VNIVERSITAT DE VALÈNCIA
Escola Tècnica Superior d'Enginyeria
Departament d´Informàtica

DEPARTAMENTO DE INFORMÁTICA
CURSO 2011-2012

Ingeniería Informática
Telemática y Sistemas de Transmisión de Datos

Tema 7

La capa de enlace y las redes de área local

Prof. Juan Manuel Orduña Huertas

Índice general

1. Introducción: problemática de acceso al medio **1**

2. Protocolos de red de área local **4**

2.1. CSMA persistente y no persistente 4

2.1.1. CSMA 1-persistente 5

2.1.2. CSMA no persistente 5

2.1.3. CSMA p-persistente 6

2.2. CSMA con detección de colisiones 6

2.3. Protocolos de reserva 9

2.3.1. Protocolo de mapa de bits 9

2.3.2. Protocolo de cuenta atrás binaria 10

3. Norma IEEE 802 para redes de área local **12**

3.1. Norma IEEE 802.3 y Ethernet 15

3.1.1. Subcapa MAC de la 802.3 15

3.2. Subcapa MAC de la IEEE 802.3z (Gigabit Ethernet) 18

3.3. NORMA IEEE 802.11 (Wifi) 19

4. La subcapa de enlace **23**

4.1. Parámetros de diseño de la capa de enlace 23

4.1.1. Entramado 24

4.1.2. Detección de errores 26

4.1.3. Control de flujo 27

4.2. Detección de errores 27

4.2.1. Circuitos de detección de errores 30

4.3. Protocolos elementales de la capa de enlace 32

4.3.1. Protocolo símplex no restringido 36

4.3.2. Protocolo símplex de parada y espera (stop & wait) 38

4.3.3. Protocolo símplex para un canal con ruido 39

4.4. Protocolos de ventana deslizante 44

4.4.1. Protocolo de ventana deslizante de un bit 47

4.4.2. Protocolo de ventana deslizante de retroceso N 47

4.5. Ejemplos de protocolos de enlace de datos 58

4.5.1. HDLC-Control de Enlace de Datos de Alto Nivel 58

4.5.2. La capa de enlace de datos en Internet: Protocolo Punto a Punto (PPP) 62

4.5.3. El nivel de enlace en Frame Relay 66

Bibliografía **68**

1 INTRODUCCIÓN: PROBLEMÁTICA DE ACCESO AL MEDIO

Una vez estudiado todo lo relacionado con el nivel físico, incluidos los medios de transmisión y los protocolos que se utilizan para comunicaciones punto a punto (el sistema telefónico y las comunicaciones serie son sistemas punto a punto), en este tema vamos a estudiar cómo se organiza el acceso al medio de transmisión en sistemas multidifusión, que es el sistema que utilizaban las primeras LAN (IEEE 802.3 con normas 10Base-5 y 10Base-2). Como ya hemos visto en el tema 6, el resto de estándares basados en la IEEE 802.3 decidieron mantener la compatibilidad a nivel de acceso al medio y capa de enlace con este estándar, y por tanto todas suponen un medio físico multidifusión.

Por sistemas multidifusión designamos a aquel sistema donde varias máquinas comparten un único canal de transmisión, de forma que cuando una máquina transmite información el resto de máquinas recibe físicamente la señal. Por tanto, en los sistemas multidifusión sólo una máquina puede estar transmitiendo a la vez, o de lo contrario se interfieren ambas señales, en lo que denominaremos colisión. La forma en cómo se organizan las máquinas para conseguir que sólo una de ellas transmita información en un momento dado es lo que veremos en este tema, así como los protocolos existentes para las redes de área local. Todos los protocolos que veremos en este tema se consideran como del nivel 2, puesto que la capa 2 del modelo ISO/OSI se encargaba de proporcionar una comunicación fiable, y el acceso al medio es necesario para una comunicación fiable.

Para explicar la problemática del acceso al medio podemos imaginar que un sistema multidifusión es análogo a un coloquio de varias personas en una habitación. Para que lo que hablan sea inteligible, sólo una de las personas puede hablar en cada instante. Sin embargo, en un coloquio las personas pueden pedir el turno de la palabra (por ejemplo levantando la mano antes de hablar) pero en los sistemas multidifusión generalmente el canal es único, sin canales de control para gestionar el accesso al canal. Con estas carac-

terísticas, lo más probable es que cuando alguien que esta hablando deja de hacerlo, si dos o más personas tienen algo que decir, hablen a la vez, haciendo inteligible la conversación y debiendo empezar a hablar de nuevo. Además en un coloquio siempre hay un moderador que es el que asigna la palabra. Sin embargo en los sistemas multidifusión normalmente el acceso al medio debe controlarse de forma distribuida, ya que cualquier estación puede desconectarse en cualquier momento.

Para solucionar el problema de qué estación accede al canal en cada instante existen dos estrategias:

- **Reparto estático (síncrono) del canal:** Consiste en multiplexar el canal en la frecuencia o en el tiempo (N subcanales con un ancho de banda de N veces menor que el ancho de banda del canal original o N ranuras de tiempo), asignando un subcanal o ranura de tiempo a cada estación de forma fija. Sin embargo, esta estrategia solo sirve cuando el número de usuarios (estaciones) conectados al canal es pequeño (N muy grande implica ranura de tiempo o subcanales muy pequeños) y **fijo**. Sin embargo, en un sistema donde una estación puede desconectarse o apagarse en cualquier instante, o donde el tráfico generado va a ráfagas, esta estrategia es ineficiente, ya que cuando una estación no tiene datos para transmitir el ancho de banda asignado a ella se pierde.
- **Reparto dinámico (asíncrono) del canal:** El canal se asigna a una estación u otra dependiendo de parámetros temporales y teniendo en cuenta solo las estaciones que desean transmitir algo.

Antes de entrar a explicar los métodos de reparto del canal de transmisión, vamos a definir unos conceptos básicos:

Trama (marco): Unidad de control de flujo de la capa de enlace. Se compone de una serie de bytes que definen unos campos de información determinados, que veremos a lo largo del tema. Cuando una estación gana el acceso al medio, transmite al menos un número entero de tramas.

Medio compartido: Todas las estaciones pueden transmitir y recibir del canal o medio compartido. Lo que cualquier estación transmite puede ser escuchado por todas las demás.

Ranuras: Supondremos o bien que el tiempo es una magnitud contínua, en cuyo caso una estación puede comenzar a transmitir en cualquier instante, o bien que el tiempo se divide en “ranuras” o intervalos discretos. En ese caso, la transmisión de una trama comienza siempre al principio de una ranura.

Colisión: Alteración de las señales producida por la transmisión simultánea de 2 ó más tramas procedentes de distintas estaciones en un medio compartido. Cuando se produce una colisión **todas** las tramas involucradas en la colisión deben ser transmitidas de nuevo, puesto que ninguna llegó a su destino correctamente.

2 PROTOCOLOS DE RED DE ÁREA LOCAL

Existe una gran variedad de protocolos de acceso múltiple (protocolos de acceso al medio para canales multidifusión) asíncronos. Este tipo de protocolos se pueden clasificar en tres grupos:

De contienda: No se realiza un control para determinar quién tiene el turno de transmisión, sino que todas las estaciones con datos para transmitir compiten por el acceso al medio. Son técnicas de control inherentemente distribuidas muy eficientes cuando el tráfico va a ráfagas. Los ejemplos más representativos son los protocolos CSMA.

Reserva: Dividen el tiempo en ranuras. Cuando una estación desea transmitir, reserva futuras ranuras de tiempo. Estas técnicas son adecuadas para un tráfico contínuo.

Rotación circular: Cada una de las estaciones que tienen datos para transmitir lo hacen por riguroso turno durante un tiempo limitado. Cuando la que tiene el turno agota su tiempo, cede el curno a la siguiente estación en la secuencia lógica. Esta técnica es eficiente cuando varias estaciones tienen datos para transmitir durante un largo periodo de tiempo. Los ejemplos más extendidos son los protocolos de paso de testigo.

2.1. CSMA persistente y no persistente

Los protocolos CSMA (del inglés *Carrier Sense Multiple Access*, acceso múltiple con detección de portadora), como su propio nombre indica, incorporan todos ellos detección de portadora. De estos, existen distintas variantes, que vamos a estudiar a continuación:

2.1.1. CSMA 1-persistente

El algoritmo de este protocolo es el siguiente: cuando una estación tiene datos para transmitir, “escucha” el canal (con el término “escucha” indicamos que la estación comprueba si existe señal portadora en el canal). Si el canal está libre (no hay portadora), entonces transmite una trama. Si el canal está ocupado, entonces la estación espera hasta que la portadora desaparece. En cuanto detecta que el canal está en reposo (sin portadora), entonces transmite una trama. Si se produce una colisión, la estación espera una cantidad aleatoria de tiempo y comienza de nuevo el algoritmo. Este algoritmo se denomina 1-persitente porque la estación transmite con probabilidad 1 cuando el canal está inactivo.

El parámetro más importante que determina el rendimiento de este protocolo es el **retardo de propagación** del canal. El rendimiento ideal sería que todas las tramas fuesen transmitidas y que no ocurriera ninguna colisión. Sin embargo, existe la posibilidad de que justo después de que una estación A comienze a transmitir, otra estación B escuche el canal para transmitir si lo encuentra libre. Si la señal portadora de A no se ha propagado aún hasta la estación B, entonces B detectará el canal como libre y transmitirá su trama, produciéndose una colisión. Por ello, cuanto mayor sea el retardo de propagación, mayor probabilidad de que esto ocurra, con lo que el rendimiento del protocolo decrecerá.

Hay que hacer notar que aunque el retardo de propagación sea nulo, existirán colisiones. Supongamos que dos estaciones A y B tienen tramas listas para ser transmitidas mientras una tercera estación C está ya transmitiendo por el canal. En cuanto C deje el canal libre, A y B comenzarán a transmitir simultáneamente, según el algoritmo descrito, produciendo una colisión.

2.1.2. CSMA no persistente

En este protocolo se intenta que las estaciones sean menos agresivas a la hora de conseguir el acceso al canal. Si una estación tiene datos para enviar, escucha el canal. Si este está libre, entonces la estación comienza a transmitir. Pero si el canal está ocupado, no lo escucha de manera contínua para tomarlo en cuanto acabe la transmisión actual, sino que espera un tiempo aleatorio y comienza de nuevo el algoritmo. Por tanto, este algoritmo consigue una mejor utilización del canal, aunque introduce mayores retardos en el envío de las tramas.

2.1.3. CSMA p-persistente

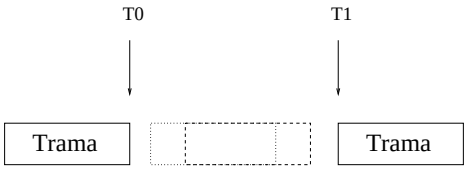
Este protocolo se aplica sólo a sistemas con el tiempo ranurado. Cuando una estación tiene datos para transmitir, escucha el canal. Si el canal está libre, la estación transmite con una probabilidad p y se espera hasta la siguiente ranura con una probabilidad $q = 1-p$. Si en esa ranura el canal también está libre, la estación transmite o se espera nuevamente con probabilidades p y q, respectivamente. Este proceso se repite hasta que la trama se transmite o hasta que el canal está ocupado. En este último caso, la estación espera un tiempo aleatorio y comienza de nuevo el algoritmo. Si la estación detecta el canal ocupado inicialmente, entonces espera hasta la siguiente ranura y aplica el algoritmo anterior.

2.2. CSMA con deteccción de colisiones

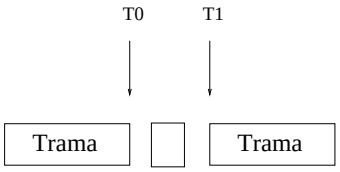
Este protocolo mejora las prestaciones conseguidas por los protocolos CSMA haciendo que las estaciones detengan la transmisión de sus tramas en cuanto detecten que ha habido una colisión, ya que no tiene sentido seguir transmitiendo si ya saben que esa trama debe ser transmitida de nuevo. La detención de la transmisión ahorra por tanto tiempo y ancho de banda del canal.

La Figura 2.1 muestra la diferencia entre una colisión en un sistema con protocolo CSMA y una colisión en un sistema CSMA/CD. En el primero, aunque la colisión es detectada, hasta que la última trama involucrada en la colisión no es transmitida totalmente, no puede transmitirse una trama válida por el canal. En el segundo caso, en cuanto se detecta la colisión todas las tramas involucradas en ella dejan de ser transmitidas, con lo que el instante $T1$ (el comienzo de la siguiente trama válida tras una colisión) ocurre mucho antes. Por tanto, la utilización del canal es mayor.

En el protocolo CSMA/CD el medio compartido puede estar en uno de los siguientes estados: periodo de transmisión, donde una estación está transmitiendo datos, periodo de contienda de 2 o más estaciones por el medio, y periodo de inactividad, donde ninguna estación desea transmitir por el canal compartido. La figura 2.2 ilustra una secuencia temporal con los 3 estados. En el punto marcado como t_0 una estación ha terminado de transmitir su trama, y cualquier estación que tenga datos para transmitir puede intentar hacerlo. Si 2 o más estaciones lo intentan, se produce una colisión. Las colisiones se detectan comparando la potencia o el ancho de pulso de la señal transmitida con el de la señal recibida. Al detectar la colisión, las estaciones involucradas detienen la transmisión y esperan un



a) Colision CSMA



b) Colision CSMA/CD

Figura 2.1: *Gráfica temporal de una colisión con protocolo CSMA (a) y con protocolo CSMA/CD (b)*

tiempo aleatorio antes de intentarlo de nuevo (suponiendo que que ninguna otra estación lo ha conseguido en ese lapso, pero nada impide que ocurra otra colisión. Así, el modelo CSMA/CD consistirá en periodos alternantes de contienda y transmisión, ocurriendo periodos de inactividad cuando ninguna estación tenga datos para transmitir.

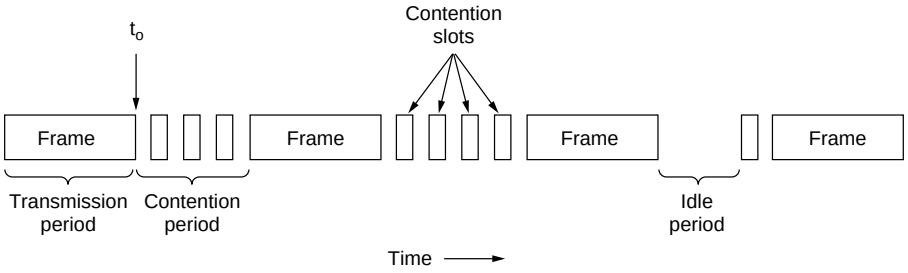


Figura 2.2: *Gráfica temporal con distintos estados del canal*

El objetivo de los protocolos es reducir al máximo el periodo de contienda. Sin embargo, el parámetro que nos va a medir el periodo de contención mínimo es el tiempo mínimo que dos estaciones cualesquiera van a tardar en darse cuenta de que sus tramas han colisionado. Este tiempo va a depender del retardo de propagación del canal, y vendrá dado por el retardo de propagación τ de las dos estaciones más alejadas, ya que ese es el peor de los casos.

En principio se podría pensar que si cualquier estación puede estar segura de haber ganado el acceso al medio si τ segundos después de comenzar a transmitir su trama, no ha detectado una colisión. No obstante, puede darse el caso que muestra la Figura 2.3

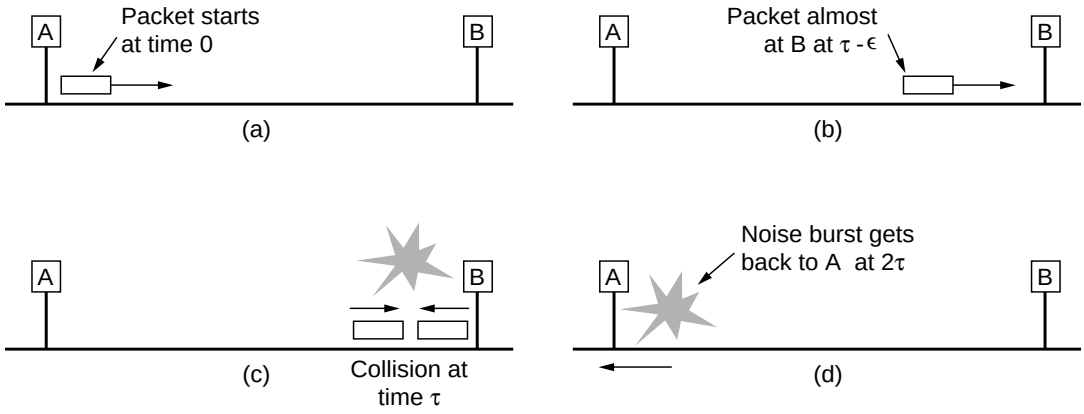


Figura 2.3: *Detección de una colisión en el peor de los casos.*

En este supuesto, la estación A comienza a transmitir una trama en el instante T_0(figura a). Después de un periodo $\tau - \varepsilon$, donde $\varepsilon \rightarrow 0$, la trama está a punto de llegar a la estación de destino B (figura b). En ese instante (figura c) la estación B escucha el canal, y como aún no le ha llegado el comienzo de la trama que envió A, entiende que el canal está libre y comienza a transmitir su trama, con lo que en el momento τ se produce la colisión. No obstante, la estación A aún no ha podido detectar la colisión. Sólo detectará la colisión (figura d) cuando la ráfaga de ruido producida por la interferencia de las dos tramas se propague hasta ella. Este retardo de propagación será de $\simeq \tau$, con lo que una estación necesita 2τ segundos para estar completamente segura de que ha ganado el acceso al medio.

De esta forma, en un protocolo CSMA/CD para que una estación sepa que una trama suya ha provocado una colisión, la estación debe de estar transmitiendo aún su trama cuando detecte la colisión. En caso contrario, la estación no tiene forma de saber si la colisión ha sido provocada por tramas emitidas por otras estaciones. Y para estar segura de que si existe colisión esta se detectará mientras la trama se está transmitiendo, la longitud de la trama debe ser tal que tarde más tiempo en ser transmitida que el doble del retardo de propagación del canal utilizado.

Otra consecuencia muy importante es que como toda estación debe monitorear el canal en busca de ruido que pueda indicar una colisión al mismo tiempo que transmite, en

CSMA/CD todos los canales son inherentemente semidúplex (los circuitos de recepción se usan cuando se transmite).

2.3. Protocolos de reserva

Las colisiones afectan negativamente a las prestaciones de la red, especialmente cuando el canal de transmisión tiene un retardo de propagación alto y las tramas tienen corta longitud. Por ello se han desarrollado protocolos que resuelven la contienda por el canal sin que se produzcan colisiones. No obstante, lo que se consigue con estos protocolos es sólamente **acotar** el periodo de contienda, aunque no eliminarlo. Sin embargo, el acotamiento del periodo de ocntienda puede ser vital para los sistemas distribuidos en tiempo real.

