LA CAPA DE ENLACE

Autor: Rogelio Montañana

3.1 INTRODUCCIÓN

La capa de enlace, que se sitúa inmediatamente encima de la capa física, se ocupa de suministrar un transporte de bits, normalmente fiable, a la capa de red. La capa de enlace solo se ocupa de equipos física y directamente conectados, sin tener conocimiento o 'conciencia' de la red en su conjunto. Esto no quiere decir que no pueda haber ningún dispositivo en el cable que conecta los dos equipos, puede haber amplificadores o repetidores; los amplificadores son dispositivos que amplifican la señal desde el punto de vista analógico, los repetidores interpretan bit a bit la información digital contenida en la señal y la regeneran de nuevo. Los amplificadores distorsionan ligeramente la señal, por lo que si se conectan muchos en serie la deformación puede llegar a ser excesiva (algo parecido a hacer una fotocopia de fotocopia muchas veces). En cambio los repetidores, al regenerar la señal digital original no introducen ninguna distorsión y por tanto por este lado se pueden encadenar en serie sin restricciones; sin embargo consideraciones del retardo introducido en la propagación de la señal también imponen un número máximo en este caso. En cualquier caso tanto los amplificadores como los repetidores son dispositivos que funcionan a nivel físico puesto que se limitan a reproducir la señal bit a bit sin alterarla ni interpretar su significado.

Un ejemplo de nivel de enlace podría ser una línea E3 que interconecta dos ordenadores y que transcurre por un sistema SDH. La línea atravesará típicamente una serie de repetidores, ADMs y probablemente cross-connect SDH; sin embargo desde el punto de vista de esos dos ordenadores todo será equivalente a una conexión E3 PDH tradicional.

Una característica importante de la capa de enlace es que los bits han de llegar a su destino en el mismo orden en que han salido; en algunos casos *puede haber errores o pérdida de bits, pero nunca debe producirse una reordenación en el camino*.

Las principales funciones que desarrolla la capa de enlace son las siguientes:

- Agrupar los bits en grupos discretos denominados tramas. Esto permite desarrollar de forma más eficiente el resto de funciones.
- Realizar la comprobación de errores mediante el código elegido, que puede ser corrector o simplemente detector. En el caso de código corrector se procede a corregir los errores, en el de un código detector la trama errónea se descarta y opcionalmente se pide retransmisión al emisor.
- Efectuar control de flujo, es decir pedir al emisor que baje el ritmo o deje momentáneamente de transmitir porque el receptor no es capaz de asimilarla información enviada.

No todas las funciones se implementan en todos los protocolos de enlace. La retransmisión de tramas erróneas y el control de flujo a menudo se implementan en las superiores (capa de red, de transporte, o incluso en la de aplicación).

La mayoría de las funciones del nivel de enlace se implementan en el hardware de los equipos. Esto hace que los protocolos de nivel de enlace se modifiquen poco con el tiempo.

Los tipos de servicio que la capa de enlace puede suministrar a la capa de red son normalmente los siguientes:

- Servicio no orientado a conexión y sin acuse de recibo
- Servicio no orientado a conexión con acuse de recibo
- Servicio orientado a conexión con acuse de recibo

En el primer caso el envío se hace 'a la buena de dios' sin esperar ninguna indicación del receptor sobre el éxito o fracaso de la operación. Este tipo de servicio es apropiado cuando la tasa de error es muy baja (redes locales o fibra óptica) y se deja la misión de comprobar la corrección de los datos transmitidos a las capas superiores (normalmente el nivel de transporte); se considera en estos casos que la probabilidad de error es tan baja que se pierde más tiempo haciendo comprobaciones inútiles que dejando esta tarea a las capas superiores. También se usa este tipo servicio cuando se quiere transmitir información en tiempo real (por ejemplo en una vídeoconferencia) y no se quiere sufrir el retraso que impondría un servicio más sofisticado en la capa de enlace (se supone que en este caso preferimos la pequeña tasa de error del medio

físico a cambio de minimizar el retardo, o dicho de otro modo si se hiciera reenvío en caso de error sería peor el remedio que la enfermedad).

En el segundo tipo de servicio se produce un acuse de recibo para cada trama enviada. De esta manera el emisor puede estar seguro de que ha llegado. Suele utilizarse en redes con mas tasa de error, por ejemplo redes inalámbricas.

El tercer servicio es el más seguro y sofisticado. El emisor y el receptor establecen una conexión explícita de antemano, las tramas a enviar se numeran y se aseguran ambos de que son recibidas todas correctamente en su destino y transmitidas a la capa de red una vez y sólo una.

En el servicio orientado a conexión se pueden distinguir tres fases: establecimiento de la conexión, envío de los datos, y terminación de la conexión. En la primera se establecen los contadores y buffers necesarios para la transmisión, en la segunda se envían los datos con las retransmisiones que sea preciso, y en la tercera se liberan los buffers y variables utilizadas.

3.2 TRAMAS

La capa de enlace agrupa los bits en paquetes discretos denominados tramas (frames) que son los que envía por la línea. A veces, como en el caso de SDH, el medio físico crea también sus propias tramas, que no tienen por que coincidir con las del nivel de enlace. Según el tipo de red la trama puede oscilar entre unos pocos y unos miles de bytes[1]. La utilización de tramas simplifica el proceso de detección y eventual corrección de errores. Una buena parte de las tareas de la capa de enlace tiene que ver con la construcción e identificación de las tramas.

Para identificar el principio y final de una trama la capa de enlace puede usar varias técnicas; las mas normales son:

- Contador de caracteres
- Caracteres indicadores de inicio y final con caracteres de relleno
- Bits indicadores de inicio y final, con bits de relleno
- Violaciones de código a nivel físico

En el primer método se utiliza un campo en la cabecera de la trama para indicar el número de caracteres de ésta. Parece lo más sencillo e intuitivo, pero tiene un serio problema: si un error afecta precisamente a la parte de la trama que indica la longitud, o si por un error en la línea se envían bits de más o de menos, todas las tramas posteriores serán mal interpretadas.

El segundo método utiliza una secuencia especial de caracteres para marcar el inicio y final de cada trama, normalmente los caracteres ASCII DLE STX para el inicio y DLE ETX para el final (DLE es Data Link Escape, STX es Start of Text y ETX End of Text). De esta forma si ocurre un error o incidente grave el receptor sólo tiene que esperar a la siguiente secuencia DLE STX o DLE ETX para saber en que punto se encuentra.

Cuando se usa este sistema para transmitir ficheros binarios es posible que por puro azar aparezcan en el fichero secuencias DLE STX o DLE ETX, lo cual provocaría la interpretación incorrecta de un principio o final de trama por parte del receptor. Para evitar esto se utiliza una técnica conocida como *relleno de caracteres* (‘character stuffing’ en inglés): el emisor cuando ve que ha de transmitir un carácter DLE proviniente de la capa de red intercala en la trama otro carácter DLE; el receptor, cuando recibe dos DLE seguidos, ya sabe que ha de quitar un DLE y pasar el otro a la capa de red.

El principal problema que tiene el uso de DLE STX y DLE ETX es su dependencia del código de caracteres ASCII. Este método no resulta adecuado para transmitir otros códigos, especialmente cuando la longitud de carácter no es de 8 bits. Tampoco es posible enviar con este sistema tramas cuyo tamaño no sea múltiplo de ocho bits. Para evitar estos problemas se ha diseñado una técnica que podríamos

[1] Aunque al hablar de capacidades, caudales, velocidades, etc. siempre se hace referencia a bits por segundo, al hablar de tramas, paquetes, etc. se habla casi siempre en bytes, kilobytes, etc.; en estos casos los prefijos Kilo, Mega, etc. tienen el significado informático de 2^{10}, 2^{20}, etc.

considerar una generalización de la anterior, consistente en utilizar una determinada secuencia de bits para indicar el inicio de una trama. Generalmente se utiliza para este fin la secuencia de bits 01111110, que se conoce como byte indicador (‘flag byte’ o ‘flag pattern’) . El receptor esta permanentemente analizando la trama que recibe buscando en ella la presencia de un flag byte, y en cuanto lo detecta sabe que ha ocurrido un inicio (o final) de trama. Aunque el flag byte tiene ocho bits el receptor no realiza el análisis byte a byte sino bit a bit, es decir la secuencia 01111110 podría suceder 'a caballo' entre dos bytes y el receptor la interpretaría como flag byte; esto permite el envío de tramas cuya longitud no sea múltiplo de ocho.

Queda por resolver el problema de que los datos a transmitir contengan en sí mismos la secuencia 01111110; en este caso se utiliza la técnica conocida como relleno de bits o inserción de bit cero (‘bit stuffing’ o ‘zero bit insertion’). Consiste en que el emisor, en cuanto detecta que el flujo de bits contiene cinco bits contiguos con valor 1, inserta automáticamente un bit con valor 0. El receptor por su parte realiza la función inversa: analiza el flujo de bits entrante y en cuanto detecta un 0 después de cinco unos contiguos lo suprime en la reconstrucción de la trama recibida. De esta forma la secuencia 01111110 no puede nunca aparecer como parte de los datos transmitidos mas que como delimitador de tramas. Si las cosas van mal y el receptor pierde noción de donde se encuentra bastará con que se ponga a la escucha de la secuencia 01111110 que le indicará el inicio o final de una trama.

La trama a transmitir incluye casi siempre, además de los datos, alguna información de control de errores, por ejemplo un código CRC como veremos más tarde. Es importante notar que el relleno de bits (o de bytes) debe aplicarse a la trama inmediatamente antes de transmitirla y después de haber calculado el CRC, ya que de lo contrario la trama no sería interpretada correctamente si el CRC contuviera por azar el carácter o secuencia delimitadora de trama.

El cuarto método de identificación de tramas, violaciones de código a nivel físico, se utiliza en determinados tipos de red local aprovechando el hecho de que determinadas secuencias de símbolos no están permitidas y por tanto no pueden ocurrir en los datos a transmitir. Este método está muy relacionado con las características del medio físico utilizado. Veremos algunos ejemplos de protocolos de enlace que utilizan este método cuando hablemos de redes locales en el capítulo 4.

