamiento también diferenciado. La introducción de prioridades constituye el procedimiento habitual para la gestión diferenciada de los diversos flujos de tráfico en base a **criterios de calidad de servicio**. Las

[20] En el caso de las redes *Frame Relay*, la señalización implícita recae en el protocolo LAPF, que tiene funcionalidades parecidas a TCP. LAPF gobierna las conexiones lógicas extremo a extremo (circuitos virtuales) en este tipo de redes, y se ubica en la capa de enlace del modelo TCP/IP.

prioridades también pueden servir para dar preferencia al tráfico de gestión de red frente al tráfico de las aplicaciones, especialmente en situaciones de fallo o bloqueo de nodos o enlaces.

Los mecanismos de control de congestión, junto con los criterios que permiten una aplicación más racionalizada de tales mecanismos (imparcialidad, calidad de servicio), se engloban bajo el concepto de **gestión de tráfico**. En particular, probablemente la gestión de tráfico más avanzada es la estandarizada en las redes ATM, donde el usuario y la red subscriben un acuerdo de tráfico en el momento de establecer una conexión. Este acuerdo especifica una serie de parámetros para el flujo de tráfico generado por el usuario, a la vez que garantiza la reserva de los recursos de transmisión y conmutación necesarios mientras el tráfico se ajuste a los parámetros acordados. La red puede denegar la reserva de recursos cuando estos son insuficientes para la calidad de servicio solicitada. Cabe señalar que la tecnología ATM surgió en los años 90 con la idea de implementar la llamada Red Digital de Servicios Integrados de Banda Ancha (RDSI-BA), destinada a ofrecer cobertura geográfica mundial a toda clase de aplicaciones y servicios. El fracaso de este proyecto frente a la ya existente y cada vez más extendida Internet, relegó la tecnología ATM a ser una más entre las disponibles. No obstante, la gestión del tráfico ATM, basada en el esquema de reservas descrito, ha sido versionada para Internet a través del protocolo RSVP (*Resource Reservation Protocol*) (RFC 2205), un protocolo satélite de IP.

8.4. EL PROTOCOLO HDLC

HDLC (*High-level Data Link Control*) es un **protocolo de enlace orientado a bit**, lo que significa que el campo de datos de las tramas es una secuencia de bits sin agrupar, a diferencia de los antiguos protocolos de enlace orientados a carácter, en los que el campo de datos de las tramas era siempre un múltiplo de 8 bits[21].

HDLC es un protocolo de enlace muy completo, cuya importancia radica en el hecho de que ha servido de base para el desarrollo de otros muchos protocolos de enlace, los cuales han tenido incluso mayor implantación que el propio HDLC. Entre los protocolos derivados de HDLC figuran los siguientes: PPP (*Point-to-Point Protocol*, utilizado en enlaces punto a punto de larga distancia mediante líneas serie, y en líneas ADSL), LAPB (*Link Access Procedure Balanced*, utilizado en las antiguas redes X.25), LAPM (*Link Access Procedure for Modems*, utilizado en la antigua norma V.42 para módems de marcación telefónica), LAPD (*Link Access Procedure for D-Channel*, utilizado en RDSI), LAPF (*Link Access Procedure for Frame Relay*, para redes *Frame Relay*), LLC (*Logical Link Control*, protocolo de enlace para todas las redes de área local estandarizadas por el IEEE) y Cisco HDLC, una solución propietaria de Cisco, también para enlaces punto a punto de larga distancia (líneas serie).

[21] Los **protocolos de enlace orientados a carácter** fueron muy populares cuando los dispositivos interconectados solamente intercambiaban texto, codificado habitualmente mediante caracteres ASCII de 8 bits. Con la transmisión de nuevos formatos de información además de texto (imágenes, audio, etc.), y la expansión de otros sistemas de codificación basados en caracteres de 16 o 32 bits (por ejemplo, Unicode), los protocolos orientados a carácter dejaron paso a los protocolos orientados a bit.

