TEMA 3: LA CAPA DE ENLACE

INDICE TEMA 3

1. FUNCIONES DE LA CAPA DE ENLACE DE DATOS

La capa de Enlace de Datos es la responsable de la transferencia de mensajes (tramas) a través del canal físico. A la vez, transforma un canal físico susceptible de provocar errores en un enlace lógico prácticamente libre de errores.

En redes de gran alcance la capa de Enlace proporciona un servicio de conexión mediante enlaces punto a punto entre cada par de nodos de la red (que estén conectados). Realiza las funciones y procedimientos necesarios para, establecer, mantener y llevar a cabo estas conexiones.

En las redes locales sus funciones tienden a ser mucho más simples, al igual que los protocolos de enlace modernos que se utilizan en la red Internet (SLIP y PPP).

Las funciones que debe realizar la capa de enlace de datos en los casos que sea necesario se pueden agrupar en los siguientes epígrafes:

a) Estructuración de Mensajes en Tramas
b) Direccionamiento
c) Control de Errores
d) Control de Transmisión y del Flujo de Datos

Estas funciones se describen en los siguientes apartados, ejemplificándolas con sus diferentes implementaciones en los protocolos de enlace clásicos: BISYNC, DDCMP, SDLC, HDLC, cuyas principales características se recogen en la tabla de la página siguiente. En los últimos apartados del capítulo se describe el protocolo PPP, admitido como estándar en la red Internet. En los apéndices se incluyen otros protocolos menores que se han utilizado tradicionalmente para la transferencia de ficheros o bloques de datos en conexiones punto a punto mediante enlaces RS-232 entre dos ETDs, ya sea vía módem o mediante un simple cable de comunicación serie.

1.1 Estructuración de Mensajes en Tramas

La *trama* es la unidad de datos que utiliza la capa de enlace. Más correctamente, según el estándar ISO/OSI, debería denominarse LPDU (Link Protocol Data Unit, Unidad de Datos del Protocolo de Enlace) de manera similar a la NPDU (Network Protocol Data Unit), la TPDU (Transport Protocol Data Unit), etc.

Las tramas generadas por la capa de enlace de datos viajan desde un nodo hasta otro que esta conectado al mismo medio físico de comunicación que el anterior. Una trama nunca salta de una subred a otra, sino que son el vehículo para transmitir datos que pueden estar viajando más allá de una subred, para atravesar un medio físico determinado.

La trama facilita la sincronización en la comunicación entre las entidades de la capa de Enlace, que consiste en la localización del comienzo y el final del bloque de información transmitido. Además, el protocolo ha de permitir la transmisión de cualquier tipo de datos que no deben de ser confundidos con información de control del protocolo aunque su codificación coincida. A esto se lo denomina *comunicación transparente*.

La información de control del protocolo viaja en la trama junto con la información a transmitir. En función de la situación que la información de control puede ocupar en la

trama, se dice que esta es *posicionalmente dependiente* (p. ej., si va siempre contenida en la cabecera de la trama) o *posicionalmente independiente* (cuando alguno de los códigos de control, puede aparecer en cualquier punto de la trama).

Existen varios métodos para lograr la transparencia de la comunicación. En este caso mencionaremos los que son o han sido más utilizados [SLOMAN 87].

	BYSYNC	**DDCMP**	**SDLC**	**HDLC**
Originador	IBM	DEC	IBM	ISO
Full-Duplex	No	Si	Si	Si
Half-Duplex	Si	Si	Si	Si
Serie	Si	Si	Si	Si
Paralelo	No	Si	No	No
Transparencia de datos	Relleno de caracteres	Cuenta	Relleno de bits	Relleno de bits
Transmisión asíncrona	No	Si	No	No
Transmisión síncrona	Si	Si	Si	Si
Punto a Punto	Si	Si	Si	Si
Multipunto	Si	Si	Si	Si
Direccionamiento de estación (multipunto)	Encabezamiento opcional	Encabezamiento	Uno o un número de octetos	Uno o un número de octetos
Detección de errores	CRC-16/12 (VRC/LRC)	CRC-16 (Hdr+Datos)	CRC-CCITT	CRC-CCITT
Detección de errores sobre	Unicamente mensajes de texto	Encabezamiento + datos por separado	Toda la trama	Toda la trama
Recuperación de errores	Parada y espera	Vuelta a atrás N	Vuelta a atrás N	Vuelta a atrás N o rechazo selectivo
Tamaño de ventana ACK	1	255	7/127	7/127
Bootstrapping	No	Si	Si	Si
Códigos de carácter	ASCII, EBCDIC, Transcode	ASCII (solamente en caracteres de control)	Cualquiera	Cualquiera
Caracteres de control	Muchos	DEL, ENQ, SYN, SOH	Flag	Flag
Formatos de trama	Numerosos	1 (3 tipos)	1 (3 tipos)	1 (3 tipos)
Control de enlace	Ecabezamiento opcional	Encabezamiento	1 ó 2 octetos	1 ó 2 octetos
Longitud del campo de datos	n x L	n x 8	n x 8	No restringida
Trama: Comienzo **Fin**	2 SYN FTX/ETB	2 SYN cuenta	Flag Flag	Flag Flag
Control de flujo	Caracteres de control	Ninguno, se descartan datos	Formatos de campo de control y mecanismos de ventanas	Trama RNR, mecanismo de ventana

1.1.1 Códigos de Control Reservados

Este método se utiliza en algunos de los viejos *protocolos orientados al carácter* (tanto los datos como la información de control se estructura en caracteres). Se habían diseñado en principio para la transmisión de textos imprimibles, donde entre los datos de

usuario no suelen aparecer caracteres de control. Un ejemplo es el código BISYNC de IBM que emplea el conjunto de caracteres ASCII donde además de los caracteres imprimibles se recogen un buen número de caracteres de control, algunos de los cuales aparecen en la siguiente tabla.

Codificación				Denominación	Descripción
Hexadecimal	Decimal	Octal	Binaria		
01	1	001	00000001	SOH	Comienzo de cabecera
02	2	002	00000010	STX	Comienzo de texto
03	3	002	00000011	ETX	Fin de texto
04	4	004	00000100	EOT	Fin de transmisión
06	6	006	00000110	ACK	Acuse de recibo correcto
10	16	020	00010000	DLE	Secuencia de escape
15	21	025	00010101	NAK	Acuse de recibo negativo
16	22	026	00010110	SYN	Carácter de sincronismo

Una de las posibles configuraciones de una trama BISYNC es la siguiente, en la cual, el campo destinado a la *cabecera* es de formato libre para el diseñador de la comunicación y el bloque denominado CRC se utiliza para la detección de errores en la trama. Como los caracteres de control STX y ETX pueden aparecer en cualquier punto de la trama, al no estar definidas las longitudes de los campos *cabecera* y *datos*, nos encontramos ante un caso en el que la información de protocolo es posicionalmente independiente.

SYN	SYN	SYN	SYN	SOH	"cabecera"	STX	"datos"	ETX	CRC

Para que la transmisión de los caracteres en el campo *datos* sea transparente, hay que enmascarar aquellos que coincidan con los que se utilizan para el control. Por ejemplo, la presencia del carácter ETX dentro del campo *datos* tendría resultados desastrosos ya que provocaría la interrupción de la lectura de la trama de forma prematura. Para evitar esto se inserta el carácter DLE delante de aquellos caracteres que puedan ser confundidos con caracteres de control. En la recepción, todo lo que va detrás de un DLE se interpreta como datos y no como carácter de control. Un DLE como dato se transmite también poniéndole otro DLE delante.

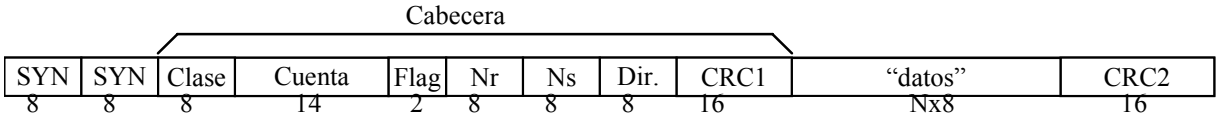
Cadena de caracteres a enviar como *datos*: a 9 STX ACK 2 6 DLE ETX 3 t y
Cadena enviada en el campo *datos* de la trama: a 9 DLE STX DLE ACK 2 6 DLE DLE DLE ETX 3 t y
Cadena recibida tras retirar los DLE: a 9 STX ACK 2 6 DLE ETX 3 t y

Como se puede comprobar, la longitud del mensaje varía al insertar el carácter DLE y además se han de ir revisando todos los datos a transmitir para insertarlo donde corresponda, lo cual, ralentiza el proceso de transmisión y hace que estos métodos sean poco eficaces para transmitir datos que no sean fundamentalmente caracteres de texto imprimibles.

1.1.2 Campo de Longitud

El *campo de longitud* es un campo dentro de la cabecera de la trama que contiene la longitud del mensaje, con lo que el receptor solo tiene que contar los bytes que van

llegando para determinar donde termina la trama. Uno de los protocolos que utiliza este método es el DDCMP desarrollado por DEC (Digital Equipment Coorporation). En este caso la cabecera no es opcional sino que tiene una estructura fija y lleva toda la información de control. Se trata pues de información de control posicionalmente dependiente. En el campo *datos* puede viajar ahora cualquier información sin problemas de transparencia, ya que en él el receptor nunca buscará caracteres de control, y el final de la trama se determinará cuando hayan llegado todos los caracteres que se esperan.



Clase: Tipo de mensaje.
Cuenta: Campo de longitud. Número de caracteres en el campo *datos*.
Flag: Control del enlace.
Nr: Número de tramas recibidas.
Ns: Número de tramas enviadas.
Dir.: Dirección de destino.

1.1.3 Inserción de Bit

Este método se utiliza en protocolos *orientados al bit* (tratan la información como bits individuales, no como caracteres. Es el caso del protocolo HDLC, el SDLC o el PPP en modo de transmisión síncrona.

Se define una única secuencia de control, denominada *Flag*, que se utiliza para delimitar el comienzo y el final de cada trama. Se trata de la secuencia 01111110. Por lo tanto, para mantener la transparencia de los datos que van entre los Flags que delimitan la trama se ha de evitar que en ella aparezcan seis bits 1 seguidos y flanqueados por dos bits 0. La técnica consiste en lo siguiente:

a) El emisor inserta un cero cada vez que detecta una secuencia de cinco bits 1 seguidos.

b) El receptor cada vez que encuentra una secuencia de cinco bits 1 seguidos comprueba si el siguiente bit, el sexto, es un 0, en cuyo caso lo elimina. Si el sexto es otro bit 1 comprueba si el siguiente bit, el séptimo, es un 0. En este caso se tratará de un Flag que marca el final de la trama. Si el séptimo bit es un 1, se habrá producido un error (aborto de la transmisión, emisor inactivo, etc.)