3.3 CONTROL DE FLUJO

Cuando dos ordenadores se comunican generalmente han de adoptarse medidas para asegurar que el emisor no satura al receptor. Si la línea entre ellos es de baja capacidad probablemente el factor limitante será la conexión, pero si es un canal rápido (por ejemplo una red local) es posible que el emisor, si es un ordenador más rápido o está menos cargado que el receptor, envíe datos a un ritmo superior al que es capaz de asimilar éste. En este caso el nivel de enlace en el receptor utilizará los buffers que tenga disponibles para intentar no perder datos, pero si el ritmo acelerado sigue durante un tiempo suficiente se producirá antes o después una pérdida de tramas por desbordamiento. En estos casos es preciso habilitar mecanismos que permitan al receptor frenar al emisor, es decir ejercer **control de flujo** sobre él. El control de flujo puede implementarse en el nivel de enlace o en niveles superiores (por ejemplo el nivel de transporte). Es importante que el control de flujo se ejerza de forma que no produzca ineficiencias en la comunicación; por ejemplo en enlaces de área extensa, donde la capacidad es un bien muy costoso, es importante mantener el nivel de ocupación del enlace tan alto como sea posible sin incurrir por ello en pérdida de tramas.

3.4 CONTROL DE ERRORES

Por desgracia el medio de transmisión utilizado en redes de ordenadores introduce errores. La tasa de errores es función de múltiples factores, pero principalmente del medio de transmisión utilizado. La fibra óptica y las redes locales suelen tener las tasas más bajas[2], mientras que las transmisiones inalámbricas con equipos móviles (GSM o LANs inalámbricas) o sobre telefonía analógica suelen tener las más altas.

[2] Por ejemplo los estándares SONET/SDH requieren en fibra óptica una tasa de error o BER (Bit Error Rate) de 10^{-12}, es decir un bit erróneo cada 10^{12} bits transmitidos.

La disciplina que estudia los errores de transmisión desde el punto de vista matemático es la teoría de la codificación. En 1950 R. W. Hamming publicó un artículo donde establecía las bases de los códigos de detección y corrección de errores; vamos a ver ahora los aspectos fundamentales de este planteamiento.

La trama que se transmite de un ordenador a otro está formada por *m* bits de datos y *r* bits redundantes, de comprobación. La trama tiene pues una longitud $n = m + r$, y forma lo que en teoría de la codificación se denomina una **palabra codificada** o **codeword** de *n* bits.

Dadas dos codewords cualesquiera, por ejemplo 10001001 y 10110001 es fácil determinar en cuantos bits difieren aplicando la operación OR exclusivo entre ambas y contando el número de bits a 1 del resultado; por ejemplo en nuestro caso difieren en 3 bits. Este valor, el número de posiciones de bit en que dos codewords difieren, se denomina **distancia de Hamming**. Si dos codewords están separadas por una distancia *d* serán necesarias *d* conversiones de un bit (por ejemplo *d* errores de un bit) para transformar una en la otra.

En la transmisión de información generalmente los datos de usuario pueden adoptar cualquier valor, por lo que en la parte de datos de la trama hay 2^m valores posibles; pero debido a la manera como se calculan los *r* bits de comprobación no están permitidas las 2^n codewords que en principio podrían formar la trama. A partir del algoritmo utilizado para calcular los bits de comprobación es posible construir todas las codewords legales, y averiguar cuales son las dos que están a menor distancia. Esta es la distancia Hamming del código completo.

La distancia Hamming de un código determina su capacidad de detección y corrección de errores. Para detectar *d* errores (es decir, *d* bits erróneos en la misma trama) es preciso que la distancia sea como mínimo de $d + 1$; de esa manera la codeword errónea no coincidirá con ninguna otra codeword válida y el receptor puede detectar la anomalía. Si se quiere un código capaz de corregir *d* errores es preciso que la distancia Hamming sea como mínimo $2d + 1$, ya que entonces la codeword errónea recibida sigue estando más cerca de la codeword original que de cualquier otra. Así por ejemplo, si la distancia Hamming del código utilizado en la corrección de errores de un protocolo determinado es de 5, entonces el protocolo podrá corregir hasta 2 errores en una trama, y detectar hasta 4.

El ejemplo más sencillo de código de detección de errores es el bit de paridad. El bit de paridad se elige de forma que mantenga la paridad (par o impar) de la codeword. El código formado con un bit de paridad tiene una distancia de 2, ya que cambiando un bit de cualquier codeword el resultado es ilegal, pero cambiando dos vuelve a serlo. Con una distancia 2 es posible detectar errores de 1 bit, pero no es posible detectar errores múltiples, ni corregir errores de ningún tipo. A cambio tiene un overhead mínimo, ya que supone añadir solamente un bit a cada codeword. Por este motivo el bit de paridad se utiliza en situaciones donde la fiabilidad es muy alta y la codeword muy pequeña, como en algunos sistemas de memoria RAM o de grabación de datos en soporte magnético.

La *eficiencia* de un código viene dada por la relación m/n; a la diferencia $1-m/n$ se la denomina *redundancia*. Por ejemplo al utilizar un bit de paridad para acceder a un byte de memoria se tiene una eficiencia de 0,8889 y una redundancia de 0,1111.

En esencia cualquier mecanismo de control de errores se basa en la inclusión de un cierto grado de redundancia, lo cual requiere un compromiso entre eficiencia y fiabilidad. Supongamos por ejemplo que deseamos transmitir una trama de 64 bits y utilizamos un bit de paridad; la eficiencia se reducirá en un 2% solamente y seremos capaces de detectar errores simples, pero los errores dobles pasarán desapercibidos; en caso de errores múltiples la probabilidad de que pasen desapercibidos es de 0,5, lo cual no es aceptable. Para mejorar la fiabilidad podemos introducir un bit de paridad cada ocho bits de datos; para esto imaginemos la trama como una matriz de 8 x 8 bits, a la cual añadimos una novena columna que son los bits de paridad; la transmisión se haría fila a fila, sin incluir la novena columna (los bits de paridad) que iría al final de la trama; el receptor reconstruiría la matriz 8 x 8 a partir de los bits recibidos y a continuación construiría la columna de bits de paridad, que luego compararía con la recibida; en caso de discrepancia se supondría un error y se pediría retransmisión. Con este sistema las probabilidades de detectar errores múltiples han aumentado, pero si los errores se producen a ráfagas (cosa muy normal en transmisión inalámbrica, por ejemplo) tendremos varios bits erróneos cerca, con lo que la probabilidad de que su bit de paridad detecte el error vuelve a ser de 0,5. Ahora bien, si en vez de calcular el bit de paridad para cada fila lo hacemos para cada columna tomaremos bits no contiguos de la trama, con lo que

la probabilidad de que un error a ráfagas pase desapercibido es mucho menor; tendrían que fallar dos bits de una misma columna y no fallar ningún otro bit, por ejemplo. Si estadísticamente la probabilidad de que un error en una columna pase desapercibido es de 0,5, la de que esta situación se de en las 8 columnas es de $0,5^8$, es decir, 0,0039. En este caso una elección inteligente de los bits de paridad ha permitido aumentar considerablemente la fiabilidad del código sin reducir su eficiencia.

El objetivo esencial de los códigos de detección o corrección de errores consiste en optimizar los algoritmos de cálculo de los bits de control para que sean capaces de detectar el máximo número de errores posible con un número razonable de bits adicionales. Las técnicas matemáticas en que se basan estos algoritmos han sido objeto de exhaustivos estudios por parte de especialistas en teoría de la codificación, y no son fácilmente mejorables; como consecuencia de esto los algoritmos de detección y corrección de errores son una parte bastante estable dentro de los sistemas de transmisión de datos.

Los códigos de corrección de errores se denominan también *corrección de errores hacia adelante* o FEC (*Forward Error Control*) y los de detección se llaman códigos de *corrección de errores hacia atrás* o por realimentación (*feedback* o *backward error control).*

3.4.1 Códigos correctores de errores

Los códigos de corrección de errores siempre tienen una eficiencia menor que los de detección para el mismo número de bits, y salvo que el medio de transmisión tenga muchos errores no salen rentables; por ejemplo, supongamos que tenemos una tasa de errores de 10^{-6} (un bit erróneo por millón) y queremos enviar tramas con 1000 bits de información útil; con corrección de errores necesitaremos 10 bits de comprobación por cada trama (eficiencia de 0,99); con detección de errores cada trama deberá llevar únicamente un bit de comprobación (0,999 de eficiencia) y tendremos que retransmitir una trama de cada 1,000 (0,999 de eficiencia) lo cual da una eficiencia total de 0,998 (0,999 * 0,999). Por este motivo los códigos correctores sólo se utilizan cuando el medio físico no es suficientemente fiable y no es posible emplear códigos detectores, como ocurre en los casos siguientes:

- El canal de comunicación es simplex, es decir la comunicación sólo es posible en un sentido; en este caso el receptor no dispone de un mecanismo que le permita pedir retransmisión.

- Se realiza una emisión broadcast o multicast; aunque fuera posible pedir retransmisión en este caso sería inaceptable que el emisor tuviera que atender a todas las solicitudes que se le planteen.

- El funcionamiento en tiempo real de la aplicación no toleraría el retardo introducido por un mecanismo de reenvío.

El más conocido y utilizado de los códigos correctores es el conocido como Reed-Solomon (RS), que se utiliza por ejemplo en las emisiones MPEG-2 de televisión digital; el uso de códigos RS representa un overhead del 10% aproximadamente, pero permite por ejemplo en el caso de transmisiones vía satélite reducir la tasa de error de 10^{-5} a 10^{-9}.

La capacidad reparadora de los códigos correctores disminuye cuando aparece una gran cantidad de errores contiguos. Por desgracia muchas fuentes de error tienen tendencia a producir errores a ráfagas (interferencias por arranque de motores, por ejemplo). Para aumentar la eficiacia de los códigos correctores se utiliza la técnica denominada interleaving consistente en calcular el código corrector a partir de una secuencia de bits que no corresponde con la secuencia a transmitir, sino que ha sido alterada de determinada forma; el receptor realizará una alteración siguiendo el mismo algoritmo antes de aplicar el código corrector; en caso de producirse un error a ráfagas durante la transmisión es muy probable que los bits erróneos no se encuentren contiguos en la secuencia alterada, con lo que la corrección podrá aplicarse de manera más efectiva. El inconveniente de aplicar interleaving a la información es que el receptor no puede entregar los bits a medida que los recibe, sino que ha de esperar a tener la secuencia completa sobre la que pueda verificar el código corrector antes de pasar los datos correspondientes; cuanto mayor sea el grado de interleaving mayor es el retrado producido por este factor. El retardo introducido por interleaving ha de ser entre 20 y 40 veces el de la ráfaga que se desea evitar. Teóricamente sería posible corregir un error a ráfagas de una duración arbitrariamente grande realizando un interleaving suficientemente grande.

