TEMA 2: La capa de enlace de datos.

2.1 Introducción.

En este tema desarrollaremos una descripción de la capa de enlace de datos del modelo de referencia OSI, equivalente a la capa nodo a red del modelo de referencia TCP/IP.

La capa de enlace de datos, que se sitúa encima de la capa física, tiene como objetivo lograr la comunicación fiable y eficiente entre dos máquinas adyacentes en la capa de enlace de datos, esto es, dos máquinas que están conectadas físicamente por un canal de comunicaciones cuya propiedad esencial es que los bits son entregados en el destino en el mismo orden en que fueron enviados por el origen.

Aunque el problema pueda padecer trivial, pues la máquina origen pone los bits en el alambre y estos llegan a la máquina destino, existen problemas tales como la comisión ocasional de errores (perdida de un bit, cambio del valor de un bit, etc.), la velocidad finita de transmisión de los datos, que la cual provoca retardos entre el momento en que se envía un bit y se recibe, etc., problemas que deben ser solucionados. De forma general, los errores y velocidades de transmisión dependen del medio de transmisión utilizado: La fibra óptica y las redes locales suelen tener las tasas más bajas de errores y las mayores velocidades de transmisión, mientras que las transmisiones inalámbricas con equipos móviles (GSM o LANs inalámbricas) o sobre telefonía analógica suelen tener las más altas de errores y menores velocidades de transmisión.

La capa de enlace de datos puede diseñarse para ofrecer distintos tipos de servicios, pudiendo variar los servicios disponibles de un sistema a otro. Generalmente, los servicios ofrecidos por la capa de enlace de datos pueden ser de tres tipos:

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

En el primer tipo de servicio, el envío se hace sin esperar ninguna indicación del receptor sobre el éxito o fracaso del envió. Este tipo de servicio es apropiado cuando la tasa de errores es muy baja y se deja la misión de comprobar la corrección de la transmisión a las capas superiores; o bien cuando se quiere transmitir información en tiempo real (voz, etc.), en cuyo caso lo importante no es recibir correctamente todas los datos sino la velocidad de transmisión de los mismos.

En el segundo tipo de servicio se produce un acuse de recibo para cada marco enviado. De esta manera el emisor puede estar seguro del éxito del envió, pudiendo reenviarlo si no ha sido recibido en un tiempo especificado. Proporcionar acuse de recibo en la capa de enlace de datos es una optimización de la comunicación, pues esta tarea puede ser asumida por la capa de transporte, la cual puede enviar un mensaje y si no llega de forma completa reenviarlo. El problema con esta estrategia es que si el

mensaje ha sido dividido por las capas inferiores en, por ejemplo, en diez tramas y solo se ha perdido una trama, la capa de transporte debe reenviar el mensaje entero, con sus diez tramas y no solo la trama perdido.

El tercer tipo de 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 transmitidos a la capa superior (capa de red) una y sólo una vez. En este tercer servicio, el servicio orientado a conexión, se pueden distinguir tres fases distintas en la transferencia:

1. Establecimiento de la conexión, haciendo que ambos lados inicialicen las variables y contadores necesarios para seguir la pista a las tramas que son recibidos y a los que no.

2. Transmisión de los datos mediante una o más tramas.

3. Cierre de la conexión, liberando las variables, los buffers y demás recursos necesarios para mantener la conexión.

Estudiaremos a partir de ahora las como especificar las tramas, como detectar y corregir errores en la transmisión y como controlar el flujo de datos, misiones fundamentales de la capa de enlace de datos.

2.2 Especificación de las tramas.

En la capa física, el envío de información se hace en forma de bits sueltos; la capa de red actúa de manera distinta: construye con los bits paquetes discretos denominados tramas (frames en inglés) que son los que envía por la línea. Según el tipo de red la trama puede oscilar entre unos pocos y unos miles de bytes. La utilización de tramas permite simplificar el proceso de detección y corrección de errores. Una buena parte de las tareas de la capa de enlace tienen que ver con la construcción e identificación de las tramas.

La división en tramas de los bits es más difícil de lo que parece a primera vista. Una manera de lograr está división es introducir intervalos de tiempo entre los tramas, a semejanza de los espacios en el texto. Sin embargo, las redes pocas veces ofrecen garantías sobre la temporización, por lo que es posible que estos intervalos sean eliminados o que puedan introducirse otros intervalos durante la transmisión.