Cadena de bits a transmitir: 01111101 00111111 01101111 11110100
Trama enviada al medio físico: 01111110 011111<u>0</u>01 0011111<u>0</u>1 01101111 1<u>0</u>1110100 01111110
Cadena de bits en el receptor: 01111101 00111111 01101111 11110100

Como se puede ver, el método de inserción de bit varia la longitud de algunos caracteres, por lo que no se puede utilizar en transmisiones asíncronas. Cuando un protocolo como el PPP ha de funcionar sobre medios de transmisión asíncronos, se habilitan otros mecanismos para mantener la transparencia de los datos. Estas técnicas se describen en el capítulo dedicado al PPP.

Al tener que revisar la información bit a bit, los procesos de transmisión y recepción se hace más lentos por lo que esta técnica resulta poco eficiente implementada en software. Sin embargo, en la mayoría de los casos la inserción y eliminación de los bits se implementa en los circuitos electrónicos de gran escala de integración (LSI) que se emplean en las interfaces físicas convirtiéndolo en un método rápido y fiable.

1.2 Direcionamiento

El direccionamiento permite identificar el origen y el destino en un enlace de datos que interconecta varios potenciales emisores o receptores de la información (ETDs), por ejemplo en una red local.

Normalmente, cada trama contiene *explícitamente* las direcciones del origen y el destino o bien el contexto de la conexión identifica *implícitamente* el origen y/o el destino de la trama.

1.2.1 Direccionamiento Implícito

En conexiones punto a punto entre dos nodos, las tramas que envía un nodo van dirigidas forzosamente al que está en el otro extremo. En este caso no es necesario que las direcciones vayan indicadas explícitamente en la trama.

1.2.2 Direccionamiento por Preselección

En algunos buses, como el IEEE-488 (un bus paralelo equivalente al HP-IB), uno de los nodos actúa como controlador y preselecciona, previamente a la transmisión de tramas, un origen (transmisor) y un destino (receptor). Entonces el origen puede enviar un número definido de mensajes hacia el destino. Las tramas no contienen explícitamente las direcciones de origen y destino, ya que ningún otro nodo puede estar activo en ese momento.

1.2.3 Direccionamiento en Sistemas con un Único Maestro (Master-Slave)

En muchos sistemas de comunicación de carácter industrial, entre los elemento conectados a un mismo medio de transmisión, existe un elemento único que funciona como *maestro* (*Master*) y los demás como *esclavos* (*Slaves*). Todos los intercambios de tramas tienen lugar entre el maestro y un esclavo o viceversa. No hay intercambios directos de tramas entre esclavos. Si estos han de intercambiar información, lo hacen a través del maestro.

En las tramas solo ha de ir explícitamente una dirección, la de origen cuando la trama va de un esclavo al maestro o la de destino cuando viaja desde el maestro hacia uno de los esclavos.

1.2.4 Direccionamiento con Varios Maestros

En este caso cualquier nodo conectado al medio de transmisión puede enviar una trama a cualquier otro. En la trama han de ir explícitamente las direcciones de origen y destino de la misma.

1.2.5 Direccionamiento de Mensajes Multidifundidos

Cuando se pueden enviar tramas dirigidas a varios de los nodos conectados al medio físico de transmisión se habla de mensajes multidifundidos. Se distinguen dos categorías, las tramas *multicast*, dirigidas a algunos de los nodos, y las *broadcast*dirigida s a todos los nodos.

En ambos casos la trama suele llevar explícitamente dos direcciones. La de origen, que identifica al nodo emisor, y la de destino, que suele ser un código especial que identifica de forma global a algunos o todos los nodos de la red. Por ejemplo, en muchas redes locales un campo de dirección de destino en la trama con todos los bits a 1 significa que la trama va dirigida a todas las estaciones.

1.3 Control de Errores

Los circuitos de comunicación a larga distancia suelen tener una tasa de errores comparativamente alta frente a otros medios como las redes locales. En ese caso la capa de Enlace de Datos es el encargado de conseguir un enlace lógico libre de errores, lo que implica tanto la detección como la corrección de errores. En el caso de las redes locales, dada la baja tasa de errores, la capa de Enlace solo realiza la detección de errores, dejando los algoritmos o mecanismos de corrección de los mismos a las capas superiores.

Los errores pueden producirse en los nodos terminales o de conmutación de la red, en las interfaces que los conectan a las líneas de transmisión o en las propias líneas de transmisión. Si bien en los dos primeros casos los errores son muy poco probables, en las líneas de transmisión los errores pueden variar entre 1 bit erróneo entre cada 10^3 (líneas de par trenzado sin apantallar, comunicaciones vía radio, ...) y 1 bit erróneo entre cada 10^9 (líneas de fibra óptica). Además, lejos de tender a desaparecer, la creciente tendencia del mercado hacia las comunicaciones sin cables hace que la presencia de errores en los medios de transmisión sea algo cada vez más habitual.

La aceptabilidad de los errores en comunicaciones depende del contenido de los datos, la utilización final de los mismos y la mayor o menor dificultad que conlleva su corrección. La corrección de errores es un tema prácticamente independiente de su detección. En general, si se dispone de la información, el método más simple para corregir un error es la retransmisión del dato. Los algoritmos de corrección de errores son costosos computacionalmente y añaden una gran sobrecarga a la información transmitida. A veces un mismo algoritmo puede servir para la detección y la corrección de errores.

1.3.1 Origen de los errores

Los errores se producen por dos tipos de fallos diferentes en su naturaleza: sucesos estáticos, cuyo comportamiento y existencia son conocidos de antemano, y sucesos transitorios, que aparecen de forma aleatoria. Como ejemplo de los primeros se pueden

citar la distorsión de señal y las pérdidas por atenuación. Dentro de la segunda clase se encuentran las perturbaciones eléctricas y electromagnéticas.

Los errores que proceden de sucesos estáticos resultan mucho más sencillos de manejar, porque sus efectos son predecibles de antemano. Así, se compensa la atenuación de altas frecuencias con amplificadores ecualizadores; las distorsiones se corrigen con cables de baja capacidad o baja inductancia; el apantallamiento de los conductores disminuye las interferencias de radiofrecuencia. En resumen, con una buena planificación se puede diseñar una línea capaz de comunicar dos puntos cualesquiera con un número mínimo de errores estáticos.

Las causas transitorias no son tan fáciles de tratar. La mayoría de los errores transitorios están producidos por interferencias eléctricas y electromágnéticas en la línea de comunicaciones. Estas interferencias se derivan de meteoros atmosféricos (rayos), de la presencia de otros dispositivos emisores de ondas electromagnéticas, encendido o apagado de fluorescentes o motores, perturbaciones en la red eléctrica, etc. Cuando la línea de comunicaciones se basa en la red telefónica, sufre una nueva categoría completa de ruidos: caídas de línea, diafonía, ecos, ...

El ruido más común, especialmente en líneas telefónicas, es el impulsivo o en ráfagas (busrt noise). Está compuesto por períodos de ruido disruptivo flanqueados por otros períodos sin ruido. La comunicación serie, por su naturaleza, es muy sensible a los ruidos impulsivos, ya que al circular la información bit a bit, el tiempo en que el dato está expuesto a la aparición de ruido es mayor. Por ejemplo, un ruido de 0,01 segundos en una comunicación a 1200 bps contamina por completo una ráfaga de 12 bits.

En general, la mayoría de los casos los errores se deben a interferencias electromagnéticas, lo que implica que los errores no se suelen presentar en forma de bits individuales, sino en forma de ráfagas de bits erróneos. En otros casos, hay ciertos errores con consecuencias más catastróficas que otros, por ejemplo, si el error se produce en el campo de dirección de destino de una trama, esta jamás llegará a su destinatario.

1.3.2 Detección de Errores

Los mecanismos de detección de errores se basan en la inclusión de información redundante en la trama, es decir, una serie de bits adicionales que representan de algún modo el contenido del mensaje transmitido, y que se envían conjuntamente con el propio mensaje. Aunque ningún método puede asegurar al cien por cien la detección de todos los errores que se pueden producir, un método muy efectivo podría ser la retransmisión de la trama una o varias veces, para ser comparada. Incluso se podrían corregir los errores detectados mediante la comparación de un número impar de transmisiones de la misma trama. Esto, lógicamente, reduce enormemente la capacidad del enlace, por lo que se buscan otras técnicas que mantengan unas buenas prestaciones pero con una cantidad menor de información redundante.

1.3.2.1 Paridad Vertical y Horizontal

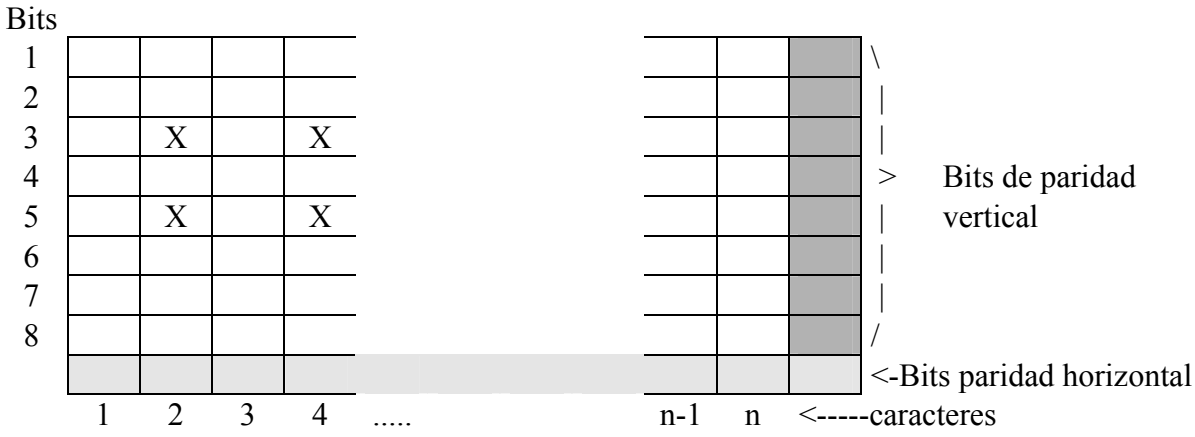
Un mecanismo de *paridad horizontal*, como el utilizado en las transmisiones asíncronas, consiste en añadir a un grupo de bits de datos (entre 5 y 8), un bit más con valor 0 ó 1 según corresponda para que el número de bits 1 total sea par (*paridad par*) o impar (*paridad impar*). Así por ejemplo, si se trabaja con paridad impar, el dato “0100001” se

transforma en el "01000011" porque se necesita el último 1 para que el byte tenga un número impar de unos. Por el contrario, si trabajásemos con paridad par el resultado sería "01000010".