Actualmente, todos los detalles de HDLC están recogidos en el estándar ISO/IEC 13239:2002. HDLC fue diseñado tanto para enlaces punto a punto como para enlaces multipunto, opciones denominadas, respectivamente, configuraciones balanceada y no balanceada. En la **configuración no balanceada**, una de las estaciones, la **estación primaria** (P), ejerce el control del enlace, y las demás son **estaciones secundarias** (S). Para esa configuración se definen 2 modos de operación:

- **NRM** (*Normal Response Mode*). Es el modo habitual dentro de la configuración balanceada. En este caso, sólo la primaria puede iniciar la comunicación con cualquiera de las secundarias. Para ello, la primaria envía una **orden** o **comando**, y la secundaria solicitada envía una **respuesta** a ese comando.
- **ARM** (*Asynchronous Response Mode*). En este caso, una secundaria puede iniciar la transmisión, pero la primaria sigue gestionando el enlace (inicialización, recuperación de errores, desconexión, etc.).

Cualquiera que sea el modo de operación dentro de la configuración no balanceada, no puede existir comunicación entre secundarias. Esta configuración tendría actualmente escaso recorrido, pero fue útil en su momento para gestionar los llamados sistemas centralizados, formados por un gran computador central y un conjunto de terminales de usuario, cada uno de ellos consistente en una combinación de teclado y pantalla.

Por otra parte, la **configuración balanceada** interconecta 2 estaciones de igual jerarquía, a cada una de las cuales se la denomina **estación combinada** (C), pues puede actuar en un momento dado como primaria y en otro como secundaria. El modo de operación en la configuración balanceada se denomina **ABM** (*Asynchronous Balanced Mode*). La **Figura 8.19** muestra las dos configuraciones descritas.

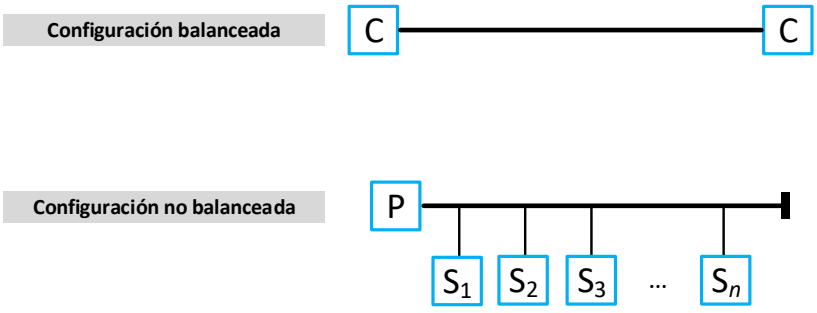


Figura 8.19. Configuraciones balanceada y no balanceada del protocolo HDLC.

En HDLC se definen 3 tipos de tramas bajo un único formato. Son las siguientes:

- **Tramas de información** (tramas-I). Estas tramas contienen un campo de datos no vacío que sirve para encapsular una unidad de información del protocolo de capa superior.
- **Tramas de supervisión** (tramas-S). El campo de datos de estas tramas está vacío, pues se utilizan exclusivamente para implementar diversas formas de acuse de recibo.

- **Tramas no numeradas** (tramas-U). Se utilizan para gestionar el enlace. Su campo de datos también está vacío.

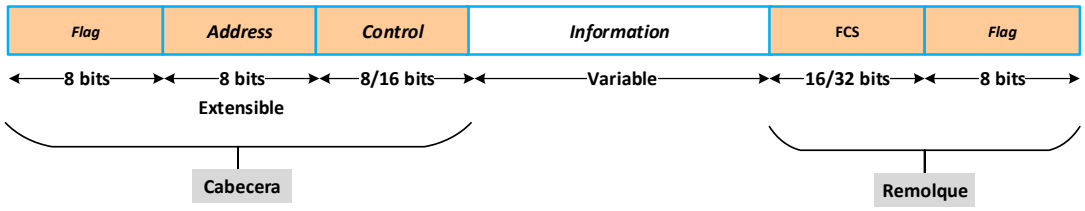


Figura 8.20. Formato común para los 3 tipos de tramas HDLC.