Otra forma de explicar el interleaving es mediante la siguiente analogía: supongamos que tenemos un fax que por una avería tiene tendencia a omitir fragmentos de líneas de texto cuando éste está escrito en formato vertical (retrato). Es bastante probable que los fragmentos omitidos correspondan a palabras enteras, en cuyo caso será muy difícil para el receptor adivinar su significado. Pero si escribimos el texto en horizontal (apaisado) los errores afectarán a letras de líneas diferentes (y por tanto de palabras diferentes), con lo que el receptor podrá fácilmente deducir las letras que faltan por el contexto. Por otro lado, mientras que cuando se enviaba en formato vertical el receptor podía ir leyendo líneas a medida que aparecían, cuando se envía en formato horizontal el receptor no puede leer nada de la página hasta que ésta le llega en su totalidad (por tanto sufre un retardo mayor en la recepción de la información).

3.4.2 Códigos detectores de errores (CRC)

El algoritmo de detección de errores más utilizado en la práctica se basa en lo que se conoce como *códigos polinómicos* (también llamados *códigos de redundancia cíclica* o *CRC, Cyclic Redundancy Check*). La idea básica es la misma que en el caso de los bits de paridad: añadir a los datos a transmitir unos bits adicionales cuyo valor se calcula a partir de los datos; la trama así construida se envía, y el receptor separa los bits de datos de la parte CRC; a partir de los datos recalcula el CRC y compara con el valor recibido; si ambos no coinciden se supone que ha habido un error y se pide retransmisión.

La aritmética polinómica tiene unas propiedades singulares que la hacen especialmente fácil de programar en sistemas digitales, por lo que es posible implementarla directamente en hardware con lo que se consigue una eficiencia elevada, cosa importante para evitar que la comunicación se ralentice por el cálculo del CRC. Veamos algunas de estas propiedades:

Supongamos la siguiente suma de polinomios:

	$X^7 + x^4 + x^3 + x + 1$	que equivale a:	10011011
+	$X^7 + x^6 + x^3 + x$	que equivale a:	11001010
	---		---
	$x^6 + x^4 + 1$	que equivale a:	01010001

Obsérvese como no se arrastra valor a la unidad superior. En la práctica el resultado de la suma es equivalente a haber efectuado un OR EXCLUSIVO bit a bit entre las dos cadenas.

En el caso de la resta la situación es idéntica:

	$X^6 + x^4 + x^2 + 1$	01010101
-	$x^7 + x^5 + x^3 + x^2 + x + 1$	10101111
	---	---
	$x^7 + x^6 + x^5 + x^4 + x^3 + x$	11111010

ya que al utilizar módulo 2 la operación de dos valores iguales siempre da 0 y dos diferentes da 1.

En el caso de la división la operación se hace como en binario con la única peculiaridad de que la resta se hace módulo 2, como acabamos de ver.

Veamos paso a paso como se utiliza todo esto en una transmisión de datos con un ejemplo concreto:

1. En primer lugar el emisor y el receptor acuerdan un generador polinómico común *G(x)*, por ejemplo $x^4 + x + 1$ (que representaremos como 10011 o *g*); el primero y último bits de un generador polinómico siempre deben ser 1. El CRC siempre tiene una longitud un bit menos que el generador polinómico utilizado, por lo que en nuestro caso será de 4 bits.

2. Supongamos ahora que el emisor desea transmitir la cadena *c1,* formada por los bits 1101011011, que podemos ver como un polinomio de grado 9 (los datos a transmitir siempre deben tener mas bits que el generador polinómico utilizado). El emisor añade cuatro bits (en principio a 0) al final de los datos a transmitir, formando la cadena *c2* 110010110110000; esto equivale a multiplicar la cadena por 2^4

3. El emisor divide la cadena *c2* por el generador polinómico acordado (10011) usando las reglas de división binaria módulo 2 que antes hemos mencionado, y calcula el resto *r*, que es en este caso 1110.

4. El emisor resta el resto *r* de la cadena *c2*, formando así la cadena *c3* 11010110111110. Obsérvese que, como la resta es una operación XOR sobre los cuatro últimos bits, en la práctica la resta se hace sencillamente sustituyendo los cuatro últimos bits de *c2* por *r*. Al restar al dividendo el resto el valor obtenido es divisible por *g*.

5. La cadena *c3* es transmitida al receptor.

6. El receptor recibe la cadena *c3* y la divide por g. Si el resultado no es cero la transmisión se considera errónea y se solicita retransmisión.

Este algoritmo, que en principio puede parecer extraño y arbitrario, tiene tres características interesantes:

- Da un resultado predecible y reproducible, por lo que aplicado sobre unos mismos datos siempre dará el mismo resultado,

- Las operaciones utilizadas (desplazamiento, XOR, etc.) lo hacen muy fácil de implementar en hardware, y

- Suministra un mecanismo extremadamente flexible y robusto para la detección de errores, a través de la elección del generador polinómico adecuado.

Los generadores polinómicos más utilizados forman parte de estándares internacionales, y son los siguientes:

CRC-12: $x^{12} + x^{11} + x^3 + x^2 + x + 1$
CRC-16: $x^{16} + x^{15} + x^2 + 1$
CRC-CCITT: $x^{16} + x^{12} + x^5 + 1$
CRC-32: $x^{32} + x^{26} + x^{23} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

CRC-12 se utiliza en los códigos con longitud de carácter de 6 bits; CRC-16 y CRC-CCITT se utilizan en conexiones WAN, mientras que CRC-32 se utiliza en conexiones LAN.

En todos los generadores utilizados aparece $x+1$ como factor, ya que esto asegura la detección de todos los errores con un número impar de bits. Un código polinómico de *r* bits detectará todos los errores a ráfagas de longitud =< *r* . Un generador como CRC-16 o CRC-CCITT detectará todos los errores simples y dobles, todos los errores con un número impar de bits, todos los errores a ráfagas de longitud 16 o menor, 99,997% de los errores a ráfagas de 17 bits y 99,998% de los errores a ráfagas de 18 o más bits.

Cabría pensar en la posibilidad de que un error alterara la trama de tal forma que el CRC de la trama errónea coincidiera con el de la trama correcta (después de todo la probabilidad de que una trama distinta tenga el mismo CRC es 1/65536 para CRCs de 16 bits); los algoritmos de cálculo de CRCs intentan conseguir que las otras posibles tramas con igual CRC se encuentren muy alejadas (en términos de distancia Hamming) por lo que tendría que producirse una gran cantidad de errores en la misma trama para que esta posibilidad pudiera darse. En todo caso, cualquier protocolo de nivel de enlace fallará si se produce un error que no pueda ser detectado por el CRC.

3.5 PROTOCOLOS DE ENLACE ELEMENTALES

La principal característica que diferencia los protocolos de nivel de enlace es su comportamiento frente a los errores. Cuando el receptor detecta una trama errónea puede hacer una de las dos cosas siguientes:

1. Descartar silenciosamente la trama errónea sin notificarlo a nadie.
2. Solicitar del emisor la retransmisión de la trama errónea.

En el primer caso, es decir cuando no se realiza retransmisión de tramas erróneas el protocolo de enlace es trivial, por lo que normalmente esta opción casi no se comenta al hablar del nivel de enlace y se le dedica poco o ningún espacio en los libros de texto. En cambio se suelen explicar con todo detalle las diversas variantes de protocolos de enlace con retransmisión. Esto provoca lógicamente que al hablar de protocolos a nivel de enlace casi siempre se piense exclusivamente en los que realizan retransmisión de tramas erróneas. Paradójicamente este tipo de protocolos de enlace es hoy en día la excepción y no la regla. Dada la elevada fiabilidad de la mayoría de los medios físicos actuales normalmente no es rentable solicitar comprobación y retransmisión de las tramas, ya que esto supondría realizar un proceso casi siempre inútil en cada nodo del trayecto. Será normalmente el protocolo de transporte el que se ocupe de solicitar la retransmisión en caso de error. En caso de error la información habrá viajado inútilmente hasta el host de destino, pero esta estrategia es más rentable cuando la tasa de errores es baja. En los casos en que la tasa de errores del medio físico es excesiva se prefiere incorporar en el nivel físico un mecanismo corrector de errores, lo cual se traduce en la práctica en un canal prácticamente libre de errores al nivel de enlace; esto es lo que ocurre por ejemplo en las comunicaciones por red conmutada vía módem gracias al estándar V.42, en las comunicaciones a través de redes GSM o en las transmisiones de televisión digital con el uso de códigos RS, como ya hemos comentado.

Hecha esta aclaración pasaremos ahora a comentar los diferentes tipos de protocolos de enlace con retransmisión.

3.5.1 Protocolo de parada y espera

Como caso más sencillo de protocolo con retransmisión tenemos el denominado de parada y espera, consistente en que el emisor espera confirmación o acuse de recibo después de cada envío y antes de efectuar el siguiente. El acuse de recibo, también llamado ACK (del inglés acknowledgement) sirve tanto para indicar que la trama ha llegado correctamente como para indicar que se está en condiciones de recibir la siguiente, es decir el protocolo incorpora también la función de control de flujo. Este tipo de protocolos donde el emisor espera una confirmación o acuse de recibo para cada dato enviado se denominan protocolos PAR (Positive Acknowledgement with Retransmission) o también ARQ (Automatic Repeat reQuest).

Cuando la trama recibida es errónea (cosa que el receptor podrá verificar gracias al CRC) no se produce ACK. Lo mismo sucede cuando la trama enviada se pierde por completo. En este caso el emisor, pasado un tiempo máximo de espera, reenvía la trama. Una optimización que se puede incorporar en el protocolo es el uso de acuse de recibo negativo o NAK (Negative Acknowledgement) cuando se recibe una trama errónea; de esta forma el emisor puede reenviar la trama sin esperar a agotar el tiempo de espera, con lo que se consigue una mayor utilización de la línea.

Supongamos que una de las veces lo que se pierde no es la trama enviada sino el mensaje de ACK; pasado el tiempo de espera el emisor concluirá erróneamente que la trama se ha perdido y la reenviará, llegando ésta duplicada al receptor; el receptor no tiene ningún mecanismo para detectar que la trama es un duplicado, por lo que pasará el duplicado al nivel de red, lo cual no está permitido en un protocolo de enlace. Una forma de que el receptor distinga los duplicados es numerar las tramas, por ejemplo con un campo de un bit podemos numerar las tramas en base 2 (0, 1, 0, 1, ...) que es suficiente para detectar los duplicados.