Mediante este método se pueden detectar todos los errores de un único bit en el grupo. Sin embargo, la probabilidad de detectar un error que afecte a varios bits es sólo 0.5, lo que no es mucho. En concreto, la paridad puede detectar únicamente errores que afectan a un número impar de bits.

La detección de errores se puede mejorar considerablemente añadiendo a cada grupo de *n* caracteres que se envían sucesivamente, uno más que contenga bits de paridad para los bits que ocupan la misma posición en los *n* caracteres transmitidos. A esto se le conoce como *paridad vertical*.

Este método combinado es capaz de detectar todos los errores de 1, 2 ó 3 bits en el bloque, todos los errores con un número impar de bits erróneos y algunos con número par, ya que por ejemplo, no detectaría un error de cuatro bits como el señalado con las "X" en la figura. Detectará también todas las ráfagas de bits erróneos de longitud igual o inferior a 8. Una ráfaga de *r* bits erróneos no implica que todos los bits de la ráfaga hayan cambiado de valor, sino solamente que el primero y el último de la ráfaga son incorrectos. Los demás pueden serlo o no.



El porcentaje de información redundante que se añade con este método en un bloque de *n* caracteres es: (n+9)/(8·n). Si el bloque es de 128 caracteres tendríamos aproximadamente un 13% de información redundante en el bloque.

	TRANSMITIDO	RECIBIDO	
	Mensaje	Error en dos bits de una fila	Error en dos bits de dos filas
C	11000011	11000011	11000011
f	01100110	01100110	01100110
y	11111001	111**00**001	111**00**001
U	01010101	01010101	010**01**101
Cód.	00001001	00001001	00001001
	El receptor calcula →	00010001 (error detectado)	00001001 (error no detectado)

La mayor virtud de las comprobaciones de paridad en E/S serie es su sencillez, lo que permite implementarla fácilmente por hardware.

1.3.2.2 Chequeo por suma: Checksum

El emisor de la trama realiza la suma de los bytes o caracteres a transmitir en todos o en determinados campos de la trama. Esta suma realizada habitualmente en módulo 256 ó 65536, generará 8 ó 16 bits respectivamente de información para el control de errores, que se añadirán al final de la trama o del campo que se supervisa. Generalmente, si el tamaño del bloque a comprobar es suficientemente largo, el byte extra resulta insignificante comparado con el margen adicional de seguridad que se logra.

Substituye generalmente al método de paridad horizontal proporcionando un nivel de detección de errores similar. Esta técnica detecta errores de 2 bits en una fila o dos filas, pero falla en la detección de un número par de bits erróneos en una columna. También es importante señalar que esta técnica no es capaz de detectar errores de secuencia: es decir, se produce una suma idéntica cuando el mensaje se envía en cualquier otro orden al azar.

TRANSMITIDO		RECIBIDO		
	Mensaje	Error en dos bits de una fila	Error dos bits en dos filas	Error de un bit en dos columnas
C	1000011	1000011	1000011	1000010
f	1100110	1100110	1100110	1100111
y	1111001	1100001	1100001	1100001
U	1010101	1010101	1001101	1001101
Cód.	101110111	101110111	101110111	101110111
El receptor calcula →		101011111 (error detectado)	101010111 (error detectado)	101110111 (error no detectado)

1.3.2.3 Códigos de Redundancia Cíclica: CRC

El uso del *código polinomial*, más conocido como *código de redundancia cíclica* (CRC), está muy extendido. Proporciona una buena detección de errores con poca información redundante adicional. Normalmente se le añaden a la trama 16 ó 32 bits de información para la detección de errores mediante el CRC.

El método consiste básicamente en que el emisor trata la información a transmitir como una cadena de bits, y la convierte en un polinomio binario (donde los valores de los coeficientes son cero o uno). Este polinomio binario se multiplica por el grado del *polinomio generador* que es conocido tanto por el emisor como por el receptor, y posteriormente se divide por el en modulo 2 (no se llevan acarreos en la suma ni la división), generándose un resto de grado una unidad inferior al polinomio generador. Los coeficientes (ceros o unos) de este resto se añaden a la trama como código de detección de errores. Un polinomio generador de orden *n* generará un código de *n* bits.

El receptor tratará a toda la trama como un polinomio (información y código de detección de errores conjuntamente) y lo dividirá en módulo dos por el mismo polinomio

generador que uso el emisor. Si el resto de la división es cero, no se habrán detectado errores en la transmisión.

Información a enviar: 10110101101 $x^{10}+x^8+x^7+x^5+x^3+x^2+1$
Polinomio generador: 10011 x^4+x+1
Polinomio a dividir: 101101011010000 $(x^{10}+x^8+x^7+x^5+x^3+x^2+1)\cdot x^4$ $x^{14}+x^{12}+x^{11}+x^9+x^7+x^6+x^4$
División en módulo 2:

```
101101011010000   |10011          x14  x12 x11  x9  x7 x6  x4   | x4+x+1
10011             10101010010     x14       x11 x10       x10+x8+x6+x4+x
 01011                                x12  x10 x9
 00000                                x12      x9 x8
  10110                                   x10  x8 x7 x6
  10011                                   x10      x7 x6
   01011                                       x8       x4
   00000                                       x8   x5  x4
    10111                                          x5
    10011                                          x5   x2 x
     01000                                             x2 x
     00000
      10001
      10011
       00100
       00000
        01000
        00000
         10000
         10011
          00110
          00000
           0110
```

Resto obtenido: 0110 x^2+x
Trama transmitida: 101101011010110 $x^{14}+x^{12}+x^{11}+x^9+x^7+x^6+x^4+x^2+x$

El ejemplo anterior muestra el proceso en el emisor utilizando un polinomio generador de orden 4 que genera un código de detección de errores de 4 bits [HALSALL 95].

Se puede comprobar que el polinomio correspondiente a la trama transmitida es divisible por el polinomio generador utilizado en el método anterior y que, por lo tanto, la transmisión estará libre de errores si, al hacer la división el receptor, el resto obtenido es cero.

El proceso descrito puede ser excesivamente laborioso implementado en software, sobre todo cuando las tramas de información son largas y los polinomios generadores de mayor orden, lo normal es implementarlo en el hardware de comunicaciones, donde un simple circuito de desplazamiento realiza la división.

Ciertos polinomios generadores son más adecuados que otros. El CCITT y el IEEE, recomiendan como polinomio generador para CRCs de 16 bits el polinomio: $x^{16}+x^{12}+x^5+1$. Un polinomio generador de orden *r* elegido adecuadamente generará un CRC de *r* bits y detectará:

a) Todos los errores de un bit.

b) Todos los errores de dos bits.
c) Todos los errores con un número impar de bits erróneos.
d) Todas las ráfagas de errores de longitud inferior a *r+1* bits.
e) $0,5^{(r-1)}$ es la probabilidad de no detectar una ráfaga de errores de *r+1* bits.
f) $0,5^{r}$ es la probabilidad de no detectar una ráfaga de errores de más de *r+1* bits.

Por último, un CRC de 16 bits en una trama de 128 bytes, solo supone un incremento del 1,6 % en su longitud. Para un valor de comprobación de 16 bits sus características de detección son:

Errores de un solo bit:	100 %
Errores de dos bits:	100 %
Errores en un número impar de bits:	100 %
Errores en ráfagas menores de 16 bits:	100 %
Errores en ráfagas de 17 bits:	99,9969 %
Todos los demás errores en ráfagas:	99,9984 %

1.3.2.4 Verificación de Bucle

Se utiliza normalmente en configuraciones en anillo, donde el receptor no retira la trama sino que la deja en el anillo para que vuelva al emisor, quien si detecta que ha habido algún error, la retransmite. En una línea punto a punto esto supondría que el receptor devuelve la trama completa al emisor para su comprobación, lo que reduciría a la mitad la capacidad del enlace. En ninguno de los dos casos el emisor puede saber si el error se ha producido en el camino de ida o en el de vuelta si no se utiliza algún método de detección adicional.

1.3.3 Corrección de Errores

El método más sencillo y utilizado de corrección de errores es la detección de los mismos por alguno de los mecanismos señalados anteriormente y la retransmisión de la trama que contiene los errores.

Sin embargo, en algunos casos se puede incluir información redundante en la trama de manera que los errores los pueda corregir el propio receptor. La cantidad de código que se añade a la trama suele estar entorno al 50% de su longitud original. Además de su coste en tiempo de computación, es difícil diseñar algoritmos que permitan corregir tanto errores de bits dispersos como grandes ráfagas de errores y su eficacia es inferior a la detección (por término medio, uno de cada mil errores no se puede corregir). Por lo tanto sólo son rentables en enlaces con tasas de error muy altas o que tienen grandes retardos, donde la petición de retransmisión de una trama supone una pérdida de tiempo muy importante. Este es el caso de algunas comunicaciones con satélites que se encuentran muy alejados de la tierra, a los que una trama puede tardar en llegar muchos segundos o incluso minutos.

1.4 Control de Transmisión y del Flujo de Datos

En este apartado se va a tratar de ejemplificar como ha de funcionar un protocolo de enlace para evitar situaciones ambiguas o comprometidas que den lugar al bloqueo o mal funcionamiento del enlace por causas achacables al protocolo, es decir, al software de comunicación. Obviamente los errores vienen generalmente provocados por algún fallo en el medio físico que da lugar a situaciones que el protocolo ha de prever cómo detectar y resolver. Las situaciones anómalas que ha de tener en cuenta el protocolo son básicamente:

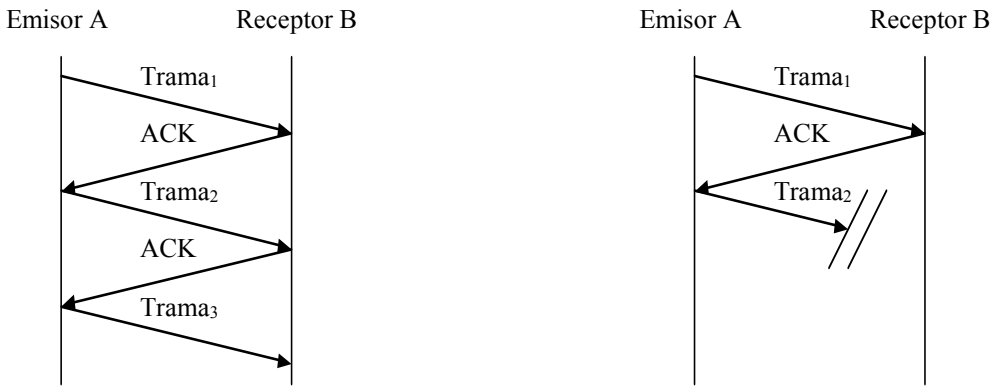
a) La pérdida de una trama
b) La aparición de tramas duplicadas
c) La llegada de tramas fuera de secuencia

Si no se tratan adecuadamente estas situaciones, la primera dará lugar a la pérdida de información en la comunicación, la segunda a la aceptación de datos erróneos como si fueran válidos y la tercera puede provocar tanto la pérdida de datos como la aceptación de datos erróneos como válidos.