Dado que es demasiado riesgo depender de la temporización para marcar el inicio y el final de una trama, se han diseñado varios métodos para ello. Veremos a continuación tres de ellos: Cuenta de caracteres, caracteres de inicio y final con caracteres de relleno y bits indicadores de inicio y final con bits de relleno.

2.2.1 Cuenta de caracteres.

Este primer método se basa en especificar en un campo del encabezado el número de caracteres de la trama. Cuando la capa de enlace de datos del destino ve la

cuenta de caracteres, sabe cuántos caracteres siguen y por tanto dónde está el final de la trama. Esto puede verse en la siguiente figura (figura 2.2.1.1).

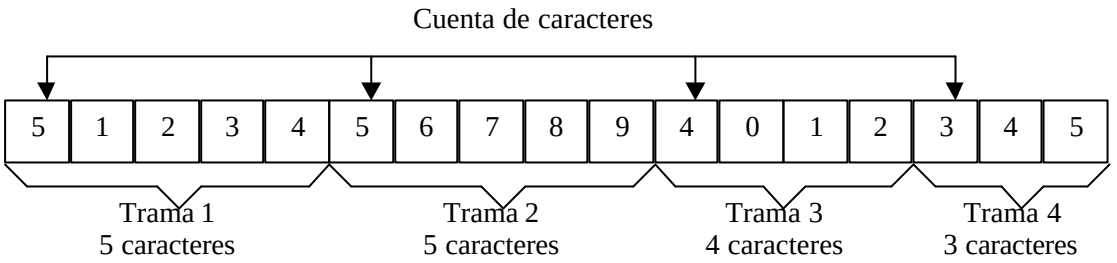

Figura 2.2.1.1: Cuenta de caracteres como especificación de las tramas.

El problema de la cuenta de caracteres es que la cuenta puede alterarse por un error de transmisión. Por ejemplo, si en el anterior ejemplo (figura 2.2.1.1) la cuenta de caracteres de la segunda trama, que tiene un valor 5 se convierte por un error en un solo bit en el valor 7, el destino perderá la sincronía y será incapaz de localizar el inicio correcto de la siguiente trama, perdiendo por tanto todas las tramas a partir de este error, tal y como puede verse en la figura 2.2.1.2.

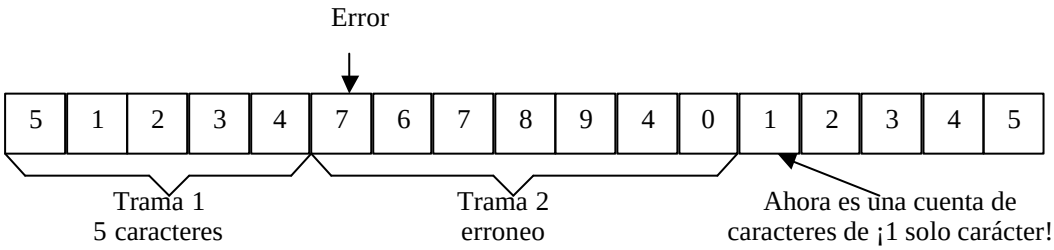

Figura 2.2.1.2: Un error en la cuenta de caracteres como especificación de las tramas.

2.2.2 Caracteres de inicio y final con caracteres de relleno.

El segundo método supera el problema de resincronización tras un error al hacer que cada trama comience con la secuencia especial de caracteres ASCII DLE STX y termine con la secuencia especial de caracteres DLE ETX, donde DLE es Data Link Escape, escape de enlace de datos, STX es Start of TeXt, inicio de texto y ETX es End of TeXt, fin de texto. De esta manera, si el destino pierde el limite de las tramas, todo lo que tiene que hacer es buscar los caracteres DLE STX o DLE ETX para encontrar el principio o el fin de una trama.

Sin embargo, este sistema presenta un importante problema. Cuando se transmiten datos binarios, como programas objeto, etc., puede ocurrir fácilmente que los caracteres correspondientes a DLE STX o DLE ETX aparezcan en los datos, lo cual interferirá en la delimitación de las tramas. Una forma de resolver esto es hacer que la capa de enlace de datos inserte un carácter ASCII DLE justo antes de cada carácter DLE que aparezca en los datos. Esto produce que un DLE STX o DLE ETX de enmarcado puede distinguirse por la ausencia o presencia de un solo DLE. Los DLE de los datos siempre se duplican. En la figura 2.2.2.1 podemos ver un ejemplo de flujo de datos antes del relleno de caracteres y después del relleno de caracteres.

