Unidad 4: La Capa de Enlace de datos

En este capítulo estudiaremos el diseño de la capa 2, la capa de enlace de datos. La propiedad fundamental de este canal es que los bits se entregan con exactitud en el mismo orden en que fueron enviados. A primera vista podría pensarse que el problema es trivial, sin embargo, los circuitos de comunicación cometen ocasionalmente errores y además tienen un retardo de propagación distinto de cero.
La capa de enlace tiene que desempeñar varias funciones específicas que incluyen: proporcionar una interfase definida con la capa de red (3) , determinar la manera en que los bits se agrupan en marcos (frames), manejar los errores de transmisión y regular el flujo de marcos para que los receptores lentos no sean abrumados por los transmisores rápidos.

Servicios proporcionados a la capa de Red

La función principal de la capa de enlace de datos es suministrar servicios a la capa de red. El principal de ellos es la transferencia de datos de la capa de red en la máquina origen a la capa de red de la máquina destino.
La capa de enlace puede ser diseñada para ofrecer varios servicios:

1. Servicio sin conexión y sin acuse: consiste en hacer que la máquina de origen envíe marcos independientes a la máquina de destino sin pedir que ésta los reconozca o acuse su recibo. No se establece una conexión previamente. Si se pierde un marco, el mismo no se intenta recuperar en la capa de enlace. Este es una clase de servicio apropiada para tasas de error muy bajas en donde la corrección de los mismos se deja para las capas superiores o también para transmisión de datos en tiempo real como la voz en donde el retardo es más crítico que unos pocos errores en los datos transmitidos. La mayoría de las LAN utilizan este tipo de servicios en su capa de enlace.
2. Servicio sin conexión con acuse: en este tipo de servicio aun no se usan conexiones pero cada marco enviado es reconocido individualmente de manera que un marco erróneo pueda ser retransmitido a petición del receptor. Es un servicio útil para canales inestables como los inalámbricos. El uso de uno u otro de estos servicios depende de la confiabilidad del canal de transmisión.
3. Servicio orientado a la conexión con acuse: con este servicio, las máquinas origen y destino establecen una conexión antes de comenzar la transmisión de los marcos. Cada marco enviado a través de la conexión está numerado, y la capa de enlace garantiza que cada marco será recibido exactamente una vez y en el orden adecuado. En este caso, se reconocen tres fases en la transmisión que son, establecimiento de la conexión, transmisión propiamente dicha y liberación de la conexión.

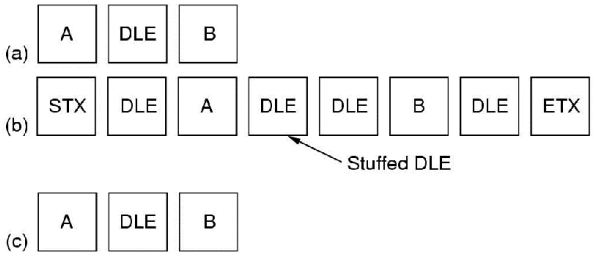
Enmarcado

A fin de proporcionar servicios fiables a la capa de red, la capa de enlace debe hacer uso de los servicios que le proporciona la capa física. Lo que hace la capa física es aceptar un flujo de bits en bruto e intentar entregarlo al destino. No se garantiza que dicho flujo esté libre de errores. Es responsabilidad de la capa de enlace detectar y corregir estos errores.
El enfoque más común es que la capa de enlace divida el flujo en marcos discretos de manera de poder hacer una comprobación de errores que viaja junto al marco y que puede ser calculada por el destino para saber si se deslizó algún error.
La división en marcos del flujo de bits no es una tarea trivial como parece, uno de los métodos usados es utilizar intervalos de tiempo para dividir los marcos, aunque esta no es una buena técnica debido a lo complejo de mantener una temporización coherente en distintos componentes de la red. Para suplir esta falencia proponemos 4 métodos:

1. Conteo de caracteres: se vale de un campo del encabezado que especifica el número de caracteres que lleva el marco. El problema que tiene es que la mayor parte de los errores alterará esta cuenta en el destino.
2. Caracteres de inicio y fin con relleno de caracteres: este segundo método supera el problema de la resincronización tras un error colocando los caracteres DLE STX[1] al comienzo del marco y DLE ETX[2] al final del marco, aun así se usa el método anterior como complemento. El problema que

[1] Data Link Exchange – Start of TeXt
[2] Data Link Exchange – End of TeXt

tiene es que cuando se transmiten datos binarios es factible que aparezcan las combinaciones de caracteres antedichas como parte de los datos a transmitir, lo que interferirá en el enmarcado. Esto suele resolverse con el *relleno de caracteres* que consiste en que el emisor agregue un carácter DLE antes del carácter DLE " accidental". De esta manera, en el lado receptor, cuando se encuentre dos DLE seguidos eliminará uno de ellos y recuperará los datos enviados tal cual. Este método tiene la desventaja de estar muy relacionado a la transmisión de datos en ASCII y se vuelve muy ineficiente para otro tipo de contenidos.