El formato general de las tramas HDLC se muestra en la **Figura 8.20**. El significado y uso de los diversos campos es el siguiente:

- ***Flag***. Es el patrón 01111110, cuya función es la sincronización a nivel de trama, es decir, actúa como delimitador de comienzo y final de trama. En HDLC es necesario señalizar la frontera de las tramas porque la longitud del campo de datos de las mismas es variable (si la trama HDLC tuviera un tamaño fijo, el receptor podría reconocer directamente todos los campos sin necesidad de delimitarla). Si una trama se transmite de forma aislada, es necesario que contenga el delimitador tanto al principio como al final de la misma; en cambio, si se transmiten varias tramas de forma seguida (***back-to-back***), el delimitador de final de una trama juega también el papel de comienzo de la siguiente.
- ***Address***. Este campo es especialmente significativo en la configuración no balanceada (aunque también, como veremos más adelante, tiene su uso en la configuración balanceada), en cuyo caso contiene siempre la dirección de una secundaria. Cuando la primaria genera la trama, se interpreta como dirección destinataria; cuando es la secundaria la que genera la trama, se interpreta como dirección origen. Su longitud es de un octeto, pero puede extenderse a un múltiplo de octetos en función del número de secundarias a gestionar. En tal caso, el primer bit de cada octeto sirve para señalizar si le sigue otro octeto a continuación (bit a 0) o si es el último del campo de dirección (bit a 1). El uso de un único campo de dirección (normal o extendido) explica el hecho de que no pueda existir comunicación entre secundarias.
- ***Control***. Este campo contiene varios sub-campos, y su longitud puede ser de 8 o 16 bits. Veremos más adelante su descripción.
- ***Information***. Es el campo de datos, que sólo está presente en las tramas-I. Como ya se apuntó anteriormente, contiene la unidad de información del protocolo de capa superior. Su longitud es variable y el estándar no especifica ninguna cota máxima para la misma.
- **FCS** (*Frame Check Sequence*). Es el campo de control de errores. Su longitud es de 16 bits (CRC-CCITT) o 32 bits (CRC-32) dependiendo de la cantidad de bits a proteger.

La **Figura 8.21** muestra el formato del campo de control, tanto en la versión normal (8 bits) como en la extendida (16 bits). El primer bit del campo de control es 0 en las tramas-I, y 1 en las tramas-S y las tramas-U. Estas dos últimas se distinguen por el segundo bit, 0 en las tramas-S y 1 en las tramas-U. El campo de control de las tramas-I incluye un sub-campo N(S) (de 3 bits en formato

normal y 7 bits en formato extendido) que permite la secuenciación de las mismas. Tengamos en cuenta que el diseño de HDLC contempla todas las variantes de las funciones de control de flujo y errores descritas en las Secciones 8.1 y 8.2, cuya correcta implementación requiere, como ya vimos, secuenciar las tramas de datos (tramas-I) y utilizar acuses de recibo numerados con el objeto de validar o no los flujos de tramas-I. En HDLC se definen 4 formas de acuse de recibo recogidas en sendas tramas-S, cuyo campo de control contiene un número de secuencia N(R) que siempre indica la siguiente trama-I que una estación espera recibir. Las 4 tramas-S se distinguen por los bits 3 y 4 del campo de control de las mismas, los cuales constituyen el sub-campo S (supervisión). Son éstas:

- **Trama RR** (*Receiver Ready*). Equivale a un ACK (positivo).
- **Trama RNR** (*Receiver Not Ready*). Equivale también a un ACK, pero al mismo tiempo indica a la estación que la recibe que detenga la transmisión de nuevas tramas-I. Por tanto, la trama RNR ejerce la doble función de control de errores y control de flujo.
- **Trama REJ** (*Reject*). Equivale a un NAK. La estación que recibe esa trama tiene que retransmitir la trama de información indicada por el acuse de recibo (trama-I cuyo N(S) es igual a N(R)), y todas las tramas de información que fueran transmitidas posteriormente (modalidad volver-atrás-N).
- **Trama SREJ** (*Selective Reject*). Equivale a un NAK de carácter selectivo. La estación que recibe esa trama sólo tiene que retransmitir la trama de información indicada por el acuse de recibo (modalidad rechazo selectivo).