Aunque la transmisión de datos ocurre únicamente en un sentido, este protocolo requiere un canal dúplex para funcionar; como la comunicación no ocurre simultáneamente un canal semi-dúplex sería suficiente.

3.5.2 Acuse de recibo 'piggybacked'

El protocolo de parada y espera que hemos visto transmitía datos en una sola dirección; el canal de retorno era utilizado únicamente para enviar los mensajes de acuse de recibo (ACK). Si tuviéramos que transmitir datos en ambas direcciones podríamos utilizar dos canales semi-dúplex con los protocolos anteriores, pero nos encontraríamos enviando en cada sentido tramas de datos mezcladas con tramas ACK.

La trama ACK contiene una cantidad mínima de información útil, pero ha de contener no obstante una serie de campos de control imprescindibles que ocupan mas bits que la propia información de ACK. Si se están transmitiendo datos en ambas direcciones resulta más eficiente, en vez de enviar el ACK solo en una trama, enviarlo dentro de una trama de datos; de esta forma el ACK viajará 'casi gratis' y se ahorrará el envío de una trama. Esta técnica se conoce con el nombre de *piggybacking* o *piggyback acknowledgement*; (en inglés piggyback significa llevar a alguien o algo a hombros o a cuestas).

Ahora bien, para 'montar' el ACK en una trama de datos es preciso que esta se envíe en un tiempo razonablemente corto respecto a cuando debería enviarse el ACK; de lo contrario el emisor, al ver que el ACK esperado no llega reenviará la trama, lo cual daría al traste con el pretendido beneficio del piggybacking; como no es posible saber de antemano cuando se va a enviar la siguiente trama de datos generalmente se adopta una solución salomónica: se espera un determinado tiempo y si el nivel de red no genera ningún paquete en ese tiempo se genera una trama ACK; en este caso el tiempo de espera debe ser sensiblemente inferior al timer de reenvío del emisor.

3.6 PROTOCOLOS DE VENTANA DESLIZANTE

Los protocolos de parada y espera son sencillos de implementar, pero son poco eficientes. Veamos un ejemplo.

Supongamos que utilizamos una línea de 64 Kb/s para enviar tramas de 640 bits de un ordenador A a otro B que se encuentra a una distancia de 2.000 Km. A tarda 10 ms en emitir cada trama (640/64000) o, dicho de otro modo, transmite 64 bits cada milisegundo. Por otro lado los bits tardan 10 ms en llegar de A a B (recordemos que la velocidad de las ondas electromagnéticas en materiales es de unos 200.000 Km/s); justo cuando llega a B el primer bit de la primera trama A termina de emitirla (en ese momento la trama esta 'en el cable'); diez milisegundos más tarde B recibe la trama en su totalidad, verifica el CRC y devuelve el ACK; suponiendo que el tiempo que tarda B en verificar el CRC y generar el ACK es despreciable la secuencia de acontecimientos es la siguiente:

Instante (ms)	Suceso en A	Suceso en B
0 ms	Emite primer bit de trama 1	Espera
10 ms	Emite último bit de trama 1; espera	Recibe primer bit de trama 1
20 ms	Espera	Recibe último bit de trama 1; envía ACK
30 ms	Recibe ACK; emite primer bit de trama 2	Espera
…	…	…

A partir de aquí el ciclo se repite. De cada 30 ms se está transmitiendo 10 ms y esperando 20 ms, es decir se está utilizando la línea con una eficiencia de 10/30 = 0,33 = 33%.

La eficiencia obtenida depende de tres parámetros: la velocidad de la línea, *v*, el tamaño de trama, *t*, y el tiempo de ida y vuelta también llamado 'round trip time'; en el caso normal de que el tiempo de ida y el de vuelta son iguales definimos τ como el tiempo de ida, por lo que el tiempo de ida y vuelta es de 2τ. Podemos derivar una expresión que nos permita calcular la eficiencia a partir de estos valores:

$$\text{Eficiencia} = \frac{t/v}{(t/v) + 2\tau}$$

Que aplicada al ejemplo anterior resulta:

$$\text{Eficiencia} = (640/64000) / (640/64000 + 2*0{,}01) = 0{,}01 / (0{,}01 + 0{,}02) = 0{,}01 / 0{,}03$$

Si en vez de una línea de 64 Kb/s hubiera sido una de 2 ,048 Mb/s la eficiencia habría sido del 1,5%. Es evidente que los protocolos de parada y espera tienen una baja eficiencia en algunos casos. El caso extremo de ineficiencia se da cuando se utilizan enlaces vía satélite, en los que el valor de 2τ puede llegar a ser de medio segundo. Para aprovechar mejor los enlaces con valores elevados del tiempo de ida y

vuelta hacen falta protocolos que permitan crear un 'pipeline', o dicho de otro modo tener varias tramas 'en ruta' por el canal de transmisión.

Al tener varias tramas simultáneamente pendientes de confirmación necesitamos un mecanismo que nos permita referirnos a cada una de ellas de manera no ambigua, ya que al recibir los ACK debemos saber a que trama se refieren. Para ello utilizamos un número de secuencia; sin embargo como el número de secuencia va a aparecer en todas las tramas y los mensajes ACK nos interesa que sea lo menor posible; por ejemplo con un contador de 3 bits podemos numerar las tramas módulo 8 (0,1,2,…,7) con lo que es posible enviar hasta siete tramas (0....6) antes de recibir el primer ACK; a partir de ese punto podemos enviar una nueva trama por cada ACK recibido. Esto es lo que se denomina un protocolo de *ventana deslizante*.

Podemos imaginar el funcionamiento del protocolo de ventana deslizante antes descrito como un círculo dividido en ocho sectores de 45º cada uno, numerados del 0 al 7; sobre el círculo hay una ventana giratoria que permite ver los sectores correspondientes a las tramas pendientes de confirmación; la ventana puede abrirse como máximo 315º, es decir permite ver hasta siete sectores correspondientes a las tramas enviadas pendientes de confirmación. Cuando se recibe un ACK se envía otra trama y la ventana gira un sector.

El protocolo de parada y espera se puede considerar como un protocolo de ventana deslizante en el que se utiliza un bit para el número de secuencia; en este caso el círculo estaría formado por dos sectores de 180º cada uno, y la ventana tendría una apertura de 180º.

Suponiendo un retardo nulo en el envío de los bits y en el proceso de las tramas en los respectivos sistemas, así como una longitud nula de las tramas ACK, el tamaño de ventana mínimo necesario *W* para poder llenar un canal de comunicación puede calcularse con la fórmula:

$$W = 2\tau * v/t + 1$$

Debiendo redondearse el valor al entero siguiente por encima. En la fórmula anterior 2τ es el tiempo (en segundos) que tarda una trama en hacer el viaje de ida y vuelta (round-trip time), *v* es la velocidad del canal de transmisión y *t* el tamaño de la trama a transmitir. Por ejemplo para el caso del ejemplo anterior donde $2\tau = 0{,}02$, v = 64000 y t = 640 obtenemos W = 3, por tanto la ventana mínima es 3; con una línea E1 (v = 2048000) W = 65.

La suposición de que los tiempos de proceso y la longitud de las tramas ACK son despreciables no es correcta, por lo que en la práctica se consigue una mejora en el rendimiento incluso para valores de *W* bastante superiores al calculado en la fórmula anterior.

Independientemente del tamaño de trama, velocidad de la línea y tiempo de ida y vuelta, un protocolo de parada y espera nunca puede conseguir un 100% de ocupación de una línea.

Cuando se utiliza un protocolo de ventana deslizante con ventana mayor que uno el emisor no actúa de forma sincronizada con el receptor; cuando el receptor detecta una trama defectuosa puede haber varias posteriores ya en camino, que llegarán irremediablemente a él, aún cuando reporte el problema inmediatamente. Existen dos posibles estrategias en este caso:

- El receptor ignora las tramas recibidas a partir de la errónea (inclusive) y solicita al emisor retransmisión de todas las tramas a partir de la errónea. Esta técnica se denomina *retroceso n*.

- El receptor descarta la trama errónea y pide retransmisión de ésta, pero acepta las tramas posteriores que hayan llegado correctamente. Esto se conoce como *repetición selectiva*.

Siguiendo con la representación en círculo de los números de secuencia podemos imaginar el comportamiento de retroceso n como una ventana de tamaño uno en el lado del receptor, es decir el receptor solo aceptará recibir las tramas en estricta secuencia e irá desplazando su ventana una posición cada vez. En el caso de repetición selectiva el receptor tiene la ventana abierta el mismo número de sectores que el emisor.

3.6.1 Protocolo de retroceso n

En retroceso n el receptor procesa las tramas en estricta secuencia, por lo que sólo necesita reservar espacio en buffers para una trama. En cambio en repetición selectiva el receptor ha de disponer de espacio en el buffer para almacenar todas las tramas de la ventana, ya que en caso de pedir retransmisión tendrá que intercalar en su sitio la trama retransmitida antes de pasar las siguientes a la capa de red (la capa de red debe recibir los paquetes estrictamente en orden).

En cualquiera de los dos casos el emisor deberá almacenar en su buffer todas las tramas que se encuentren dentro de la ventana, ya que en cualquier momento el receptor puede solicitar la retransmisión de alguna de ellas.