3. Indicadores de inicio y fin con relleno de bits: el tercer método es similar al anterior con la diferencia de que se maneja a nivel de bits, y el marco no tiene una longitud fija. Cada marco comienza y termina con un patrón especial 01111110, llamado byte **indicador**. Cada vez que la capa de enlace del transmisor encuentra cinco unos consecutivos, automáticamente inserta un 0 en la cadena de bits. Este relleno de bits es análogo al relleno de caracteres del método anterior. Así, cuando el receptor encuentra cinco unos seguidos de un cero automáticamente elimina el cero obteniendo la cadena original. Si el receptor pierde la pista del comienzo o final del marco solo debe explorar la entrada de caracteres en busca de las secuencias indicadoras que nunca podrán repetirse en los datos.
4. Violaciones de codificación en la capa física: es un método poco usado y consiste en cambiar el estado de medio bit con el objetivo de detectar el comienzo y fin del marco.

Control de Errores

Habiendo resuelto el problema de marcar el inicio y el final de cada marco, llegamos al siguiente problema: cómo asegurar que todos los marcos sean entregados finalmente a la capa de red destino, en el orden apropiado y sin errores.
Si encaramos el caso de la llegada de los marcos, observamos que la mejor solución sería que el receptor enviase al emisor un paquete especial confirmando la correcta llegada de cada marco. Esta solución tiene la desventaja de que si un marco llegara a perderse por completo, el receptor ni siquiera se enteraría, para solucionar este tema, se utilizan temporizadores que se disparan en el emisor con la transmisión de cada paquete. Si en un tiempo prudencial, no se recibió una confirmación o una petición de retransmisión, el emisor vuelve a transmitir el marco. Aun este método tiene la falla de que en caso de pérdida de un paquete de confirmación, el emisor retransmite un marco innecesario, generando una duplicidad en el receptor. Para solucionar esto será necesario introducir un número de secuencia en los marcos de manera que el receptor pueda controlar el número y el orden de los mismos.
Códigos de detección y corrección de errores
Los diseñadores de redes han desarrollado dos estrategias básicas para manejar los errores. Una, denominada *detección de errores* implica incluir en la información transmitida suficiente información para que el receptor pueda saber que ha ocurrido un error (sin saber cuál) y pida una retransmisión de la información con error. La otra implica incluir una cantidad mayor de información redundante tal que permita a receptor no solo detectar el error sino corregirlo (si hablamos de lógica binaria, solo hace falta saber dónde está el error y cambiar el valor actual por el inverso).
Es evidente que el porcentaje de información redundante necesaria para corregir errores es mucho mayor que el necesario para detectarlo por lo que es claro que en pro de la eficiencia del sistema, si tenemos un medio de transmisión fiable será más apta la detección con retransmisión pero si la cantidad de errores es considerable, será preferible usar códigos de corrección.

Detección de errores

En la actualidad, el método más usado para la detección de errores es el llamado *código polinómico* o *código de redundancia* cíclica **CRC**. Este se basa en el tratamiento de una cadena de bits como representaciones de polinomios con coeficientes de 0 y 1 solamente. El emisor y el recepto debe acordar un polinomio generador fijo

antes de comenzar la transmisión. La idea es anexar una suma de comprobación al final del marco de tal manera que el polinomio representado por el marco más la suma de comprobación sea divisible por el polinomio generador. Es decir, si al realizar este cociente (en binario), queda un residuo, ha ocurrido un error.

Corrección de errores

En 1950 R. W. Hamming publicó un artículo donde establecía las bases de los códigos de detección y corrección de errores; vamos a ver ahora los aspectos fundamentales de esta teoría.