En los protocolos que vamos a ver supondremos que hay N estaciones, cada una con una dirección única de 0 a $N-1$ incorporada al hardware.

2.3.1. Protocolo de mapa de bits

Con este protocolo cada periodo de contienda consiste exactamente en N ranuras de tiempo, tal como muestra la Figura 2.4. Si la estación 0 tiene una trama para enviar, entonces transmitirá un bit “1” durante la ranura 0. No está permitido a ninguna otra estación transmitir durante esa ranura. Igualmente, e independientemente de lo que haga la estación 0, la estación 1 es la única que puede poner transmitir en la ranura 1. Dicha estación pondrá un bit “1” si tiene una trama para transmitir, y un bit “0” si no tiene datos para transmitir. En general, la estación j puede anunciar que tiene una trama para transmitir poniendo un bit “1” en la ranura de tiempo j, $0 \leq j \leq N-1$. Una vez transcurridas las N ranuras, cada estación tiene conocimiento de cuáles son las estaciones que desean transmitir. En ese momento, las estaciones comienzan a transmitir por orden numérico (Figura 2.4). Este tipo de protocolos en los que las estaciones “anuncian” su deseo de transmitir se denominan **protocolos de reserva**.

Como todas las estaciones estan de acuerdo en el orden en el que deben transmitir, no existen colisiones. Una vez que la estación con mayor numeración haya transmitido su trama , comienza otro periodo de contienda de N bits.

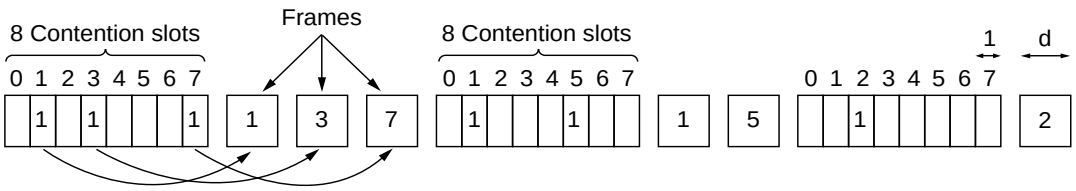


Figura 2.4: *Protocolo de mapa de bits.*

2.3.2. Protocolo de cuenta atrás binaria

Uno de los problemas que presenta el protocolo de mapa de bits es la sobrecarga de N bits extra que introduce en la red en cada periodo de contienda. Por ello se desarrolló este protocolo. En él, cada estación tiene una dirección binaria. Cada estación que desea usar el canal difunde su dirección como una cadena de bits, comenzando por el bit de mayor peso. Todas las estaciones tienen una dirección con el mismo número de bits. A los bits en cada posición de la dirección que las estaciones van transmitiendo al canal, se les aplica una OR lógica (simplemente usando una codificación tal que el “0” lógico sea tensión nula podemos tener ya la OR lógica). Cuando una estación comprueba que en una posición de bit de orden alto su bit es “0” y al hacer la OR lógica en el canal queda un “1”, se da por vencida y no continúa poniendo sus bits de menor peso.

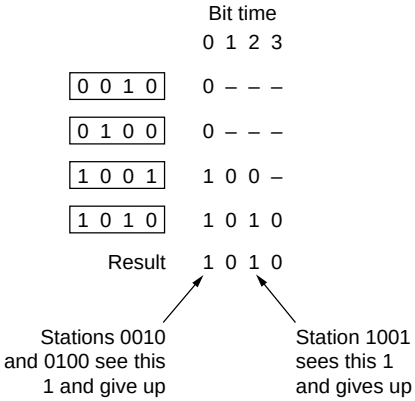


Figura 2.5: *Cuenta atrás binaria (guiones significan silencio).*

La Figura 2.5 muestra un ejemplo de este protocolo. En este ejemplo las direccio-

nes son de 4 bits, y las estaciones que tienen tramas para enviar son las estaciones 0010, 0100, 1001 y 1010. En el periodo correspodiente al primer bit, las estaciones pondrán en el canal 0, 0, 1 y 1, respectivamente. La operación OR lógica de estos bits resulta en un “1”. Por tanto, las estaciones 0010 y 0100 saben que una estación con una dirección mayor está compitiendo por el canal, y se dan por vencidas. En el segundo bit de mayor peso, estas estaciones ya no emiten nada, sólo emiten las estaciones 1001 y 1010. Ambas ponen un bit “0” en el canal, y como la OR lógica resultante es tambien un “0”, ambas estaciones emiten su tercer bit más significativo. En este, la estación 1001 emite un “0” y el resultado de la Or lógica es un “1”. Por tanto, sólo la estación 1010 emite su cuarto bit, resultando la ganadora del acceso al medio (es la que tiene la dirección mayor de todas las que desean acceder al medio).

3 NORMA IEEE 802 PARA REDES DE ÁREA LOCAL

En este capítulo del tema vamos a ver cómo se aplican en las redes de área local los protocolos que hemos visto de forma teórica en el capítulo anterior. Estos están definidos en una normativa que desarrolló la IEEE (ver *http://www.ieee.org*), y que se conoce como norma **IEEE 802**. Esta norma recoge todos los estándares para redes de área local.

El modelo de referencia de la IEEE 802 separa la capa de enlace definida en el modelo ISO en 2 subcapas. La capa superior se denomina LLc (del inglés *Logical Link Control*), mientras que la subcapa inferior la denomina subcapa MAC (del inglés *Medium Access Control*). Esta última subcapa es la que se encarga de la gestión del acceso al medio. Así, las redes de área local se rigen por este esquema, que define tanto la capa de acceso al medio como las especificaciones del nivel físico de las redes de área local. La figura 3.1 muestra este modelo de referencia.

Es muy importante destacar que los diferentes estándares difieren en la capa física y en la subcapa MAC (subcapa de acceso al medio), pero son totalmente compatibles en la subcapa superior de la capa de enlace de datos (capa 2 del modelo ISO). La normativa IEEE 802 consta de múltiples normas, puesto que la IEEE tiene múltiples comités de trabajo que van desarrollando cada estándar. Los más destacados son los siguientes:

- 802: Arquitectura y cuestiones generales
- 802.1: Aspectos comunes: puentes, gestión, redes locales virtuales, etc.
- 802.2: Logical Link Control (LLC)
- 802.3: Redes CSMA/CD (Ethernet)
- 802.4: Redes Token-Passing Bus

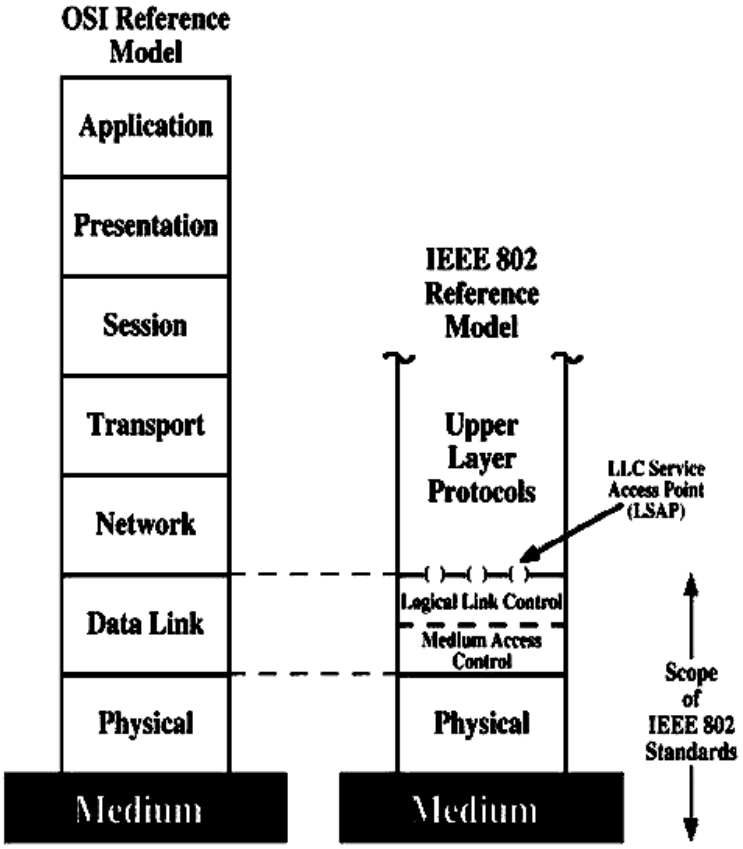


Figura 3.1: *Modelo de referencia IEEE 802*

- 802.5: Redes Token Ring
- 802.6: Redes MAN DQDB (Distributed Queue Dual Bus), actualmente en hibernación e inactivo
- 802.7: Grupo asesor en redes de banda ancha
- 802.8 Grupo asesor en fibras ópticas
- 802.9: Redes de servicios Integrados (Iso-Ethernet)
- 802.10: Seguridad en LAN/MAN
- 802.11: LANs inalámbricas
- 802.12: Redes Demand Priority (100VG-AnyLAN)
- 802.14: Redes de TV por cable
- 802.15: Redes Bluetooth

Los grupos de trabajo 802 no son algo estático; continuamente se están planteando para su estandarización nuevas técnicas y protocolos, nuevos medios físicos, etc. Cuando surge una nueva propuesta el grupo de trabajo correspondiente nombra un grupo de estudio que la analiza, y si el informe es favorable se crea un 'subgrupo' de trabajo (llamado oficialmente proyecto) que eventualmente propone una adenda al estándar para su aprobación. Los proyectos se identifican por letras añadidas al grupo de trabajo del que provienen. A título de ejemplo detallamos a continuación algunos de los proyectos más relevantes del comité 802:

- 802.1D: puentes transparentes
- 802.1G: puentes remotos
- 802.1p: Filtrado por clase de tráfico (Calidad de Servicio)
- 802.1Q: Redes locales virtuales (VLANs)
- 802.3u: Fast Ethernet
- 802.3x. Ethernet Full dúplex y control de flujo
- 802.3z: Gigabit Ethernet

- 802.3ab: Gigabit Ethernet en cable UTP-5
- 802.3ad: Agregación de enlaces

El 802 describe aspectos generales y arquitectura. El 802.1 estandariza aspectos que son comunes a todas las LANs, como son la gestión y el funcionamiento de puentes. El 802.2 describe la subcapa LLC (Logical Link Control), también común a todas las redes 802. La mayoría de los demás grupos de trabajo (802.3, .4, .5, .6, .9, .11 y .12) tienen que ver con diversas tecnologías de red local. Cada uno de ellos especifica el nivel físico y la subcapa MAC correspondiente. Por ejemplo el estándar 802.3 describe el nivel físico y el subnivel MAC de la red con protocolo MAC CSMA/CD, mas conocida como Ethernet. Dada la cantidad de diferentes tecnologías existentes, en este tema vamos a explicar sólo las ya vistas en el Tema 6.

3.1. Norma IEEE 802.3 y Ethernet

Esta norma define un estándar CSMA/CD 1-persistente. Se confunde a menudo con la norma **Ethernet**, aunque no es la misma. La norma Ethernet fué dessarrollada por la compañía Xerox e inicialmente definía un protocolo CSMA/CD de 2,94 Mbps para conectar más de 100 estaciones a un cable de 1 km. Este sistema tuvo tanto éxito que la Xerox, DEC e Intel diseñaron un estándar para una Ethernet a 10 Mbps. Este estándar formó la base de la norma 802.3. Al extenderse tanto las redes basadas en la norma 802.3, se han extendido con el nombre de Ethernet.

3.1.1. Subcapa MAC de la 802.3

La norma 802.3 define una estructura de **trama** para encapsular la información que circula por la capa física. De esta forma, cada vez que el computador transmite información por la capa física no se transmite un flujo contínuo de bits, sino una serie de tramas, cada una de las cuales está compuesta por los campos que aparecen en la Figura 3.2.

La Figura 3.2 muestra 2 formatos de trama: la estructura original de DIX (DEC, Intel, Xerox, que fueron los que definieron Ethernet) y el formato de trama definido en el IEEE 802.3.

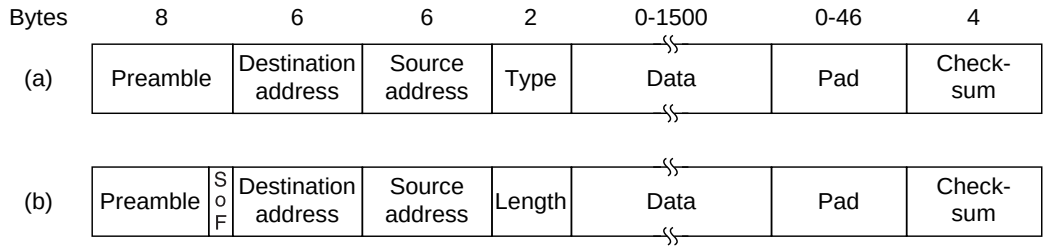


Figura 3.2: *Formatos de trama a) Ethernet DIX b) 802.3.*

En la trama DIX, cada trama se inicial con un **Preámbulo** de 8 bytes de longitud, todos ellos con el patrón “10101010”. La codificación Manchester de estos bits produce una onda cuadrada de 10 MHz de frecuencia durante 6,4 μseg. para que el reloj del receptor se sincronize con el del emisor.

A continuación vienen dos direcciones, una para el **destino** y una para el **origen**. Aunque se permiten direcciones de 2 y de 6 bytes, para transmitir a 10 Mbps ya sólo se usaban direcciones de 6 bytes. En la actualidad consiste en una dirección de 6 bytes que identifica de manera unívoca la tarjeta de red. Esta dirección viene asignada de fábrica. De los 6 bytes de dirección, cuando el más significativo (el bit 47) está a “1” significa que la trama es **multicast** (va dirigida a más de una estación). Cuando los 6 bytes son todos “1” entonces significa que la trama es una trama **broadcast**, es decir, va dirigida a todas las estaciones de la red. Por ello, este bit también se denomina bit Ï/G”(de Individual/Grupo)

Además, el bit de orden 46 se emplea para distinguir las direcciones locales de las direcciones globales. Las direcciones locales son asignadas por el administrador de la red, mientras que las globales son asignadas por el IEEE para asegurar que no haya más de dos direcciones iguales en todo el mundo. La capa superior, la de red, es la que se encarga de encontrar el camino adecuado para llegar hasta el destino. Por ello, este bit también se denomina bit L/G (de Local/Global) ó U/L (Universal/Local).

Hay que destacar que esta dirección de red es distinta de lo que se conoce como “dirección internet”, “dirección IP” o simplemente “dirección de la estación”. Una estación determinada tendrá una dirección ethernet como la del ejemplo anterior y si además utiliza el protocolo TCP/IP entonces tendrá además una dirección IP, que tiene el formato “147.156.1.4”. No obstante, el protocolo TCP/IP y los distintos protocolos de red pertenecen a los niveles ISO de red y de transporte, y por tanto este tema será estudiado en la asignatura de cuarto curso “Redes de computadores”.

A continuación viene el campo de **Tipo** o *Ethertype*, que indica al receptor qué hacer con la trama. Hay que tener en cuenta que es posible utilizar múltiples protocolos de la capa de red al mismo tiempo en la misma máquina, por lo que cuando llega una trama Ethernet el kernel del sistema oprativo debe saber a cuál hay que entregarle la trama. Así pues, gracias a este campo la red Ethernet puede soportar múltiples protocolos de red.

A continuación viene el campo de **Datos**, que puede contener de 0 a 1500 bytes. Este límite se eligió de forma arbitraria cuando se definió el estándar DIX, principalmente porque un transceptor necesita suficiente memoria RAM para mantener toda la trama, y en 1978 la RAM era muy cara.

Además también hay una longitud mínima del campo de datos. por eso existe el **Campo de relleno en inglés, *PAD***. La norma 802.3 utiliza el protocolo CSMA/CD con detección de colisión, en el que una estación para de transmitir en cuanto detecta una colisión. Como ya hemos visto, este protocolo requiere que la trama tenga un tamaño mínimo igual al doble del retardo de propagación del canal. De lo contrario, se puede producir una colisión cuando el emisor ya ha acabado de transmitir, y por tanto el emisor no se enterará de que su trama produjo colisión. Para él la transmisión fué correcta porque mientras él transmitió no hubo colisión. Por tanto, no intentará enviarla de nuevo.

Para evitar este caso, el tiempo mínimo que una estación debe estar transmitiendo una trama (el doble del retardo de propagación de una red 10Base-5 de 5 segmentos de 500 metros con 4 repetidores, y transmitiendo a 10 Mbps) es aproximadamente de 50 microsegundos en el peor de los casos. A 10 Mbps. un bit tiene un periodo de 100 nseg. y por tanto 500 bits es el tamaño de trama más pequeño. Para agregar un margen de seguridad, se redondeó el tamaño a 512 bits, es decir, 64 bytes. Ello quiere decir que cualquier trama Ethernet válida debe tener 64 bytes desde el preámbulo hasta el checksum. Como el campo de datos puede tener 0 bytes, y el resto de campo suman como minimo 14 bytes (originalmente la dir. origen y destino podían tener 2 bytes, sumando 8 del preámbulo, 4 de las 2 direcciones y 2 del tipo = 14 bytes), el campo de relleno es un campo que puede tener desde 0 a 46 bytes, dependiendo de la longitud del campo de datos y del formato de las direcciones origen y destino.

Cuando el doble del tiempo de propagación es **mayor** que la longitud mínima de la trama en bits, se puede producir lo que se conoce como **Colisiones tardías:** cuando la colisión se detecta despues de que se hayan transmitido los primeros 64 bytes de la trama.

Por otra parte, cuando una misma trama genera más de una colisión al intentar ganar el medio a este hecho se le conoce como **múltiples colisiones**. Llevado a un caso peor, se

denomina **excesivas colisiones** al hecho de que una trama haya generado 16 colisiones consecutivas en sus primeros 16 intentos de acceso al medio. Todos estos datos se pueden comprobar en un monitor de la red.

El campo final de una trama Ethernet es el campo de verificación de trama o **Checksum**. Es el código detector de errores (Código Cíclico Redundante o CRC) asociado a cada trama. El funcionamiento de este código se estudia más adelante.

Cuando la IEEE estandarizó Ethernet hizo 2 cambios al formato DIX, como aparece en la Figura 3.2. Primero, redujo el preámbulo de 8 a 7 bytes, y utilizó el último byte del preámbulo para un delimitador de inicio de trama. Y segundo, cambió el campo tipo por el campo longitud del campo de datos. Al eliminar el campo tipo el receptor ya no podía de qué tipo era la trama, pero eso se resolvió usando los primeros bits del campo de datos . Pero eso se detalla en el control lógico del enlace (LLC) más adelante.