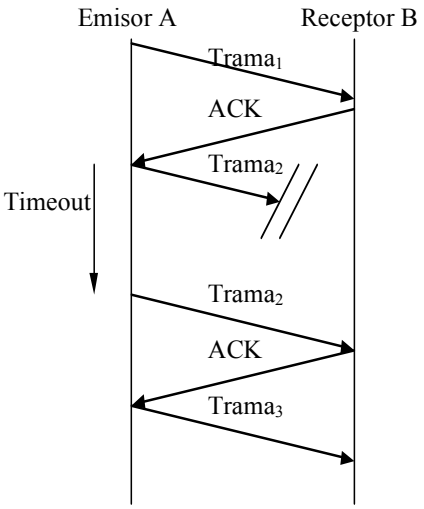
1.4.1 Protocolo Simple de Enlace de Datos

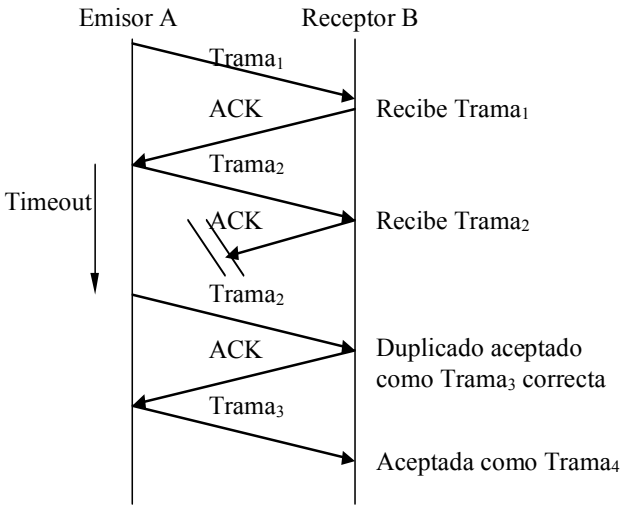
Se supone que se trabaja con un protocolo de enlace que solo envía tramas con información, sin establecer ni liberar conexiones. Si ninguna de las situaciones anómalas planteadas anteriormente se pudiera dar, el protocolo podría ser muy simple. Únicamente para evitar desbordar al receptor, el emisor enviaría una trama y esperaría un acuse de recibo antes de enviar la siguiente [SLOMAN 87].

Pero si por alguna razón una de las tramas no llega, llega incorrecta o un ACK no llega a su destino la comunicación se quedará bloqueada y el emisor ya no enviará más tramas.



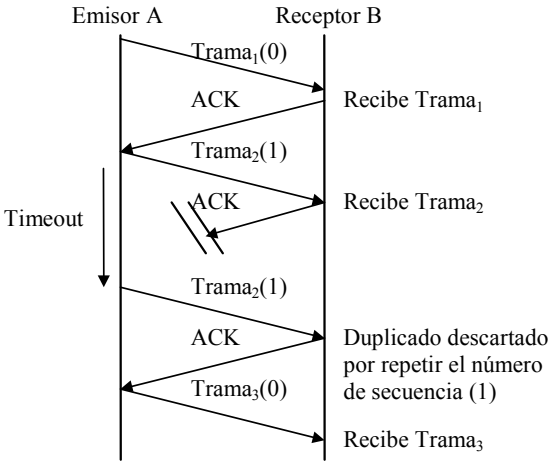
Es fácil subsanar este problema si el emisor es capaz de decidir el reenvío de la última trama si no ha recibido el ACK en un tiempo razonable. A este tiempo se le denomina *tiempo de espera* o *timeout* y su duración óptima se estima entre una y dos veces el tiempo de retardo medio necesario para el envío, procesado y respuesta a una trama.

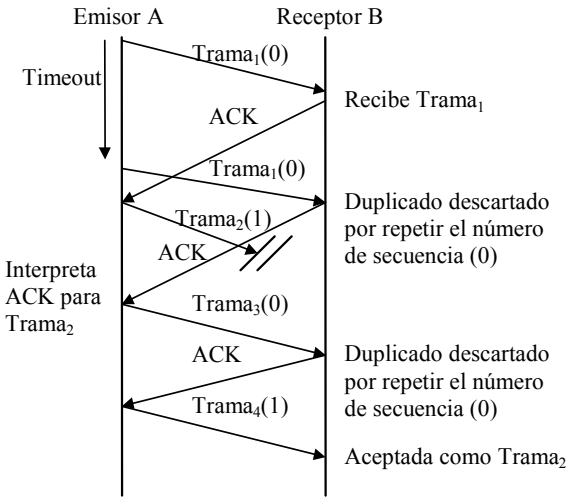




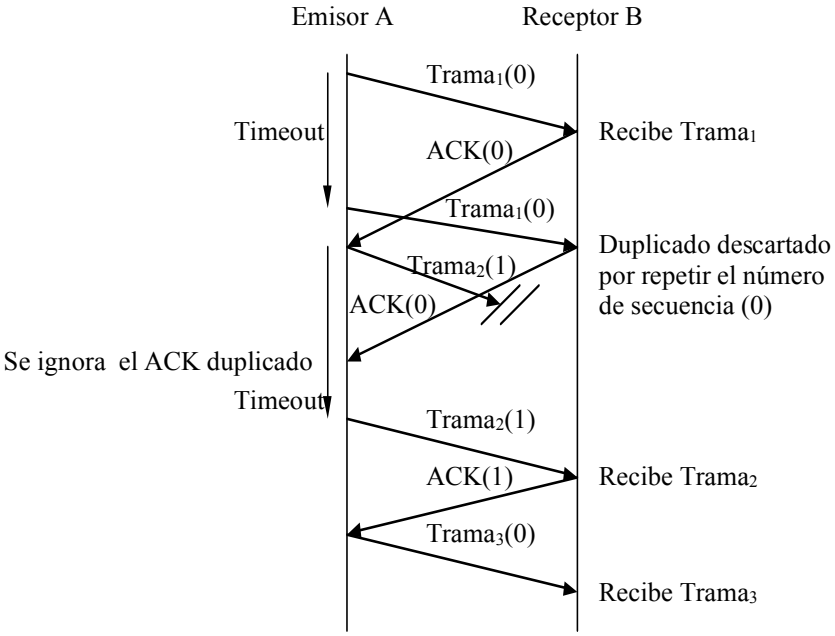
A pesar de la mejora, en el caso de que la trama pérdida sea un ACK el receptor puede aceptar una trama retransmitida por el emisor tras el timeout como si fuese una nueva trama.

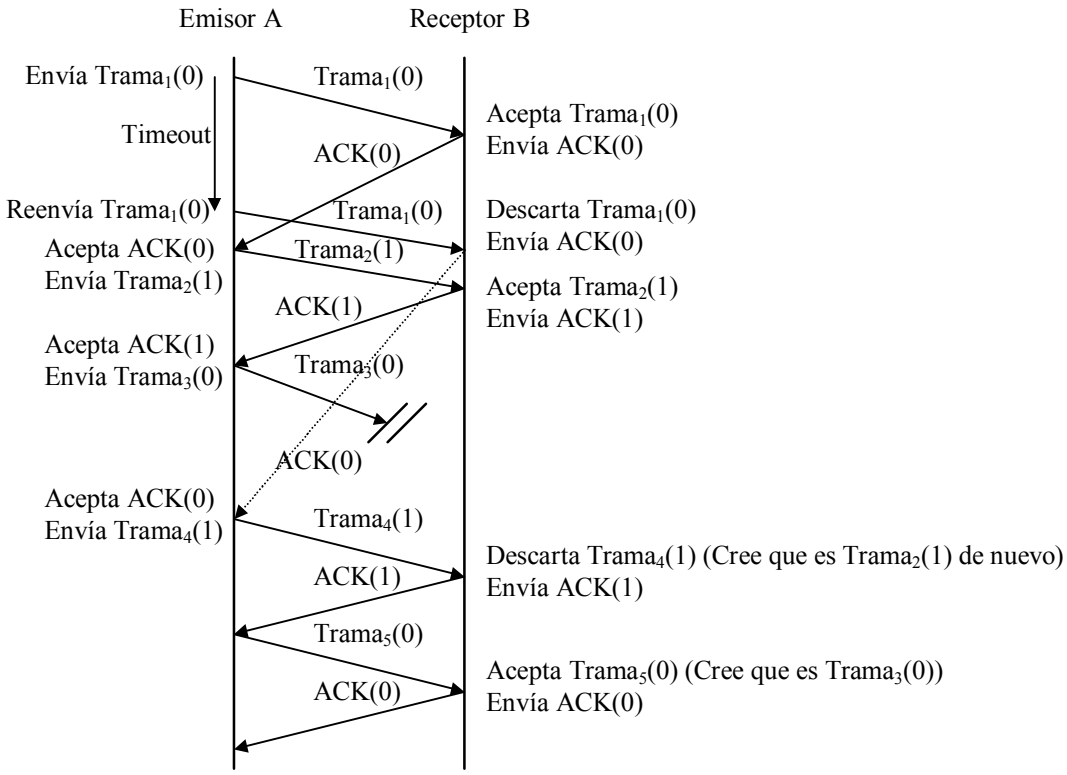
De nuevo la solución es simple: se deben numerar las tramas para que el receptor no las confunda. Supóngase que las tramas se numeran con un único bit, es decir, se envía una trama con el número 0, luego una con el número 1, la siguiente tendrá de nuevo el número 0, etc. Los números de secuencia de trama (entre paréntesis en las figuras) generalmente tienen un rango más amplio, pero se ha elegido de esta forma para facilitar la comprensión de los siguientes ejemplos.