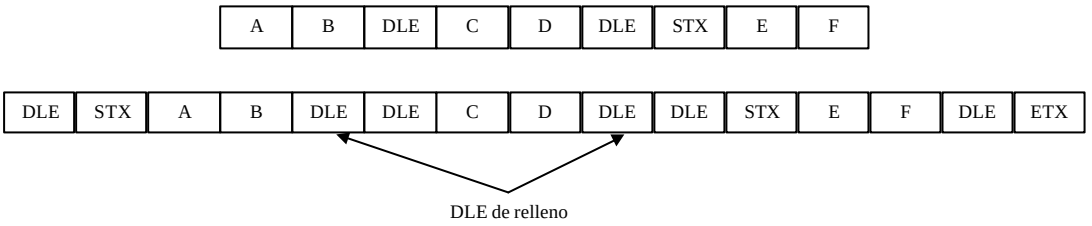

Figura 2.2.2.1: Datos enviados por la capa de red y datos después del relleno de caracteres por la capa de enlace de datos.

El principal problema que tiene el uso de esta técnica de enmarcado 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. Además, tampoco es posible enviar tramas cuyo tamaño no sea múltiplo de ocho bits.

2.2.3 Bits indicadores de inicio y final con bits de relleno.

Para evitar los problemas anteriormente expuestos, se diseño un tercer método, que podríamos considerar una generalización del anterior, consistente en utilizar una determinada secuencia de bits para indicar el inicio y final de una trama. De esta forma se permiten tramas de cualquier tamaño y el método es independiente del código ASCII.

La técnica funciona de la siguiente manera: Cada marco comienza y termina con un patrón especial de bits, el patrón 01111110, llamado byte *indicador*. El receptor esta permanentemente analizando en la secuencia de datos que recibe la presencia de un byte indicador y en cuanto lo detecta sabe que ha ocurrido el inicio (o final) de una trama. Aunque el byte indicador tiene ocho bits, el receptor no realiza el análisis byte a byte, sino bit a bit, por lo cual la secuencia 01111110 puede aparecer “a caballo” entre dos bytes y el receptor lo interpretará como la aparición de un byte indicador, permitiendo este hecho el envío de tramas cuya longitud no es múltiplo de ocho bits.

En este enfoque queda por resolver el problema de que los datos a transmitir contengan en sí mismos la secuencia de bits 01111110. Para evitar la aparición de esta secuencia en los datos se emplea la técnica conocida como relleno de bits o inserción de bit cero (bit stuffing o zero bit insertion). Está técnica consiste en que el emisor, cuando detecta en el flujo de bits de salida que cinco bits contiguos tienen el valor 1, inserta automáticamente un bit con valor 0. De esta forma la secuencia 01111110 no puede nunca aparecer como parte de los datos a transmitir. El receptor, por su parte, realiza la función inversa, analizando el flujo de bits entrante y suprimiendo de los datos el 0 recibido después de cinco 1 consecutivos. Un ejemplo de relleno de bits puede verse a continuación en la figura (2.2.3.1).

Con el relleno de bits, si el receptor pierde la noción de donde se encuentra, bastará con que espere la llegada de la secuencia 01111110 para conocer que se encuentra al principio o final de una trama, pues el byte indicador solo puede aparecer en los límites de las tramas y nunca en los datos.

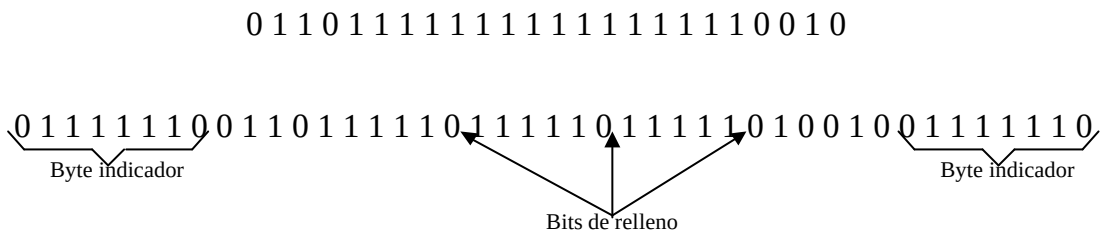

Figura 2.2.3.1: Especificación de tramas mediante relleno de bits.