3.2. Subcapa MAC de la IEEE 802.3z (Gigabit Ethernet)

Gigabit Ethernet utiliza el mismo formato de trama y el mismo protocolo de acceso al medio (CSMA/CD) que los estándares 802.3 y 802.3u (Ethernet y Fast Ethernet). No obstante, introduce 2 mejoras:

Extensión de la portadora: Consiste en añadir una serie de símbolos al final de la trama MAC de tal forma que la trama resultante tenga una duración equivalente a 4.096 bits (mucho mayor que los 512 bits (64 bytes) exigidos en los estándares 802.3 y 802.3u. El objetivo es que la longitud de la trama sea mayor que el el doble del tiempo de propagación cuando se transmite a 1 Gpbs.

Ráfagas de tramas: Consiste en permitir la transmisión consecutiva de varias tramas cortas sin necesidad de dejar el control del CSMA/CD. De esta forma se evita la redundancia y gasto que supone la extensión de la portadora. Obviamente, esta mejora sólo se puede aplicar en el caso de que una estación tenga preparadas para transmitir varias tramas pequeñas.

Gigabit Ethernet define un concentrador compartido o conmutado. En el concentrador compartido sí se utilizan las mejoras indicadas, pero en el concentrador conmutado que cada segmento es un medio dedicado, y no se utilizan dichas mejoras. De hecho, no hay que luchar para acceder al medio compartido.

3.3. NORMA IEEE 802.11 (Wifi)

Típicamente una LAN inalámbrica está formada por un conjunto de estaciones base, unidas entre sí por algún tipo de cable, y una serie de estaciones móviles que comunican con la estación base más próxima. El conjunto de estaciones base forma en realidad un sistema celular en miniatura.

La transmisión se realiza mediante ondas electromagnéticas no guiadas y se utiliza un protocolo CSMA. El alcance de cada estación es limitado. Este conjunto de características permiten optimizar el acceso al medio, ya que se pueden producir situaciones como las que se describen a continuación:

Supongamos cuatro estaciones A, B, C y D situadas en línea y separados 10 metros cada una de la siguiente:

$$\text{A} <\!\!—_{10m.}\!\!—> \text{B} <\!\!—_{10m.}\!\!—> \text{C} <\!\!—_{10m.}\!\!—> \text{D}$$

Supongamos también que el alcance máximo de cada estación es de 12 metros. Ahora imaginemos que implementamos un protocolo CSMA para su comunicación. La secuencia de sucesos para transmitir una trama podría ser la siguiente:

1. A desea transmitir datos a B; al detectar el medio lo encuentra libre y empieza la transmisión.

2. Con A transmitiendo C desea transmitir datos hacia B; detecta el medio y lo encuentra libre (C no escucha a A pues esta a 20m de distancia), por tanto C empieza a transmitir.

El resultado es una colisión en el receptor (B) que no es detectada ni por A ni por C. Esto se conoce como el problema de la **estación oculta**.

Imaginemos ahora la misma distribución de estaciones y otra secuencia de sucesos:

1. B desea transmitir datos hacia A, detecta el medio libre e inicia la transmisión.

2. A continuación C desea transmitir datos hacia D; como detecta que B está transmitiendo se espera a que termine para evitar una colisión.

El resultado es que una transmisión que en principio podría haberse hecho sin interferencias (ya que A no puede escuchar a C y D no puede escuchar a B) no se lleva a cabo, reduciendo así la eficiencia del sistema. Esto se conoce como el problema de la **estación expuesta**. Además la mayoría de los estándares de radio son semidúplex, con lo que no pueden transmitir y escuchar el medio a la vez. Por ello, la 802.11 no puede usar CSMA/CD.

Para solucionar este problema, la 802.11 soporta 2 modos de funcionamiento, **DCF** y **PCF** (de *Función de Coordinación Distribuida* y *Función de Coordinación Puntual*. El primero no usa ningún tipo de control central (igual que ethernet) pero el sgundo usa a la estación base como control central. Por eso PCF es opcional.

Cuando funciona en modo DCF, la 802.11 usa el protocolo **CSMA/CA** (CA = Collision Avoidance o evitación de colisiones), que se basa en el MACAW (Multiple Access with Collision Avoidance for Wireless) es el protocolo MAC que ha servido de base para el estándar IEEE 802.11. Este protocolo especifica el funcionamiento de LANs inalámbricas. La figura 3.3 muestra el uso que hace el CSMA/CA de la detección del canal virtual.

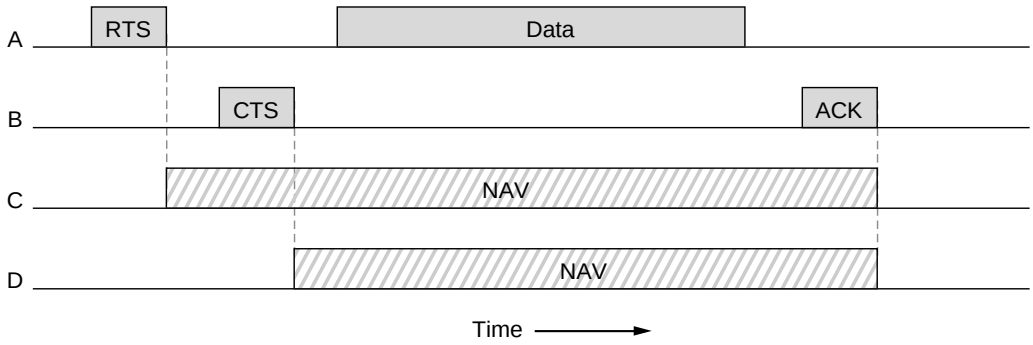


Figura 3.3: *Detección de canal virtual usando CSMA/CA*

En este ejemplo, A decide enviar datos a B, pero se resuelven los dos problemas antes mencionados de la siguiente forma:

1. Cuando una estación tiene una trama que transmitir antes de enviarla envía una trama pequeña de aviso (de 30 bytes) denominada RTS (Request To Send). La trama RTS

contiene información sobre la longitud de la trama que se pretende transmitir y la estación de destino.

2. Al recibir la trama RTS la estación de destino, si está en condiciones de recibir la transmisión, responde con otra trama denominada CTS (Clear To Send). La trama CTS también indica la longitud de la trama que se va a recibir.

Ahora apliquemos este protocolo al caso de la estación oculta para ver que ocurre:

1. A transmite una trama RTS a B indicando la longitud de trama que desea enviarle.
2. B responde con una trama CTS que también especifica la longitud de la trama. En este momento C capta la respuesta de B, por lo que se percata de que va a tener lugar una transmisión en la que B actuará de receptor y sabe que deberá permanecer en silencio durante el tiempo que dure la transmisión (C sabe lo que durará pues conoce la longitud de la trama y la velocidad de la red). Por tanto, durante ese tiempo no transmite nada y se evita la colisión.
3. A envía a B la trama correspondiente.
4. Además, como C y D detectan el CTS de B (en la figura 3.3 se supone que el alcance de cada antena emisora es de 22 mts. en lugar de 12 m.), se autoimponen un tiempo de canal virtual ocupado, indicado por NAV (Vector de Asignación de Red)

En el caso de la estación expuesta ocurriría lo siguiente:

1. B transmite a A una trama RTS indicando que quiere enviarle datos. En ese momento C se entera de las intenciones de B.
2. A devuelve a B una trama CTS. Entretanto C, que ha captado el RTS pero no el correspondiente CTS, comprende que aunque detecta que B está transmitiendo el destinatario está fuera de su alcance, por lo que puede comunicar con D cuando quiera, sin esperar a que B termine.

Para solucionar el problema de los canales ruidosos, la IEEE 802.11 permite dividir las tramas en fragmentos, cada uno con su propio CRC. Cada fragmento se numera de forma individual y su recepción se confirma utilizando un protocolo de parada y espera

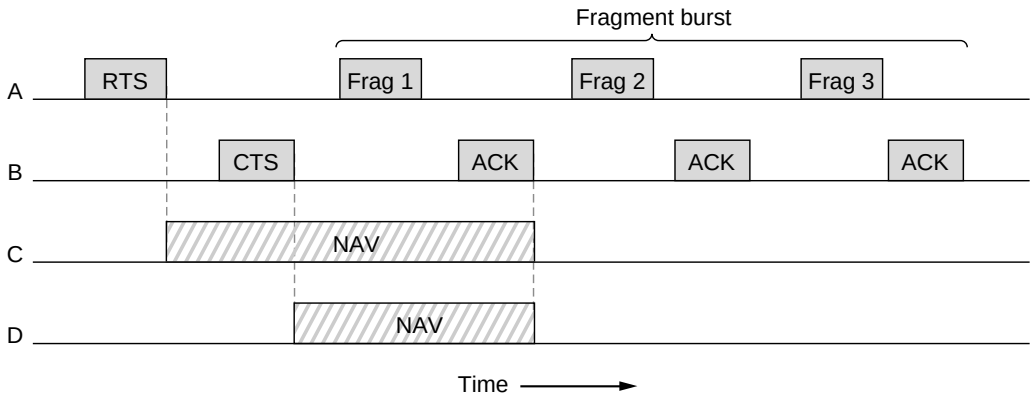


Figura 3.4: *Ráfagas de fragmentos*

(que veremos más adelante). Una vez adquirido el canal mediante RTS y CTS se pueden enviar múltiples fragmentos en una **ráfaga de fragmentos**, tal como muestra la Figura 3.4

La fragmentación incrementa la velocidad real de transporte, limitando las retransmisiones a la fragmentos erróneos, en lugar de a la trama completa.

4 LA SUBCAPA DE ENLACE

La capa de enlace tiene como objetivo

Lograr una comunicación fiable y eficiente entre dos máquinas adyacentes, donde adyacente quiere decir que están conectadas por un canal o canales de comunicaciones que actúan conceptualmente como un cable, es decir, que el canal **entrega los bits en el mismo orden que los recibe**.

Aunque a primera vista este objetivo parece trivial, hay muchas implicaciones que lo hacen complejo:

- El circuito de comunicación puede generar errores.
- La velocidad de transmisión es finita.
- Hay un retardo de propagación inherente al canal.
- Un emisor rápido no debe saturar a uno lento.

Por todo ello, primero hay que determinar los objetivos o funciones que van a determinar el diseño de la capa de enlace.

4.1. Parámetros de diseño de la capa de enlace

La capa de enlace debe desempeñar las siguientes funciones:

- Proporcionar un interfaz bien definido con la capa de red.

- Detectar (y opcionalmente corregir) los posibles errores del canal de comunicación.
- Controlar el flujo de datos.

Para cumplir con estas metas, la capa de enlace toma de la capa de red un **paquete** y lo encapsula como carga útil en una **trama**, que es la unidad de control de flujo de la capa de enlace. De esta forma, tal y como muestra la Figura 4.1, cada trama contiene un encabezado, un campo de carga útil y un terminador o final.

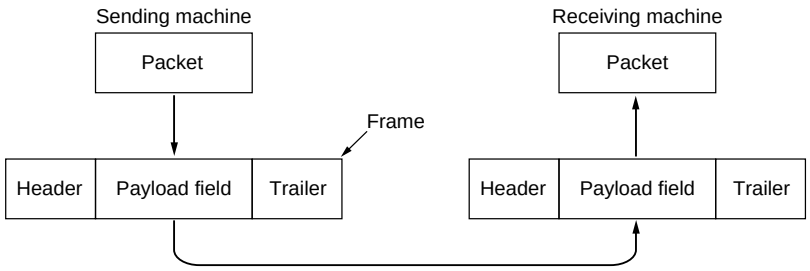


Figura 4.1: *Relación entre paquetes y tramas.*

4.1.1. Entramado

La capa física proporciona a la capa de enlace de datos un flujo de bits. La capa de enlace divide ese flujo de bits en tramas separadas y calcula el CRC de cada trama.

La división del flujo de bits en tramas no es trivial, y existen distintos métodos para conseguir este objetivo:

El primer método es el de **Cuenta de caracteres**. La figura 4.2 muestra un ejemplo de esta técnica. Consiste en añadir un campo de encabezado para especificar el número de caracteres de la trama. Así, cuando la capa de enlace de datos del destino lee esta cabecera, sabe dónde termina la trama. El problema de esta técnica es cuando por un error en el canal de transmisión se altera el campo de cabecera. En ese caso el receptor pierde la sincronía e interpreta mal esa y las siguientes tramas, tal como aparece en la Figura 4.2 b).

El segundo método se denomina método de **bandera** o **flag**. Este método se ilustra en la Figura 4.3. Este método evita el problema de de resincronizar los inicios de trama después de un error haciendo que cada trama comienze y termine con caracteres especiales

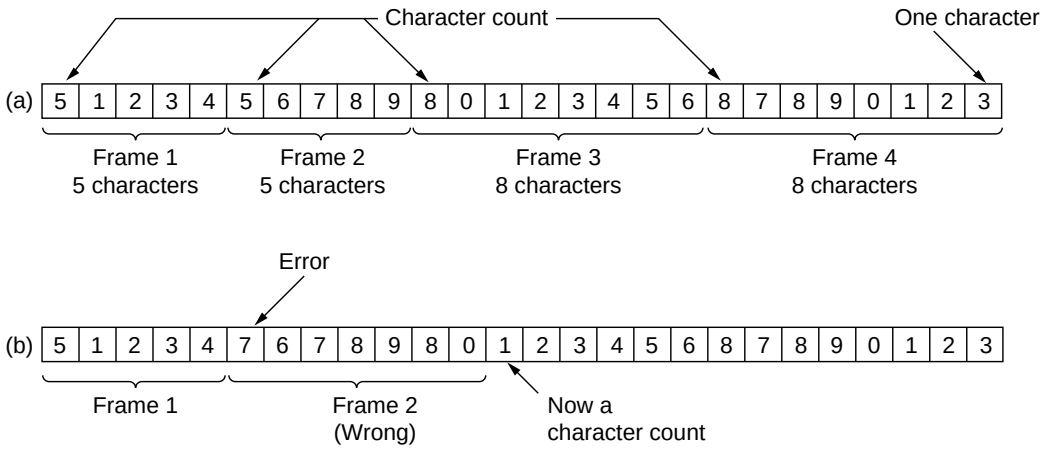


Figura 4.2: *Cuenta de caracteres.*

o delimitadores. Así, aunque el receptor pierda la sincronía puede buscar las banderas para delimitar la trama. Dos banderas consecutivas delimitan el final de una trama y el comienzo de la siguiente.

No obstante, cuando se usa este método para transmitir datos binarios surge un problema, porque es relativamente fácil que en la carga útil aparezca una secuencia de bits con el mismo patrón que el byte de bandera. Para resolver esto, la capa de enlace del emisro inserta un caracter especial (ESC) justo antes de la bandera .ªccidental"de los datos. La capa de enlace del receptor quita ese caracter de escape antes de entregar la carga útil a la capa de red. Por ello, esta técnica se denomina también de **rellenado de caracteres (byte stuffing)**.

Si resulta que también aparece un patrón de bits ESC en los datos, la capa de enlace también añade otro byte de escape. Por tanto, cualquier byte de escape individual es parte de una secuencia de escape, mientras que uno doble indica que un patron escape apareció accidentalmente en los datos.

La desventaja de esta técnica es que está ligada al uso de caracteres de 8 bits, y hay sistemas (como el Unicode) que usan caracteres con distinto núemro de bits. Por eso, la técnica de **rellenado de bits (bit stuffing)** permite que las tramas tengan un número arbitrario de bits. Esta técnica es análoga al rellenado de caracteres, y utiliza como delimitador el patrón 01111110. De la misma forma que la otra técnica, esta no permite que en los datos aparezcan 5 unos consecutivos en los datos, la capa de enlace del emisor añade un 0.

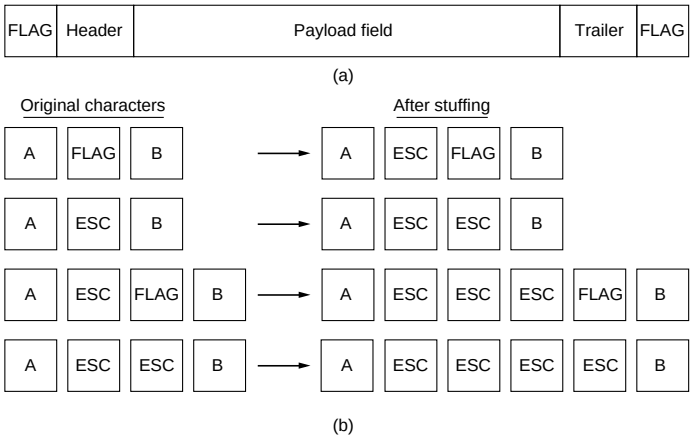


Figura 4.3: *a) Trama delimitada por banderas b) Ejemplos de secuencias de bytes con relleno de caracteres.*

Cuando el receptor ve 5 unos seguidos de un 0 automáticamente retira el 0. De esta forma el relleno de bits también es completamente transparente para la capa de enlace de ambas computadoras. La Figura 4.4 muestra un ejemplo de esta técnica

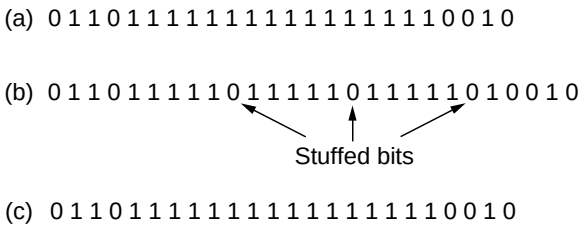


Figura 4.4: *Rellenado de bits: a) Datos originales b) Datos en la línea c) Datos en la capa de red del receptor*

4.1.2. Detección de errores

Una de las funciones de la capa de enlace es detectar los errores **pero no reenviar las tramas erróneas detectadas**. Esto se hace en los niveles superiores. El único caso donde el nivel de enlace reenvía las tramas erróneas es en las redes X.25, es decir, **prácticamente nunca**. Sin embargo, la capa de enlace está preparada para ello.

Lo que sí es necesario es que la capa de enlace detecte los errores que puedan producirse en las tramas como consecuencia de fallos en los circuitos físicos de la transmisión. Hay que tener en cuenta que incluso puede desaparecer una trama completa, debido a una ráfaga de errores. La manera habitual de asegurar una entrega fiable de los datos es proporcionar realimentación al emisor sobre lo que ha ocurrido al otro lado de la línea. Es decir, enviar tramas de control que contengan **confirmaciones** (positivas o negativas) de las tramas que han llegado.

Como además se pueden perder tramas completas, hay que recurrir a **temporizadores** en el emisor, de forma que si se pierde una trama este no se quede esperando su confirmación indefinidamente. Pero ocurre que también se pueden perder las tramas de control que confirman la recepción de una trama, con lo que al saltar el temporizador se volvería a mandar la misma trama y por tanto se duplicaría. Para que esto no ocurra, es necesario asignar **números de secuencia** a las tramas.