El marco que se transmite de un ordenador a otro está formado por *m* bits de datos y *r* bits redundantes, de comprobación. El marco tiene entonces una longitud $n = m + r$, y forma lo que en teoría de la codificación se denomina una **palabra codificada** o **codeword** de *n* bits.
Dadas dos codewords cualesquiera, por ejemplo 10001001 y 10110001 es fácil determinar en cuantos bits difieren aplicando la operación OR exclusivo entre ambas y contando el número de bits a 1 del resultado; por ejemplo en nuestro caso difieren en 3 bits. Este valor, el número de posiciones de bit en que dos codewords difieren, se denomina **distancia de Hamming**. Si dos codewords están separadas por una distancia *d* serán necesarias *d* conversiones de un bit (por ejemplo *d* errores de un bit) para transformar una en la otra.
El ejemplo más sencillo de código de detección de errores es el bit de paridad. El bit de paridad se elige de forma que mantenga la paridad (par o impar) de la codeword. El código formado con un bit de paridad tiene una distancia de 2, ya que cambiando un bit de cualquier codeword el resultado es ilegal, pero cambiando dos vuelve a serlo. Con una distancia 2 es posible detectar errores de 1 bit, pero no detectar errores múltiples. Tampoco es posible corregir errores con este código. A cambio se trata de un código de mínimo overhead, ya que supone añadir solamente un bit a cada codeword. Por este motivo el bit de paridad se utiliza en situaciones donde la fiabilidad es muy alta y la codeword muy pequeña, como sistemas de memoria RAM o grabación de datos en soporte magnético.
La *eficiencia* de un código viene dada por la relación *m/n*; a la diferencia *1-m/n* se la denomina *redundancia*. Por ejemplo al utilizar un bit de paridad para acceder a un byte de memoria se tiene una eficiencia de 0,8889 y una redundancia de 0,1111.
Los códigos de corrección de errores tienen una eficiencia menor que los de detección para el mismo número de bits, y salvo que el medio de transmisión tenga muchos errores no son rentables; por ejemplo, supongamos que tenemos una tasa de errores de 10^{-6} (un bit erróneo por millón) y queremos enviar marcos con 1000 bits de información útil; con corrección de errores necesitaremos 10 bits de comprobación por cada marco (eficiencia de 0,99); con detección de errores cada marco deberá llevar únicamente un bit de comprobación (0,999 de eficiencia) y tendremos que retransmitir un marco de cada 1,000 (0,999 de eficiencia) lo cual da una eficiencia total de 0,998 (0,999 * 0,999). Por este motivo los códigos de corrección de errores solo suelen utilizarse cuando el medio de transmisión es simplex, ya que en ese caso el receptor no puede solicitar retransmisión. En ocasiones los códigos de corrección de errores se denominan *corrección de errores hacia adelante* (*forward error control*) y los de detección se llaman códigos de *corrección de errores hacia atrás* o por realimentación (*feedback* o *backward error control)*

PROTOCOLOS ELEMENTALES A NIVEL DE ENLACE

Una vez resuelto el problema de como verificar que la transmisión se ha producido sin errores, vamos a estudiar en detalle los mecanismos que se utilizan para la transmisión de la información al nivel de la capa de enlace entre el emisor y el receptor. En la práctica es bastante normal que las funciones de la capa física y la capa de enlace se implemente en hardware, y sean ejecutadas por un chip de entrada/salida que atiende a la puerta física correspondiente.

Un protocolo simplex sin restricciones

Inicialmente vamos a suponer que la comunicación es perfecta, sin errores, por lo que no tenemos que preocuparnos de comprobar el CRC ni de retransmisiones. El receptor está siempre disponible y preparado para recibir los marcos con un espacio de buffer infinito, por lo que no hemos de efectuar control de flujo. El emisor está siempre preparado para transmitir cualquier cosa que reciba de la capa de red. En este caso el único evento posible es *frame_arrival*.