2.3 Control de errores.

Una vez resuelto el problema de indicar el inicio y final de una trama, surge el siguiente problema: Cómo asegurar que todas las tramas sean entregados, a la capa de red del ordenador destino, en el orden apropiado. Un transmisor puede dedicarse a enviar tramas sin importarle si están llegando en el caso de un servicio no orientado a conexión y sin acuse de recibo, pero esto no sería correcto para un servicio orientado a conexión y con acuse de recibo.

La forma más simple de asegurar la entrega correcta de los datos es proporcionar al transmisor información sobre lo que ocurre al otro lado de la línea. La forma más simple es enviar del receptor al transmisor tramas de control que indiquen si una trama de entrada a llegado de forma correcta o no. De esta forma el transmisor sabe si una trama ha llegado de forma correcta o debe retransmitirlo.

Sin embargo, debido a errores en la línea, una trama puede desaparecer completamente de la línea. En ese caso, el receptor nunca informará de que no le ha llegado correctamente el marco, pues no ha recibido nada, y el emisor no sabrá que debe reenviarlo. Para evitar este problema se suele establecer un tiempo máximo (timer) en el que el receptor deberá reaccionar, adoptando el emisor medidas en caso contrario (por ejemplo reenviar la trama). La elección del timer adecuado a cada circunstancia es muy importante: Si se elige muy corto se producirán retransmisiones innecesarias, y si se elige muy largo se perderá mucho tiempo (y por tanto mucha eficiencia del canal) esperando confirmaciones inexistentes. Además, en el caso de reenviar tramas por expiración del timer puede suceder que el receptor reciba correctamente la misma trama dos veces; en este caso deben habilitarse los mecanismos necesarios para asegurar que a la capa de red no se le pasará la trama duplicada.

Los sistemas de control de errores han evolucionado en dos sentidos. El primero se basa en incluir suficiente información redundante en cada trama transmitida para que el receptor pueda deducir lo que debió ser la trama transmitida. El segundo sentido se basa en incluir sólo suficiente redundancia para que el receptor sepa que ha ocurrido un error (pero no qué error) y entonces solicite la retransmisión. La primera estrategia usa códigos de corrección de errores y la segunda códigos de detección de errores. Los códigos de corrección de errores se utilizan generalmente cuando el servicio ofrecido es no orientado a conexión y sin acuse de recibo, pues en este caso no existe medio de informar al emisor del error producido. Los códigos de detección de errores se usan de forma general en los otros dos servicios, en los cuales existe acuse de recibo y el emisor tiene la posibilidad de reenviar la trama errónea.

Veremos a continuación una introducción a la distancia de Hamming para pasar con posterioridad a ver ambos sistemas de control de errores.

2.3.1 Distancia de Hamming.

Supongamos una trama con m bits de datos (es decir, de mensaje) y r bits redundantes o de comprobación. Sea n la longitud total (es decir n=m+r). A esta unidad de n bits que contiene datos y bits de comprobación la denominaremos como *palabra código* de n bits.

Dadas dos palabras código cualquiera (por ejemplo 10001001 y 10110001), es posible determinar cuántos bits correspondientes difieren (en este caso difieren en tres bits). La cantidad de posiciones de bit en las que difieren dos palabras código se conoce como *distancia de Hamming*. Su significado es que, si dos palabras código están separadas una distancia de Hamming d, se requerirán d errores de un bit para convertir una en la otra.

En la mayoría de las aplicaciones de transmisión de datos, todos los 2^m mensajes de datos posibles son usados, pero debido a la manera en que se calculan los bits de comprobación, no se usan todas las 2^n palabras código posibles. Por ello, conocido el algoritmo de cálculo de los bits de comprobación, es posible construir la lista completa de palabras código legales y, en esta lista, encontrar las dos palabras código cuya distancia de Hamming es mínima. Esta distancia mínima es la distancia de Hamming de todo el código.

Las propiedades de detección y corrección de errores de un código dependen de su distancia de Hamming, de forma que para detectar d errores se necesita un código con distancia d+1, pues con tal código no hay manera de que d errores de un bit puedan cambiar una palabra código válida a otra. De manera parecida, para corregir d errores se necesita un código con distancia 2d+1, pues así las palabras código legales están tan separadas que, aún con d cambios, la palabra código original sigue estando más cercana que cualquier otra palabra código, por lo que puede determinarse de manera biunívoca.