4.1.3. Control de flujo

Otro de los objetivos de la capa de enlace es realizar el control de flujo, es decir, qué hacer con un emisor que quiere transmitir datos sistemáticamente y a mayor velocidad que la que peude aceptar un receptor. En general, existen 2 métodos o estrategias. Una de ellas es el control de flujo basado en retroalimentación , donde el receptor realimenta al emisor con información de control bien autorizándolo a enviar más datos, o bien indicándole su estado. La otra estrategia es el control de flujo basado en tasa, donde el protocolo tiene un mecanismo integrado que limita la velocidad de transmisión de los datos. En la capa de enlace de datos se utiliza el control de flujo por realimentación, como veremos más adelante.

4.2. Detección de errores

Los errores de transmisión son inevitables en cualquier medio físico de transmisión, y hay que diseñar mecanismos para tratarlos. Además, como resultado de los procesos físicos que los generan, algunos medios como la radio suelen generar los errores a ráfagas, y no de manera individual.

En la práctica, la capa de enlace usa el **código cíclico redundante, código poli-**

nomial, código de redundancia cíclica o CRC. Estos códigos se basan en considerar las cadenas de bits como representaciones de polinomios con coeficientes 0 y 1 solamente. Así, una trama de k bits se considera como una lista de coeficientes de un polinomio de k términos (o sea, de grado $k-1$), cuyos términos van (de izquierda a derecha) de x^{k-1} a x^0. Así, una cadena de k bits representa a un polinomio de grado $k-1$. Por ejemplo, la cadena 111010 representa a un polinomio de 6 términos , con coeficientes 1,1,1,0,0 y 1. Es decir, representa al polinomio $x^5 + x^4 + x^3 + x^0$

La aritmética polinomial o módulo 2 se realiza según las reglas de campos algebraicos, de tal forma que no existen acarreos en la suma ni préstamos en la resta. Es decir, tanto la suma como la resta de polinomios se hace mdiante una OR-Exclusiva. Los fundamentos teóricos de esta técnica se explican en la asignatura TIC. La división se realiza de la misma forma que en binario, excepto que la resta es módulo 2 (OR-Exclusiva). Se dice que un divisor "cabe" en un dividendo si este tiene tantos bits significativos como el divisor.

Cuando se utiliza un CRC, tanto el emisor como el receptor deben ponerse de acuerdo en el **polinomio generador**, $G(x)$. Por supuesto, los bits de mayor y de menor orden de este polinomio deben ser un 1. Para calcular el CRC de una trama de m bits, correpondiente al polinomio $M(x)$, la trama debe ser más larga que el polinomio generador.

La idea es incluir una suma de verificación (en inglés **checksum**) al final de la trama de tal manera que el polinomio representado por la trama con suma de verificación sea divisible entre $G(x)$. Cuando el receptor recibe la trama con suma de verificación, intenta dividirla entre $G(x)$. Si hay un resto, ha habido un error de transmisión.

El algorimo para calcular la suma de verificación o CRC es el siguiente:

1. Sea r el grado de G(x). Hay que añadir r bits cero al final de la trama, para que ahora contenga $m + r$ bits y corresponda al polinomio $M(x)x^r$.
2. Dividir la cadena de bits correspondiente a $M(x)x^r$ entre la correspondiente a $G(x)$ usando una división módulo 2.
3. Restar el resto de la división anterior (que siempre es de r o menos bits) a la cadena de bits correspondiente a $M(x)x^r$ usando una resta módulo 2. El resultado es la trama con CRC que debe transmitirse, $T(x)$.

En la Figura 4.5 se ilustra el cálculo para una trama 1101011011 utilizando el generador $G(x) = x^4 + x + 1$.

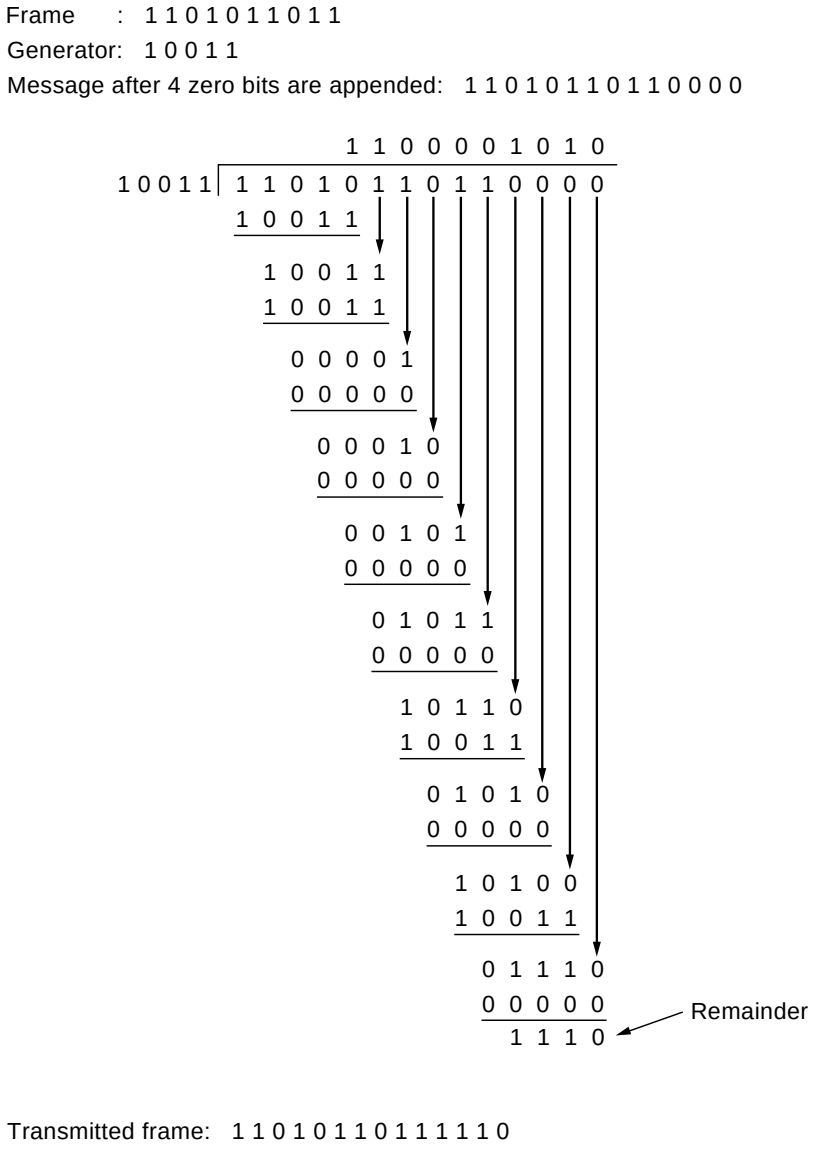


Figura 4.5: *Cálculo de un CRC para añadir a una trama*

Ejercicio opcional: Calcular el proceso inverso, es decir, suponer que llega la trama 11010110111110 y calcular el resto al dividir dicha trama pro el mismo polinomio generador $G(x) = x^4 + x + 1$.

Hay que destacar que $T(x)$ es divisible (módulo 2) entre $G(x)$. En cualquier problema de división, si se resta el residuo del dividendo, lo que queda es divisible entre el divisor. Por ejemplo, en base 10, si se divide 28 entre 5, el residuo es 3. Si se resta 3 a 28, lo que queda (25) es divisible entre 5.

Por tanto, para detectar si la cadena de bits (trama que le ha llegado al receptor contiene algún error, este lo que debe hacer es dividir $T(x)$ entre el polinomio generador $G(x)$. Si el resto es cero, entonces la trama es correcta. De lo contrario, contiene un error en uno o más bits, y por tanto debe ser retransmitida.

Se puede demostrar que un código polinomial con r bits de verificación detecta todos los errores en ráfaga de longitud menor o igual a r. También puede demostrarse que cuando ocurre una ráfaga de errores mayor que $r + 1$, la probabilidad de que una trama incorrecta no sea detectada es de $1/2^r$, suponiendo que todos los patrones de bits sean igualmente probables. El IEEE 802.3 ha estandarizado el polinomio generador

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + 1$$

que entre otras propiedades se puede demostrar que detecta todos los errores en ráfaga menor de 32 bits y todas las ráfagas que afecten a un número par de bits.

4.2.1. Circuitos de detección de errores

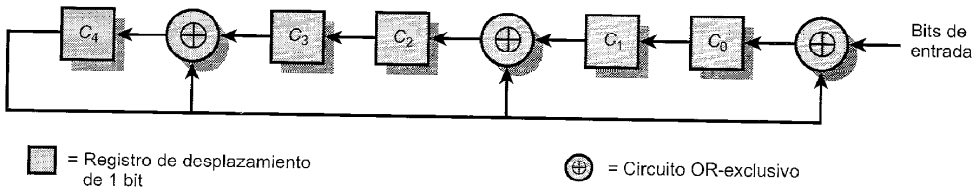
El procedimiento de detección de errores mediante CRC se puede representar, y de hecho implementar, con un circuito divisor formado por puertas «exclusive-OR» y un registro de desplazamiento. Todo el registro utiliza una señal de reloj común, que provoca un desplazamiento de un bit a lo largo de todo el registro.

El circuito se realiza de la siguiente manera:

1. El registro contendrá r bits, igual al grado del polinomio generador del CRC.
2. Hay n puertas exclusive-OR. La presencia o ausencia de puerta corresponderá con la presencia o ausencia del término correspondiente en el polinomio generador $G(x)$,

excluyendo al término x^r. Este término corresponde a la realimentación, y por tanto a la segunda entrada en cada una de las puertas.

Este circuito se explica mejor considerando un caso particular, como el ejemplo que se muestra en la Figura 4.6. En este ejemplo se usa un polinomio generador $G(x) = x^5 + x^4 + x^2 + 1$, que corresponde a la cadena de **6 bits** 110101, y un mensaje $M(x) = x^9 + x^7 + x^3 + x^2 + 1$, que corresponde a la cadena de **10 bits** 1010001101.



(a) Implementación mediante registro de desplazamiento

	C_4	C_3	C_2	C_1	C_0	$C_4 \oplus C_3$	$C_4 \oplus C_1$	$C_4 \oplus$ entrada	entrada	
Inicialización	0	0	0	0	0	0	0	1	1	
Paso1	0	0	0	0	1	0	0	0	0	
Paso2	0	0	0	1	0	0	1	1	1	
Paso3	0	0	1	0	1	0	0	0	0	
Paso4	0	1	0	1	0	1	1	0	0	
Paso5	1	0	1	0	0	1	1	1	0	Mensaje a enviar
Paso6	1	1	1	0	1	0	1	0	1	
Paso7	0	1	1	1	0	1	1	1	1	
Paso8	1	1	1	0	1	0	1	1	0	
Paso9	0	1	1	1	1	1	1	1	1	
Paso10	1	1	1	1	1	0	0	1	0	
Paso11	0	1	0	1	1	1	1	0	0	
Paso12	1	0	1	1	0	1	0	1	0	Cinco ceros añadidos
Paso13	1	1	0	0	1	0	1	1	0	
Paso14	0	0	1	1	1	0	1	0	0	
Paso15	0	1	1	1	0	1	1	0	—	

(b) Ejemplo con entrada 1010001101

Figura 4.6: *a) Circuito generador y detector de errores b)Cálculo de un CRC*

En la Figura 4.6 se muestra la realización del registro de desplazamiento. El proceso comienza con la puesta a cero de todo el registro. El mensaje o dividendo se introduce a continuación, bit a bit, comenzando con el bit más significativo. Cada fila de la tabla muestra los valores almacenados en los cinco elementos de memoria del registro de desplazamiento, así como los valores que aparecerán en las salidas de las tres puertas exclusive-OR.

Debido a que no hay realimentación hasta que un 1 del dividendo aparezca en el extremo más significativo del registro, las primeras cinco operaciones son simplemente desplazamientos. Siempre que un "1"llegue al extremo izquierdo del registro (C_4),se resta l («exclusive-OR») del segundo (C_3), cuarto (C_1) y sexto (entrada) bit del siguiente desplazamiento. Esto es idéntico al procedimíento de la división mencionado anteriormente. El procedimiento continua para todos los bits del mensaje más los 5 bits ($r = 5$) igual a cero. Estos últimos son para desplazar M cinco posiciones a la izquierda y dar cabida al CRC. Tras procesar el último bit, el registro de desplazamiento contendrá el resto que puede ser transmitido (se muestra sombreado), es dcir, el CRC generado.

En el receptor, se utiliza la misma lógica. Cada bit de la trama $T(x)$ se introducirá en el registro de desplazamiento. Si no ha habido errores, al final el registro de desplazamiento debería contener todas las posiciones igual a cero.

4.3. Protocolos elementales de la capa de enlace

Antes de estudiar los protocolos hay que destacar los supuestos implícitos del modelo de comunicaciones que vamos a utilizar. Primero, estamos suponiendo que en las capas fisica, de enlace de datos y de red hay procesos independientes que se comunican pasando mensajes de un lado a otro. En muchos casos, los procesos de las capas fisica y de enlace de datos se ejecutan en un procesador dentro de un chip especial de E/S y los de la capa de red lo hacen en la CPU principal. Sin embargo, también puede haber otras implementaciones

Otro supuesto clave es que la máquina A desea mandar un flujo considerable de datos a la máquina B usando un servicio confiable orientado a la conexión. Después consideraremos el caso en que B también quiere mandar datos a A de manera simultánea. Se ha supuesto que A tiene un suministro infinito de datos listos para ser enviados y nunca tiene que esperar a que se produzcan datos. Cuando la capa de enlace de datos le solicita datos, la capa de red siempre es capaz de proporcionarlos de inmediato. (Esta restricción también se desechará posteriormente.)

También supondremos que las máquinas no fallan. Es decir, estos protocolos manejan errores de comunicación, pero no los problemas causados por computadoras que fallan y se reinician. En lo que concierne a la capa de enlace de datos, el paquete que se le pasa a través de la interfaz desde la capa de red es de datos puros, que deben ser entregados bit por bit a la capa de red del destino. El hecho de que la capa de red del destino pueda interpretar parte del paquete como un encabezado no es de importancia para la capa de enlace

de datos.

Cuando la capa de enlace de datos acepta un paquete, lo encapsula en una trama agregándole un encabezado y un terminador de enlace de datos, tal como aparece en la figura 4.7. Por lo tanto, una trama consiste en un paquete incorporado, cierta información de control (en el encabezado) y una suma de verificación (en la cola). Supondremos que existen procedimientos adecuados como To_physical_layer para enviar una trama y From_physical_layer para recibir una trama. El hardware emisor calcula y agrega el CRC, por lo que el software de la capa de enlace de datos no necesita preocuparse por ella.

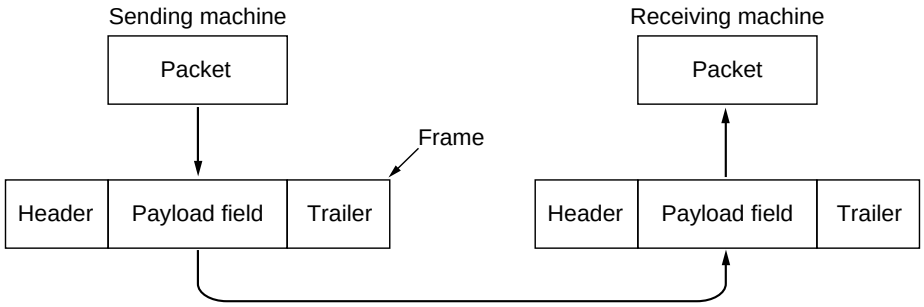


Figura 4.7: *Encapsulamiento de datos en la capa de enlace*

Inicialmente el receptor sólo está esperando que ocurra algo. En los protocolos indicaremos que la capa de enlace de datos está en espera de que ocurra algo con la llamada al procedimiento wait_for_event(&event). Este procedimiento sólo regresa cuando ocurre algo (por ejemplo, cuando llega una trama), y en ese caso la variable event indica lo que ha ocurrido. El grupo de eventos posibles difiere para cada uno de los diferentes protocolos que describiremos, y se definirán por separado para cada protocolo.

Cuando llega una trama al receptor, el hardware calcula la suma de verificación. Si ésta es incorrecta (es decir, si hubo un error de transmisión), se le informa a la capa de enlace de datos (event = checksum_err). Si la trama entrante llega sin daño, también se le informa a la capa de enlace de datos (event =frame_arrival) para que pueda adquirirla mediante from_physical_layer. Tan pronto como la capa de enlace de datos receptora adquiere una trama sin daños, revisa la información de control del encabezado y, si todo está bien, pasa la parte que corresponde al paquete a la capa de red. Para mantener independientes el protocolo de red con el protocolo de enlace de datos, **nunca** se entrega una cabecera de trama a la capa de red.

En la figura 4.8 se muestran algunas declaraciones comunes (en C) para muchos de

los protocolos que se analizarán después. Se definen cinco estructuras de datos: boolean, seq_nr, packet, frame_kind y frame. Un seq_Nr (número de secuencia) es un entero pequeño que sirve para numerar las tramas, a fin de distinguirlas. Estos números de secuencia van de 0 hasta Max_sec (inclusive), que se define en cada protocolo que lo necesita. Un packet es la unidad de intercambio de información entre la capa de red y la de enlace de datos en la misma máquina, o entre entidades iguales de la capa de red. Un frame está compuesto de cuatro campos: kind, Seq, ack e info. Los primeros tres contienen información de control y el último puede contener los datos por transferir. Estos campos de control constituyen en conjunto el encabezado de la trama.

El campo kind indica si hay datos en la trama, porque algunos de los protocolos distinguen entre las tramas que contienen exclusivamente información de control y los que también contienen datos. Los campos seq y ack se emplean para números de secuencia y confirmaciones de recepción, respectivamente; El campo info de una trama de datos contiene un solo paquete; el campo info de una trama de control no se usa.

En la figura 4.8 también se listan varios procedimientos que son librerías. El procedimiento wait_for_event se queda en un ciclo cerrado esperando que algo ocurra. Los procedimientos To_Network_layer y From_Network_layer sirven para que la capa de enlace de datos pase paquetes a la capa de red y acepta paquetes de ella, respectivamente.

En la mayoría de los protocolos suponemos un canal inestable que pierde tramas completas ocasionalmente. Para poder recuperarlas, la capa de enlace de datos emisora debe arrancar un temporizador o reloj interno cada vez que envía una trama. Si no obtiene respuesta tras transcurrir cierto intervalo de tiempo predeterminado, el temporizador expira y la capa de enlace de datos recibe una señal de interrupción. Esto se maneja permitiendo que el procedimiento wait_for_event devuelva event = timeout. Los procedimientos start_timer y stop_timer inician y detienen, respectivamente, el temporizador. Las terminaciones del temporizador sólo son posibles cuando éste se encuentra en funcionamiento. Se permite explícitamente llamar a start_timer cuando el temporizador está funcionando; eso sirve para hacer que el temporizador termine después de haber transcurrido un intervalo completo de temporización (a menos que se restablezca o apague antes).