Un protocolo simplex de parada y espera

En una primera aproximación a la vida real supongamos ahora que el receptor no siempre está disponible para recibir, por tener ocupado su buffer de entrada; esto puede ocurrir bien porque la capa de enlace no sea capaz de 'digerir' los marcos con la suficiente rapidez, o porque la capa de red del receptor no sea lo bastante rápida (podría ser que el mismo proceso tenga que atender varias líneas, por ejemplo).
En este caso lo más sencillo es que el emisor espere confirmación después de enviar cada marco , de forma que sólo después de recibir la confirmación envíe la siguiente. De esta manera se garantiza la no saturación del receptor. Esto es lo que se conoce como un protocolo de *parada y espera.*
Aunque la transmisión de datos ocurre únicamente en un sentido, este protocolo requiere un canal dúplex para funcionar; como la comunicación no ocurre simultáneamente un canal semi-dúplex sería suficiente.
Un protocolo simplex para un canal ruidoso
Siguiendo en nuestro proceso de aproximar el protocolo a la realidad contemplemos ahora la posibilidad de que el canal de comunicación no sea perfecto; los marcos pueden alterarse debido al ruido de la comunicación, o incluso perderse por completo. Gracias a la presencia del campo CRC el receptor podrá detectar la llegada de un marco defectuoso, en cuyo caso pedirá retransmisión. La posibilidad de que un marco se pierda por completo introduce una complicación adicional, ya que si esto le ocurre por ejemplo a un acuse de recibo el emisor pasado un tiempo razonable enviará el marco de nuevo pensando que no ha llegado la primera vez, lo cual produciría un marco duplicado que el receptor pasaría a la capa de red; esto es inaceptable para cualquier protocolo.
Para poder reconocer cuando un marco llega duplicado lo más sencillo es numerarlos; en nuestro caso el emisor no enviará un marco hasta recibir el acuse de recibo de la anterior, por lo que nos bastaría con numerar los marcos con un campo de un solo bit como 0,1,0,1, etc. (módulo 2). Los 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).
De nuevo, la transmisión solo ocurre en un sentido a la vez, por lo que un canal semi-dúplex sería suficiente. Ha sido necesario introducir un *timer* en el emisor para evitar la posibilidad de que al perderse un marco se quedara esperando eternamente la confirmación; así pues la retransmisión se hará por dos circunstancias: recepción errónea o agotamiento del plazo establecido. El receptor no requiere establecer ningún timer; simplemente analiza si el marco recibido es el esperado, en cuyo caso lo pasa a la capa de red, o de lo contrario lo ignora; en cualquier caso envía un marco de acuse de recibo al emisor.
Obsérvese que en este protocolo el receptor no realiza la comprobación del campo CRC; para él todo marco que reciba de la capa física es correcto; se supone que los marcos pueden llegar o perderse, pero no llegar de forma parcial o alterada. La realidad no es así, como ya sabemos. Podemos considerar que en nuestro caso hay un nivel inferior que se ocupa de comprobar el CRC, y que descarta el marco en caso de detectar cualquier error. De esta forma el efecto sería equivalente a la suposición simplista de que los marcos o no llegan o llegan perfectamente. Generalmente todo lo relativo al control de errores y cálculo de CRCs se realiza en rutinas implementadas en el hardware de comunicaciones de los equipos, y es por tanto relativamente independiente del protocolo utilizado. En cierto modo podemos considerar el cálculo de CRCs como la parte 'baja' de la capa de enlace.

Protocolo de ventana corrediza

Los protocolos que hemos visto hasta ahora transmitían datos en una sola dirección; el canal de retorno era utilizado únicamente para enviar marcos de acuse de recibo (ACK) cuyo contenido era irrelevante. Si tuviéramos que transmitir datos en ambas direcciones podríamos utilizar dos canales semi-dúplex con los protocolos anteriores, pero sería más eficiente utilizar el canal semi-dúplex ya existente para enviar en cada sentido tanto marcos de datos como de ACK; el campo *kind* nos permitirá diferenciar unas de otras.
Aún más eficiente sería, en vez de generar una ACK de manera automática cada vez que se recibe algo, esperar a enviarla cuando haya información útil que enviar; en tal caso el ACK viajaría 'gratis' y se ahorraría el envío de un marco . 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 un marco de datos es preciso que este llegue pronto, o de lo contrario el emisor reenviará el marco, lo cual eliminaría el pretendido beneficio del piggybacking; como no es posible saber de antemano cuando va a llegar el siguiente paquete de la capa de red, generalmente se adopta una solución salomónica: se espera un determinado tiempo y si no llega ningún paquete en ese tiempo se genera una ACK; en este caso el tiempo de espera debe ser sensiblemente inferior al timer del emisor.
En los protocolos de ventana corrediza, cada marco de salida contiene un número de secuencia con un intervalo que va de 0 hasta algún máximo. El máximo es generalmente $2^n - 1$ por lo que el número de secuencia cabe perfectamente en un campo de n bits

La esencia de todos los protocolos de ventana corrediza es que, en cualquier instante, el transmisor mantiene un grupo de números de secuencia que corresponde a los marcos que tiene permitido enviar. Se dice que esto marcos caen dentro de la **ventana transmisora**. De manera semejante, el receptor mantiene una **ventana receptora**. No es necesario que ambas ventanas tengan el mismo tamaño aunque generalmente es así.