Como ejemplo sencillo de código de detección de errores, considere un código en el que se agrega un bit de paridad a los datos. El bit de paridad se escoge de forma que la cantidad de bits 1 en la palabra código sea par (o impar). De esta forma, cuando se desea enviar 10110101 con paridad par se envía 101101011, y cuando se desea enviar 10110001 con paridad también par se envía 101100010. Un código con un solo bit de paridad tiene una distancia de Hamming de 2, pues cualquier error de un bit produce una palabra código con la paridad equivocada.

Como ejemplo de corrección de errores, consideremos un código con sólo cuatro palabras código validas: 0000000000, 0000011111, 1111100000 y 1111111111. Este código tiene una distancia de Hamming de 5, lo que significa que es posible corregir errores dobles. Si llega la palabra código 0000000111, el receptor sabe que el original debió ser 0000011111. Sin embargo, si un triple error cambia 000000000 a 0000000111, el error no se corregirá adecuadamente.

2.3.2 Códigos de corrección de errores.

Una vez hemos estudiado la distancia de Hamming, estamos en condiciones de diseñar códigos de corrección de errores para un código con m bits de mensaje y r bits de comprobación que permitirá la corrección de todos los errores individuales. Cada uno de los 2^m mensajes legales tiene n palabras código ilegales a una distancia 1 de él. Éstas se forman invirtiendo sistemáticamente cada uno de los n bits de la palabra código de n bits formada a partir de él. Dado que la cantidad de patrones de bits es 2^n, debemos tener $(n+1)2^m <= 2^n$. Usando que n=m+r, este requisito se convierte en $(m+r+1) <= 2^r$. Dado m, esto impone un límite inferior a la cantidad de bits de comprobación r necesarios para corregir errores individuales.

Este límite inferior teórico puede lograrse usando un método desarrollado por el propio Hamming. Los bits de la palabra código se numeran consecutivamente, comenzando por el bit 1 a la izquierda. Los bits que son potencias de 2 (1, 2, 4, 8, 16, etc.) son bits de comprobación. El resto (3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, etc,) se rellenan con los m bits de datos. Cada bit de comprobación obliga a que la paridad de un grupo de bits, incluyéndolo a él mismo, sea par (o impar). Un bit puede estar incluido en varios cálculos de paridad. Así, el bit de datos situado en la posición k es comprobado por los bits de paridad resultante de escribir k como suma de potencias de 2. Por ejemplo, 11=1+2+8 y 13=1+4+8, con lo cual el bit 11 esta comprobado por los bits de paridad 1, 2 y 8 y el bit 13 por los bits de paridad 1, 4 y 8.

Al llegar una palabra código, el receptor inicializa a cero un contador y examina cada bit de comprobación k (k=1, 2, 4, 8, ...) para ver si tiene la paridad correcta. Si no, suma k al contador. Si el contador es igual a cero tras haber examinado todos los bits de comprobación (es decir, si todos fueron correctos), la palabra código se acepta como válida. Si el contador es diferente de cero, contiene el número del bit incorrecto.

Por ejemplo, supongamos que deseamos mandar el símbolo ASCII “a” (valor decimal 97). Su representación binaria en código ASCII de 8 bits es 00110001. Debemos usar por tanto una palabra código de 12 bits, pues m=8 y r=4. Veamos el cálculo de los bits de comprobación. Para ello descompongamos los 8 primeros números de la representación de los datos como suma de potencias de dos: 3=1+2, 5=1+4, 6=2+4, 7=1+2+4, 9=1+8, 10=2+8, 11=1+2+8, 12=4+8. Por tanto los bits de comprobación se calculan como: b(1)=P(3, 5, 7, 9, 11), b(2)=P(3, 6, 7, 10, 11), b(4)=P(5, 6, 7, 12) y b(8)=P(9, 10, 11, 12), donde b(k) representa el bit de comprobación k y P(a, b, ...) es la función paridad de los bits a, b, etc. Entonces, la palabra código a enviar, si usamos paridad par, es: 100101110001, donde los subrayados son los bits de comprobación.