Este protocolo también falla produciendo pérdidas de datos si por alguna razón se produce un timeout antes de la llegada de un ACK correspondiente a una trama correcta.. En ese caso el ACK correspondiente a la retransmisión de la primera trama puede engañar al emisor (el receptor descarta la trama recibida pero tiene que volver a enviar un ACK), haciéndole creer que la segunda trama (que se ha perdido) ha llegado con éxito al receptor. Al haberse completado el ciclo de los números de secuencia al receptor llega por tercera vez una trama con número de secuencia (0), por lo que es descartada. La siguiente trama en ser recibida es la cuarta, que lleva el siguiente número de secuencia que espera el receptor, el (1). El receptor cree haber recibido todas las tramas cuando en realidad le faltan dos.





Obviamente, utilizando un rango de números de secuencia más amplio es más difícil que se produzca la situación anterior. En lugar de numerar las tramas en módulo 2 (0 y 1) se pueden numerar en módulo 8 (del 0 al 7), en módulo 128 (del 0 al 127), etc. Pero el riesgo siempre existe si no se mejora el protocolo y hay que tener en cuenta que en algunos enlaces los retardos pueden ser mucho mayores de lo esperado en determinadas circunstancias. La solución más simple es que los ACK se numeren también con el número de secuencia de la trama a la que dan el acuse de recibo. Con esto tenemos ya un protocolo

completo que funcionará bajo cualquier situación anómala de las indicadas en los puntos a) y b) mencionados anteriormente, garantizando en envío secuencial de tramas sin pérdidas ni duplicados.

Aún se pueden presentar problemas si se producen situaciones como la mencionada en el punto c), la llegada de tramas fuera de secuencia, en el caso de que el rango de los números de secuencia no sea suficiente para asumir los retardos que se puedan producir en la transmisión. Esta situación se puede observar en la última figura.

Las mejoras en el protocolo que se describen a continuación, no mejoran la robustez del protocolo, si no sus prestaciones en cuanto al aprovechamiento de la velocidad del enlace.

1.4.2 Pipelining

Esta técnica consiste en enviar más de una trama antes de recibir un acuse de recibo. Si las tramas se numeran en modulo *N* se podrán enviar teóricamente *N-1* tramas antes de recibir un acuse de recibo. Aunque lo que en realidad limita su número es el tamaño del buffer del emisor, ya que ha de mantener ahí las tramas para su posible retransmisión, y del receptor, que ha de tener suficiente espacio para almacenarlas por si no tiene tiempo de ir procesándolas a medida que van llegando.

Al no tener que esperar innecesariamente por cada acuse de recibo se aprovecha mejor la capacidad del enlace, más aún, si un ACK puede servir para confirmar la recepción de más de una trama.

1.4.3 Acuse de Recibo Negativo (NAK)

Si una trama llega al receptor con errores, este puede advertir al emisor del error enviándole un NAK sin que este tenga que esperar a la finalización del timeout. Se aumenta así la velocidad de la comunicación. El NAK puede indicar también la causa del rechazo de la trama al emisor con el objeto de realizar estadísticas de los errores que se producen. Ante la recepción de un NAK el protocolo puede responder con dos mecanismos diferentes.

1.4.3.1 Vuelta Atrás

Conocido como *pullback NAK*. Si se han recibido las tramas con los números de secuencia 4, 5, 6 y 7, y se recibe un NAK para la trama 5, el emisor retransmite de nuevo las tramas 5, 6 y 7, aunque estas últimas hubieran llegado correctamente al receptor.

1.4.3.2 Repetición Selectiva

Conocido como *selective repeat*. Dada la situación indicada en el párrafo anterior, el emisor solo retransmitirá la trama 5. Aunque esto representa un notable ahorro en la capacidad del enlace, su complejidad no se justifica sino se trata de un enlace con una tasa de errores elevada.

1.4.4 Piggyback Acknowledgement

En comunicaciones donde la información fluye en ambos sentidos, sobre todo si es en modo full-duplex, se consigue un notable aumento de la capacidad del enlace si a las tramas que se envían con información se les incorpora el acuse de recibo de las que se han recibido. En el caso de que no haya información que enviar se envía una trama sólo con el ACK.

La trama llevará entonces dos números de secuencia: el que corresponde a la trama que se envía y el correspondiente al acuse de recibo de la última trama recibida correctamente.

1.4.5 Control del Flujo

El control del flujo de datos es un problema que surge cuando hay dos dispositivos intercambiando datos y el dispositivo que está recibiendo los datos no es capaz de tratar esos datos con la misma rapidez con la que le están llegando. Si no se toma ninguna medida, parte de los datos se perderán por error de solapamiento.

Las soluciones a tomar para que este error no ocurra son bastante variadas. La más sencilla sería disminuir la velocidad de transmisión de la comunicación hasta un valor en el cual no haya acumulación de datos en el otro dispositivo, es decir, igualar la velocidad de tratamiento de los datos a la velocidad de transmisión. Sin embargo, esta solución obliga a ralentizar el equipo emisor de datos, de modo que las prestaciones del sistema decaerían.

Otra posibilidad es disponer de una memoria intermedia que sirva de amortiguador en la llegada de datos de modo que, aunque los datos no estén aún tratados, sí estén almacenados para su uso posterior. Esta solución es bastante fácil de implementar, pero tiene el defecto de que esta limitada por la cantidad de memoria RAM de que disponga el dispositivo receptor de datos. Además, podría darse el caso de que la información a mandar, fuese bastante superior a la capacidad de este buffer, de modo que los datos que le llegasen después de que el buffer se llenase, se perderían.

Se ha de buscar por tanto una solución mejor que las dos anteriores. La solución es que exista una comunicación bidireccional entre los dos dispositivos intercomunicados. De modo que cuando el receptor, esté próximo a llenar su buffer, se lo indique al dispositivo emisor y éste cese de mandar datos. Cuando de nuevo existe sitio suficiente en el buffer de recepción, el receptor se lo indica al emisor y éste reanuda el flujo de datos. De esta manera los tiempos de espera del emisor, los puede dedicar éste a otras tareas y el aprovechamiento de la CPU es mayor.

La solución apuntada en el párrafo anterior es la que se conoce como control del flujo. Existen dos formas diferentes de hacer el control del flujo: *control hardware* y *control software.*

1.4.5.1 Control Hardware

Consiste en utilizar líneas dispuestas para ese fin como las que tiene la conexión RS-232-C. Este método de control del flujo de transmisión utiliza líneas del puerto serie para parar o reanudar el flujo de datos y por tanto el cable de comunicaciones, además de

las tres líneas fundamentales de la conexión serie: emisión, recepción y masa, ha de llevar algún hilo más para transmitir las señales de control.

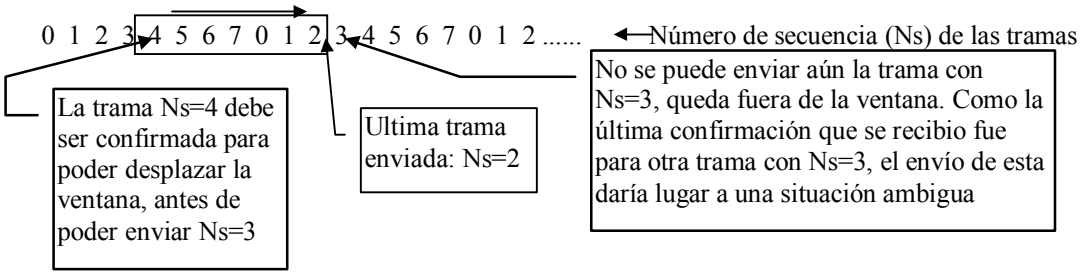
En el caso más sencillo de que la comunicación sea en un solo sentido, por ejemplo con una impresora, bastaría con la utilización de una línea más. Esta línea la gobernaría la impresora y su misión sería la de un semáforo. Por ejemplo, utilizando los niveles eléctricos reales que usa la norma serie RS-232-C, si esta línea está a una tensión positiva de 15 V. (0 lógico) indicaría que la impresora está en condiciones de recibir datos, y si por el contrario está a -15 V. (1 lógico) indicaría que no se le deben enviar más datos por el momento.

Si la comunicación es en ambos sentidos, entonces necesitaríamos al menos dos líneas de control, una que actuaría de semáforo en un sentido y la otra en el otro. Las líneas se han de elegir que vayan de una salida a una entrada, para que la lectura sea válida y además se debe tratar de utilizar las que la norma RS-232-C recomienda para este fin.

1.4.5.2 Control Software

La otra forma de control del flujo consiste en enviar a través de la línea de comunicación caracteres de control o información en las tramas que indican al otro dispositivo el estado del receptor. La utilización de un control software de la transmisión permite una mayor versatilidad del protocolo de comunicaciones y por otra parte se tiene mayor independencia del medio físico utilizado. Así por ejemplo, con un protocolo exclusivamente hardware sería bastante difícil hacer una comunicación vía telefónica, ya que las señales auxiliares de control se tendrían que emular de alguna manera.

Las formas más sencillas de control de flujo por software son el empleo de un protocolo como el XON/XOFF que se verá más adelante o como la espera de confirmación antes del envío mediante un ACK o similar como se indicaba en el ejemplo del apartado 1.4.1.



Un mecanismo más sofisticado y muy empleado es el de la *ventana deslizante*. La ventana determina cuantos mensajes pueden estar pendientes de confirmación y su tamaño se ajusta a la capacidad del buffer del receptor para almacenar tramas. El tamaño máximo de la ventana está además limitado por el tamaño del número de secuencia que se utiliza para numerar las tramas. Si las tramas se numeran con tres bits (en modulo 8, del 0 al 7), se podrán enviar hasta siete tramas sin esperar acuse de recibo y sin que el protocolo falle. Si el número de secuencia es de 7 bits (modulo 128, del 0 al 127) se podrán enviar hasta 127 tramas si es que el buffer del receptor tiene capacidad para ellas. Normalmente, si el tamaño no es prefijado por el protocolo, en el establecimiento del enlace el emisor y receptor negociarán el tamaño de la ventana atendiendo a las características del elemento que ofrece menos prestaciones [HALSALL 95].