Los procedimientos star_ack_timer y stop_ack_timer controlan un temporizador auxiliar usado para generar confirmaciones de recepción en ciertas condiciones. Los procedimientos enable_Network_layer y disable_network_layer se usan en los protocolos más complicados, en los que ya no suponemos que la capa de red siempre tiene paquetes que enviar. Cuando la capa de enlace de datos habilita a la capa de red ésta tiene permitido interrumpir cuando tenga que enviar un paquete. Esto lo indicamos con event = network_layer_ready.

```
#define MAX_PKT 1024                    /* determines packet size in bytes */

typedef enum {false, true} boolean;     /* boolean type */
typedef unsigned int seq_nr;            /* sequence or ack numbers */
typedef struct {unsigned char data[MAX_PKT];} packet;/*   packet definition */
typedef enum {data, ack, nak} frame_kind; /* frame_kind definition */

typedef struct {                        /* frames are transported in this layer */
  frame_kind kind;                      /* what kind of a frame is it? */
  seq_nr seq;                           /* sequence number */
  seq_nr ack;                           /* acknowledgement number */
  packet info;                          /* the network layer packet */
} frame;

/* Wait for an event to happen; return its type in event. */
void wait_for_event(event_type *event);

/* Fetch a packet from the network layer for transmission on the channel. */
void from_network_layer(packet *p);

/* Deliver information from an inbound frame to the network layer. */
void to_network_layer(packet *p);

/* Go get an inbound frame from the physical layer and copy it to r. */
void from_physical_layer(frame *r);

/* Pass the frame to the physical layer for transmission. */
void to_physical_layer(frame *s);

/* Start the clock running and enable the timeout event. */
void start_timer(seq_nr k);

/* Stop the clock and disable the timeout event. */
void stop_timer(seq_nr k);

/* Start an auxiliary timer and enable the ack_timeout event. */
void start_ack_timer(void);

/* Stop the auxiliary timer and disable the ack_timeout event. */
void stop_ack_timer(void);

/* Allow the network layer to cause a network_layer_ready event. */
void enable_network_layer(void);

/* Forbid the network layer from causing a network_layer_ready event. */
void disable_network_layer(void);

/* Macro inc is expanded in-line: Increment k circularly. */
#define inc(k) if (k < MAX_SEQ) k = k + 1; else k = 0
```

Figura 4.8: *Estructuras y funciones usadas en los protocolos*

Cuando una capa de red está inhabilitada, no puede causar tales eventos. Así la capa de enlace puede evitar que la capa de red la sature con paquetes para los que no tiene espacio de búfer.

Los números de secuencia de las tramas siempre están en el intervalo de 0 a Max_sec (inclusive), donde Max_sec es diferente para los distintos protocolos. Con frecuencia es necesario avanzar circularmente en 1 un núnero de secuencia (por ejemplo, Max_sec va seguido de 0). La macro inc lleva a cabo este incremento. Esta función se ha definido como macro porque se usa dentro de la ruta critica. Las declaraciones de la figura 4.8 son parte de todos los protocolos que siguen.

4.3.1. Protocolo símplex no restringido

Como ejemplo inicial consideraremos un protocolo que es lo más sencillo posible:

- Los datos se transmiten sólo en una dirección.
- Las capas de red tanto del emisor como de] receptor siempre están listas.
- El tiempo de procesamiento puede ignorarse.
- Hay un espacio infinito de búfer.
- Lo mejor de todo, el canal de comunicación entre las capas de enlace de datos nunca tiene problemas. Ni daña ni pierde las tramas.

Este protocolo, que es completamente irreal, se muestra en la figura 4.9.

El protocolo consiste en dos procedimientos diferentes, uno emisor y uno receptor. El emisor se ejecuta en la capa de enlace de datos de la máquina de origen y el receptor se ejecuta en la capa de enlace de datos de la máquina de destino. No se usan números de secuencia ni confirmaciones de recepción, por lo que no se necesita Max_sec. El único tipo de evento posible es frame_arrival (es decir, la llegada de una trama sin daños).

El emisor está en un ciclo while infinito que sólo envía datos al canal tan rápidamente como puede. El cuerpo del ciclo consiste en tres acciones: obtener un paquete de la (siempre dispuesta) capa de red, construir una trama de salida usando la variable s y enviar la trama a su destino. Este protocolo sólo utiliza el campo info de la trama, pues los demás

```
/* Protocol 1 (utopia) provides for data transmission in one direction only, from
   sender to receiver.  The communication channel is assumed to be error free
   and the receiver is assumed to be able to process all the input infinitely quickly.
   Consequently, the sender just sits in a loop pumping data out onto the line as
   fast as it can. */

typedef enum {frame_arrival} event_type;
#include "protocol.h"

void sender1(void)
{
  frame s;                          /* buffer for an outbound frame */
  packet buffer;                    /* buffer for an outbound packet */

  while (true) {
      from_network_layer(&buffer); /* go get something to send */
      s.info = buffer;              /* copy it into s for transmission */
      to_physical_layer(&s);        /* send it on its way */
  }                                 /* Tomorrow, and tomorrow, and tomorrow,
                                       Creeps in this petty pace from day to day
                                       To the last syllable of recorded time.
                                           - Macbeth, V, v */
}


void receiver1(void)
{
  frame r;
  event_type event;                 /* filled in by wait, but not used here */

  while (true) {
      wait_for_event(&event);       /* only possibility is frame_arrival */
      from_physical_layer(&r);      /* go get the inbound frame */
      to_network_layer(&r.info);    /* pass the data to the network layer */
  }
}
```

Figura 4.9: *Protocolo simplex no restringido*

campos tienen que ver con el control de errores y de flujo, y aquí no hay restricciones de control de errores ni de flujo.

El receptor también es sencillo. Inicialmente espera que algo ocurra, siendo la única posibilidad la llegada de una trama sin daños. En algún momento, la trama llega y el procedimiento wait_for_event regresa, conteniendo event el valor frame_arrival (que de todos modos se ignora). La llamada a from_physicl_layer elimina la trama recién llegada del búfer de hardware y la coloca en la variable r, en donde el código receptor pueda obtenerla. Por último, la parte de datos se pasa a la de red y la capa de enlace de datos se retira para esperar la siguiente trama.

4.3.2. Protocolo símplex de parada y espera (stop & wait)

Ahora omitirernos el supuesto más irreal hecho en el protocolo l: la presencia en la capa de enlace de datos receptora de una cantidad infinita de espacio de búfer en el cual almacenar todas las tramas de entrada mientras esperan su respectivo turno (o o que es equivalente, la velocidad infinita de procesaiento de la capa de red receptora). Aún suponemos que el canal de comunicaciones está libre de errores y que el tráfico de datos es símplex.

El problema principal que a resolver aquí es cómo evitar que el emisor sature al receptor enviando datos a mayor velocidad de la que este último puede procesarlos. En esencia, si el receptor requiere un tiempo t para ejecutar From_physical_layer más To_network_layer, el emisor debe transmitir a una tasa media menor que una trama por tiempo t. Es más, si suponemos que en el hardware del receptor no se realiza de manera automática el almacenamiento en el búfer y el encolamiento, el emisor nunca debe transmitir una trama nueva hasta que la vieja haya sido obtenida por From_physical_layer, para que lo nuevo no sobrescriba lo antiguo.

En ciertas circunstancias restringidas (por ejemplo, transmisión síncrona y una capa de enlace de datos receptora dedicada por completo a procesar la línea de entrada única), el emisor podría introducir simplemente un retardo en el protocolo l y así reducir su velocidad lo suficiente para evitar que se sature el receptor. Sin embargo, es más común que la capa de enlace de datos tenga varios canales a los cuales atender, y el intervalo de tiempo entre la llegada de una trama y su procesamiento puede variar en forma considerable. Si los diseñadores de la red pueden calcular el comportamiento de peor caso del receptor, podrán programar al emisor para que transmita con tanta lentitud que, aun si cada trama suhe el retardo máximo, no haya desbordamientos. El problema con este método es que es dema-

siado conservador. Conduce a un aprovechamiiento del ancho de banda muy por debajo del óptimo, a menos que el mejor caso y el peor sean iguales (es decir, la variación en el tiempo de reacción de la capa de enlace de datos sea pequeña).

Una solución más general para este dilema es hacer que el receptor proporcione retroalimentación al emisor. Tras haber pasado un paquete a su capa de red, el receptor devuelve al emisor una pequeña trama ficticia que, de hecho, autoriza al emisor para transmitir la siguiente trama. Tras haber enviado una trama, el protocolo exige que el emisor espere basta que llegue la pequeña trama ficticia (es decir, la confirmación de recepción). Los protocolos en los que el emisor envía una trama y luego espera una confirmación de re- cepción antes de continuar se denominan de parada y espera.

En la figura 4.10 se muestra el código de lo que podría ser un ejemplo de un protocolo símplex de parada y espera.

Aunque el tráfico de datos en este ejemplo es símplex y por tanto va sólo desde el emisor al receptor, las tramas viajan en ambas direcciones. En consecuencia, el canal de comunicación entre las dos capas de enlace de datos necesita tener capacidad de transferencia de información bidireccional. Sin embargo, este protocolo implica una altemancia estricta de flujo: primero el emisor envía una trama y después el receptor envía una trama. Por tanto, aquí sería suficiente un canal fisico semidúplex.

La única diferencia entre el receiver1 y el receiver2 es que, tras entregar un paquete a la capa de red receiver2 devuelve al emisor una trama de confirmación de recepción antes de entrar nuevamente en el ciclo de espera. Puesto que sólo es importante la llegada de la trama en el emisor, no su contenido, el receptor no necesita poner ninguna información específica en él.

4.3.3. Protocolo símplex para un canal con ruido

Ahora consideremos la situación normal de un canal de comunicación con errores. Las tramas pueden llegar dañadas o perderse por completo. Sin embargo, suponemos que si una trama se daña en tránsito, el hardware del receptor detectará esto cuando calcule el CRC. Si la trama está dañada de tal manera que pese a ello la suma de verificación sea correcta, un caso excesivamente improbable, este protocolo (y todos los demás) puede fallar (es decir, entregar un paquete incorrecto a la capa de red).

A primera vista puede parecer que funcionaría una variación del protocolo 2: añadir

```
/* Protocol 2 (stop-and-wait) also provides for a one-directional flow of data from
   sender to receiver. The communication channel is once again assumed to be error
   free, as in protocol 1. However, this time, the receiver has only a finite buffer
   capacity and a finite processing speed, so the protocol must explicitly prevent
   the sender from flooding the receiver with data faster than it can be handled. */

typedef enum {frame_arrival} event_type;
#include "protocol.h"

void sender2(void)
{
  frame s;                          /* buffer for an outbound frame */
  packet buffer;                    /* buffer for an outbound packet */
  event_type event;                 /* frame_arrival is the only possibility */

  while (true) {
      from_network_layer(&buffer); /* go get something to send */
      s.info = buffer;              /* copy it into s for transmission */
      to_physical_layer(&s);        /* bye-bye little frame */
      wait_for_event(&event);       /* do not proceed until given the go ahead */
  }
}

void receiver2(void)
{
  frame r, s;                       /* buffers for frames */
  event_type event;                 /* frame_arrival is the only possibility */
  while (true) {
      wait_for_event(&event);       /* only possibility is frame_arrival */
      from_physical_layer(&r);      /* go get the inbound frame */
      to_network_layer(&r.info);    /* pass the data to the network layer */
      to_physical_layer(&s);        /* send a dummy frame to awaken sender */
  }
}
```

Figura 4.10: *Protocolo simplex de parada y espera*

un temporizador. El emisor podría enviar una trama, pero el receptor sólo enviaría una trama de confirmación de recepción si los datos llegaran correctamente. Si llegara una trama dañada al receptor se desecharía. Poco después, el temporizador del emisor expiraría y se enviaría la trama de nuevo. Este proceso se repetiría hasta que la trama por fin llegara intacta. El esquema anterior tiene un defecto crucial. Sea la siguiente secuencia de eventos:

1. La capa de red de A entrega el paquete l a su capa de enlace de datos. El paquete se recibe correctamente en B y se pasa a la capa de red de B. B devuelve a A una trama de confirmación de recepción.

2. La trama de confirmación de recepción se pierde por completo. Nunca llega. El canal no hace distinciones en el sentido de circulación de las tramas, y puede dañar o perder tramas en cualquier sentido.

3. El temporizador de la capa de enlace de datos de A expira. Al no haber recibido una confirmación de recepción, supone (incorrectamente) que su trama de datos se ha perdido o dañado, y envía otra vez la trama que contiene el paquete l.

4. La trama duplicada también llega bien a la capa de enlace de datos de B y de ahí se pasa de manera inadvertida a la capa de red. Es decir, el protocolo fallará.

Es claro que lo que se necesita es alguna manera de que el receptor sea capaz de distinguir entre una trama que está viendo por primera vez y una retransmisión. La forma evidente de lograr esto es hacer que el emisor ponga un número de secuencia en el encabezado de cada trama que envía. Así el receptor puede examinar el número de secuencia de cada trama que llega para ver si es una trma nueva o un duplicado que debe descartarse.

Dado que es deseable que el encabezado de las tramas sea pequeño, surge la pregunta: ¿cuál es la cantidad mínima de bits necesarios para el número de secuencia? La única ambigüedad de este protocolo es entre una trama m y la siguiente trama $m + l$, ya que cada trama tiene que ser reconocida. Si la trama m se pierde o se daña, el receptor no confirmará su recepción y el emisor seguirá tratando de enviarla. Una vez que la trama se recibe correctamente, el receptor devuelve una confirmación de recepción al emisor. Es aquí donde surge el problema potencial. Dependiendo de si el emisor recibe correctamente la trama de confirmación de recepción, tratará de enviar m o $m + l$. El evento que indica al emisor que puede enviar $m + 2$ es la llegada de una confirmación de recepción de $m + l$. Pero esto implica que m se recibió de manera correcta, y además que su confirmación de recepción fue recibida correctamente por el emisor (de otra manera, el emisor no habría

comenzado con $m + l$. Como consecuencia, la única ambigüedad es entre una trama y su antecesor o sucesor inmediatos.

Por lo tanto, basta con un número de secuencia de l bit (O o l). En cada instante, el receptor espera un número de secuencia en particular. Cualquier trama de entrada que contenga un número de secuencia equivocado se rechaza como duplicado. Cuando llega una trama que contiene el número de secuencia correcto, se acepta y se pasa a la capa de red y el número de secuencia esperado se incrementa módulo 2 (es decir, O se vuelve l y l se vuelve O). En la figura 4.11 3-12 se muestra un ejemplo de este tipo de protocolo. Los protocolos en los que el emisor espera una confirmación de recepción positiva antes de avanzar al siguiente elemento de datos suelen llamarse PAR (Confirmación de Recepción Positiva con Retransmisión) o ARQ (Solicitud Automática de Repetición).

Al igual que el protocolo 2, éste también transmite datos en una sola dirección. El protocolo 3 difiere de sus antecesores en que tanto el emisor como el receptor tienen una variable cuyo valor se recuerda mientras la capa de enlace de datos está en estado de espera. El emisor recuerda el número de secuencia de la siguiente trama a enviar en Next_frame_to_send; el receptor recuerda el número de secuencia de la siguiente trama esperada frame_expected. Cada protocolo tiene una fase de inicialización corta antes de entrar en el ciclo infinito.

Tras transmitir una trama, el emisor arranca el temporizador. Si éste ya se estaba ejecutando, se resetea para conceder otro intervalo completo de temporización. Dicho intervalo debe escogerse de modo que haya suficiente tiempo para que la trama llegue al receptor, éste la procese en el peor caso y la confirmación de recepción llegue al emisor. Sólo cuando ha transcurrido ese intervalo de tiempo se puede suponer con seguridad que se ha perdido la trama transmitida o su confirmación de recepción, y que se debe enviar un duplicado. Si el intervalo establecido es muy pequeño, el emisor transmitirá tramas innecesarias. Si bien estas tramas adicionales no afectarán a la corrección del protocolo, sí dañarán las prestaciones. Tras transmitir una trama y arrancar el temporizador, el emisor espera que ocurra un evento. Hay tres posibilidades: llega una trama de confirmación de recepción sin daño, llega una trama de confirmación de recepción dañada o expira el temporizador. Si recibe una confirmación de recepción válida, el emisor obtiene el siguiente paquete de la capa de red y lo coloca en el búfer, sobrescribiendo el paquete previo. También avanza el número de secuencia. Si llega una trama dañada o no llega ninguna, ni el búfer ni el número de secuencia cambia, con el fin de enviar un duplicado. Cuando llega una trama válida al receptor, su número de secuencia se verifica para saber si es un duplicado. Si no lo es, se acepta, se pasa a la capa de red y se genera una confirmación de recepción. Los duplicados

```
/* Protocol 3 (par) allows unidirectional data flow over an unreliable channel. */

#define MAX_SEQ 1                       /* must be 1 for protocol 3 */
typedef enum  {frame_arrival, cksum_err, timeout} event_type;
#include "protocol.h"

void sender3(void)
{
  seq_nr next_frame_to_send;            /* seq number of next outgoing frame */
  frame s;                              /* scratch variable */
  packet buffer;                        /* buffer for an outbound packet */
  event_type event;

  next_frame_to_send = 0;               /* initialize outbound sequence numbers */
  from_network_layer(&buffer);          /* fetch first packet */
  while (true) {
      s.info = buffer;                  /* construct a frame for transmission */
      s.seq = next_frame_to_send;       /* insert sequence number in frame */
      to_physical_layer(&s);            /* send it on its way */
      start_timer(s.seq);               /* if answer takes too long, time out */
      wait_for_event(&event);           /* frame_arrival, cksum_err, timeout */
      if (event == frame_arrival) {
          from_physical_layer(&s);      /* get the acknowledgement */
          if (s.ack == next_frame_to_send) {
              stop_timer(s.ack);        /* turn the timer off */
              from_network_layer(&buffer);     /* get the next one to send */
              inc(next_frame_to_send);     /* invert next_frame_to_send */
          }
      }
  }
}

void receiver3(void)
{
  seq_nr frame_expected;
  frame r, s;
  event_type event;

  frame_expected = 0;
  while (true) {
      wait_for_event(&event);           /* possibilities: frame_arrival, cksum_err */
      if (event == frame_arrival) {     /* a valid frame has arrived. */
          from_physical_layer(&r);      /* go get the newly arrived frame */
          if (r.seq == frame_expected) {/* this is what we have been waiting for. */
              to_network_layer(&r.info);    /* pass the data to the network layer */
              inc(frame_expected); /* next time expect the other sequence nr */
          }
          s.ack = 1 – frame_expected;   /* tell which frame is being acked */
          to_physical_layer(&s);        /* send acknowledgement */
      }
  }
}
```

Figura 4.11: *Protocolo simplex para canal ruidoso*

y las tramas dañadas no se pasan a la capa de red.