Con un número de secuencia de *n* bits se puede tener como máximo una ventana de 2^n-1 tramas, no de 2^n. Por ejemplo, con un número de secuencia de tres bits el emisor puede enviar como máximo siete tramas sin esperar contestación. De esta forma se garantiza que el número de trama recibido en dos ACK sucesivos siempre será distinto y no habrá duda posible en la detección de duplicados que pudieran producirse por la expiración prematura de timers; si se utilizara una ventana de 2^n se podrían producir conflictos; por ejemplo si tenemos un bit para el número de secuencia y permitimos una ventana de tamaño 2 en el emisor podría ocurrir lo siguiente:

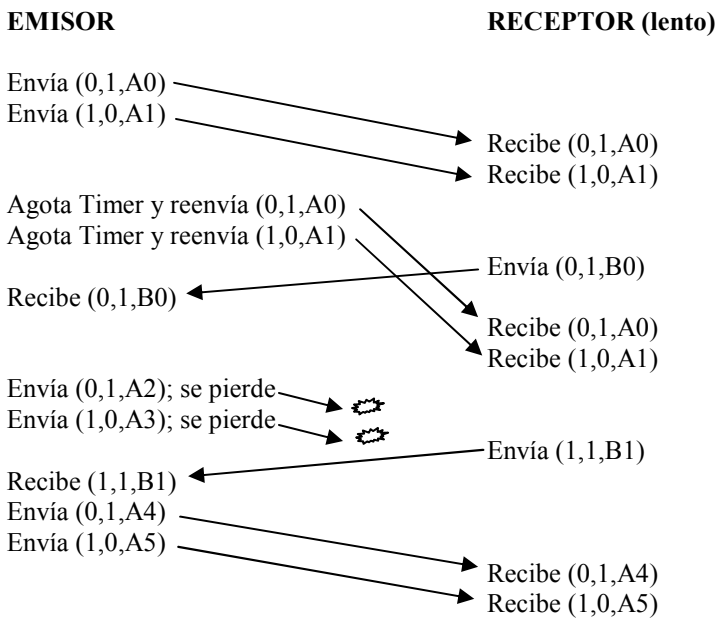

Figura 3.1.- Ejemplo del conflicto producido al utilizar una ventana de tamaño 2 con un número de secuencia de un bit. La notación utilizada corresponde a *(num_sec, num_ack, trama).*

Con lo que el receptor obtendría duplicadas las tramas A0 y A1 sin posibilidad de detectarlo

3.6.2 Protocolo con repetición selectiva

La repetición selectiva aprovecha las tramas correctas que llegan después de la errónea, y pide al emisor que retransmita únicamente esta trama. Como los paquetes se han de transferir en orden a la capa de red cuando falla una trama el receptor ha de conservar en buffers todos los paquetes posteriores hasta conseguir correctamente la que falta; en la práctica esto requiere tener un buffer lo suficientemente grande para almacenar un número de tramas igual al tamaño de la ventana, ya que se podría perder la primera trama de la ventana y recibirse correctamente el resto, en cuyo caso habría de conservarlas hasta recibir correctamente la primera.

La posibilidad de una recepción no secuencial de tramas plantea algunos problemas nuevos. Por ejemplo, supongamos que con un número de secuencia de tres bits el emisor envía las tramas 0 a 6, las cuales son recibidas correctamente. Entonces el receptor realiza las siguientes acciones:

1. Las transmite a la capa de red,
2. Libera los buffers correspondientes
3. Avanza la ventana para poder recibir siete tramas más, cuyos números de secuencia podrán ser 7,0,1,2,3,4,5
4. Envía un ACK para las tramas 0 a 6 recibidas

Imaginemos ahora que el ACK no llega al emisor. Éste supondrá que ninguna de las tramas ha llegado, por lo que las reenviará todas de nuevo (tramas 0 a 6). De estas las tramas 0 a 5 se encuentran dentro de la ventana del receptor y son por tanto aceptadas; la trama 6 está fuera de rango y es ignorada. En procesamiento secuencial el receptor no aceptaría estas tramas si no recibiera antes la trama 7 pendiente, pero con retransmisión selectiva las tramas fuera de orden se aceptan y se pide retransmisión de la trama 7; una vez recibida ésta se pasaría a la capa de red seguida de las tramas 0 a 5 antes recibidas, que serían duplicados de las anteriores. Los duplicados no detectados serían pasados al nivel de red, con lo que el protocolo es erróneo.

La solución a este conflicto está en evitar que un mismo número de secuencia pueda aparecer en dos ventanas consecutivas. Por ejemplo con un número de secuencia de 4 bits (0-15) y tamaño de ventana 8 la ventana del receptor sería inicialmente 0-7, después 8-15, 0-7 y así sucesivamente. Al no coincidir ningún número de secuencia entre ventanas contiguas se puede efectuar el proceso no secuencial de tramas sin que ocurra el conflicto anterior. El valor máximo de la ventana para un protocolo de repetición selectiva en el caso general es (MAX_SEQ+1)/2.

Aunque el número de secuencia en repetición selectiva se duplica respecto a retroceso n el número de tramas que hay que mantener en el buffer no necesita ser superior al tamaño de ventana, ya que este será el número máximo de tramas que habrá que manejar en cualquier circunstancia.

Como es lógico la técnica de repetición selectiva da lugar a protocolos más complejos que la de retroceso n, y requiere mayor espacio de buffers en el receptor. Sin embargo, cuando las líneas de transmisión tienen una tasa de errores elevada la repetición selectiva da un mejor rendimiento, ya que permite aprovechar todas las tramas correctamente transmitidas. La decisión de cual utilizar se debería tomar valorando en cada caso la importancia de estos factores: complejidad, espacio en buffers, tasa de errores y eficiencia.

3.7 PROTOCOLOS DE NIVEL DE ENLACE REALES

Después de haber visto la teoría describiremos ahora algunos de los protocolos de enlace más comúnmente utilizados.

3.7.1 HDLC - High-level Data Link Control

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; cada uno de estos organismos estandarizó el protocolo introduciendo sus propias variantes sobre la propuesta inicial. 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 3.1. Dada su importancia comentaremos ahora los aspectos más relevantes del protocolo HDLC.

Protocolo	Nombre completo	Utilización
HDLC	High level Data Link Control	Estándar ISO
ADCCP	Advanced Data Communications Control Procedure	Estándar ANSI
LLC	Logical Link Control	Estándar IEEE 802.2 para LANs
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 (V.32, V.34, etc.)
PPP	Point to Point Protocol	Estándar Internet
SDLC	Synchronous Data Link Protocol	SNA (IBM)

Tabla 3.1.- Algunos miembros de la familia de protocolos de enlace 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 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 tabla 3.2.

Campo	Tamaño (bits)	Valor
Delimitador	8	01111110
Dirección	8	Variable
Control	8	Variable
Datos	>=0	Variable
Checksum	16	Variable
Delimitador	8	01111110

Tabla 3.2.- Estructura de trama de HDLC

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 01111110111111011111110….

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 siguiente:

Bits →	1	3	1	3
	0	SEQ	P/F	NEXT

- 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 siguiente:

Bits →	2		2	1	3
	1	0	ORDEN	P/F	NEXT

Según el valor del subcampo ORDEN 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 siguiente:

Bits →	2		2	1	3
	1	1	ORDEN1/2	P/F	ORDEN2/2

En conjunto hay 5 bits que sirven para especificar diversos comandos que no detallaremos.

3.7.2 El nivel de enlace en la Internet

El modelo TCP/IP en sí mismo dice muy poco acerca del nivel de enlace. Esto es en parte consecuencia del principio básico de interoperabilidad que como hemos dicho estuvo presente en el diseño de TCP/IP. Durante muchos años TCP/IP no tuvo un protocolo de nivel de enlace propio o 'característico' y se ha limitado a transmitir datos utilizando otras redes de la mejor manera posible. Por ejemplo en la tabla 3.3 aparecen los estándares Internet que especifican el transporte de datagramas sobre algunas tecnologías de red muy conocidas. Resulta sorprendente que hasta 1990 no hubiera un protocolo estandarizado a nivel de enlace para el envío de datagramas sobre líneas punto a punto. Esto es lo que se intentó resolver con el protocolo PPP.

Medio	RFC	Año
X.25	877, 1356	1983
Ethernet	894	1984
802.x	1042	1988
FDDI	1188, 1390	1990
PPP	1171, 1663	1990
Frame Relay	1490	1993
ATM	1483, 1577	1994

Tabla 3.3.- Especificación del transporte de datagramas IP sobre diversas tecnologías

3.7.2.1 PPP (Point to Point Protocol)

PPP fue el desarrollo de un grupo de trabajo del IETF. El protocolo, elaborado en 1990, se encuentra especificado en los RFC 1661, 1662 y 1663.

PPP ha sido diseñado para ser muy flexible, por ello incluye un protocolo especial, denominado LCP (Link Control Protocol), que se ocupa de negociar una serie de parámetros en el momento de establecer la conexión con el sistema remoto.

Como puede verse en la tabla 3.4 la estructura de trama de PPP se basa en la de HDLC, salvo por el hecho de que se trata de un protocolo orientado a carácter, por lo que la longitud de la trama ha de ser un número entero de bytes

Campo	Tamaño (bytes)	Valor
Delimitador	1	01111110
Dirección	1	11111111
Control	1	00000011
Protocolo	1 ó 2	Protocolo
Datos	>=0	Variable
Checksum	2 ó 4	Variable
Delimitador	1	01111110

Tabla 3.4.- Estructura de una trama PPP

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.

Salvo que se negocie una transmisión fiable los campos dirección y control contienen siempre la secuencia 1111111100000011. Para no transmitir estos dos bytes de información inútil en todas las tramas generalmente LCP negocia la supresión de estos dos bytes de las tramas al inicio de la sesión (salvo que se pida 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.

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. El protocolo de autenticación más utilizado, conocido como CHAP (Challenge Handshake Protocol) utiliza el siguiente mecanismo:

1. El usuario se identifica ante el servidor con su código de usuario correspondiente.

2. El servidor envía al usuario una secuencia de caracteres arbitrariamente generada que el usuario debe transformar mediante un algoritmo de encriptado, usando como clave de encriptado su password.

3. Entretanto el servidor realiza el mismo proceso, es decir encripta la secuencia de caracteres seleccionada utilizando como clave de encriptado la password del usuario que se intenta identificar.

4. El usuario envía al servidor la secuencia encriptada y éste la compara con la suya; si ambas coinciden el servidor concluye que el usuario se ha identificado satisfactoriamente.

El uso de CHAP permite una identificación segura del usuario sin tener que enviar passwords por la red, evitando así los problemas de seguridad que esto supondría.

3.7.3 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 tabla 3.5:

Campo	Tamaño (bytes)	Valor
Delimitador	1	01111110
Dirección	2	Variable
Datos	0-8188	Variable
Checksum	2	Variable
Delimitador	1	01111110

Tabla 3.5.- Estructura de una trama LAPF (Frame Relay) normal

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.

3.7.4 El nivel de enlace en ATM

Como ya comentamos en el capítulo 1, lo que para nuestro modelo híbrido OSI-TCP/IP es el nivel de enlace corresponde en el modelo ATM a lo que se denomina la subcapa TC (Transmission Convergence, convergencia de la transmisión) y que en dicho modelo se incluye como parte de la capa física. La tarea fundamental de la subcapa TC es la obtención de las celdas provinientes de la capa ATM (capa de red) y su transformación en una secuencia de bits que pasa a la subcapa PMD (Physical Media Dependent) la cual los transmite por el medio físico, realizando por tanto la subcapa PMD la función del nivel físico en nuestro modelo.

3.7.4.1 Celdas de 48 bytes + cabecera

Merece la pena detenernos un momento a comentar la forma como se llegó a la decisión sobre el tamaño de las celdas ATM, ya que refleja el mecanismo que a menudo se sigue en la elaboración de estándares internacionales en materia de comunicaciones.