La **Figura 8.21** también pone de manifiesto la presencia del sub-campo N(R) en el campo de control de las tramas-I, con las mismas dos opciones de longitud (3 en formato normal y 7 en formato extendido), y exactamente con el mismo significado que en el campo de control de las tramas-S. De esta manera se pueden aprovechar las tramas-I para transportar acuses de recibo referidos al flujo opuesto, evitando el consumo de recursos derivado del envío de tramas-S específicas. A esta operación de imbricación de información de control de un flujo de datos en el flujo de datos opuesto se la conoce como ***piggybacking***. En consecuencia, el uso de tramas-S queda relegado a la situación en que un receptor quiere dar acuse de recibo y no tiene trama-I por enviar.

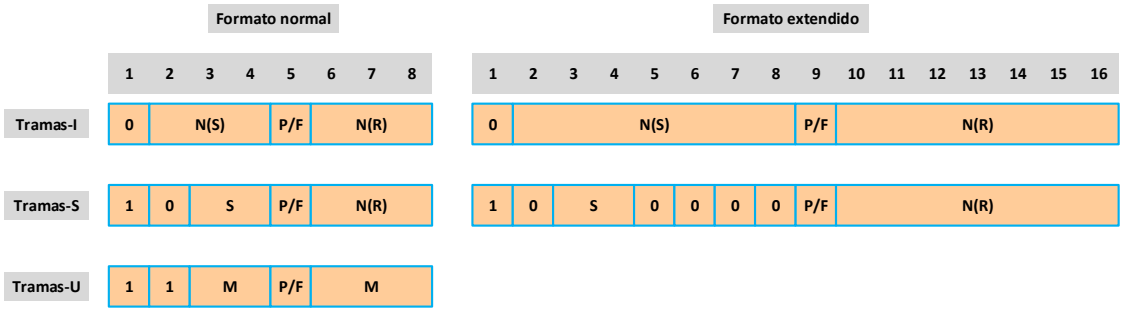


Figura 8.21. Formato normal y extendido del campo de control de los tres tipos de tramas HDLC. Los bits $5-8$ del campo de control en formato extendido de las tramas-S no se utilizan (están siempre fijados a 0).

Las tramas-U no contienen números de secuencia, de ahí su denominación. Por la misma razón, el campo de control de las mismas no admite extensión. Estas tramas ejecutan funciones relacionadas con la gestión del enlace. Precisamente, los 5 bits M del campo de control de las tramas-U sirven para codificar las distintas funciones (hasta un máximo de 32, aunque algunas combinaciones binarias no están definidas).