1.4.6 Sincronización y Supervisión del Protocolo

Las tareas de sincronización y supervisión son sobre todo necesarias en enlaces que requieren el establecimiento y liberación de conexiones. En estos enlaces se envían tramas que no contienen información a transmitir, sino códigos para el control del enlace que son interpretados por la propia capa de Enlace. Generalmente, estas tramas no van numeradas con números de secuencia como las que contienen información, por lo que a veces se las denomina *tramas sin numerar*. Sus funciones suelen ser:

a) Establecimiento de la conexión: llamadas por módem, detección del interlocutor activo, negociación de parámetros de la comunicación, etc.
b) Mantenimiento de la conexión: chequeo periódico del estado del enlace, recuperación y resincronización de la comunicación tras errores o fallos temporales, etc.
c) Liberación de la conexión: liberación del enlace, desactivación de llamadas por módem, etc

2. EL PROTOCOLO PPP: POINT TO POINT PROTOCOL

El protocolo PPP, siglas correspondientes a *Point to Point Protocol*, o lo que es lo mismo, Protocolo Punto a Punto, es como su nombre indica un protocolo que se desarrollo para comunicaciones punto a punto. ¿Que le diferencia pues de los mencionados hasta este momento? (HDLC, SDLC, etc.). Su característica diferenciadora más importante es que se trata de un protocolo diseñado para dar servicio a los enlaces punto a punto de la red Internet, adecuándose a las características de ésta y a las nuevas tecnologías empleadas en ella. Se trata de un protocolo de enlace que puede servir de transporte para múltiples protocolos de red y que contiene mecanismos para establecimiento, negociación y liberación de los enlaces. Aunque esta basado en el HDLC, le supera al adecuarse mejor a las nuevas líneas de comunicación, más rápidas y con menos errores, lo que permite simplificar algunas funciones del HDLC. Se convierte además en un estándar abierto al estar recogido en los RFCs (Request For Coments), documentos que recogen el conjunto de estándares abiertos de la red Internet. Con ello supone también una mejora respecto al SLIP, un protocolo no estándar utilizado ampliamente por los usuarios que se conectaban mediante un módem y una línea telefónica a un router de la red Internet para ganar el acceso a la misma. A continuación se recoge un extracto del RFC 1331, una de las primeras definiciones del protocolo PPP que en la actualidad se recoge en los RFCs 1661, 1662, 1663 y 1717.

EXTRACTO DEL RFC 1331
==========

Network Working Group
Request for Comments: 1331
Obsoletes: RFCs 1171, 1172

W. Simpson
Daydreamer
May 1992

The Point-to-Point Protocol (PPP) for the Transmission of Multi-protocol Datagrams over Point-to-Point Links

Status of this Memo

This RFC specifies an IAB standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "IAB Official Protocol Standards" for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Abstract

The Point-to-Point Protocol (PPP) provides a method for transmitting datagrams over serial point-to-point links. PPP is comprised of three main components:

1. A method for encapsulating datagrams over serial links.

2. A Link Control Protocol (LCP) for establishing, configuring, and testing the data-link connection.

3. A family of Network Control Protocols (NCPs) for establishing and configuring different network-layer protocols.

This document defines the PPP encapsulation scheme, together with the PPP Link Control Protocol (LCP), an extensible option negotiation protocol which is able to negotiate a rich assortment of configuration parameters and provides additional management functions.

This RFC is a product of the Point-to-Point Protocol Working Group of the Internet Engineering Task Force (IETF). Comments on this memo should be submitted to the ietf-ppp@ucdavis.edu mailing list.

1. Introduction

Motivation

In the last few years, the Internet has seen explosive growth in the number of hosts supporting TCP/IP. The vast majority of these hosts are connected to Local Area Networks (LANs) of various types, Ethernet being the most common. Most of the other hosts are connected through Wide Area Networks (WANs) such as X.25 style Public Data Networks (PDNs). Relatively few of these hosts are connected with simple point-to-point (i.e., serial) links. Yet, point-to-point links are among the oldest methods of data communications and almost every host supports point-to-point connections. For example, asynchronous RS-232-C [1] interfaces are essentially ubiquitous.

Encapsulation

One reason for the small number of point-to-point IP links is the lack of a standard encapsulation protocol. There are plenty of non-standard (and at least one de facto standard) encapsulation protocols available, but there is not one which has been agreed upon as an Internet Standard. By contrast, standard encapsulation schemes do exist for the transmission of datagrams over most popular LANs.

PPP provides an encapsulation protocol over both bit-oriented synchronous links and asynchronous links with 8 bits of data and no parity. These links MUST be full-duplex, but MAY be either dedicated or circuit-switched. PPP uses HDLC as a basis for the encapsulation.

PPP has been carefully designed to retain compatibility with most commonly used supporting hardware. In addition, an escape mechanism is specified to allow control data such as XON/XOFF to be transmitted transparently over the link, and to remove spurious control data which may be injected into the link by intervening hardware and software.

The PPP encapsulation also provides for multiplexing of different network-layer protocols simultaneously over the same link. It is intended that PPP provide a common solution for easy connection of a wide variety of hosts, bridges and routers.

Some protocols expect error free transmission, and either provide error detection only on a conditional basis, or do not provide it at all. PPP uses the HDLC Frame Check Sequence for error detection. This is commonly available in hardware

implementations, and a software implementation is provided.

By default, only 8 additional octets are necessary to form the encapsulation. In environments where bandwidth is at a premium, the encapsulation may be shortened to as few as 2 octets. To support high speed hardware implementations, PPP provides that the default encapsulation header and information fields fall on 32-bit boundaries, and allows the trailer to be padded to an arbitrary boundary.

Link Control Protocol

More importantly, the Point-to-Point Protocol defines more than just an encapsulation scheme. In order to be sufficiently versatile to be portable to a wide variety of environments, PPP provides a Link Control Protocol (LCP). The LCP is used to automatically agree upon the encapsulation format options, handle varying limits on sizes of packets, authenticate the identity of its peer on the link, determine when a link is functioning properly and when it is defunct, detect a looped-back link and other common misconfiguration errors, and terminate the link.

Network Control Protocols

Point-to-Point links tend to exacerbate many problems with the current family of network protocols. For instance, assignment and management of IP addresses, which is a problem even in LAN environments, is especially difficult over circuit-switched point-to-point links (such as dial-up modem servers). These problems are handled by a family of Network Control Protocols (NCPs), which each manage the specific needs required by their respective network-layer protocols. These NCPs are defined in other documents.

Configuration

It is intended that PPP be easy to configure. By design, the standard defaults should handle all common configurations. The implementor may specify improvements to the default configuration, which are automatically communicated to the peer without operator intervention. Finally, the operator may explicitly configure options for the link which enable the link to operate in environments where it would otherwise be impossible.

This self-configuration is implemented through an extensible option negotiation mechanism, wherein each end of the link describes to the other its capabilities and requirements. Although the option negotiation mechanism described in this document is specified in terms of the Link Control Protocol (LCP), the same facilities may be used by the Internet Protocol Control Protocol (IPCP) and others in the family of NCPs.

2. Physical Layer Requirements

The Point-to-Point Protocol is capable of operating across any DTE/DCE interfaz (e.g., EIA RS-232-C, EIA RS-422, EIA RS-423 and CCITT V.35). The only absolute requirement imposed by PPP is the provision of a full-duplex circuit, either dedicated or circuit-switched, which can operate in either an asynchronous (start/stop) or synchronous bit-serial mode, transparent to PPP Data Link Layer frames. PPP does not impose any restrictions regarding transmission rate, other than those imposed by the particular DTE/DCE interfaz in use.

PPP does not require any particular synchronous encoding, such as FM, NRZ, or NRZI.

Implementation Note:

NRZ is currently most widely available, and on that basis is recommended as a default. When configuration of the encoding is allowed, NRZI is recommended as an alternative, because of its relative immunity to signal inversion configuration errors.

PPP does not require the use of modem control signals, such as Request To Send (RTS), Clear To Send (CTS), Data Carrier Detect (DCD), and Data Terminal Ready (DTR).

Implementation Note:

When available, using such signals can allow greater functionality and performance. In particular, such signals SHOULD be used to signal the Up and Down events in the Option Negotiation Automaton (described below).

3. The Data Link Layer

The Point-to-Point Protocol uses the principles, terminology, and frame structure of the International Organization For Standardization's (ISO) High-level Data Link Control (HDLC) procedures (ISO 3309-1979 [2]), as modified by ISO 3309:1984/PDAD1 "Addendum 1: Start/stop transmission" [5]. ISO 3309-1979 specifies the HDLC frame structure for use in synchronous environments. ISO 3309:1984/PDAD1 specifies proposed modifications to ISO 3309-1979 to allow its use in asynchronous environments.

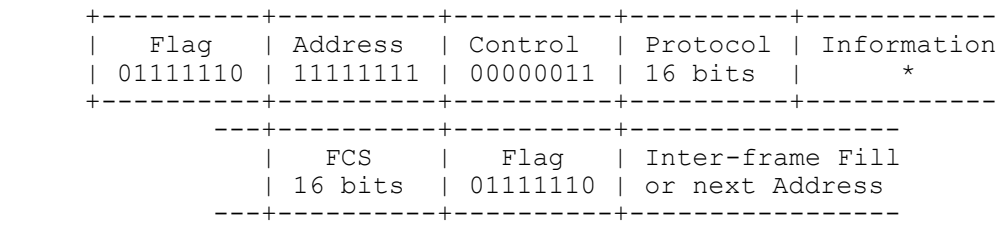
The PPP control procedures use the definitions and Control field encodings standardized in ISO 4335-1979 [3] and ISO 4335-1979/Addendum 1-1979 [4]. The PPP frame structure is also consistent with CCITT Recommendation X.25 LAPB [6], since that too is based on HDLC.

The purpose of this memo is not to document what is already standardized in ISO 3309. We assume that the reader is already familiar with HDLC, or has access to a copy of [2] or [6]. Instead, this paper attempts to give a concise summary and point out specific options and features used by PPP. Since "Addendum 1: Start/stop transmission", is not yet standardized and widely available, it is summarized in Appendix A.

To remain consistent with standard Internet practice, and avoid confusion for people used to reading RFCs, all binary numbers in the following descriptions are in Most Significant Bit to Least Significant Bit order, reading from left to right, unless otherwise indicated. Note that this is contrary to standard ISO and CCITT practice which orders bits as transmitted (i.e., network bit order). Keep this in mind when comparing this document with the international standards documents.

3.1. Frame Format

A summary of the standard PPP frame structure is shown below. This figure does not include start/stop bits (for asynchronous links), nor any bits or octets inserted for transparency. The fields are transmitted from left to right.



Inter-frame Time Fill

For asynchronous links, inter-frame time fill SHOULD be accomplished in the same manner as inter-octet time fill, by transmitting continuous "1" bits (mark-hold state).

For synchronous links, the Flag Sequence SHOULD be transmitted during inter-frame time fill. There is no provision for inter-octet time fill.