En el comité de la CCITT que elaboraba los estándares ATM existían dos grupos claramente diferenciados. Por un lado estaban los fabricantes de ordenadores y las empresas y organismos interesados en usar ATM para transmitir datos; estos eran reacios a utilizar un tamaño de celda pequeño, ya que esto introduce un elevado costo de proceso y una pérdida considerable de capacidad debido a las cabeceras que necesariamente ha de llevar cada celda. Este grupo proponía un tamaño de celda de 128 bytes

En la postura contraria se encontraban las PTTs europeas, cuyo objetivo era utilizar ATM para transmitir conversaciones telefónicas. Además de utilizar la técnica habitual PCM para digitalizar una conversación telefónica en un canal de 64 Kb/s, en ATM es bastante frecuente utilizar técnicas de compresión (por ejemplo la denominada ADPCM) que permiten meter el canal habitual en tan solo 32, o incluso 24 Kb/s. De esta forma es posible aprovechar aun mas la capacidad disponible.

Las PTTs proponían utilizar celdas de 16 bytes, ya que así una conversación podría generar una celda cada 2 ms si se usaba PCM, o cada 4 o 6 ms si se empleaba ADPCM. Con celdas de 128 bytes como proponían los fabricantes de ordenadores costaría 16 ms llenar una celda con una conversación PCM, y 32 o 48 ms con ADPCM; desde el punto de vista de los operadores europeos esto planteaba un problema importante, ya que el tiempo de llenado de la celda aumenta el retardo en la comunicación; cuando el retardo total extremo a extremo es mayor de 20 ms el efecto del eco producido es perceptible, ya que el retardo es equivalente al de una conexión de larga distancia; por tanto es preciso instalar costosos equipos de cancelación de eco.

Las compañías telefónicas estadounidenses no tenían ningún problema con la utilización de celdas de 128 bytes, ya que con distancias de miles de kilómetros y retardos de más de 30 ms en las comunicaciones costa a costa estaban ya acostumbrados desde hacía tiempo a instalar canceladores de eco en sus líneas. Pero las PTTs europeas, al trabajar con distancias menores de 2.000 Km, no han instalado canceladores de eco y en caso de haber optado por celdas de 128 bytes se habrían visto obligadas a hacer costosas inversiones, o a renunciar a la posibilidad de utilizar sistemas de compresión para transmitir la voz.

Después de muchas negociaciones cada bando cedió un poco en sus pretensiones. Las PTT accedieron a subir a 32 bytes el tamaño de celda, mientras que los fabricantes de ordenadores bajaron a 64 bytes. En ese momento la CCITT decidió zanjar la discusión partiendo la diferencia y fijando la celda en 48 bytes (más cabecera). Así utilizando ADPCM a 24 Kb/s el retardo puede llegar a ser de 18ms, que esta muy cerca del límite de 20 ms para que se produzca el eco (hay que tomar en cuenta que además habrá alguna longitud de cable cuyo retardo también influye).

3.7.4.2 Transmisión de celdas

Cada celda ATM tiene 5 bytes de cabecera, el último de los cuales es un checksum de los otros cuatro. La subcapa TC se ocupa de calcular el valor de dicho byte utilizando el polinomio generador $x^8 + x^2 + x + 1$. Este campo checksum se denomina HEC (Header Error Control).

La razón de hacer checksum de la cabecera únicamente es acelerar el proceso de cálculo; se supone que los niveles superiores harán corrección de errores si lo consideran apropiado (algunas aplicaciones, como el vídeo o audio, pueden soportar sin problemas una pequeña tasa de errores). También debemos tomar en cuenta el hecho de que ATM se diseñó pensando en las fibras ópticas, que son un medio de transmisión altamente fiable. Hay estudios que demuestran que la gran mayoría de los (ya pocos) errores que se producen en fibras ópticas son errores simples. El HEC detecta todos los errores simples y el 90% de los errores múltiples.

Una vez está en su sitio el HEC la celda está lista para transmisión. Existen dos tipos de medios de transmisión, los asíncronos y los síncronos. Los asíncronos simplemente transmiten cada celda cuando está preparada. Los síncronos por el contrario tienen que transmitir celdas con una periodicidad fija, y en

caso de no haber celdas útiles preparadas envían celdas de relleno o inútiles (también llamadas 'idle' cells).

Otro tipo de celdas 'anormales' (es decir, sin datos) son las denominadas celdas OAM (Operation And Maintenance). Estas son utilizadas por los conmutadores ATM para intercambiar información de control sobre la red, con la que es posible hacer labores de mantenimiento, tales como gestión de averías y de rendimiento. Sirven también para transmitir información del estado de la red, por ejemplo del grado de congestión. También se utilizan celdas OAM para 'saltar' el espacio ocupado por la información de control de una trama SONET/SDH.

3.7.4.3 Recepción de celdas

En el lado receptor la subcapa TC ha de tomar el flujo de bits entrante, localizar el principio y final de cada celda, verificar el HEC (y descartar las celdas inválidas), procesar las celdas OAM y las celdas inútiles, y pasar a la capa ATM las celdas de datos.

La detección del principio y final de cada celda se hace por mecanismos completamente distintos a los utilizados en HDLC. No existe ninguna secuencia de bits característica del principio y final de cada celda, pero si se sabe que cada celda ocupa exactamente 53 x 8 = 424 bits, por lo que una vez localizada una será fácil encontrar las siguientes. La clave para encontrar la primera celda está en el HEC: en recepción la subcapa TC captura 40 bits de la secuencia de entrada y parte de la hipótesis de que sea un principio de celda válido; para comprobarlo calcula el HEC aplicando el polinomio sobre los cuatro primeros bytes y comparando el resultado con el quinto; si el cálculo no es correcto desplaza la secuencia un bit y repite el cálculo; repitiendo este proceso como máximo 424 veces el TC localiza finalmente el principio de una celda, y a partir de ella todas las que le siguen.

Con un HEC de tan solo 8 bits la probabilidad de que un conjunto de bits elegido al azar resulte ser un HEC válido es de 1/256, lo cual no es despreciable. Por ello el receptor cuando localiza un HEC válido, para asegurarse de que el resultado no ha sido fruto de la casualidad, repite el cálculo con el HEC siguiente (53 bytes después); haciendo esta comprobación con varias celdas sucesivas se puede reducir a un nivel despreciable la probabilidad de que el acierto haya sido pura casualidad. Por ejemplo si el HEC obtenido es correcto en cinco celdas consecutivas la probabilidad de que esto sea fruto de la casualidad es de $1/256^5$, o sea 10^{-12} aproximadamente.

Una vez localizado el principio de una celda el receptor ya puede sin problemas localizar todas las demás por su posición relativa, siempre que se mantenga el sincronismo. Podría ocurrir que como consecuencia de un error se introdujera o eliminara un bit en la secuencia, con lo que el receptor perdería el sincronismo. En tal caso el primer síntoma sería un HEC erróneo, pero un HEC erróneo también puede producirse por un bit erróneo, cosa más normal que un bit de más o de menos. Por esto cuando la TC detecta un HEC erróneo no supone inmediatamente que ha perdido el sincronismo; en principio considera que ha sido un bit erróneo, y se pone alerta ante la posibilidad de que la siguiente celda dé también un HEC erróneo, en cuyo caso la sospecha de pérdida de sincronismo crece. Si varias celdas consecutivas tienen un HEC erróneo la TC supone que ha perdido el sincronismo y empieza de nuevo el proceso antes descrito de detección de principio de celda.

Cabe pensar en la posibilidad de que un usuario genere, con o sin intención, flujos de datos con secuencias de 5 bytes que al incluirlos en celdas ATM contuvieran sistemáticamente HECs válidos; entonces la TC podría interpretar erróneamente la cabecera de celdas, y por tanto los datos. Para evitar esta posibilidad se altera el orden de los datos a enviar antes de transmitirlos de acuerdo con un patrón preestablecido; en el lado del receptor se aplica a los datos recibidos un patrón simétrico de reordenación, de forma que los datos se regeneran en el mismo orden en el que el usuario los envió. Este proceso, denominado ‘scrambling’ se realiza de forma que sea transparente y no afecte a los datos enviados por el usuario.

3.8 EJERCICIOS

1. Indique si es verdadera o falsa cada una de las siguientes afirmaciones:

a) En el funcionamiento normal de PPP en caso de detectar un error en el CRC el receptor pide retransmisión de la trama correspondiente.

b) En los protocolos a nivel de enlace las tramas siempre se numeran, aun cuando el protocolo no contemple la posibilidad de retransmisión en caso de pérdida o error en la trama.

c) Los protocolos de ventana deslizante que utilizan repetición selectiva son mas eficientes que los de retroceso n, pero requieren un mayor espacio para buffers en el lado del transmisor.

d) La decisión de fijar el tamaño de la celda ATM en 53 bytes se tomó en base a razones fundamentalmente políticas, no atendiendo a criterios técnicos.

2. Cuando el medio de transmisión tiene muchos errores es conveniente enviar tramas pequeñas, para reducir la cantidad de datos transmitidos; por el contrario cuando hay pocos errores es mejor utilizar tramas grandes, para reducir la cantidad de cabeceras. A la vista de esto cabría deducir que ATM que tiene una trama pequeña (tamaño de celda 53 bytes) fue diseñado para medios con muchos errrores. Es correcta esta conclusión?

3. Se ha de transmitir mediante el protocolo HDLC la siguiente secuencia de bits:

011011110111111011111100

Diga cual sería la cadena de bits que realmente se transmitiría en el campo de datos de la trama. ¿Plantearía algún problema el hecho de que la longitud de la cadena a transmitir (22 bits) no sea un número entero de octetos?

4. Los CRCs utilizados en líneas WAN son normalmente de 16 bits. Calcule la probabilidad de que una cadena de 16 bits elegida al azar sea un CRC correcto para una trama.

5. Deduzca una expresión que permita calcular el tamaño de ventana mínimo necesario para poder aprovechar completamente una línea de transmisión con una capacidad de v bits por segundo, un tamaño de trama de t bits y una longitud de l Kilómetros. El medio de transmisión es fibra óptica. Suponga que los tiempos de proceso de las tramas son despreciables, así como la longitud de las tramas ACK. Suponga también que la contribución al tiempo de propagación de los equipos intermedios (repetidores, amplificadores, etc.) es despreciable. Utilice dicha fórmula para calcular el tamaño de ventana mínimo para una línea E1 con tramas de 2000 bits y una distancia de 200 Km.