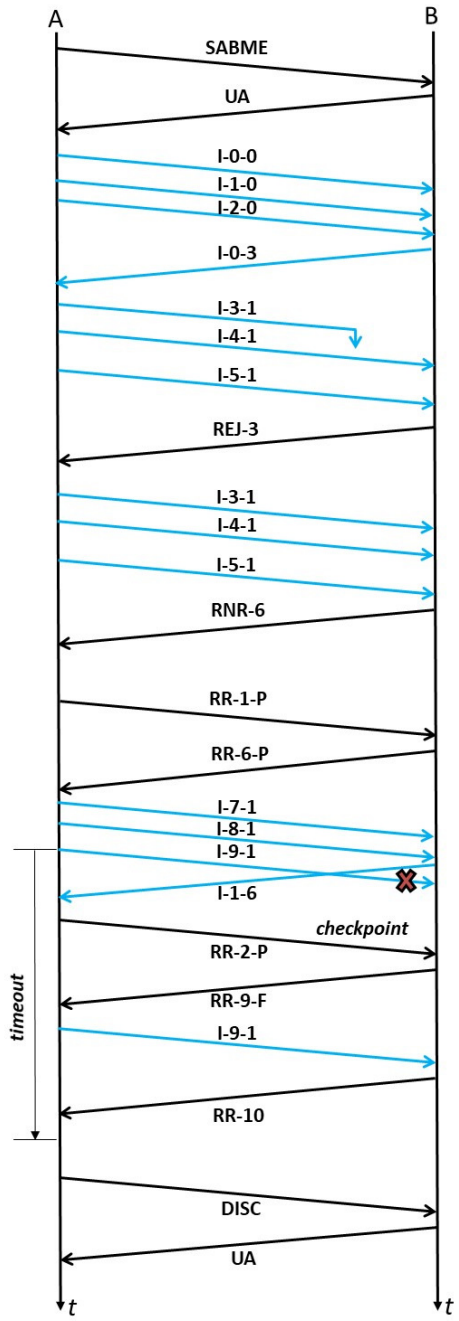


Figura 8.22. Ejemplos básicos de intercambios de tramas en HDLC.

Finalmente, todos los campos de control contienen el bit P/F (*Poll/Final*). Se trata de un bit con una doble función, y se activa siempre a 1. Su interpretación es muy sencilla en las configuraciones no balanceadas: es activado como bit P por la estación primaria cuando envía un comando a una secundaria, y es activado como bit F por una secundaria cuando envía la última trama de respuesta a un comando de la primaria. Pero también se utiliza en las configuraciones balanceadas, donde las estaciones, al ser combinadas, pueden jugar el doble papel. Así, supongamos una configuración balanceada de dos estaciones A y B, en la que A quiere forzar una respuesta de B. Para ello, A envía una trama a B con el bit P/F activado como bit P. En este momento, A ejerce el papel de primaria y B el de secundaria. La trama de respuesta de B, o la última de las tramas de respuesta de B, será recibida por A con el bit P/F activado como bit F. No obstante, en un intercambio de tramas simultáneo, B puede estar ejerciendo también el papel de primaria y A el de secundaria, lo cual llevaría a situaciones de ambigüedad en la interpretación del bit P/F si no se hiciera uso de alguna información de control adicional. Ahí es donde entra en juego el campo de dirección de las tramas, y su uso específico en las configuraciones balanceadas. Cuando la estación A envía una trama como comando (bit P/F activado como bit P), al igual que ocurre en las configuraciones no balanceadas, el campo de dirección de dicha trama contiene la dirección de B, pues en este momento A ejerce el papel de primaria y B el de secundaria. Por la misma razón, la trama de respuesta de B, con el bit P/F activado como bit F, contendrá su propia dirección (dirección de B). En un diálogo con los papeles intercambiados, las tramas contendrán la dirección de A. En definitiva, cuando una estación recibe una trama con el bit P/F activado, si la dirección que aparece en esa trama es la de la propia estación, significa que la trama es un comando, el bit P/F está activado como bit P, y la estación tiene que enviar una respuesta; por el contrario, si la dirección que aparece corresponde a la estación origen, la trama es una respuesta y el bit P/F está activado como bit F.

En la **Tabla 8.5** se recogen las tramas más importantes del protocolo HDLC, acompañadas de una breve descripción. La tabla también indica si la trama se utiliza siempre como comando (C), siempre como respuesta (R), o si puede funcionar como comando o respuesta (C/R).

La descripción más completa de un protocolo es mediante una máquina de estados finita; no obstante, nos limitaremos al diagrama temporal de la **Figura 8.22**, que describe algunos de los intercambios más representativos. En particular, el diagrama muestra el uso del bit P/F en configuración balanceada, y especialmente el uso de ese bit como elemento central del mecanismo básico de retransmisión en HDLC (*checkpoint*): cuando la estación A ha enviado un grupo de tramas de información ($7-9$ en el caso de la figura), puede enviar una trama RR con el bit P/F activado y la dirección de la estación B en el campo de dirección de la misma (por tanto, se trata de un comando) para forzar el acuse de recibo por parte de la estación B. De esta manera se aceleran las retransmisiones en caso de que sean necesarias.