Los códigos de Hamming sólo pueden corregir errores individuales. Sin embargo, existe una forma para que los códigos de Hamming corrijan varios errores consecutivos. Se dispone como matriz una secuencia de k palabras código consecutivas, con una palabra código por fila. Una vez dispuesta la matriz, se envía como columnas, comenzando por la columna de la izquierda. Cuando la trama llega al receptor, la matriz se reconstruye, una columna a la vez. Si sucede un error de longitud k, cuando mucho habrá afectado a 1 bit de cada una de las k palabras código, por lo cual, y como el código de Hamming puede corregir un error de una palabra código, puede restaurarse la totalidad del bloque. Este método usa kr bits de comprobación para inmunizar bloques

de km bits de datos contra una sola ráfaga de errores (varios errores consecutivos) de longitud k o menos. Un ejemplo de uso del código de Hamming para corregir errores en ráfaga puede verse en la figura siguiente (figura 2.3.2.1).

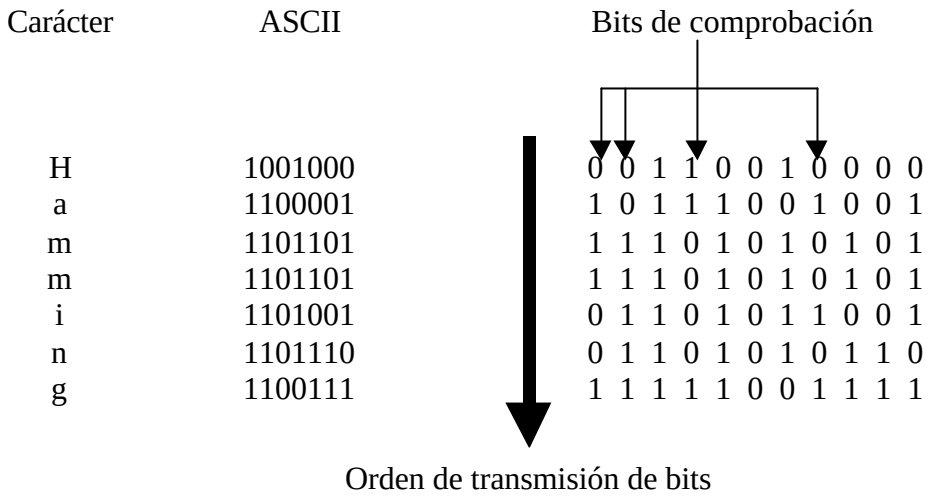

Figura 2.3.2.1: Uso del código de Hamming para corregir varios errores consecutivos.

2.3.3 Códigos de detección de errores.

Con mayor frecuencia en las transmisiones se prefiere la detección de errores seguida de la retransmisión porque es más eficiente. Supongamos errores aislados con una tasa de errores de 10^{-6} por bit y un tamaño de marco de 1.000 bits. Para proporcionar corrección de errores en bloques de 1.000 bits necesitamos 10 bits de comprobación; un megabit de datos requerirá por tanto 10.000 bits de comprobación. Para comprobar una trama con 1 bit de error basta con un bit de paridad por bloque. Por cada 1.000 bloques se tendrá que transmitir un bloque extra (1.001 bits). El gasto extra del método de detección de errores y retransmisión es de sólo 2.001 bits por megabit de datos, contra 10.000 bits con un código de Hamming.

Si se agrega una solo bit de paridad a un bloque y el bloque viene alterado por una ráfaga de errores prolongada, la probabilidad de que se detecte el error es de 0.5, lo que difícilmente es aceptable. Puede mejorarse la probabilidad considerando a cada bloque como una matriz rectangular de n bits de ancho y k bits de alto. Se calcula un bit de paridad para cada columna y se agrega a la matriz como última fila. La matriz se transmite entonces fila por fila. Si al llegar el bloque el receptor comprueba que algún bit de paridad está mal, solicita la retransmisión del bloque.

Este método puede detectar una sola ráfaga de duración n, pues sólo se cambiará un bit por columna. Sin embargo, toda ráfaga de duración n+1 puede pasar sin ser detectada. Si el bloque se altera por múltiples ráfagas de error, la probabilidad de que cualquiera de las n columnas tenga, por accidente, la paridad correcta es 0.5, por lo que la probabilidad de aceptar un bloque alterado es de 2^{-n}.

En la práctica, el método anterior no suele usarse, siendo sustituido por el *código polinómico*, también conocido como *código de redundancia cíclica o código CRC*. Los códigos polinómicos se basan en el tratamiento de cadenas de bits como

representaciones de polinomios con coeficientes 0 y 1 solamente. Un marco de k bits se considera como la lista de coeficientes de un polinomio que van de x^{k-1} a x^0. El bit mayor (más a la izquierda) es el coeficiente de x^{k-1}, el siguiente bit es el de x^{k-2} y así sucesivamente. Por ejemplo, 110001 tiene 6 bits y por tanto representa un polinomio de seis términos con coeficientes 1, 1, 0, 0, 0 y 1: $x^5+x^4+x^0$.