6. Se quiere establecer un enlace E1 entre dos equipos mediante fibra óptica utilizando el protocolo HDLC con retroceso n y trama normal (no extendida). Se sabe que todas las tramas tendrán un tamaño (incluida la información de control) de 1 KByte. Suponiendo que todo el retardo se debe a la transmisión de la señal por la fibra diga cual será la distancia máxima a la que el enlace podrá operar al 100% de su capacidad. Considere despreciables los tiempos de generación de tramas y acuses de recibo (ACKs).

7. La red académica y de investigación de la Comunidad Valenciana se conecta a Internet mediante un PVC ATM suministrado por el servicio Gigacom de Telefónica. El equipo que se conecta físicamente a la línea OC3 de Telefónica es un conmutador ATM marca Cisco modelo Lightstream 1010, propiedad de la Universidad de Valencia, que comunica con otro conmutador ATM, este ya de

la red Gigacom; a partir de ahí la red Gigacom se ocupa de encaminar las celdas correctamente hacia su destino, en el otro extremo del PVC.

En el conmutador ATM de la Universidad de Valencia el comando SHOW INTERFACE muestra una serie de contadores que permiten averiguar, entre otras cosas, el estado de la conexión ATM a nivel de enlace. La ejecución de dicho comando en la consola del conmutador ha dado el siguiente resultado:

SHOW INTERFACE ATM0/1/2

```
  ATM0/1/2 is up, line protocol is up
  Hardware is oc3suni
  Description: Linea de acceso con RedIRIS (Servicio GIGACOM)
  MTU 4470 bytes, sub MTU 0, BW 156250 Kbit, DLY 0 usec, rely 255/255, load 1/255
  Encapsulation ATM, loopback not set, keepalive not set
  Last input 00:00:00, output 00:00:00, output hang never
  Last clearing of "show interface" counters 4h
  Queueing strategy: fifo
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  1 minute input rate 3131000 bits/sec, 7381 packets/sec
  1 minute output rate 748000 bits/sec, 1768 packets/sec
->   59253444 packets input, 3140432532 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants
->   22 input errors, 23 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     45065854 packets output, 2388490262 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets
     0 output buffer failures, 0 output buffers swapped out
```

Como puede ver por el campo 'Last clearing of "show interface" counters' los contadores llevan 4 horas sin ser puestos a cero. En el campo 'packets input' se indica el número de celdas ATM que han entrado por esa interfaz en ese tiempo, y en el campo 'CRC' aparece el número de errores de CRC que han ocurrido (obviamente el CRC solo se puede comprobar en el sentido entrante).

a) Calcule a partir de estos datos el BER (Bit Error Rate) correspondiente al enlace físico entre el conmutador ATM de la Universidad de Valencia y el que le da acceso en Telefónica al servicio Gigacom.

b) Usando como muestra este período de 4 horas calcule el número medio de celdas por hora que entran en la Comunidad Valenciana con un valor erróneo en el campo payload sin ser detectadas.

Suponga en todo momento que los bits erróneos se reparten de forma perfectamente homogénea en el flujo de bits transmitidos.

8. Suponga que tiene que transmitir un fichero de 1 MByte de un host a otro a través de una línea de 64 Kb/s. Cada octeto del fichero tiene el código decimal 80 (carácter ASCII 'P'). De alguna manera ha conseguido diseñar un protocolo que le permite transmitir el fichero en tramas HDLC, enviando en cada trama 2000 bits de información útil (en el campo datos), sin tener que añadir información de control adicional para los niveles superiores al de enlace. Calcule el número de tramas que ha de transmitir, la cantidad total de bits transmitidos y el tiempo que dura la transmisión. Suponga que la línea no tiene errores y que el tamaño de ventana utilizado es lo bastante grande como para que el emisor no suspenda la transmisión mientras espera recibir los ACK.

Suponga que tiene que transmitir otro fichero cuyos octetos contienen el código ASCII 231 (decimal). Cambiarían en algo los cálculos anteriores?

Suponga ahora que los ordenadores utilizan el código EBCDIC, en vez del código ASCII. Cambiarían en algo los cálculos? (Aunque sea código EBCDIC los octetos del fichero siguen conteniendo el valor 80, o 231 según el caso).

9. Una empresa de comunicaciones encarga a un programador diseñar un protocolo de enlace, de parada y espera, utilizando tramas con formato HDLC estándar, que sirva tanto para transmisión por fibra óptica como por GSM. Los expertos en fibras ópticas le aconsejan utilizar tramas lo más grande

posibles, ya que así se reducirá el overhead debido a la información de control; por el contrario, los responsables de transmisión GSM piden que las tramas sean pequeñas, pues de lo contrario la probabilidad de tramas erróneas aumenta, se producirán muchas retransmisiones y la eficiencia disminuirá.

Al recibir consejos tan contradictorios el programador decide utilizar un algoritmo autoadaptativo que empiece utilizando un tamaño de trama intermedio y analice las retransmisiones producidas; a partir de ese dato calculará la tasa de error o BER (Bit Error Rate) y ajustará el tamaño de trama al valor óptimo según las circunstancias en cada momento.

Deduzca una fórmula que permita al programador calcular el tamaño de trama óptimo en función del BER. Aplique entonces dicha fórmula para calcular el tamaño de trama óptimo para un BER de 10^{-3}. Por tamaño de trama óptimo se entiende aquel que de una mayor eficiencia de transferencia neta.

Haga las siguientes suposiciones:

a) El emisor siempre dispone de datos a enviar.
b) No se produce nunca relleno de bits.
c) Los tiempos de generación, transmisión y propagación de las tramas (tanto de datos como de ACK) y de sus checksum se consideran despreciables.
d) Considere que los bits erróneos se distribuyen de forma perfectamente homogénea en el flujo de bits transmitidos.

Discuta o comente de forma *cualitativa*, que influencia tendría en el tamaño de trama óptimo la supresión de la suposición c), es decir, la consideración de los tiempos de transmisión y propagación de las tramas (el tiempo de transmisión es lo que tarda el emisor en enviar la trama, el tiempo de propagación es lo que tarda el primer bit de ésta en llegar al receptor). Si para aumentar la eficiencia el programador decidiera utilizar un protocolo retroceso n con ventana 7 en el emisor cual sería, también de forma cualitativa, la consecuencia en el tamaño de trama óptimo.

Discuta o comente de forma *cualitativa* que influencia tendría en el tamaño de trama óptimo la supresión de la suposición d), es decir, el tomar en consideración el reparto no homogéneo de los bits erróneos (como de hecho ocurre en la realidad).

Suponga ahora que los bits erróneos se producen según una distribución de Poisson. Deduzca la fórmula que permita calcular el tamaño de trama óptimo en este caso. Aplíquela al caso de un BER de 10^{-3} y compare el resultado con el obtenido anteriormente.

INFORMACIÓN SUPLEMENTARIA:

a) El BER de un canal de comunicación se define como:

$$BER = \frac{\text{bits erróneos transmitidos}}{\text{bits totales transmitidos}}$$

b) En HDLC un mismo delimitador (01111110) puede ser simultáneamente final de una trama y principio de la siguiente.

3.9 SOLUCIONES

S1.-

a) **Falsa**. Por defecto PPP no suministra transmisión fiable, con lo que si hay un error en el CRC la trama es simplemente descartada y no retransmitida.

b) **Falsa**. Si no se prevé realizar retransmisiones no es necesario numerar las tramas, y hacerlo sería una tarea inútil. Por ejemplo el modo normal de funcionamiento de PPP no efectúa retransmisiones y utiliza el valor 00000011 en el campo control en todas las tramas, sin numerarlas. Por esto cuando una trama tiene este valor en el campo control se la denomina trama 'no numerada'.

c) **Falsa**. En el transmisor ambos tipos de protocolo requieren el mismo espacio en buffers

d) **Verdadera**. El valor de 53 bytes (48 + cabecera) lo adoptó la ITU como solución de compromiso entre el requerimiento planteado por las empresas de telecomunicaciones europeas (PTTs), que querían un tamaño de celda pequeño, y los fabricantes de equipos de comunicaciones y ordenadores que preferían uno grande.

S2.-

Desde luego que no. ATM fue diseñado pensando en el uso de medios de transmisión altamente fiables, hasta el punto de que no se verifican errores de transmisión salvo en la información de cabecera, que se considera crítica. El tamaño de celda de ATM se eligió pequeño para poder 'colar' con facilidad tráfico de alta prioridad (por ejemplo tráfico en tiempo real) que pueda aparecer en una línea de transmisión ocupada con tráfico menos urgente.

S3.-

El emisor intercala un bit a cero siempre que en el flujo de datos a enviar encuentra cinco unos seguidos (independientemente de cual sea el valor del siguiente bit en el flujo). Por tanto la cadena transmitida sería:

011011110111110**0**0111111**0**100

(donde se han marcado en negrita los bits de relleno). No es problema que la cadena no sea un número entero de octetos, ya que HDLC es un protocolo orientado al bit.

S4.-

La probabilidad es de 1 en 65536 (2^{16}), es decir de 0,0000153.

S5.-

Para tener la línea siempre ocupada es preciso tener un tamaño de ventana que nos permita enviar datos durante el tiempo de ida y vuelta (2τ) igual a dos veces el tiempo de propagación (τ), sin esperar a un ACK. Como la velocidad de la luz en la fibra óptica es de aproximadamente 200.000 Km/s el valor de 2τ en segundos lo podemos calcular como:

$2\tau = 2 * l / 200000$

donde l es la longitud de la fibra en kilómetros.

El número de bits que se pueden transmitir en ese intervalo de tiempo será $v * 2\tau$, por lo que el número de tramas necesario será igual a $(v * 2\tau) / t$, redondeado al valor entero superior. Sin embargo no hemos tomado en consideración hasta aquí el hecho de que el ACK no se genera en cuanto llega el primer bit de la trama, sino cuando llega el último. Esto hace que el tamaño de ventana mínimo para conseguir que la línea no pare sea uno mas que el número de tramas necesarias para 'llenar' de datos la línea. Resumiendo:

Tamaño de ventana = (v * 2 * *l*) / (200000 * *t*) + 1

Debiendo redondearse al entero inmediato superior.

Para una línea E1 (v = 2,048 Mb/s) con tramas de 2000 bits y una distancia de 200 Km será:

2048000 * 2 * 200 / (200000 * 2000) + 1 = 3,048

El tamaño mínimo de ventana en este caso sería **4**.

S6.-

La trama normal HDLC utiliza tres bits del campo control para el número de secuencia, por lo que éste puede tomar 8 valores (entre 0 y 7). Al utilizar retroceso n el máximo tamaño de ventana es uno menos que el número de secuencia, o sea 7 en este caso.