Implementation Note:

Mark idle (continuous ones) SHOULD NOT be used for idle synchronous inter-frame time fill. However, certain types of circuit-switched links require the use of mark idle, particularly those that calculate accounting based on bit activity. When mark idle is used on a synchronous link, the implementation MUST ensure at least 15 consecutive "1" bits between Flags, and that the Flag Sequence is generated at the beginning and end of a frame.

Flag Sequence

The Flag Sequence is a single octet and indicates the beginning or end of a frame. The Flag Sequence consists of the binary sequence 01111110 (hexadecimal 0x7e).

The Flag is a frame separator. Only one Flag is required between two frames. Two consecutive Flags constitute an empty frame, which is ignored.

Implementation Note:

The "shared zero mode" Flag Sequence "0111111101111110" SHOULD NOT be used. When not avoidable, such an implementation MUST ensure that the first Flag Sequence detected (the end of the frame) is promptly communicated to the link layer.

Address Field

The Address field is a single octet and contains the binary sequence 11111111 (hexadecimal 0xff), the All-Stations address. PPP does not assign individual station addresses. The All-Stations address MUST always be recognized and received. The use of other address lengths and values may be defined at a later time, or by prior agreement. Frames with unrecognized Addresses SHOULD be silently discarded, and reported through the normal network management facility.

Control Field

The Control field is a single octet and contains the binary sequence 00000011 (hexadecimal 0x03), the Unnumbered Information (UI) command with the P/F bit set to zero. Frames with other Control field values SHOULD be silently discarded.

Protocol Field

The Protocol field is two octets and its value identifies the protocol encapsulated in the Information field of the frame.

This Protocol field is defined by PPP and is not a field defined by HDLC. However, the Protocol field is consistent with the ISO 3309 extension mechanism for Address fields. All Protocols MUST be odd; the least significant bit of the least significant octet MUST equal "1". Also, all Protocols MUST be assigned such that the least significant bit of the most significant octet equals "0". Frames received which don't comply with these rules MUST be considered as having an unrecognized Protocol, and handled as specified by the LCP. The Protocol field is transmitted and received most significant octet first.

Protocol field values in the "0---" to "3---" range identify the network-layer protocol of specific datagrams, and values in the "8---" to "b---" range identify datagrams belonging to the associated Network Control Protocols (NCPs), if any.

Protocol field values in the "4---" to "7---" range are used for protocols with low volume traffic which have no associated NCP. Protocol field values in the "c---" to "f---" range identify

datagrams as link-layer Control Protocols (such as LCP).

The most up-to-date values of the Protocol field are specified in the most recent "Assigned Numbers" RFC [11]. Current values are assigned as follows:

Value (in hex)	Protocol Name
0001 to 001f	reserved (transparency inefficient)
0021	Internet Protocol
0023	OSI Network Layer
0025	Xerox NS IDP
0027	DECnet Phase IV
0029	Appletalk
002b	Novell IPX
002d	Van Jacobson Compressed TCP/IP
002f	Van Jacobson Uncompressed TCP/IP
0031	Bridging PDU
0033	Stream Protocol (ST-II)
0035	Banyan Vines
0037	reserved (until 1993)
00ff	reserved (compression inefficient)
0201	802.1d Hello Packets
0231	Luxcom
0233	Sigma Network Systems
8021	Internet Protocol Control Protocol
8023	OSI Network Layer Control Protocol
8025	Xerox NS IDP Control Protocol
8027	DECnet Phase IV Control Protocol
8029	Appletalk Control Protocol
802b	Novell IPX Control Protocol
802d	Reserved
802f	Reserved
8031	Bridging NCP
8033	Stream Protocol Control Protocol
8035	Banyan Vines Control Protocol
c021	Link Control Protocol
c023	Password Authentication Protocol
c025	Link Quality Report
c223	Challenge Handshake Authentication Protocol

Developers of new protocols MUST obtain a number from the Internet Assigned Numbers Authority (IANA), at IANA@isi.edu.

Information Field

The Information field is zero or more octets. The Information field contains the datagram for the protocol specified in the Protocol field. The end of the Information field is found by locating the closing Flag Sequence and allowing two octets for the Frame Check Sequence field. The default maximum length of the Information field is 1500 octets. By negotiation, consenting PPP implementations may use other values for the maximum Information field length.

On transmission, the Information field may be padded with an arbitrary number of octets up to the maximum length. It is the responsibility of each protocol to disambiguate padding octets from real information.

Frame Check Sequence (FCS) Field

The Frame Check Sequence field is normally 16 bits (two octets). The use of other FCS lengths may be defined at a later time, or by prior agreement.

The FCS field is calculated over all bits of the Address, Control, Protocol and Information fields not including any start and stop bits (asynchronous) and any bits (synchronous) or octets (asynchronous) inserted for transparency. This does not include the Flag Sequences or the FCS field itself. The FCS is transmitted with the coefficient of the highest term first.

Note: When octets are received which are flagged in the Async-Control-Character-Map, they are discarded before calculating the FCS. See the description in Appendix A.

For more information on the specification of the FCS, see ISO 3309 [2] or CCITT X.25 [6].

Note: A fast, table-driven implementation of the 16-bit FCS algorithm is shown in Appendix B. This implementation is based on [7], [8], and [9].

Modifications to the Basic Frame Format

The Link Control Protocol can negotiate modifications to the standard PPP frame structure. However, modified frames will always be clearly distinguishable from standard frames.

A. Asynchronous HDLC

This appendix summarizes the modifications to ISO 3309-1979 proposed in ISO 3309:1984/PDAD1, as applied in the Point-to-Point Protocol. These modifications allow HDLC to be used with 8-bit asynchronous links.

Transmission Considerations

All octets are transmitted with one start bit, eight bits of data, and one stop bit. There is no provision in either PPP or ISO 3309:1984/PDAD1 for seven bit asynchronous links.

Flag Sequence

The Flag Sequence is a single octet and indicates the beginning or end of a frame. The Flag Sequence consists of the binary sequence 01111110 (hexadecimal 0x7e).

Transparency

On asynchronous links, a character stuffing procedure is used. The Control Escape octet is defined as binary 01111101 (hexadecimal 0x7d) where the bit positions are numbered 87654321 (not 76543210, BEWARE).

After FCS computation, the transmitter examines the entire frame between the two Flag Sequences. Each Flag Sequence, Control Escape octet and octet with value less than hexadecimal 0x20 which is flagged in the Remote Async-Control-Character-Map is replaced by a two octet sequence consisting of the Control Escape octet and the original octet with bit 6 complemented (i.e., exclusive-or'd with hexadecimal 0x20).

Prior to FCS computation, the receiver examines the entire frame between the two Flag Sequences. Each octet with value less than hexadecimal 0x20 is checked. If it is flagged in the Local Async-Control-Character-Map, it is simply removed (it may have been inserted by intervening data communications equipment). For each Control Escape octet, that octet is also removed, but bit 6 of the following octet is complemented. A Control Escape octet immediately preceding the closing Flag Sequence indicates an invalid frame.

> Note: The inclusion of all octets less than hexadecimal 0x20 allows all ASCII control characters [10] excluding DEL (Delete) to be transparently communicated through almost all known data communications equipment.

The transmitter may also send octets with value in the range 0x40 through 0xff (except 0x5e) in Control Escape format. Since these octet values are not negotiable, this does not solve the problem of receivers which cannot handle all non-control characters. Also, since the technique does not affect the 8th bit, this does not solve problems for communications links that can send only 7-bit characters.

A few examples may make this more clear. Packet data is transmitted on the link as follows:

```
0x7e is encoded as 0x7d, 0x5e.
0x7d is encoded as 0x7d, 0x5d.
0x01 is encoded as 0x7d, 0x21.
```

Some modems with software flow control may intercept outgoing DC1 and DC3 ignoring the 8th (parity) bit. This data would be transmitted on the link as follows:

```
0x11 is encoded as 0x7d, 0x31.
0x13 is encoded as 0x7d, 0x33.
0x91 is encoded as 0x7d, 0xb1.
0x93 is encoded as 0x7d, 0xb3.
```

Aborting a Transmission

On asynchronous links, frames may be aborted by transmitting a "0" stop bit where a "1" bit is expected (framing error) or by transmitting a Control Escape octet followed immediately by a closing Flag Sequence.

Time Fill

On asynchronous links, inter-octet and inter-frame time fill MUST be accomplished by transmitting continuous "1" bits (mark-hold state).

Note: On asynchronous links, inter-frame time fill can be viewed as extended inter-octet time fill. Doing so can save one octet for every frame, decreasing delay and increasing bandwidth. This is possible since a Flag Sequence may serve as both a frame close and a frame begin. After having received any frame, an idle receiver will always be in a frame begin state.

Robust transmitters should avoid using this trick over-zealously since the price for decreased delay is decreased reliability. Noisy links may cause the receiver to receive

garbage characters and interpret them as part of an incoming frame. If the transmitter does not transmit a new opening Flag Sequence before sending the next frame, then that frame will be appended to the noise characters causing an invalid frame (with high reliability). Transmitters should avoid this by transmitting an open Flag Sequence whenever "appreciable time" has elapsed since the prior closing Flag Sequence. It is suggested that implementations will achieve the best results by always sending an opening Flag Sequence if the new frame is not back-to-back with the last. The maximum value for "appreciable time" is likely to be no greater than the typing rate of a slow to average typist, say 1 second.

3. APENDICE : PROTOCOLOS SIMPLES PARA TANSFERENCIAS DIRECTAS

Se tratan en este apartado algunos protocolos de enlace muy conocidos que a diferencia de los mencionados anteriormente no tienen como fin el servir como soporte de enlace para transportar información generada por las capas superiores de la arquitectura de comunicaciones en un enlace punto a punto determinado. Muy al contrario, se desarrollaron como protocolos de comunicación para aplicaciones muy concretas (y generalmente muy simples) que sólo requieren enlaces punto a punto sin saltos entre redes donde no hace falta una capa con algoritmos de encaminamiento o cosas similares [CAMPBELL 84].

3.1 Protocolo XON/XOFF

Básicamente este protocolo consiste en usar estos dos caracteres para controlar el flujo de caracteres. El carácter XON es el código 17, mientras que el XOFF es el 19. Estos caracteres se les conoce también por su nombre de la tabla ASCII. Así el carácter XON (ASCII 17) se denomina algunas veces DC1, y el carácter XOFF (ASCII 19) se le conoce también por DC3. Desde el punto de vista del teclado de un ordenador estos caracteres se suelen corresponder con control-S (XOFF) y con control-Q (XON).