4.4. Protocolos de ventana deslizante

En los protocolos previos, las tramas de datos se transmiten en una sola dirección. Una manera de lograr una transmisión de datos full dúplex es tener dos canales de comunicación separados y utilizar cada uno para tráfico de datos símplex (en diferentes direcciones). Si se hace esto, tenemos dos circuitos fisicos separados, cada uno con un canal ”de ida”(para datos) y un canal ”de retorno”(para confirmaciones de recepción). En ambos casos, el ancho de banda del canal usado para confirmaciones de recepción se desperdicia casi por completo. El usuario está pagando dos circuitos, pero sólo usa la capacidad de uno. Por eso, es mejor utilizar el mismo circuito para datos en ambas direcciones. Después de todo, en los protocolos 2 y 3 ya se usaba para transmitir tramas en ambos sentidos, y el canal de retorno tiene la misma capacidad que el canal de ida. Por ello, en este modelo las tramas de datos de A a B se mezclan con las tramas de confirmación de recepción de A a B. Analizando el campo de tipo (kind) en el encabezado de una trama de enviada, el receptor puede saber si la trama es de datos o de confirmación de recepción.

Aunque el entrelazado de datos y de tramas de control en el mismo circuito es una mejora respecto al uso de dos circuitos fisico separados, se puede lograr otra mejora. Cuando llega una trama de datos, en lugar de enviar inmediatamente una trama de control independiente, el receptor se espera hasta que la capa de red le pasa el siguiente paquete. La confirmación de recepción se adjunta a la trama de datos de salida (usando el campo ack del encabezado de la trama). Así la confirmación de recepción viaja gratuitamente en la siguiente trama de datos de salida.

La técnica de retardar temporalmente las confirmaciones de recepción para que puedan viajar en la siguiente trama de datos de salida se conoce como superposición o *piggybacking*. La ventaja principal de usar la supeposición es un mejor aprovechamiento del ancho de banda disponible del canal. El campo ack del encabezado de la trama ocupa sólo unos cuantos bits, mientras que una trama aparte requeriría de un encabezado, la confirmación de recepción y una suma de verificación. Además, el envío de menos tramas implica menos interrupciones de ”ha llegado tramaτ tal vez menos búfer en el receptor, dependiendo de la manera en que esté organizado el software del receptor. En el siguiente protocolo que examinaremos, el campo de superposición ocupa sólo l bit en el encabezado de la trama y pocas veces ocupa más de algunos bits.

Sin embargo, la superposición introduce una complicación inexistente en las confirmaciones de recepción independientes. ¿Cuánto tiempo debe esperar la capa de enlace de datos un paquete al cual superponer la confirmación de recepción? Si la capa de enlace de datos espera más tiempo del que tarda en terminar el temporizador del emisor, la trama será retransmitida, frustrando el propósito de enviar confirmaciones de recepción. Por ello, lo que se hace es que si aparece rápidamente un nuevo paquete a enviar, la confirmación de recepción se superpone a él; si no ha llegado ningún paquete nuevo al final de este periodo, la capa de enlace de datos manda una trama de confirmación de recepción independiente.

Los siguientes tres protocolos son bidireccionales y pertenecen a una clase llamada protocolos de ventana deslizante. Los tres difieren enhe ellos en la eficiencia, complejidad y requisitos de búfer, como se analizará más adelante. En ellos, al igual que en todos los protocolos de ventana deslizante, cada trama de salida contiene un número de secuencia, que va desde 0 hasta algún número máximo. Por lo general, éste es $2^n - l$, por lo que el número de secuencia encaja perfectamente en un campo de n bits. El protocolo de ventana deslizante de parada y espera utiliza n = l, y restringe los números de secuencia a 0 y l, pero las versiones más refinadas pueden utilizar un n arbitrario.

La esencia de todos los protocolos de ventana deslizante es

1. En cualquier instante, el emisor mantiene un grupo de números de secuencia que corresponde a las tramas que tiene permitido enviar. Se dice que estas tramas caen dentro de la ventana emisora.

2. De manera semejante, el receptor mantiene una ventana receptora correspondiente al grupo de tramas que tiene permitido aceptar.

3. La ventana del emisor y la del receptor no necesitan tener los mismos límites inferior y superior, ni siquiera el mismo tamaño. En algunos protocolos las ventanas son de tamaño fijo, pero en otros pueden crecer y disminuir a medida que se envían y reciben las tramas.

4. Aunque estos protocolos dan a la capa de enlace de datos mayor libertad en cuanto al orden en que puede enviar y recibir tramas, se conserva el requisito de que el protocolo debe entregar los paquetes a la capa de red del destino en el mismo orden en que se pasaron a la capa de enlace de datos de la máquina emisora. Tampoco hemos cambiado el requisito de que el canal de comunicación es ”de tipo cable”, es decir, que entrega todas las tramas en el orden en que fueron enviadas.

5. Los números de secuencia en la ventana del emisor representan tramas enviadas o que pueden ser enviadas, pero cuya recepción aún no se ha confirmado. Cuando llega un paquete nuevo de la capa de red se le da el siguiente número secuencial mayor, y el extremo superior de la ventana avanza en uno. Al llegar una confirmación de recepción, el extremo inferior avanza en uno. De esta manera, la ventana mantiene continuamente una lista de bamas sin confirmación de recepción. En la figura 4.12 se muestra un ejemplo.

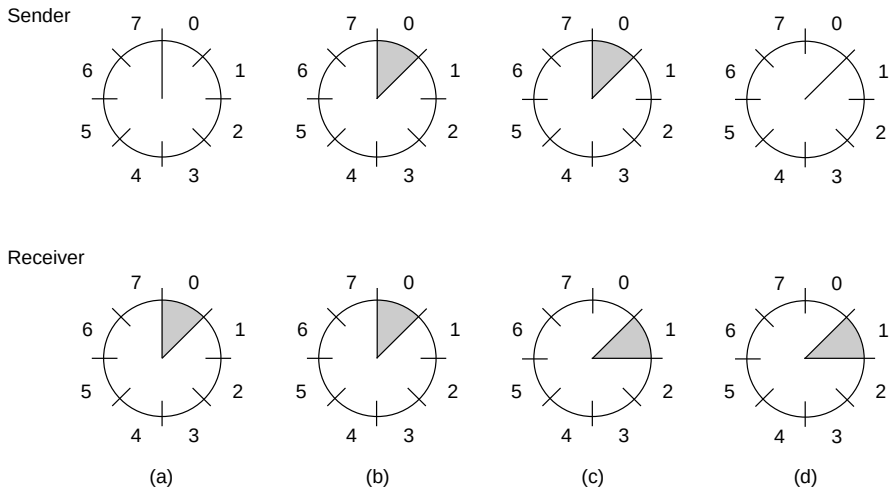


Figura 4.12: *Ventana deslizante de tamaño 1, con un número de secuencia de 3 bits. (a) Al inicio. (b) Tras la transmisión de la primera trama. (c) Tras la recepción de la primera trama. (d) Tras recibir la primera confirmación de recepción*

6. Dado que las tramas que están en la ventana del emisor pueden perderse o dañarse en tránsito, el emisor debe mantener todas estas tramas en su memoria para su posible retransmisión. Por lo tanto, si el tamaño máximo de la ventana es n, el emisor necesita n búferes para contener las tramas sin confirmación de recepción. Si la ventana llega a crecer a su tamaño máximo, la capa de enlace de datos emisora deberá hacer que la capa de red se detenga hasta que se libere otro búfer.

7. La ventana de la capa de enlace de datos receptora corresponde a las tramas que puede aceptar. Toda trama que caiga fuera de la ventana se descartará sin comentarios. Cuando se recibe la trama cuyo número de secuencia es igual al extremo inferior de la ventana, se pasa a la capa de red, se genera una confirmación de recepción y

se avanza la ventana en uno. A diferencia de la ventana del emisor, la ventana del receptor conserva siempre el mismo tamaño inicial.

8. Un tamaño de ventana de 1 significa que la capa de enlace de datos sólo acepta tramas en orden, pero con ventanas más grandes esto no es así. La capa de red en contraste, siempre recibe los datos en el orden correcto, sin importar el tamaño de la ventana de la capa de enlace de datos.

4.4.1. Protocolo de ventana deslizante de un bit

Este protocolo utiliza parada y espera, ya que el emisor envía una trama y espera su confirmación de recepción antes de transmitir la siguiente. En la figura 4.13 se muestra este protocolo.

Este protocolo comienza por definir algunas variables. Next_frame_to_send indica qué trama está tratando de enviar el emisor. De manera semejante, frame_expected indica qué trana espera el receptor. En ambos casos, 0 y 1 son las únicas posibilidades. Normamente, una de las dos capas de enlace de datos es la que comienza a transmitir la primera trama. Es decir, **sólo uno de los programas de capa de enlace de datos debe contener las llamadas de procedimiento to_physical_layer y start_timer fuera del ciclo principal**. De lo contrario puede surgir una situación peculiar que termina duplicando paquetes en la mitad de las tramas **aunque no existan errores de transmisión**.

La máquina que arranca obtiene el primer paquete de su capa de red, construye una trama y la envía. Al llegar esta (o cualquier) trama, la capa de enlace de datos receptora la revisa para saber si es un duplicado, igual que en el protocolo 3. Si la trama es la esperada, se pasa a la capa de red y la ventana del receptor se corre hacia arriba. El campo de confirmación de recepción contiene el número de la última trama recibida sin error. Si este número concuerda con el de secuencia de la trama que está tratando de enviar el emisor, éste sabe que ha terminado con la trama almacenada en el búfer y que puede obtener el siguiente paquete de su capa de red. Si el número de secuencia no concuerda, debe continuar intentando enviar la misma trama. Por cada trama que se recibe, se devuelve una.

4.4.2. Protocolo de ventana deslizante de retroceso N

Hasta ahora hemos supuesto que el tiempo de propagación necesario para que una trama llegue al receptor más el necesario para que la confirmación de recepción regrese

```
/* Protocol 4 (sliding window) is bidirectional. */

#define MAX_SEQ 1                       /* must be 1 for protocol 4 */
typedef enum {frame_arrival, cksum_err, timeout} event_type;
#include "protocol.h"
void protocol4 (void)
{
  seq_nr next_frame_to_send;            /* 0 or 1 only */
  seq_nr frame_expected;                /* 0 or 1 only */
  frame r, s;                           /* scratch variables */
  packet buffer;                        /* current packet being sent */
  event_type event;
  next_frame_to_send = 0;               /* next frame on the outbound stream */
  frame_expected = 0;                   /* frame expected next */
  from_network_layer(&buffer);          /* fetch a packet from the network layer */
  s.info = buffer;                      /* prepare to send the initial frame */
  s.seq = next_frame_to_send;           /* insert sequence number into frame */
  s.ack = 1 – frame_expected;           /* piggybacked ack */
  to_physical_layer(&s);                /* transmit the frame */
  start_timer(s.seq);                   /* start the timer running */
  while (true) {
      wait_for_event(&event);           /* frame_arrival, cksum_err, or timeout */
      if (event == frame_arrival) {     /* a frame has arrived undamaged. */
          from_physical_layer(&r);      /* go get it */
          if (r.seq == frame_expected) { /* handle inbound frame stream. */
              to_network_layer(&r.info);    /* pass packet to network layer */
              inc(frame_expected);      /* invert seq number expected next */
          }
          if (r.ack == next_frame_to_send) {        /* handle outbound frame stream. */
              stop_timer(r.ack);        /* turn the timer off */
              from_network_layer(&buffer);      /* fetch new pkt from network layer */
              inc(next_frame_to_send);/* invert sender's sequence number */
          }
      }
      s.info = buffer;                  /* construct outbound frame */
      s.seq = next_frame_to_send;       /* insert sequence number into it */
      s.ack = 1 – frame_expected;       /* seq number of last received frame */
      to_physical_layer(&s);            /* transmit a frame */
      start_timer(s.seq);               /* start the timer running */
  }
}
```

Figura 4.13: *Protocolo de ventana deslizante de 1 bit*

es insignificante. A veces esta suposición es totalmente falsa. En estas situaciones el tiempo de viaje de ida y vuelta puede tener implicaciones importantes para la eficiencia del aprovechamiento del ancho de banda.

Por ejemplo, sea un canal de satélite de 50 kbps de velocidad de transmisión, y con un retardo de propagación de ida y vuelta de 500 mseg. Por tanto,

$$T_{BIT} = \frac{1}{50000} = 0,00002 seg.$$

Imaginemos que intentamos utilizar el protocolo 4 para enviar tramas de 1000 bits por medio del satélite. Por tanto, el tiempo que el emisor tarda en poner una trama en el canal es

$$T_{TRAMA} = T_{BIT} \times 1000 bits = 0,02 seg. = 20 mseg.$$

Es decir, el emisor empieza a enviar la primera trama en $t = 0$. En $t = 20$ mseg la trama ha sido enviada por completo. En las mejores circunstancias (sin esperas en el receptor y una trama de confirmación de recepción corta), no es sino hasta $t = 270 mseg$ que la trama ha llegado por completo al receptor, hasta $t = 520 mseg$ no llega la confirmación de recepción de regreso al emisor. Esto implica que el emisor estuvo bloqueado durante el 500/520 = 96 % del tiempo. En otras palabras, sólo se usó el 4 % del ancho de banda disponible.

La combinación de un tiempo de propagación grande, un ancho de banda alto y una longitud de tramas corta es desastrosa para la eficiencia de la red. Este problema puede verse como una consecuencia de la regla que requiere que el emisor espere una confirmación de recepción antes de enviar otra trama. Si relajamos esa restricción se puede lograr una mejor eficiencia. Básicamente la solución está en permitir que el emisor envíe hasta w tramas antes de bloquearse, en lugar de sólo 1. Con una selección adecuada de w, el emisor podrá transmitir trmas continuamente durante un tiempo igual al tiempo de tránsito de ida y vuelta sin llenar la ventana.

En el ejemplo anterior, w debe ser de cuando menos 26. El emisor comienza por enviar la trama 0, como antes. Para cuando ha terminado de enviar 26 tramas, en $t = 520$, llega la confirmación de recepción de la trama 0. A partir de entonces, las confirmaciones de recepción llegarán cada 20 mseg, por lo que el emisor siempre tendrá permiso de continuar justo cuando lo necesita. En todo momento hay 25 o 26 tramas pendientes de confirmación de recepción. Dicho de otra manera, el tamaño máximo de la ventana del emisor es de 26.

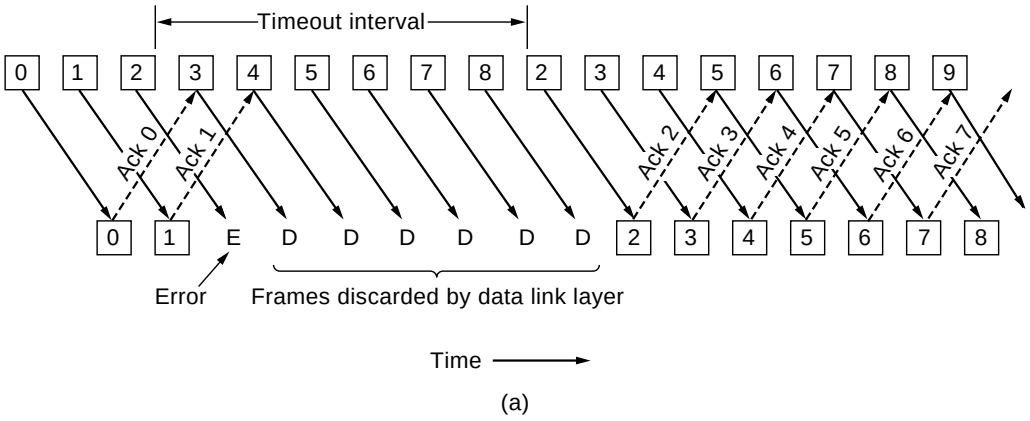
La necesidad de una ventana grande en el lado emisor se presenta cuando el producto del ancho de banda por el retardo del viaje de ida y vuelta es grande. Si el ancho de banda es alto, incluso para un retardo moderado, el emisor agotará su ventana rápidamente a menos que tenga una ventana grande. Si el retardo es grande (por ejemplo, en un canal de satélite geoestacionario), el emisor agotará su ventana incluso con un ancho de banda moderado. El producto de estos dos factores indica básicamente cuál es la capacidad del canal, y el emisor necesita la capacidad de llenarlo sin detenerse para poder funcionar con una eficiencia máxima. Esta técnica se conoce como **canalización**.

Si la capacidad del canal es de b bits/seg, el tamaño de la trama de L bits y el tiempo de propagación de ida y vuelta de R segundos, el tiempo requerido para transmitir una sola trama es de L/b segundos. Una vez que ha sido enviado el último bit de una trama de datos, hay un retardo de $R/2$ antes de que llegue ese bit al receptor y un retardo de cuando menos $R/2$ para que la confirmación de recepción llegue de regreso, lo que da un retardo total de R. En parada y espera, la línea está ocupada durante L/b e inactiva durante R, dando una utilización de la línea de $L/(L + bR)$

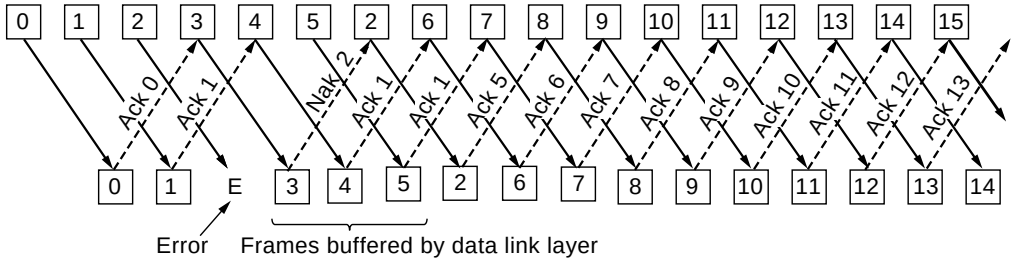
Si L ¡bR, la eficiencia será menor que 50 %. Ya que siempre hay un retardo diferente de cero para que la confirmación de recepción se propague de regreso, en principio la canalización puede servir para mantener ocupada la línea durante este intervalo, pero si el intervalo es pequeño, la complejidad adicional no justifica el esfuerzo.

El envío de tramas en canalización por un canal de comunicación inestable presenta problemas serios. Primero, ¿Qué ocurre si una trama a la mitad de una serie larga de tramas se daña o pierde?. Llegarán grandes cantidades de tramas sucesivas al receptor antes de que el emisor se entere de que algo anda mal. Cuando llega una trama dañada al receptor, obviamente debe desccartarse, pero, ¿qué debe hacerse con las tramas correctas que le siguen?. La capa de enlace de datos receptora está obligada a entregar paquetes a la capa de red en secuencia. En la figura 4.14 se muestra el efecto de la canalización en la recuperación de un error.