La distancia máxima a la que el enlace podrá operar al 100 % de eficiencia será aquella que permita tener en el 'cable' tres tramas de datos, ya que así aseguramos poder 'llenar' de datos el cable (tres tramas en cada sentido) y tener una trama mas para el ACK, por las razones expuestas en el problema 5.

El tiempo que tarda en transmitirse una trama de 1 Kbyte (8192 bits) es:

(8192 bits) / (2048000 bits/s) = 0,004 s = 4 milisegundos

Así pues, la distancia para que quepan 3 tramas será la correspondiente a un tiempo de propagación de 12 milisegundos, o sea:

200000 Km/s * 0,012 s = **2400 Km**

S7.-

a) La única parte de las celdas ATM que contiene un CRC son los 5 bytes de la cabecera (el quinto byte es un CRC de los otros cuatro). La probabilidad de que haya algún bit erróneo en la cabecera es 40 veces superior al BER, por tanto:

   BER = (Errores_CRC / total_de_celdas) / 40 = (23 / 59253444) / 40 = **$9{,}7 \times 10^{-9}$**

b) Total de celdas que entran por hora: 59253444 / 4 = $14{,}813 \times 10^{6}$

   El campo payload no contiene CRC. Por tanto todas las celdas erróneas serán aceptadas. Como el campo payload tiene un tamaño de 48 bytes la proporción de celdas con el payload erróneo será 48*8=384 veces superior al BER.

   Celdas con el payload erróneo aceptadas por hora: $14{,}813 \times 10^{6} \times 9{,}7 \times 10^{-9} \times 384$ = **55**

Los datos mostrados en este ejercicio fueron obtenidos de un caso real. Después de muchas averiguaciones se descubrió que había una avería en el conversor de fibra multimodo-monomodo conectado a esta interfaz. Desde que se resolvió ese problema la tasa de error es prácticamente cero, como es normal.

S8.-

1 MByte = 1024*1024*8 = 8388608 bits

Tramas transmitidas: 8388608/2000 = 4194,3 = **4195 tramas**

Serán 4194 tramas de 2000 bits y una de 608 bits.

Bits transmitidos: Como nos dicen que el emisor transmite de forma continuada suponemos que solo hay un delimitador entre trama y trama, con lo que la información de control es 40 bits por trama:

Información de control: 40 * 4195 =	167800
Delimitador del final:	8
Datos:	8388608
TOTAL:	**8556416**

Tiempo de transmisión: 8556416/64000 = **133,69 segundos**

Con el carácter ASCII 231 (11100111) es preciso hacer relleno de bits en la unión entre cada carácter y el siguiente (1110011111100111…); por tanto las tramas de 2000 bits (250 caracteres) llevarán 249 bits de relleno y la de 608 (76 caracteres) llevará 75.

El número de tramas será el mismo que en el caso anterior.

El número de bits de relleno será:

4194 * 249 + 75 = 1044381

El número de **bits transmitidos** en este caso será: 8556416 + 1044381 = **9600797**

Y el **tiempo de transmisión**: 9600797/64000 = **150,01 segundos**

El uso de código EBCDIC no cambiaría los resultados anteriores, ya que el relleno de bits se haría de la misma forma en ambos casos independientemente del código utilizado.

S9.-

Vamos a deducir la fórmula para calcular el tamaño de trama óptimo en función del BER y de la información de overhead.

- **t** : longitud total de la trama
- **ov**: información de overhead (flag, dirección, control y checksum)
- **BER**: Bit Error Rate
- $\mathbf{E_{ov}}$: eficiencia de la transferencia tomando en cuenta solamente el overhead, suponiendo una línea sin errores, BER = 0)
- $\mathbf{E_{BER}}$: eficiencia de la transferencia tomando en cuenta únicamente la tasa de error, suponiendo un protocolo sin overhead, ov = 0)
- **E** = eficiencia real (tomando en cuenta overhead y errores)

La parte de datos de la trama (parte útil) es *t - ov*; por tanto E_{ov} será:

$$E_{ov} = \frac{t - ov}{t} = 1 - \frac{ov}{t}$$

BER es la probabilidad de que un bit sea erróneo; por tanto la probabilidad de que haya un bit erróneo en una trama es *t*BER*, por lo que E_{BER} será:

$$E_{BER} = 1 - t * BER$$

La eficiencia real tomando estos factores en cuenta será pues el producto de las dos expresiones anteriores:

$$E = E_{ov} * E_{BER} = \left(1 - \frac{ov}{t}\right) * (1 - t*BER) = 1 + ov * BER - BER * t - \frac{ov}{t}$$

Para calcular el tamaño de trama que da una eficiencia máxima derivamos *E* respecto de *t* e igualamos a cero:

$$\frac{dE}{dt} = -BER + \frac{ov}{t^2} = 0$$

$$\frac{ov}{t^2} = BER$$

t = SQRT (ov/BER)

Para saber si se trata de un máximo o de un mínimo calculamos la segunda derivada:

$$\frac{d^2E}{dt} = -2\,\frac{ov}{t^3}$$

y vemos que es negativa para todo valor positivo de *ov* y *t*; se trata por tanto de un máximo.

Con BER = 10^{-3} y ov = 40 bits (HDLC) obtenemos:

t = SQRT (40 * 1000) = **200 bits**

La eficiencia será en este caso:

E(t=200) = (1 - 40/200) * (1 - 200 * 10^{-3}) = 0,8 * 0,8 = 0,64 = **64%**

Para comprobar que es un máximo calculemos la eficiencia para 199 y 201 bits:

E(199) = 0,79899 * 0,801 = 0,63999

E(201) = 0,80099 * 0,799 = 0,63999

Al ser el tiempo de proceso y de envío mayores que cero se producen esperas en la línea que reducen la eficiencia. El tiempo de envío tiene dos componentes: el **tiempo de transmisión (**lo que se tarda en enviar la trama) que depende de la velocidad de la línea y la longitud de la trama, y el **tiempo de propagación o retardo** (lo que tardan cada bit en llegar al destino una vez puesto en la línea) que depende de la distancia y de los equipos intermedios; el tiempo de transmisión es directamente proporcional al tamaño de la trama para una velocidad dada, por lo que *no afecta el tamaño óptimo*; el tiempo de propagación es el causante de las esperas por ACKs en un protocolo de parada y espera como el que nos ocupa, y es independiente del tamaño de trama, por lo que con un tamaño de trama mayor se tendrá menos tiempo de espera y mejorará la eficiencia; por tanto *tenderá a incrementar el tamaño de trama óptimo*. El tiempo de proceso tendrá una componente proporcional al tamaño de trama y una constante (cálculo del checksum, construcción de cabeceras, etc.); de manera análoga a

lo ocurrido con el tiempo de transmisión la parte proporcional no influirá en el tamaño óptimo, mientras que la consideración de la parte constante tenderá de nuevo a *incrementar el tamaño óptimo de la trama*.

Con un protocolo de ventana deslizante de tamaño 7 se produce un pipeline en la línea, con lo que el tiempo de propagación y el tiempo de transmisión se solapan; de esta forma se pueden reducir (o incluso eliminar) las esperas en la línea. Esto tenderá a reducir el tamaño de trama óptimo respecto a un protocolo de parada y espera. Por otro lado, el hecho de funcionar con retroceso n también provoca que el tamaño de trama óptimo sea menor ya que en caso de producirse un error se habrán de retransmitir normalmente varias tramas (hasta 7 en el caso mas desfavorable).

El reparto no homogéneo de los bits erróneos supone que estos se concentren en algun caso; habrá entonces alguna trama que tendrá más bits erróneos de lo que le corresponde, y a cambio alguna trama tendrá menos bits erróneos que los que en principio le corresponderían. Esto hace que de vez en cuando una trama que en principio debería tener un error sea correcta. El efecto será equivalente a una reducción del BER, por lo que *el tamaño de trama óptimo aumentará*.

Suponiendo que los bits erróneos se dan según una distribución de Poisson la probabilidad de que una trama de longitud *t* bits tenga exactamente *r* bits erróneos es:

$$p(x=r) = \frac{(t*BER)^r}{r!} e^{-t*BER}$$

La probabilidad para *r = 0* (trama correcta) es:

$$p(x=0) = e^{-t*BER}$$

Con lo que:

$$E_{BER} = e^{-t*BER}$$

$$E = E_{ov}*E_{BER} = (1 - ov/t) * e^{-t*BER}$$

Para calcular el valor de t que maximiza E derivamos e igualamos a cero:

$$\frac{dE}{dt} = \frac{ov}{t^2} e^{-t*BER} + \left(1 - \frac{ov}{t} \right) (-BER) * e^{-t*BER} = 0$$

$$\left(\frac{ov}{t^2} - BER + \frac{BER * ov}{t} \right) * e^{-t*BER} = 0$$

$$\left(\frac{ov}{t^2} - BER + \frac{BER * ov}{t} \right) = 0$$

$$BER*t^2 - (BER*ov) *t - ov = 0$$

$$t = \frac{BER*ov \pm SQRT ((BER*ov)^2 + 4*BER*ov)}{2 * BER}$$

La solución negativa da *t<0*, por lo que es absurda y se descarta.

$$t = \frac{BER*ov + BER * ov * SQRT(1+4/(BER*ov))}{2 * BER}$$

Dividiendo todo por *BER*:

$$t = \frac{ov + ov * SQRT(1+4/(BER*ov))}{2}$$

Para *ov* = 40 bits y BER = 10^{-3} obtenemos:

t = 220,998 = **221 bits**; E = 0,656610 = **65,66 %**

Comprobamos pues que el tamaño de trama y la eficiencia óptimos son algo mayores que los calculados antes con la suposición simplista.

Como normalmente BER<<1, entonces 4/(BER*ov) >>1 por lo que se puede realizar la siguiente aproximación:

$$t = \frac{ov + ov * SQRT(4/(BER*ov)}{2 * BER} = \mathbf{ov/2 + SQRT\ (ov/BER)}$$

Para BER pequeño el término predominante es el segundo, con lo que la solución tiende a la obtenida inicialmente conforme el valor de BER se reduce; como era de esperar cuanto mas pequeño es BER menor es el error introducido por la suposición del reparto homogéneo.

Con esta fórmula simplificada obtenemos para *ov* = 40 bits y BER = 10^{-3} los valores:

t = **220 bits**; E = 0,6566 = **65,66 %**

Para t = 200 obtenemos E = 0,6550