Control de Flujo

Otro tema importante en el diseño de la capa de enlace (y también en las superiores) es el control de flujo que pretende evitar que un emisor rápido llegue a saturar a un receptor lento.
Se conocen varios esquemas de control de flujo, pero la mayoría se basa en el mismo principio. El protocolo contiene reglas bien definidas respecto al momento en que un transmisor puede enviar un marco. Estas reglas con frecuencia prohíben el envío de marcos hasta que el receptor lo haya autorizado implícita o explícitamente. Por ejemplo, cuando se establece una conexión, el receptor podría decir. " Puedes enviarme n marcos ahora, pero tras transmitirlos, no envíes más hasta que te lo indique".

La capa de enlace en Internet

La función más importante de la capa de enlace en internet es la de proveer conectividad punto apunto en enlaces dedicados y no dedicados. Fundamentalmente trataremos dos protocolos que son ampliamente utilizados en conexiones temporales del tipo telefónico. SLIP y PPP

SLIP: IP de línea serial
SLIP es el protocolo más antiguo y fue diseñado en 1984 con el objetivo de conectar estaciones SUN a Internet a través de líneas de discado y modems. La estación solo envía paquetes IP en bruto a través de la línea, con un byte indicador (0xC0) al final para delimitar el marco. Para evitar problemas con la aparición del carácter dentro de los datos, se usa el relleno de caracteres que ya vimos anteriormente. Algunas de las versiones actuales poseen una suerte de compresión de encabezados que mejora un poco la eficiencia del protocolo.
Aunque es utilizado ampliamente, SLIP tiene algunos serios problemas. En primer lugar, SLIP no efectúa ninguna clase de detección o corrección de errores por lo que deja esta tarea a las capas superiores. En segundo lugar solo reconoce el protocolo IP lo que se convierte cada vez en mayor limitación. En tercer lugar, SLIP no permite la asignación dinámica de IP por lo que ambos equipos deberán preacordar las mismas. En cuarto lugar, SLIP no posee ningún sistema de validación de usuarios.

PPP: Protocolo de punto a punto
Para mejorar la situación, el IETF estableció un grupo de investigación para desarrollar un protocolo que solucionara los problemas del SLIP. El resultado es el PPP que realiza detección de errores, reconoce múltiples protocolos, permite la negociación de direcciones IP en el momento de la conexión, permite la validación de autenticidad. En particular en referencia a la capa de enlace proporciona 3 cosas muy importantes:

1. Un método de enmarcado sin ambigüedades que además maneja la detección de errores.
2. Un protocolo adicional de control de enlace para activar líneas, probarlas, negociar opciones y desactivarlas en forma ordenada cuando ya no son necesarias. Este protocolo se llama LCP (Link Control Protocol= Protocolo de Control de Enlace).
3. Un mecanismo para negociar opciones de capa de red con independencia del protocolo de red usado.

LA SUBCAPA DE ACCESO AL MEDIO

Como ya dijimos anteriormente, la Subcapa de acceso al medio es muy importante en redes que usan canales de difusión como las LAN. En este tipo de redes, el asunto clave es determinar quién puede usar el canal cuando hay competencia por él. Los protocolos usados para determinar quién accede a un canal multiacceso pertenecen a esta subcapa de la capa de enlace llamada subcapa MAC (Médium Access Control). Esta subcapa se encuentra en la parte inferior de la capa de enlace.

Asignación estática de canal

Es la manera más sencilla de repartir un canal entre varios usuarios competidores. Normalmente se utiliza la Multiplexión por División de Frecuencia. Es decir, con N usuarios, el ancho de banda disponible se divide en N

partes iguales y se asigna una parte a cada usuario. Cuando hay un número chico de usuarios fijos con carga permanente, este método resulta simple y eficiente.
Sin embargo, cuando el número de usuarios es grande, o el tráfico es grande y varía constantemente, o cuando el tráfico es en ráfagas, el método se vuelve muy ineficiente, considerando que cuando un usuario está inactivo, el ancho de banda asignado al mismo siempre se pierde.