La generación y tratamiento matemático de los códigos de redundancia cíclica excede el nivel de este curso, pero basta indicar que un código polinómico con r bits de comprobación detectará todos los errores en ráfaga de longitud <=r y puede además determinarse que para errores mayores que r, la probabilidad de que un marco incorrecto no sea detectado es 2^{-r}.

En la actualidad, existen cuatro polinomios que se han convertido en estándares internacionales:

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

Los cuatro tienen x+1 como factor primo. CRC-12 se usa cuando la longitud de un carácter es de 6 bits, CRC-16 y CRC-CCITT con caracteres de 8 bits y CRC-32 se usa para redes LAN.

2.4 Control de flujo.

Cuando dos ordenadores se comunican generalmente han de adoptarse medidas para asegurar que lo hacen al mismo ritmo. Si la línea entre ellos es de baja velocidad probablemente el factor que limite la velocidad será la conexión, pero si es un canal rápido (por ejemplo una red local) a menudo la velocidad de transmisión efectiva vendrá marcada por el más lento (o el más cargado) de los dos ordenadores. Es preciso habilitar mecanismos que garanticen que el ordenador rápido no abrumará al lento hasta el punto de que se pierdan tramas. Respetando esta restricción los ordenadores deben intentar comunicar a la máxima velocidad que permita el canal físico.

El mecanismo que se ocupa de controlar el ritmo de la transmisión para asegurar que ésta es soportable por ambas partes se denomina *control de flujo.* Generalmente se implementa por procedimientos que permitan al receptor informar al emisor de su situación. Veremos a continuación varios de estos mecanismos de control de flujo que se basan en este principio de información del receptor al emisor.

2.4.1 Protocolo de parada y espera.

Como caso más sencillo 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. 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. La comunicación puede realizarse mediante un canal semi-dúplex, pues la comunicación no ocurre simultáneamente en los dos sentidos.

2.4.2 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.

2.4.3 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 Kbps 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 (640/64.000) o, dicho de otro modo, transmite 64 bits cada milisegundo. Por otro lado los bits tardan 10 ms en llegar de A a B (puesto 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 puede verse en la figura 2.4.3.1. En dicha figura puede verse que de cada 30 ms se está transmitiendo 10 ms y esperando 20 ms, con lo cual la eficiencia de utilización de la línea es del 33%.

Instante (ms)	Suceso en A	Suceso en B
0	Emite primer bit de trama 1	Espera
10	Emite último bit de trama 1	Recibe primer bit de trama 1
20	Espera	Recibe último bit de trama 1 Envía ACK
30	Recibe ACK Emite primer bit de trama 2	Espera
...	...	...

Figura 2.4.3.1: Ejemplo de secuencia de acontecimientos en un protocolo de parada y espera.

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 obtener una expresión que nos permita calcular la eficiencia a partir de estos valores:

$$Eficiencia = \frac{\frac{t}{v}}{\frac{t}{v} + 2t} = \frac{t}{t + 2vt}$$

Que aplicada al ejemplo anterior nos proporciona la eficiencia del 33% indicada anteriormente.

Si en vez de una línea de 64 Kbps hubiera sido una de 2.048 Mbps 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 "buffer", 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 podemos utilizar un número de secuencia, que nos interesa que sea del menor tamaño posible, pues va a aparecer en todas las tramas y mensajes ACK. Así, 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.* 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. Con un número de secuencia de *n* bits se puede tener como máximo una ventana de 2^n-1 tramas, no de 2^n, pues 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; pues si se utilizara una ventana de 2^n podrían producirse conflictos.

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:

$$W = 1 + \frac{2vt}{t}$$

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 2τ=0,02, *v*=64.000 y *t*=640 obtenemos W=3, por tanto la ventana mínima es 3; con una línea de tipo E1 (*v*=2.048.000) W=65.

La suposición que realizamos 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.*

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.

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 buffer correspondientes.

3. Avanza la ventana para poder recibir siete tramas más, cuyos números de secuencia serán 7, 0, 1, 2, 3, 4 y 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.