El funcionamiento de este protocolo es el siguiente: cuando el receptor del mensaje desea que el emisor detenga el flujo de datos, manda un carácter XOFF (carácter de pausa) y el emisor al recibirlo detiene la emisión del mensaje. Hay que tener en cuenta que desde que se manda el carácter XOFF hasta que se interrumpe la emisión de datos, aún pueden llegar algunos datos. Por tanto no se debe esperar a tener el buffer totalmente lleno para mandar el XOFF, sino que lo habitual es mandarlo cuando, por ejemplo, está a un 75% de su capacidad.

Para que el flujo se reanude, el emisor debe recibir un carácter XON. Este carácter lo manda el receptor cuando tiene suficiente espacio en su buffer de recepción, por ejemplo cuando su nivel de llenado es del 25% .

Este protocolo funciona muy bien cuando se trata de transmitir ficheros de texto, ya que los caracteres XON (ASCII 17) y XOFF (ASCII 19) no forman parte de los caracteres usados normalmente en este tipo de ficheros. De hecho, uno de sus usos más comunes ha sido el de servir como protocolo para el envío de caracteres imprimibles hacia impresoras.

En cambio, cuando se están transmitiendo datos binarios la comunicación ha de ser half-duplex para que en un momento dado sólo esté emitiendo uno. En este caso, aunque se detecte el carácter XON o XOFF en el que emite, no se les debe dar ningún significado, ya que el que emite nunca va a tener problemas de desbordamiento del buffer. Sin embargo, al que en ese momento está recibiendo se le debe dar un tratamiento inverso. Del receptor se debe ignorar cualquier carácter, que no sea el XON o XOFF, que serán con los que regule la transmisión. En el momento que los papeles se cambien, esto es, que el emisor pase a la escucha y el receptor pase a ser el que habla, el tratamiento debe ser el contrario, para adecuarlo a la nueva situación.

Evidentemente, en el caso de la transmisión full-duplex, como ambos dispositivos están emitiendo a la vez, nos sería imposible diferenciar si realmente se nos está enviando un código de control o un dato simplemente. Por tanto para comunicaciones full-duplex necesitamos protocolos más sofisticados.

3.2 Protocolo de línea completa ETX/ACK

Este protocolo está encauzado a la transmisión de líneas de caracteres, a diferencia del anterior que está pensado para trabajar con caracteres individuales. El protocolo ETX/ACK consiste simplemente en pedir confirmación al final de cada línea de texto transmitida.

Para ello al final de cada línea se manda el carácter ETX (End of Text) que corresponde al ASCII 3. Si la recepción ha sido correcta y se está en disposición de recibir la siguiente línea, entonces el receptor responde con el carácter ACK (ACKnowledgment) que corresponde al ASCII 6, de forma que cuando el emisor recibe el ACK manda la siguiente línea.

Este protocolo es famoso por que fue el que usaron originalmente la familia de computadoras IBM VM/370. Sin embargo adolece del mismo defecto que el protocolo XON/XOFF, esto es, que funciona correctamente en comunicaciones de caracteres de texto (half o full-duplex) o binarios en modo half-duplex, pero no tiene nada que hacer en comunicaciones binarias full-duplex.

3.3 Protocolos de Transferencia de Ficheros

En este bloque se incluyen ejemplos de protocolos que controlan la transferencia de bloques arbitrarios de datos. Aunque pueden tener otras aplicaciones, su aplicación más habitual dentro de las comunicaciones asíncronas es la transferencia de ficheros. Prácticamente cualquier protocolo de transferencia de ficheros que se use en microordenadores utiliza una unidad básica llamada paquete: agrupación de varios elementos o campos formados por bytes. De estos campos, tan sólo uno contiene información. Los demás, campos de servicio, almacenan la información necesaria para que el receptor compruebe la ausencia de errores en el paquete. Lo habitual es encontrar un campo de firma del paquete que suele comenzar con el byte SOH, un número de secuencia de paquete, un campo de datos y un valor de comprobación.

3.3.1 Protocolo XMODEM

El protocolo XMODEM fue desarrollado en 1977 por Ward Christensen, el cual lo cedió para utilización pública. En honor suyo este protocolo también se le conoce como protocolo Christensen. Así mismo, Christensen desarrolló un programa de emulación de terminales llamado MÓDEM (en la actualidad MODEM7xx, donde xx es la versión), que también es de dominio público y se convirtió en un standard en el mundo de las comunicaciones. Hoy en día es raro el programa de comunicaciones por módem que no soporta el protocolo XMODEM.

El protocolo XMODEM es del tipo de los que mandan bloques de datos de cada vez. La comprobación de errores se realiza mediante una simple suma aritmética de los bytes

enviados en módulo 256. El número de bytes de datos es de 128 en cada bloque y cuando se detecta un error se repite el bloque enviado.

La estructura de cada bloque es la siguiente: el primer byte de cada bloque es el carácter SOH (Start Of Header, comienzo de cabecera), que corresponde al ASCII 1. A continuación se envía otro byte que nos indica el número de orden del bloque dentro del mensaje (en módulo 256). El primer mensaje lleva el número 1. El siguiente byte es el complemento a uno del anterior para detectar posibles errores en ese campo.

SOH	Ns	Co. Ns	"128 bytes de datos"	Checksum

A continuación se mandan 128 bytes de datos, en los cuales no tenemos ninguna restricción, es decir, que pueden ser datos binarios o cualquier carácter, sin que existan problemas de malinterpretación. Por último se envía el byte de Checksum (en módulo 256) de los 128 bytes enviados en campo de datos para el control de errores.

En cuanto a la manera de gestionar el desarrollo de la comunicación, hay que diferenciar tres fases: comienzo, envío de bloques y fin de la transmisión. En cada fase, el emisor y el receptor tienen un comportamiento complementario, que a continuación explicaremos.

Para que la transmisión comience, el receptor debe mandar el carácter NAK (ASCII 21) y esperar que le lleguen datos. Si no recibe el comienzo del bloque (SOH ASCII 1) después de 10 segundos, manda otro NAK. Por otra parte, cuando el emisor detecte el carácter NAK, comienza a enviar el bloque. Una vez enviado espera la respuesta del receptor y si esta es ACK (ASCII 6) indica que el receptor no ha detectado errores y se manda el siguiente paquete. Si la respuesta del receptor es NAK, se repite el paquete.

Cuando el emisor manda el último paquete, envía el carácter EOT (ASCII 4, End Of Transmisión) y espera que el receptor mande un ACK. Cuando esto ocurre el proceso se da por finalizado. El proceso se puede cancelar en cualquier momento mandando el carácter CAN (ASCII 24).

Para detectar los errores, el receptor calcula la suma de los caracteres que le van llegando, en módulo 256, y si no coincide con el checksum mandado entonces responde con un carácter NAK, con lo que el emisor repetirá el bloque.

El principal problema que plantea este protocolo de comunicación es el hecho de que el formato de los bloques tenga longitud fija, ya que en el último bloque parte va a ir rellenado con caracteres, para completar su longitud. Estos caracteres en algunos casos pueden ser motivo de error.

3.3.2 Protocolo Kermit

Este protocolo se diseñó con el principal objetivo de ser independiente del hardware utilizado. De hecho fue utilizado para transmitir ficheros entre una computadora DEC SYSTEM 20 y un IBM 370. Fue escrito por Frank Da Cruz y Bill Catchings en 1981 y es de dominio público sin ningún propósito comercial.

Las principales características de este protocolo son las siguientes: La longitud de los bloque es variable y se indica en el segundo carácter del bloque. Existen diferentes tipos de bloques, de cabecera, de datos, de acuse de recibo, etc., de modo que ambos dispositivos no

intercambian caracteres sueltos, sino que siempre mandan bloques. Los caracteres que se mandan son siempre imprimibles, esto implica que sean mayores del ASCII 32. Para mandar caracteres menores, se codifican de manera que aumente su ASCII y se descodifican en el receptor.

El hecho de no mandar caracteres con códigos ASCII menores al 32 es debido a que en algunos sistemas estos caracteres pueden tener asignada una misión especial que nos perturbe el funcionamiento de la transmisión.

El protocolo Kermit utiliza dos estrategias para tratar a los caracteres menores de 32, una se llama *caracterización* y la otra *codificación*.

La caracterización consiste en sumar 32 al carácter y para descaracterizarlo será por tanto preciso restar 32. Este método no se emplea con los bytes de datos, ya que para datos mayores de 223 no valdría.

Los bytes de datos no se codifican todos, sino que sólo se codifican los menores de 32 y para ello se hace la función XOR con el dato y con 64. Para descodificarlo se hace la función XOR con 64 de nuevo. Así mismo, delante del dato codificado se inserta el carácter #. Si realmente se manda el carácter # como dato, entonces se manda dos veces.

La estructura de cada bloque es la siguiente: primero se manda el carácter SOH (carácter 1, el único carácter menor que 32 que no se modifica). A continuación la longitud del mensaje "caracterizada" (longitud+32). Después un carácter que indica el tipo de mensaje. Por ejemplo la "S" indica paquete inicial, la "D" indica Datos, la "Y" indica acuse de recibo (ack), la "N" indica No reconocimiento, la "Z" indica fin de fichero, etc. A continuación se mandan los datos, codificando los menores de 32, y por último el checksum "caracterizado".

SOH	Long.	Tipo	"datos"	Checksum

El desarrollo de una comunicación es el siguiente: la comunicación la inicia el receptor mandando un NAK y a continuación el emisor manda el *bloque inicial*, en el que se indican los parámetros de la transmisión (longitud máxima del mensaje, tiempo máximo de espera de respuesta etc.). Después se espera a que el receptor mande el paquete de acuse de recibo correcto o incorrecto y si no lo reconoce se manda de nuevo. Este proceso se hace después de enviar cada bloque.

Una vez mandado el bloque inicial, se manda el *bloque de cabecera de fichero*, donde se inicia el nombre del fichero. A continuación se mandan los bloques de datos hasta acabar el fichero. Por último se manda el *bloque de final del fichero*. Si se desea mandar otro fichero se manda de nuevo el bloque de cabecera y se repite el proceso a partir de ese punto. Si no se desea mandar más ficheros, se manda el *bloque de final de transmisión* para finalizar el proceso.

4. BIBLIOGRAFÍA

[SLOMAN 87]
Sloman, M.; Kramer, J. (1987)
Distributed Systems and Computer Networks.
Prentice-Hall.

[HALSALL 95]
Halsall, F. (1995).
Data Communications, Computer Networks and Open Systems.
Addison-Wesley.

[SIMPSON 92]
Simpson, W. (1992)
RFC 1331. The Point-to-Point Protocol (PPP).
Network Working Group.

[CAMPBELL 84]
Campbell, J. (1984).
El libro del RS 232.
Anaya.