Para concluir esta descripción del protocolo HDLC, la **Figura 8.23** muestra su encaje en la arquitectura TCP/IP, para lo cual resulta más adecuado en este caso el modelo de 5 capas de la misma. La razón es que el estándar HDLC no incluye ninguna especificación de capa física, pudiendo ser ejecutado sobre múltiples versiones de la misma. Por ejemplo, el protocolo Cisco HDLC se

distribuye como una componente software en los *routers* Cisco, constituyendo sólo una opción de configuración de los enlaces serie, además de otras tales como el protocolo PPP.

Tabla 8.5. Catálogo de las principales tramas HDLC.

Tipología	Denominación	C/R	Descripción
Trama-I		C/R	Encapsula una PDU de capa superior
Trama-S	RR (*Receiver Ready*)	C/R	Acuse de recibo positivo; preparado para recibir trama-I
	RNR (*Receiver Not Ready*)	C/R	Acuse de recibo positivo; no preparado para recibir más tramas-I
	REJ (*Reject*)	C/R	Acuse de recibo negativo; volver-atrás-N
	SREJ (*Selective Reject*)	C/R	Acuse de recibo negativo; rechazo selectivo
Trama-U	SNRM (*Set Normal Response Mode*)	C	Iniciar modo NRM con campo de control normal (números de secuencia de 3 bits)
	SNRME (*Set Normal Response Mode-Extended*)	C	Iniciar modo NRM con campo de control extendido (números de secuencia de 7 bits)
	SABM (*Set Asynchronous Balanced Mode*)	C	Iniciar modo ABM con campo de control normal (números de secuencia de 3 bits)
	SABME (*Set Asynchronous Balanced Mode-Extended*)	C	Iniciar modo ABM con campo de control extendido (números de secuencia de 7 bits)
	SARM (*Set Asynchronous Response Mode*)	C	Iniciar modo ARM con campo de control normal (números de secuencia de 3 bits)
	SARME (*Set Asynchronous Response Mode-Extended*)	C	Iniciar modo ARM con campo de control extendido (números de secuencia de 7 bits)
	DISC (*Disconnect*)	C	Terminar conexión lógica
	UA (*Unnumbered Acknowledgment*)	R	ACK no numerado (sólo utilizable como respuesta a tramas-U)
	DM (*Disconnected Mode*)	R	La estación está en modo desconectado
	UP (*Unnumbered Poll*)	C	Solicitud de información de control
	UI (*Unnumbered Information*)	C/R	Envío de información de control
	RSET (*Reset*)	C	Resetea la conexión (proceso de recuperación); N(S) y N(R) se ponen a 0
	XID (*Exchange Identification*)	C/R	Solicitud de estado/Informe de estado
	TEST (*Test*)	C/R	Intercambio de campos de información idénticos para test de la conexión
	FRMR (*Frame Reject*)	R	Acuse de recibo de trama inaceptable

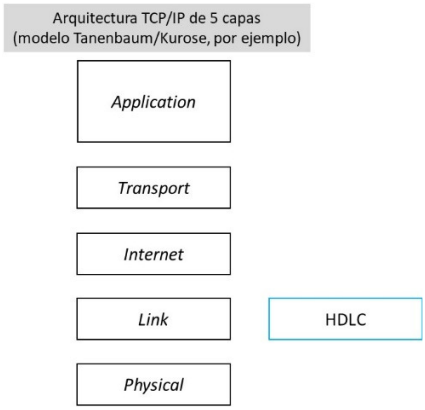


Figura 8.23. Encaje del protocolo HDLC en la arquitectura TCP/IP.