Hay dos métodos básicos para manejar los errores durante la canalización. Una manera, llamada **retroceso n**, es que el receptor simplemente descarte todas las tramas subsecuentes a la errónea, sin enviar confirmaciones de recepción para las tramas descartadas. Esta estrategia corresponde a una ventana de recepción de tamaño 1. Es decir, la capa de enlace de datos se niega a aceptar cualquier trama excepto la siguiente que debe entregar a la capa de red. Si la ventana del emisor se llena antes de terminar el temporizador, el canal comenzará a vaciarse. En algún momento, el emisor terminará de esperar y retransmitirá en orden todas las tramas cuya recepción aún no se haya confirmado, comenzando



(a)



(b)

Figura 4.14: *Canalización y recuperación de un error. (a) Efecto de un error cuando el tamaño de la ventana del receptor es de 1. (b) Efecto de un error cuando el tamaño de la ventana del receptor es grande.*

por la dañada o perdida. Esta estrategia puede desperdiciar bastante ancho de banda si la tasa de errores es alta. En la figura 4.14(a) se muestra el retroceso N en el caso en que la ventana del receptor es grande.

La otra estrategia general para el manejo de errores cuando las tramas se colocan en canalizaciones se conoce como **repetición selectiva**. Cuando se utiliza, se descarta una trama dañada recibida, pero las tramas en buen estado recibidas después de ésa se almacenan en el búfer. Cuando el emisor termina, sólo la última trama sin confirmación se retransmite. Si la trama llega correctamente, el receptor puede entregar a la capa de red en secuencia, todas las tramas que ha almacenado en el búfer. La repetición selectiva con frecuencia se combina con el hecho de que el receptor envíe una confirmación de recepción negativa (NAK) cuando detecta un error, por ejemplo, cuando recibe un error de suma de verificación o una trama en desorden. Las confirmaciones de recepción negativas estimulan la retransmisión antes de que el temporizador correspondiente expire y, por lo tanto, mejoran el rendimiento. En la figura 4.14(b), las tramas 0 y 1 se reciben correctamente y la trama 2 se pierde. Cuando la trama 3 llega al receptor, su capa de enlace de datos observa que falta una trama, por lo que devuelve una NAK para la trama 2 pero almacena la trama 3. Cuando las tramas 4 y 5 llegan, también son almacenadas por la capa de enlace de datos en lugar de pasarse a la capa de red. En algún momento, la NAK 2 llega al emisor, que inmediatamente reenvía la trama 2. Cuando llega, la capa de enlace de datos ahora tiene 2, 3, 4 y 5 y ya las puede pasar a la capa de red en el orden correcto. También puede confirmar la recepción de todas las tramas hasta e incluyendo la 5, como se muestra en la figura.

Si la NAK se perdiera, en algún momento el temporizador del emisor expirará para la trama 2 y la enviará (sólo a ella), pero eso puede tardar un poco más. En efecto, la NAK acelera la retransmisión de una trama específica. La repetición selectiva corresponde a una ventana del receptor mayor que 1 . Esta estrategia puede requerir cantidades grandes de memoria en la capa de enlace de datos si la ventana es grande. Estas dos estrategias altemativas son compromisos entre el ancho de banda y el espacio de búfer en la capa de enlace de datos.

En la figuras 4.4.2 y 4.15 se muestra un protocolo de canalización en el que la capa de enlace de datos receptora sólo acepta tramas en orden; las tramas siguientes a un error son descartadas. En la figuras 4.4.2 y 4.16 se muestra un protocolo de repetición selectiva.

La recepción no secuencial introduce ciertos problemas que no se presentan en los protocolos en los que las tramas sólo se aceptan en orden. Vamos a ver un ejemplo. Supongamos que tenemos un número de secuencia de tres bits, por lo que se permite al emisor enviar hasta siete tramas antes de recibir una confirmación de recepción. Inicialmente las

```
/* Protocol 5 (go back n) allows multiple outstanding frames. The sender may transmit up
   to MAX_SEQ frames without waiting for an ack. In addition, unlike in the previous
   protocols, the network layer is not assumed to have a new packet all the time. Instead,
   the network layer causes a network_layer_ready event when there is a packet to send. */

#define MAX_SEQ 7                       /* should be 2^n – 1 */
typedef enum {frame_arrival, cksum_err, timeout, network_layer_ready} event_type;
#include "protocol.h"

static boolean between(seq_nr a, seq_nr b, seq_nr c)
{
/* Return true if a <=b < c circularly; false otherwise. */
  if (((a <= b) && (b < c)) || ((c < a) && (a <= b)) || ((b < c) && (c < a)))
       return(true);
    else
       return(false);
}

static void send_data(seq_nr frame_nr, seq_nr frame_expected, packet buffer[ ])
{
/* Construct and send a data frame. */
  frame s;                              /* scratch variable */

  s.info = buffer[frame_nr];            /* insert packet into frame */
  s.seq = frame_nr;                     /* insert sequence number into frame */
  s.ack = (frame_expected + MAX_SEQ) % (MAX_SEQ + 1);/* piggyback ack */
  to_physical_layer(&s);                /* transmit the frame */
  start_timer(frame_nr);                /* start the timer running */
}

void protocol5(void)
{
  seq_nr next_frame_to_send;            /* MAX_SEQ > 1; used for outbound stream */
  seq_nr ack_expected;                  /* oldest frame as yet unacknowledged */
  seq_nr frame_expected;                /* next frame expected on inbound stream */
  frame r;                              /* scratch variable */
  packet buffer[MAX_SEQ + 1];           /* buffers for the outbound stream */
  seq_nr nbuffered;                     /* # output buffers currently in use */
  seq_nr i;                             /* used to index into the buffer array */
  event_type event;

  enable_network_layer();               /* allow network_layer_ready events */
  ack_expected = 0;                     /* next ack expected inbound */
  next_frame_to_send = 0;               /* next frame going out */
  frame_expected = 0;                   /* number of frame expected inbound */
  nbuffered = 0;                        /* initially no packets are buffered */
```

```
  while (true) {
    wait_for_event(&event);              /* four possibilities: see event_type above */

    switch(event) {
      case network_layer_ready:          /* the network layer has a packet to send */
           /* Accept, save, and transmit a new frame. */
           from_network_layer(&buffer[next_frame_to_send]); /* fetch new packet */
           nbuffered = nbuffered + 1;    /* expand the sender's window */
           send_data(next_frame_to_send, frame_expected, buffer);/* transmit the frame */
           inc(next_frame_to_send);      /* advance sender's upper window edge */
           break;

      case frame_arrival:                /* a data or control frame has arrived */
           from_physical_layer(&r);      /* get incoming frame from physical layer */

           if (r.seq == frame_expected) {
                /* Frames are accepted only in order. */
                to_network_layer(&r.info); /* pass packet to network layer */
                inc(frame_expected);     /* advance lower edge of receiver's window */
            }

            /* Ack n implies n – 1, n – 2, etc.  Check for this. */
           while (between(ack_expected, r.ack, next_frame_to_send)) {
                /* Handle piggybacked ack. */
                nbuffered = nbuffered – 1; /* one frame fewer buffered */
                stop_timer(ack_expected);     /* frame arrived intact; stop timer */
                inc(ack_expected);       /* contract sender's window */
           }
           break;

      case cksum_err: break;             /* just ignore bad frames */

      case timeout:                      /* trouble; retransmit all outstanding frames */
           next_frame_to_send = ack_expected;  /* start retransmitting here */
           for (i = 1; i <= nbuffered; i++) {
                send_data(next_frame_to_send, frame_expected, buffer);/* resend 1 frame */
                inc(next_frame_to_send); /* prepare to send the next one */
           }

    }

    if (nbuffered < MAX_SEQ)
           enable_network_layer();
    else
           disable_network_layer();
  }
}
```

Figura 4.15: *Protocolo de retroceso N*

```
/* Protocol 6 (selective repeat) accepts frames out of order but passes packets to the
   network layer in order. Associated with each outstanding frame is a timer. When the timer
   expires, only that frame is retransmitted, not all the outstanding frames, as in protocol 5. */

#define MAX_SEQ 7                       /* should be 2^n – 1 */
#define NR_BUFS ((MAX_SEQ + 1)/2)
typedef enum {frame_arrival, cksum_err, timeout, network_layer_ready, ack_timeout} event_type;
#include "protocol.h"
boolean no_nak = true;                  /* no nak has been sent yet */
seq_nr oldest_frame = MAX_SEQ + 1;      /* initial value is only for the simulator */

static boolean between(seq_nr a, seq_nr b, seq_nr c)
{
/* Same as between in protocol5, but shorter and more obscure. */
  return ((a <= b) && (b < c)) || ((c < a) && (a <= b)) || ((b < c) && (c < a));
}

static void send_frame(frame_kind fk, seq_nr frame_nr, seq_nr frame_expected, packet buffer[ ])
{
/* Construct and send a data, ack, or nak frame. */
  frame s;                              /* scratch variable */

  s.kind = fk;                          /* kind == data, ack, or nak */
  if (fk == data) s.info = buffer[frame_nr % NR_BUFS];
  s.seq = frame_nr;                     /* only meaningful for data frames */
  s.ack = (frame_expected + MAX_SEQ) % (MAX_SEQ + 1);
  if (fk == nak) no_nak = false;        /* one nak per frame, please */
  to_physical_layer(&s);                /* transmit the frame */
  if (fk == data) start_timer(frame_nr % NR_BUFS);
  stop_ack_timer();                     /* no need for separate ack frame */
}

void protocol6(void)
{
  seq_nr ack_expected;                  /* lower edge of sender's window */
  seq_nr next_frame_to_send;            /* upper edge of sender's window + 1 */
  seq_nr frame_expected;                /* lower edge of receiver's window */
  seq_nr too_far;                       /* upper edge of receiver's window + 1 */
  int i;                                /* index into buffer pool */
  frame r;                              /* scratch variable */
  packet out_buf[NR_BUFS];              /* buffers for the outbound stream */
  packet in_buf[NR_BUFS];               /* buffers for the inbound stream */
  boolean arrived[NR_BUFS];             /* inbound bit map */
  seq_nr nbuffered;                     /* how many output buffers currently used */
  event_type event;

  enable_network_layer();               /* initialize */
  ack_expected = 0;                     /* next ack expected on the inbound stream */
  next_frame_to_send = 0;               /* number of next outgoing frame */
  frame_expected = 0;
  too_far = NR_BUFS;
  nbuffered = 0;                        /* initially no packets are buffered */
  for (i = 0; i < NR_BUFS; i++) arrived[i] = false;
```

```
while (true) {
  wait_for_event(&event);                 /* five possibilities: see event_type above */
  switch(event) {
    case network_layer_ready:             /* accept, save, and transmit a new frame */
         nbuffered = nbuffered + 1;       /* expand the window */
         from_network_layer(&out_buf[next_frame_to_send % NR_BUFS]); /* fetch new packet */
         send_frame(data, next_frame_to_send, frame_expected, out_buf);/* transmit the frame */
         inc(next_frame_to_send);         /* advance upper window edge */
         break;

    case frame_arrival:                   /* a data or control frame has arrived */
         from_physical_layer(&r);         /* fetch incoming frame from physical layer */
         if (r.kind == data) {
              /* An undamaged frame has arrived. */
              if ((r.seq != frame_expected) && no_nak)
                 send_frame(nak, 0, frame_expected, out_buf); else start_ack_timer();
              if (between(frame_expected, r.seq, too_far) && (arrived[r.seq%NR_BUFS] == false)) {
                   /* Frames may be accepted in any order. */
                   arrived[r.seq % NR_BUFS] = true;/* mark buffer as full */
                   in_buf[r.seq % NR_BUFS] = r.info;/* insert data into buffer */
                   while (arrived[frame_expected % NR_BUFS]) {
                        /* Pass frames and advance window. */
                        to_network_layer(&in_buf[frame_expected % NR_BUFS]);
                        no_nak = true;
                        arrived[frame_expected % NR_BUFS] = false;
                        inc(frame_expected);  /* advance lower edge of receiver's window */
                        inc(too_far);     /* advance upper edge of receiver's window */
                        start_ack_timer();  /* to see if a separate ack is needed */
                   }
              }
         }
         if((r.kind==nak) && between(ack_expected,(r.ack+1)%(MAX_SEQ+1),next_frame_to_send))
              send_frame(data, (r.ack+1) % (MAX_SEQ + 1), frame_expected, out_buf);

         while (between(ack_expected, r.ack, next_frame_to_send)) {
              nbuffered = nbuffered – 1;    /* handle piggybacked ack */
              stop_timer(ack_expected % NR_BUFS);/* frame arrived intact */
              inc(ack_expected);          /* advance lower edge of sender's window */
         }
         break;
    case cksum_err:
         if (no_nak) send_frame(nak, 0, frame_expected, out_buf); /* damaged frame */
         break;
    case timeout:
         send_frame(data, oldest_frame, frame_expected, out_buf); /* we timed out */
         break;
    case ack_timeout:
         send_frame(ack,0,frame_expected, out_buf);/* ack timer expired; send ack */
  }
  if (nbuffered < NR_BUFS) enable_network_layer(); else disable_network_layer();
 }
}
```

Figura 4.16: *Protocolo de repetición selectiva*

ventanas del emisor y del receptor están como se muestra en la figura 4.17(a).

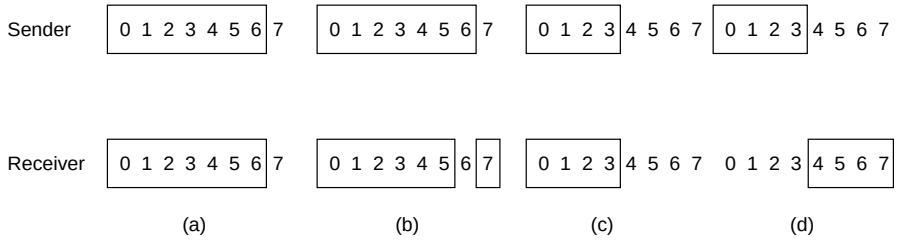


Figura 4.17: *(a) Situación original con una ventana de tamaño 7. (b) Después de que se han enviado y recibido siete tramas, pero su recepción no se ha confirmado. (c) Situación inicial con un tamaño de ventana de 4. (d) Después de que se han enviado y recibido cuatro tramas, pero su recepción no se ha confirmado.*

El emisor ahora transmite las tramas 0 a 6. La ventana del receptor le permite aceptar cualquier trma con un número de secuencia entre 0 y 6, inclusive. Las siete tramas llegan correctamente, por lo que el receptor confirma su recepción y avanza su ventana para permitir la recepción de 7, 0, l, 2, 3, 4 o 5, como se muestra en la figura 4.17(b). Los siete búferes se marcan como vacíos.

Si en este punto se borran todas las confirmaciones de recepción, en algún momento termina el temporizador del emisor y retransmite la trama 0. Cuando esta trama llega al receptor, se efectúa una verificación para saber si está dentro de la ventana del receptor. Desgraciadamente, en la figura 4.17(b) la trama 0 está dentro de la nueva ventana, por lo que se acepta. El receptor envía una confirmación de recepción, superpuesta, para la trama 6, ya que se han recibido de la 0 a la 6. El emisor se entera que todas sus tramas transmitidas han llegado de manera correcta, por lo que avanza su ventana y envía de inmediato las tramas 7, 0, l, 2, 3, 4 y 5. El receptor aceptará la trama 7 y el paquete de ésta se pasará directamente a la capa de red. Inmediatamente después, la capa de enlace de datos receptora revisa si ya tiene una trama 0 válida, descubre que sí y pasa el paquete que contiene a la capa de red. En consecuencia, la capa de red obtiene un paquete incorrecto, y falla el protocolo.

La esencia del problema es que una vez que el receptor ha avanzado su ventana, el nuevo intervalo de números de secuencia válidos se solapa con el anterior. En consecuencia, el siguiente grupo de tramas podría ser de tramas duplicadas (si se perdieron todas las confirmaciones de recepción) o de nuevas (si se recibieron todas las confirmaciones de recepción). El receptor no tiene manera de distinguir entre estos dos casos.

La solución de este problema es asegurarse que, una vez que el emisor haya avanzado su ventana, no haya solape con la ventana original. Para ello, el tamaño máximo de la ventana debe ser como máximo la mitad del intervalo de los números de secuencia, como en las figura 4.17(c) y figura 4.17(d). Por ejemplo, si se utilizan 4 bits para los números de secuencia, éstos tendrán un intervalo de 0 a 15. Sólo ocho tramas sin confumación de recepción deben estar pendientes en cualquier instante. De esa manera si el receptor apenas ba aceptado las tramas 0 a 7 y ha avanzado su ventana para permitir la aceptación de las tramas 8 a 15, puede distinguir sin ambigüedades si las tramas subsiguientes son retransmisiones (0 a 7) o nuevas (8 a 15). En general, el tamaño de la ventana para el protocolo 6 será (Max_sec + 1)/2. Por lo tanto, para números de secuencia de 3 bits, el tamaño de ventana es 4.

4.5. Ejemplos de protocolos de enlace de datos

Vamos a estudiar dos protocolos de los más amplicamente utilizados en la capa de enlace de datos. El primero, HDLC, es un protocolo clásico orientado a bits cuyas variantes se han utilizado durante décadas en muchas aplicaciones. El segundo, PPP, es un protocolo de enlace utilizado para conectar a Internet los computadores domésticos.

4.5.1. HDLC-Control de Enlace de Datos de Alto Nivel

Usando como base la técnica de ventana deslizante que hemos descrito IBM desarrolló en 1972 un protocolo de enlace denominado SDLC (Synchronous Data Link Control Protocol) para las redes SNA. Posteriormente IBM propuso SDLC para su estandarización a ANSI e ISO; En particular el protocolo desarrollado por ISO se denominó HDLC (High level Data Link Control) e introducía diversas mejoras sobre el protocolo originalmente desarrollado por IBM. La inmensa mayoría de los protocolos de enlace utilizados actualmente son subconjuntos del HDLC; una lista de los más representativos aparece en la tabla 4.1. Dada su importancia comentaremos ahora los aspectos más relevantes del protocolo HDLC.

HDLC puede ofrecer dos tipos de servicio:

No orientado a conexión y sin acuse de recibo. En este caso el receptor simplemente comprobará el CRC y descartará la trama si detecta que es errónea, pero no enviará ninguna

Protocolo	Nombre completo	Uso
HDLC	High level Data Link Control	Estándar ISO
ADCCP	Advanced Data Communications Control Procedure	Estándar ANSI
LLC	Logical Link Control	Estándar IEEE 802.2
LAP-B	Link Access Procedure Balanced	X.25
LAP-D	Link Access Procedure D-channel	RDSI (Señalización)
LAP-F	Link Access Procedure for Frame Mode Bearer Services	Frame Relay
LAP-M	Link Access Procedure - Modem	Módems RTC
PPP	Point to Point Protocol	Estándar Internet
SDLC	Synchronous Data Link Protocol	SNA (IBM)

Cuadro 4.1: Algunos miembros de la familia de protocolos HDLC

notificación de este hecho al emisor. Como era de esperar en este caso el protocolo es muy simple.