Asignación dinámica del canal

Como su nombre lo indica, se trata de encontrar la mejor manera de ajustar dinámicamente el reparto del canal único de comunicación. Todo lo realizado en este aspecto se basa en 5 supuestos que vamos a describir:

1. **Modelo de estación**. El modelo consiste en N estaciones independientes, cada una con la posibilidad de generar marcos. Una vez que la estación ha generado el marco, está se detiene hasta tanto haya podido transmitir el marco satisfactoriamente, luego de lo cual vuelve a generar otro marco si es necesario.
2. **Supuesto de canal único**. Hay un solo canal disponible para todas la estaciones. En lo referente al hardware todas las estaciones son iguales aunque por software es posible asignar prioridades.
3. **Supuesto de colisión**. Si dos marcos se transmiten al mismo tiempo, los mismos se solapan y la transmisión resultante es errónea. Este evento se llama **colisión**. Todas las estaciones pueden detectar colisiones. No existe otro tipo de errores en el canal.
4. Supuestos sobre el tiempo.
    a. **Tiempo continuo**. La transmisión de un marco puede comenzar en cualquier momento. No hay un reloj maestro que divida el tiempo en intervalos discretos.
    b. **Tiempo ranurado**. El tiempo se divide en intervalos discretos (ranuras). La transmisión de los marcos siempre comienza al inicio de una ranura. Una ranura puede contener 0, 1 o más marcos, correspondientes a una ranura inactiva, una transmisión con éxito , o una colisión, respectivamente.
5. Detección de actividad en el medio.
    a. **Detección de portadora**. Las estaciones pueden saber si el canal está en uso antes de intentar usarlo. Si se detecta que el canal está usado, ninguna estación intentará usarlo hasta que regrese a la inactividad.
    b. **Sin detección de portadora**. Las estaciones no pueden detectar de antemano si el canal está usado. Solo transmiten y después pueden determinar si la transmisión tuvo éxito.

EJEMPLOS DE PROTOCOLOS DE ACCESO MULTIPLE

Aloha puro
La idea básica de ALOHA (nombre que proviene del primer experimento de redes de radio en Hawai), es permitir que los usuarios transmitan cuando tengan datos que enviar. Por supuesto habrá colisiones y las estaciones tendrán la capacidad de conocer si hubo error o no. En caso de que el marco enviado sea destruido, la estación esperará un tiempo aleatorio y volverá a intentar la transmisión. El rendimiento de un sistema de este tipo no supera el 18%.

Aloha ranurado
La idea de este sistema es mejorar el rendimiento dividiendo el tiempo de transmisión en intervalos discretos, correspondientes cada uno a la duración de la transmisión promedio de un marco. Este concepto obliga a establecer una estación como "de sincronización" que se encargará de emitir una señal que marque el comienzo de cada intervalo. En este sistema, las estaciones no pueden transmitir en cualquier momento sino que deberán esperar hasta el comienzo de una ranura. El rendimiento de este sistema alcanza el 37%.

Protocolo de acceso múltiple con detección de portadora
Los protocolos en los que las estaciones detectan previamente una transmisión en progreso se llaman **protocolos de detección de portadora**. Veremos ahora un par de ejemplos de estos protocolos:

1. **CSMA persistente y no persistente**. CSMA es la abreviación en ingles de *Acceso Múltiple con detección de portadora* . En este protocolo, las estaciones tienen la capacidad de escuchar el canal para ver si otra estación está transmitiendo. En el caso persistente, la estación continua la escucha del canal hasta tanto se desocupe para transmitir. En el caso del no persistente, la

estación espera un tiempo aleatorio antes de volver a escuchar el canal. En ambos casos si al escuchar el canal, este se encuentra desocupado, la estación comienza la transmisión. Este protocolo alcanza un rendimiento máximo del 50%.

2. **CSMA con detección de colisiones**. Es una mejora del anterior ya que supone que las estaciones aborten la transmisión tan pronto como detecten la colisión ahorrando tiempo y ancho de banda. Una vez que una estación detecta la colisión, aborta la transmisión, espera un tiempo aleatorio e intenta de nuevo. Un ejemplo de este tipo de protocolo es el IEEE 802.3 altamente conocido.

Protocolo de LAN inalámbrica.
Cuando hablamos de redes inalámbricas hemos de tener en cuenta un factor adicional respecto a las redes alambradas; esto es el hecho de que en general, una red de este tipo tiene varias celdas de tipo celular y no podemos suponer que el medio de transmisión sea el mismo para todos los nodos celulares recetores de las señales móviles. Para ejemplificar esto veamos la figura siguiente:

Suponga que A quiere enviar un marco a D, de acuerdo al alcance de A, el canal estará desocupado pero no tiene forma de saber si el canal cercano a D está siendo utilizado por C o cualquier otro nodo.
Los protocolos de LAN inalámbrica MACA, MACW, GSM, CDMA y TDMA son algunas de la soluciones a estos problemas.