Orientado a conexión con acuse de recibo. En este caso se utilizará un mecanismo de ventana deslizante con retroceso n (o repetición selectiva en algunos casos). El número de secuencia es normalmente de tres bits, aunque algunas también se contempla la posibilidad de utilizar números de secuencia de 7 bits. En todos los casos el acuse de recibo viaja a ser posible en tramas de datos (ACK piggybacked).

Actualmente se utiliza casi siempre el servicio no orientado a conexión.

La estructura de la trama HDLC es la que aparece en la figura 4.18

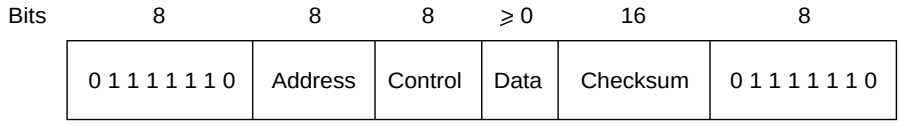


Figura 4.18: *Formato de trama para protocolos orientados a bits*

La trama se delimita mediante la secuencia 01111110, y para asegurar la transparencia de datos se utiliza relleno de bits (bit stuffing), es decir, se intercala un bit a 0 cuando en la parte de datos aparece una secuencia de cinco bits a 1, procediendo de modo inverso en el lado receptor. En sistemas síncronos cuando la línea no está transmitiendo información útil se envía continuamente la secuencia 011111101111110111111O.

Cada trama puede tener cualquier longitud a partir de 32 bits (sin contar los delimitadores), pudiendo no ser múltiplo de 8, ya que no se presupone una estructura de bytes. Por esto se suele decir que HDLC es un protocolo orientado al bit (en contraste con los que requieren que la longitud de la trama sea múltiplo de 8, que se denominan orientados al byte).

El campo checksum es un CRC que utiliza el generador polinómico CRC-CCITT.

El campo datos, también llamado en ocasiones carga útil (payload) puede o no estar presente; puede contener cualquier información y tener cualquier longitud, si bien la eficiencia del checksum disminuye cuando la longitud aumenta.

El campo dirección solo se utiliza en líneas multipunto. Las líneas multipunto son unas líneas hoy en día poco utilizadas en las que varios ordenadores o terminales comparten una misma línea física; dado que las líneas multipunto son a fin de cuentas un medio compartido para resolver el problema de acceso al medio se designa un ordenador que actúa como 'moderador' dando el turno de palabra a los demás. El campo dirección permite identificar a cual de todos los ordenadores o terminales que comparten la línea va dirigida la trama. Cuando se quiere enviar una trama a todas las estaciones (envío broadcast) se utiliza la dirección 11111111. En líneas punto a punto el campo dirección suele contener la dirección broadcast.

El campo control es realmente el corazón del protocolo. Cuando el primer bit es un cero indica que se trata de una trama de datos, también llamada de información. En ese caso la estructura de este campo es la de la figura 4.19 a).

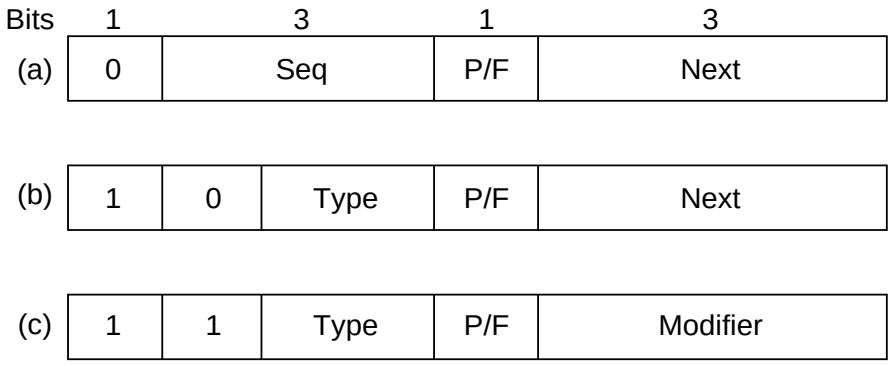


Figura 4.19: *Campo de control para los distintos tipos de trama*

El subcampo SEQ contiene el número de secuencia de la trama.

El subcampo P/F (Polling/Final) solo se utiliza en líneas multipunto y no lo comentaremos.

El subcampo NEXT contiene el ACK (piggybacked); el convenio en este caso es que el ACK indica la siguiente trama que se espera recibir, no la última recibida (evidentemente se supone que esa trama habrá sido recibida correctamente).

Cuando el primer bit del campo control es un 1 y el segundo un 0 se trata de una **trama de supervisión**. Estas tramas se utilizan para enviar los ACK cuando no hay tráfico de datos suficiente y también para algunos mensajes de control. La estructura que tienen es la de la figura 4.19 b).

Según el valor del subcampo Type las tramas de supervisión podrán enviar los siguientes cuatro comandos:

00 (Tipo 0): RECEIVE READY. Este es el nombre que recibe en el estándar el acuse de recibo (ACK). Este comando se utiliza cuando no hay tráfico de retorno suficiente para utilizar piggybacking.

01 (Tipo 1): REJECT. Corresponde al acuse de recibo negativo (NAK). Solicita retransmisión de una trama, y no acepta ninguna otra entretanto. Se utiliza cuando se emplea el mecanismo de retroceso n. La trama solicitada se especifica en el campo NEXT.

10 (Tipo 2): RECEIVE NOT READY. Indica un acuse de recibo (ACK) pero además solicita la suspensión del envío para evitar la saturación del receptor (control de flujo); el receptor enviará este mensaje cuando vea que tiene poco espacio para buffers. Para que la retransmisión se reanude el receptor deberá enviar más tarde un RECEIVE READY, REJECT o ciertas tramas de control.

11 (Tipo 3): SELECTIVE REJECT. Se utiliza para solicitar retransmisión de una trama determinada cuando se está empleando retransmisión selectiva. En este caso por tanto la ventana del emisor con un número de secuencia de tres bits no puede ser mayor de 4. Este mecanismo solo está previsto en HDLC y ADCCP, no en SDLC ni LAP-B.

En todos los casos el subcampo NEXT indica la siguiente trama que se espera recibir (o la que no se espera recibir en el caso de RECEIVE NOT READY).

En HDLC y LAP-B existe un tipo de trama extendida en la que los números de secuencia son de 7 bits; en este caso es posible utilizar un tamaño de ventana de hasta 127 en la técnica de retroceso n o de 64 con la de repetición selectiva.

Existe en HDLC un tercer tipo de trama que es el que se da cuando los dos primeros bits son 1; este tipo de tramas se denomina **no numeradas** y se utilizan para dos funciones completamente diferentes:

- Establecer determinados parámetros de inicialización del protocolo.
- Cuando se utiliza el servicio no orientado a conexión, es decir sin ACK. En este caso no es necesario numerar las tramas ya que no se pedirá retransmisión en ningún caso, de ahí la denominación de trama no numerada.

La estructura del campo control en las tramas no numeradas es la que aparece en la figura 4.19 c). En conjunto hay 5 bits que sirven para especificar diversos comandos que no detallaremos.

4.5.2. La capa de enlace de datos en Internet: Protocolo Punto a Punto (PPP)

Internet consiste en máquinas individuales (hosts y encaminadores) y la infraestructura de comunicación que las conecta. Dentro de un solo edificio, las LANs se usan ampliamente para la interconexión, pero la mayor parte de la infraestructura de área amplia está construida a partir de líneas alquiladas punto a punto. En la práctica, la comunicación punto a punto se utiliza principalmente en dos situaciones. Primero, miles de organizaciones tienen una o más LANs, cada una con cierta cantidad de hosts (computadoras personales, estaciones de trabajo, servidores, etc.) junto con un encaminador o router. Por lo general, todas las conexiones al mundo exterior pasan a través de uno o dos encaminadores que tienen líneas alquiladas punto a punto a encaminadores distantes. Son estos encaminadores y sus líneas arrendadas los que conforman las subredes de comunicación sobre las que está construida Internet.

La segunda situación en la que las líneas punto a punto desempeñan un papel principal en Internet son los millones de personas que tienen conexiones domésticas a Internet a través de módems y líneas de acceso telefónico. Generalmente lo que ocurre es que el PC doméstico del usuario se conecta a un encaminador del proveedor de servicios de Internet y luego actúa como host de Internet. La figura 4.20 ilustra esta situación.

Tanto para la conexión por línea alquilada de encaminador a encaminador como para la conexión de acceso telefónico de host a encaminador, en la línea se requiere un

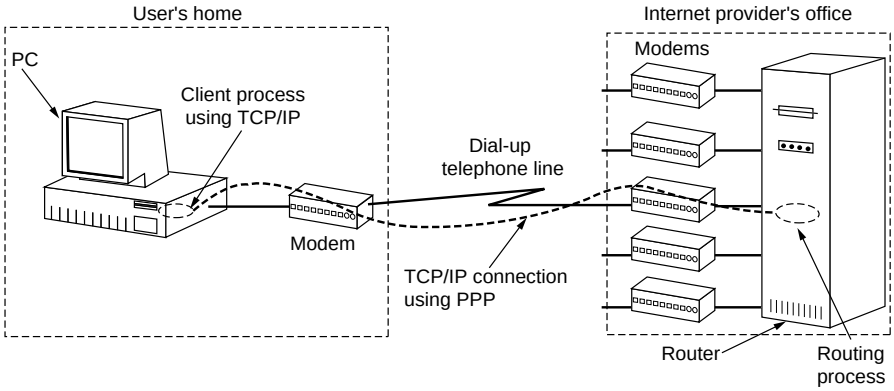


Figura 4.20: *Computador personal doméstico que funciona como host de Internet.*

protocolo de enlace de datos punto a punto para el entramado, el control de errores y las demás funciones de la capa de enlace de datos que hemos estudiado en este capítulo. El que se utiliza en Intemet se conoce como PPP.

Internet necesita de un protocolo punto a punto para diversos propósitos, entre ellos para el tráfico encaminador a encaminador y tráfico usuario doméstico a ISP. Este protocolo es PPP (Protocolo Punto a Punto), que se define en el RFC 1661 y que se ha desarrollado más en varios otros RFCs (por ejemplo, los RFCs 1662 y 1663). PPP realiza detección de errores, **soporta múltiples protocolos**, permite la negociación de direcciones de IP en el momento de la conexión, permite la autenticación y tiene muchas otras funciones. PPP proporciona tres características:

1. Un método de entramado que delimita final de una trama y el inicio de la siguiente. El formato de trama también maneja la detección de errores.

2. Un protocolo de control de enlace para activar líneas, probarlas, negociar opciones y desactivarlas ordenadamente cuando ya no son necesarias. Este protocolo se llama LCP (Protocolo de Control de Enlace). Admite circuitos síncronos y asíncronos y codificaciones orientadas a bits y a caracteres.

3. Un mecanismo para negociar opciones de capa de red con independencia del protocolo de red usado. El método escogido consiste en tener un NCP (Protocolo de Control de Red) distinto para cada protocolo de capa de red soportado.

Para ver la manera en que encajan estas piezas, consideremos la situación típica de

un usuario doméstico llamando al proveedor de servicios de Internet para convertir un PC doméstico en un host temporal de Internet.

1. El PC llama iniciamente al encaminador del proveedor a través de un módem (cable módem o módem ADSL).
2. Una vez que el módem del encaminador ha contestado el teléfono y ha establecido una conexión fisica, el PC manda al encaminador una serie de paquetes LGP en el campo de carga útil de una o más tramas PPP. Estos paquetes y sus respuestas seleccionan los parámetros PPP por usar.
3. Una vez que se han acordado estos parámetros, se envía una serie de paquetes NCP para configurar la capa de red. Por lo general el PC requiere ejecutar una pila de protocolos TCP/IP, por lo que necesita una dirección IP.
4. No hay suficientes direcciones IP para todos, por lo que normalmente cada proveedor de Internet tiene un bloque de ellas y asigna de manera dinámica una a cada PC que se acaba de conectar para que la use durante su sesión. Si un proveedor posee n direcciones IP, puede tener hasta n máquinas conectadas en forma simultánea, pero su base total de clientes puede ser muchas veces mayor. Se utiliza el NCP para IP para asignar la dirección IP.
5. En este momento, el PC es ya un host de Internet y puede enviar y recibir paquetes IP, igual que los host permanentes.
6. Cuando el usuario ha terminado, se utiliza NCP para desmantelar la conexión de la capa de red y liberar la dirección IP. Luego se usa LCP para cancelar la conexión de la capa de enlace de datos. Por último, la computadora indica al módem que cuelgue el teléfono, liberando la conexión de la capa fisica.

El formato de trama de PPP se escogió de modo que fuera muy parecido al de HDLC. La diferencia principal entre PPP y HDLC es que el primero está orientado a caracteres, no a bits. En particular, PPP usa el relleno de bytes en las líneas de acceso telefónico con módem, por lo que todas las tramas tienen un número entero de bytes. No es posible enviar una trama que conste de 30.25 bytes, como con HDLC. No sólo pueden mandarse tramas PPP a través de líneas de acceso telefónico, sino que también pueden enviarse a través de SONET o de líneas HDLC auténticas orientadas a bits (por ejemplo, para conexiones entre encaminadores). El formato de trama PPP se muestra en la figura 4.21.

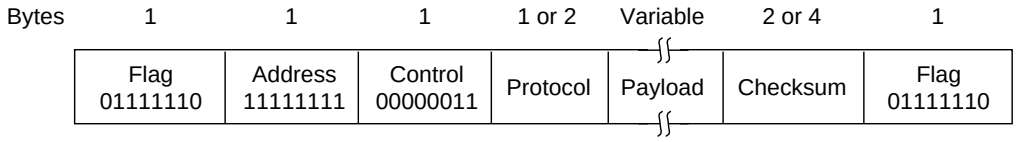


Figura 4.21: *Formato de trama completa PPP para el modo de operación no numerado.*

En función de las características del medio físico se aplicará relleno de bits, como en HDLC, o relleno de bytes (por ejemplo para transmisión por medios asíncronos). En cualquier caso el delimitador de trama es la secuencia 01111110.

El campo dirección no se utiliza. Siempre vale 11111111.

El campo control tiene por defecto el valor 00000011. Por defecto PPP suministra un servicio no orientado a conexión no fiable, es decir sin números de secuencia y acuses de recibo. Aunque no es lo normal, en el momento de establecer la conexión LCP puede negociar una transmisión fiable.

El campo protocolo establece a que tipo de protocolo pertenece el paquete recibido de la capa de red. De esta forma PPP permite establecer una comunicación multiprotocolo, es decir puede utilizarse para transmitir paquetes pertenecientes a diferentes protocolos del nivel de red entre dos ordenadores simultáneamente. Entre las posibilidades se encuentra IP, IPX(Novell), Appletalk, DECNET, OSI y otros. Los protocolos que comieNan con un bit 0 son de capa de red, como IPX, OSI CLNP, o XNS. Los que comienzan con un bit 1 se utilizan para negociar otros protocolos. Entre éstos están LCP y un NGP diferente para cada protocolo de capa de red soportado. El tamaño predeterminado del campo de protocolo es de 2 bytes, pero puede negociarse a 1 byte usando LCP.

El campo datos es de una longitud variable hasta un máximo que negocia LCP al establecer la conexión; por defecto el tamaño máximo de trama es de 1500 bytes.

El campo checksum es normalmente de 2 bytes, pero puede ser de 4 si se negocia.

PPP puede utilizarse sobre medios físicos muy diversos, por ejemplo conexiones mediante módem/RTC, RDSI, líneas dedicadas, o incluso por conexiones SONET/SDH de alta velocidad (155 o 622 Mb/s por ejemplo).

Igual que ocurre en la vida real, la negociación entre dos LCPs puede dar lugar a que todos los valores propuestos sean aceptados por la otra parte, o sólo algunos de ellos. El

protocolo establece mecanismos que permiten a los LCPs dialogar para llegar a un consenso en caso de discrepancia.

Existe otro componente de PPP que es el NCP (Network Control Protocol). Este se encarga de negociar los parámetros específicos para cada protocolo utilizado. Por ejemplo, en el caso de una conexión IP desde un usuario conectado vía módem le asigna dinámicamente una dirección IP; esto es especialmente útil cuando (como normalmente ocurre) el número de direcciones IP disponibles es menor que el número de usuarios del servicio (aunque por supuesto el número de direcciones IP disponibles debe ser suficiente para poder asignar una diferente a cada usuario simultáneo).

LCP permite utilizar diversos protocolos de autenticación, es decir que permiten validar al ordenador que llama (mediante el uso de claves tipo usuario/password). Esto resulta especialmente útil en el caso de conexiones por RTC, por ejemplo para proveedores de servicios Internet que han de facturar a sus usuarios en función del tiempo de conexión.

4.5.3. El nivel de enlace en Frame Relay

Frame Relay utiliza a nivel de enlace el protocolo denominado **LAPF (Link Access Procedure for Frame-Mode Bearer Services)**, que podemos traducir como 'procedimiento de acceso al enlace para servicios portadores en modo trama'). LAPF es una variante simplificada de HDLC que suprime la parte correspondiente al reenvío de tramas en caso de error; se construye un CRC que permite al destinatario comprobar que la trama ha llegado correctamente, pero si detecta algún error descarta silenciosamente la trama sin informar de ello al emisor (se confía en que los niveles superiores se ocuparán de ello); además LAPF requiere que la longitud de la trama a transmitir sea siempre un número entero de octetos. En general LAPF es bastante similar a PPP.

LAPF practica el relleno de bits para conseguir la transparencia de los datos. Existen dos tipos de trama, la de control y la normal que es la que se utiliza para enviar datos. La estructura de una trama LAPF normal es la que aparece en la figura 4.22

La esencia del funcionamiento de Frame Relay se encuentra en el campo Dirección, cuya estructura describiremos cuando volvamos a hablar de Frame Relay en el nivel de red.

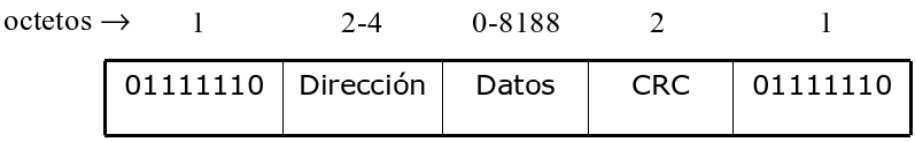


Figura 4.22: *Estructura de una trama LAPF (Frame Relay) normal.*

Bibliografía

[1] Uyless Black, *"Data Networks: Concepts, Theory and Practice", Ed. Prentice-Hall, 1989.*

[2] William Stallings, *"Data and computer communications", Ed. Prentice Hall, 8ª ed., 2008.*

[3] Andrew S. Tanenbaum, *"Computer Networks", Ed. Prentice Hall, 3ª ed., 2003.*

[4] Wayne Tomasi, *"Advanced electronic Communication Systems", Ed. Prentice Hall, 2ª ed., 1996.*